

[Design and Analysis of Algorithms]

Design and Analysis of Algorithms (CSc 523)

Samujjwal Bhandari

Central Department of Computer Science and Information Technology (CDCSIT)

Tribhuvan University, Kirtipur,

Kathmandu, Nepal.

Objectives of this Course

- Introduce data structures that are fundamental in computer science
- Introduce mathematical foundation for analysis of algorithms
- To know why we need design and analysis knowledge
- Techniques for algorithms analysis and their classifications
- Alternative ways of algorithm design
- Analyze some of the well known algorithms
- Design some kind of algorithm and analyze them

Course Introduction

- Mathematical tools for design and analysis of algorithms
- Data structure overviews
- Complexity analysis and recurrence relations
- Divide and conquer paradigm: quick sort, matrix multiplication, etc. and its application
- Greedy paradigm: minimum spanning trees-prim's, Kruskal's shortest path algorithm, etc. and its application
- Dynamic Programming: Floyd's algorithm, string matching problems, etc.
- Graph algorithms: BFS, DFS, Finding cycle in a graph, etc.
- Geometric algorithms: line segment intersection, point inclusion, etc.
- Introduction to NP-complete problems, cook's theorem, approximation algorithms.

- **Textbook:** "Introduction To Algorithms", Cormen, Leiserson, Rivest, Stein.

Prerequisites

Simple mathematical tools like recurrences, set, proof techniques, big O notation, etc.

Data structures lists, stacks, queues, graphs, trees etc.

Programming knowledge (any one).

Evaluation Criterion

Assignments.	10%
2 mid term assessments.	10%
1 final exam.	80%

Note: No late submissions of assignments are marked.

Note: The mid term assessments will cover all the completed materials before exam day.

Algorithms

An algorithm is a finite set of computational instructions, each instruction can be executed in finite time, to perform computation or problem solving by giving some value, or set of values as input to produce some value, or set of values as output. Algorithms are not dependent on a particular machine, programming language or compilers i.e. algorithms run in same manner everywhere. So the algorithm is a mathematical object where the algorithms are assumed to be run under machine with unlimited capacity.

Examples of problems:

You are given two numbers, how do you find the Greatest Common Divisor.

Given an array of numbers, how do you sort them?

We need algorithms to understand the basic concepts of the Computer Science, programming. Where the computations are done and to understand the input output relation of the problem we must be able to understand the steps involved in getting output(s) from the given input(s).

You need designing concepts of the algorithms because if you only study the algorithms then you are bound to those algorithms and selection among the available algorithms. However if you have knowledge about design then you can attempt to improve the performance using different design principles.

The analysis of the algorithms gives a good insight of the algorithms under study. Analysis of algorithms tries to answer few questions like; is the algorithm correct? i.e. the algorithm generates the required result or not?, does the algorithm terminate for all the inputs under problem domain? (*More on this later*). The other issues of analysis are efficiency, optimality, etc. So knowing the different aspects of different algorithms on the similar problem domain we can choose the better algorithm for our need. This can be done by knowing the resources needed for the algorithm for its execution.

Two most important resources are the time and the space. Both of the resources are measures in terms of complexity for time instead of absolute time we consider growth

rate (time complexity), similarly for space instead of absolute value growth rate of memory needed is considered (space complexity).

Algorithms Properties

Input(s)/output(s): There must be some inputs from the standard set of inputs and an algorithm's execution must produce outputs(s).

Definiteness: Each step must be clear and unambiguous.

Finiteness: Algorithms must terminate after finite time or steps.

Correctness: Correct set of output values must be produced from the each set of inputs.

Effectiveness: Each step must be carried out in finite time.

Here we deal with correctness and finiteness.

Expressing Algorithms

There are many ways of expressing algorithms; the order of ease of expression is natural language, pseudo code and real programming language syntax. In this course I intermix the natural language and pseudo code convention.

Random Access Machine Model

This RAM model is the base model for our study of design and analysis of algorithms to have design and analysis in machine independent scenario. In this model each basic operations (+, -) takes 1 step, loops and subroutines are not basic operations. Each memory reference is 1 step. We measure run time of algorithm by counting the steps.

Best, Worst and Average case

Best case complexity gives lower bound on the running time of the algorithm for any instance of input(s). This indicates that the algorithm can never have lower running time than best case for particular class of problems.

Worst case complexity gives upper bound on the running time of the algorithm for all the instances of the input(s). This insures that no input can overcome the running time limit posed by worst case complexity.

Average case complexity gives average number of steps required on any instance(s) of the input(s).

In our study we concentrate on worst case complexity only.

Example 1: Fibonacci Numbers

Input: n

Output: n^{th} Fibonacci number.

Algorithm: assume a as first(previous) and b as second(current) numbers

fib(n)

```
{  
     $a = 0, b = 1, f = 1;$   
    for( $i = 2; i \leq n; i++$ )  
    {  
         $f = a + b;$   
         $a = b;$   
         $b = f;$   
    }  
    return  $f;$   
}
```

Correctness

The algorithm adds previous and current values i.e. a and b , and produces n th Fibonacci number ($n > 0$) after $n-2$ iterations. So at each step the generated value will be correct. The algorithm terminates after $n-2$ iterations.

Efficiency

Time Complexity: The algorithm above iterates up to $n-2$ times, so time complexity is $O(n)$.

Space Complexity: The space complexity is constant i.e. $O(1)$.

Next is the recursive version of fib(n) as rfib(n)

```
rfib(n)
{
    if(n<2)
        return 1 ;
    else
        return( rfib(n-2)+rfib(n-1))
}
```

Correctness:

The above algorithm $rfib(n)$, for $n > 0$ produces correct output for $n=1, 2, \dots$ up to n and recursive call do obtain $rfib(1)$ and $rfib(2)$ as terminating condition i.e. every $rfib(n)$ is recursively computed unless n becomes 0 or 1.

Efficiency

Time Complexity: In the above algorithm $T(0) = t(1) = 2$ and $T(n) = T(n-2) + T(n-1) +$

3. By induction we can prove $T(n) > T(n-1)$ holds for all $n \geq 1$, and $T(n) > T(n-2)$. Using this show that $T(n)$ is at least $2^{n/2}$ i.e. $T(n) = (2^{n/2})$.

Space Complexity: The recursive call needs stack as temporary storage and that call to stack during execution of the above algorithm never exceeds n , so the space complexity is $O(n)$.

Exercises

1. You are given 12 balls of identical shape all of which, but one, are of same weight. You are also given a balance. Find the ball of the irregular weight and its type of irregularity (heavier or lighter), by using balance only 4 times.
2. Consider the **searching problem**:
Input: A sequence of n numbers $A = \langle a_1, a_2, \dots, a_n \rangle$ and a value v .
Output: An index i such that $v = A[i]$ or the special value NIL if v doesn't appear in A .
Write pseudo code for **linear search**, which scans through the sequence, looking for v .
3. Explore the word algorithm and its history.

[Graph Algorithms]

Design and Analysis of Algorithms (CSc 523)

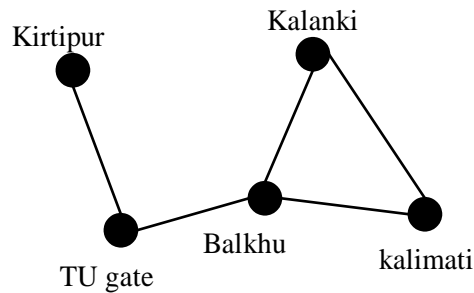
Samujjwal Bhandari

Central Department of Computer Science and Information Technology (CDCSIT)

Tribhuvan University, Kirtipur,

Kathmandu, Nepal.

Graph is a pair $G = (V,E)$ where V denotes a set of vertices and E denotes the set of edges connecting two vertices. Many natural problems can be explained using graph for example modeling road network, electronic circuits, etc. The example below shows the road network.



Representing Graphs

Generally we represent graph in two ways namely adjacency lists and adjacency matrix.

Both ways can be applied to represent any kind of graph i.e. directed and undirected.

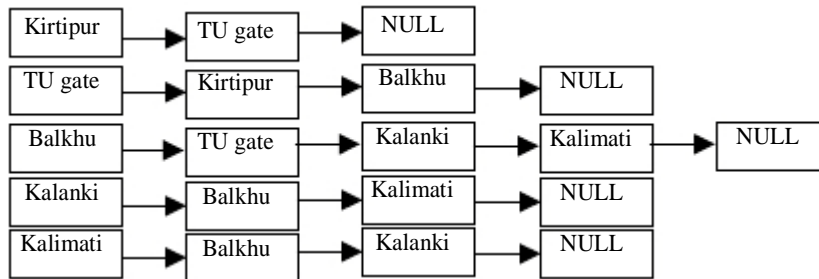
An **adjacency matrix** is an $n \cdot n$ matrix M where $M[i,j] = 1$ if there is an edge from vertex i to vertex j and $M[i,j]=0$ if there is not. Adjacency matrices are the simplest way to represent graphs. This representation takes $O(n^2)$ space regardless of the structure of the graph. So, if we have larger number of nodes say 100000 then we must have space for $100000^2 = 10,000,000,000$ and this is quite a lot space to work on. The adjacency matrix representation of a graph for the above given road network graph is given below. Take the order {Kirtipur, TU gate, Balkhu, Kalanki, Kalimati}

0	1	0	0	0
1	0	1	0	0
0	1	0	1	1
0	0	1	0	1
0	0	1	1	0

It is very easy to see whether there is edge from a vertex to another vertex ($O(1)$ time), what about space? Especially when the graph is sparse or undirected.

If **adjacency list** representation of a graph contains an array of size n such that every

vertex that has edge between the vertex denoted by the vertex with array position is added as a list with the corresponding array element. The example below gives the adjacency list representation of the above road network graph.



Searching for some edge (i,j) required $O(d)$ time where d is the degree of i vertex.

Some points:

To test if (x, y) is in graph adjacency matrices are faster.

To find the degree of a vertex adjacency list is good

For edge insertion and deletion adjacency matrix takes $O(1)$ time where as adjacency list takes $O(d)$ time.

Breadth First Search

This is one of the simplest methods of graph searching. Choose some vertex arbitrarily as a root. Add all the vertices and edges that are incident in the root. The new vertices added will become the vertices at the level 1 of the BFS tree. Form the set of the added vertices of level 1, find other vertices, such that they are connected by edges at level 1 vertices.

Follow the above step until all the vertices are added.

Algorithm:

$BFS(G,s)$ // s is start vertex

{

$T = \{s\};$

$L = \square;$ // an empty queue

$Enqueue(L,s);$

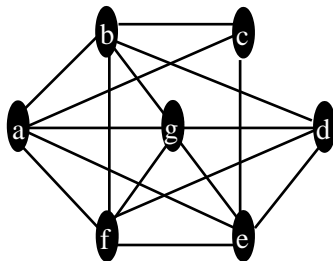
```

while (L !=  $\square$ )
{
v = dequeue(L);
  for each neighbor w to v
    if ( w  $\square$  L and w  $\square$  T )
    {
      enqueue( L,w);
      T = T * {w}; //put edge {v,w} also
    }
}
}

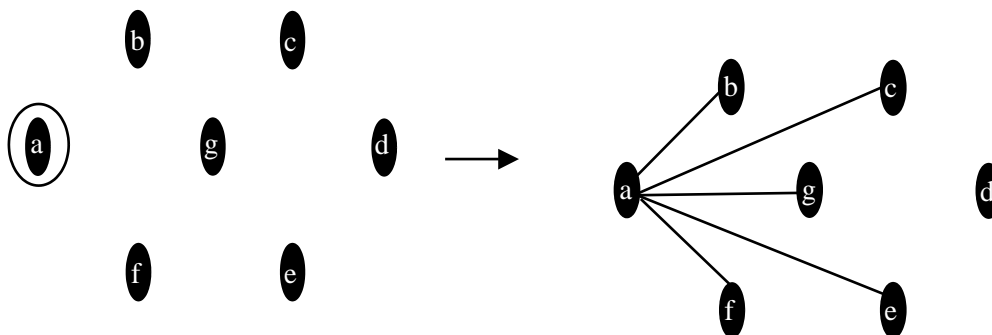
```

Example:

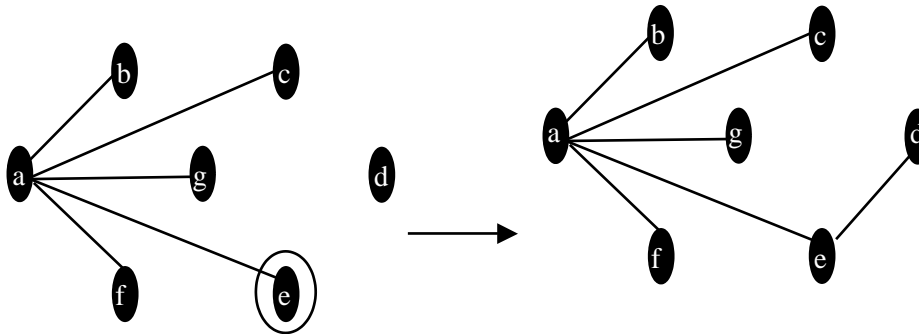
Use breadth first search to find a BFS tree of the following graph.

**Solution:**

Choose a as initial vertex then we have



Order the vertices of level 1 i.e. {b, c, g, e, f}. Say order be {e, f, g, b, c}.



Analysis:

From the algorithm above all the vertices are put once in the queue and they are accessed. For each accessed vertex from the queue their adjacent vertices are looked for and this can be done in $O(n)$ time (for the worst case the graph is complete). This computation for all the possible vertices that may be in the queue i.e. n , produce complexity of an algorithm as $O(n^2)$. Also we can write the complexity as $O(E+V)$.

Depth First Search

This is another technique that can be used to search the graph. Choose a vertex as a root and form a path by starting at a root vertex by successively adding vertices and edges, where the added edges are incident with the last vertex in the path and is not repeated. This process is continued until no possible path can be formed. If the path contains all the vertices then the tree consisting this path is DFS tree. Otherwise, we must add other edges and vertices. For this move back from the last vertex that is met in the previous path and find whether it is possible to find new path starting from the vertex just met. If there is such a path continue the process above. If this cannot be done, move back to another vertex and repeat the process. The whole process is continued until all the vertices are met. This method of search is also called **backtracking**.

Algorithm:

```
DFS(G,s)
```

```
{
```

```
  T = {s};
```

```
  Traverse(s);
```

```
}
```

```
Traverse(v)
```

```
{
```

```
  for each w adjacent to v and not yet in T
```

```
  {
```

```
    T = T * {w}; //put edge {v,w} also
```

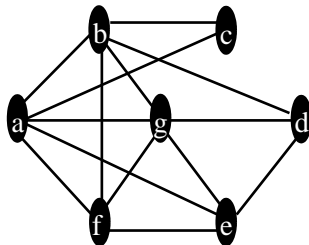
```
    Traverse(w);
```

```
  }
```

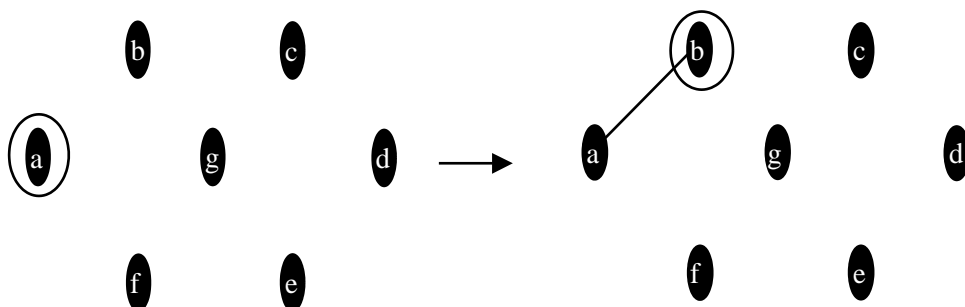
```
}
```

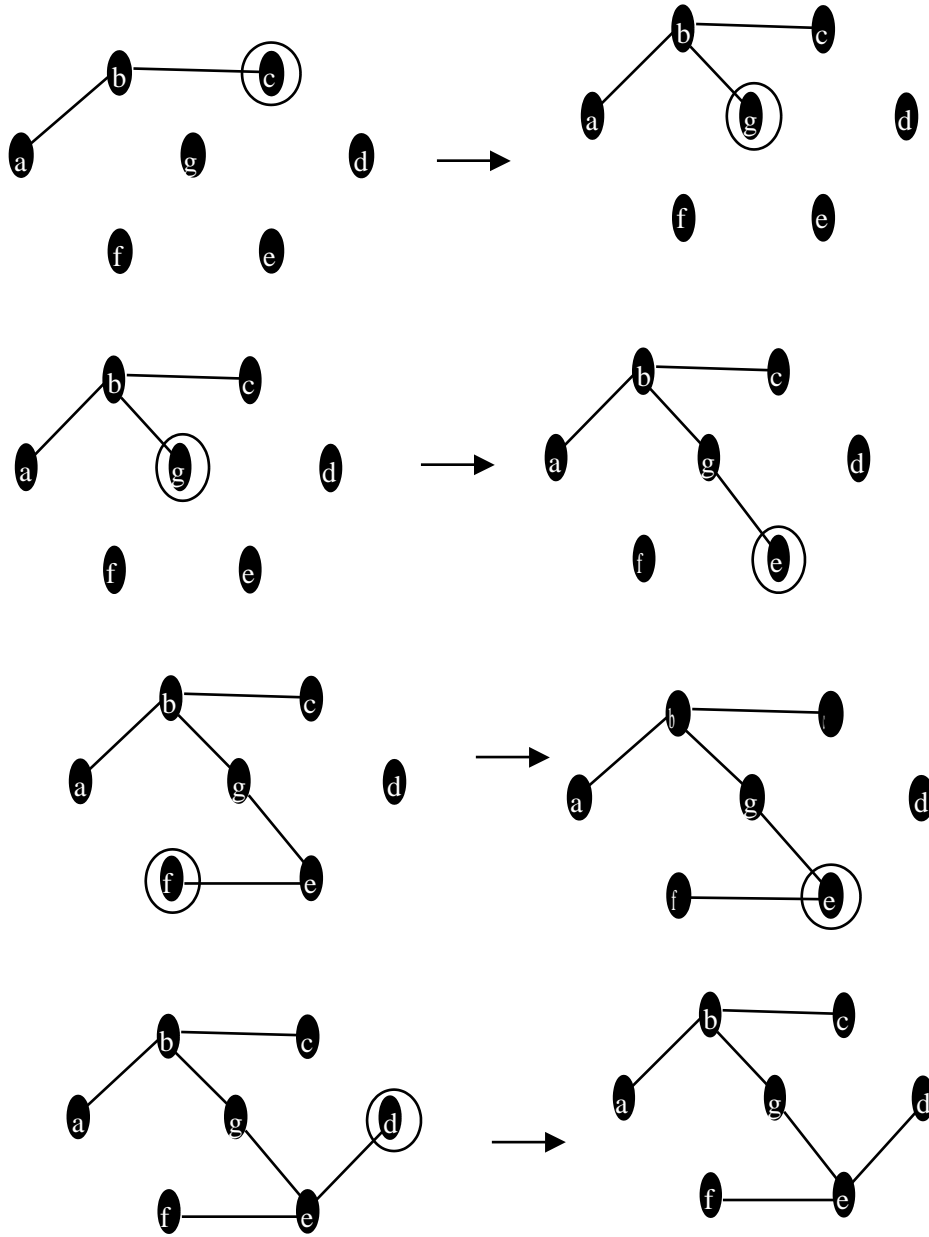
Example:

Use depth first search to find a spanning tree of the following graph.

**Solution:**

Choose a as initial vertex then we have





Analysis:

The complexity of the algorithm is greatly affected by **Traverse** function we can write its running time in terms of the relation $T(n) = T(n-1) + O(n)$, here $O(n)$ is for each vertex at most all the vertices are checked (for loop). At each recursive call a vertex is decreased. Solving this we can find that the complexity of an algorithm is $O(n^2)$. Also we can write the complexity as $O(E+V)$.

Minimum Spanning Tree

Given an undirected graph $G = (V, E)$, a subgraph $T = (V, E')$ of G is a spanning tree if and only if T is a tree. The MST is a spanning tree of a connected weighted graph such that the total sum of the weights of all edges $e \in E'$ is minimum amongst all the sum of edges that would give a spanning tree.

Kruskal's Algorithm

The problem of finding MST can be solved by using Kruskal's algorithm. The idea behind this algorithm is that you put the set of edges from the given graph $G = (V, E)$ in nondecreasing order of their weights. The selection of each edge in sequence then guarantees that the total cost that would form will be the minimum. Note that we have G as a graph, V as a set of n vertices and E as set of edges of graph G .

Algorithm:

KruskalMST(G)

{

T = {V} // forest of n nodes

S = set of edges sorted in nondecreasing order of weight

while(|T| < n-1 and E != ∅)

{

Select (u,v) from S in order

Remove (u,v) from E

if((u,v) doesnot create a cycle in T)

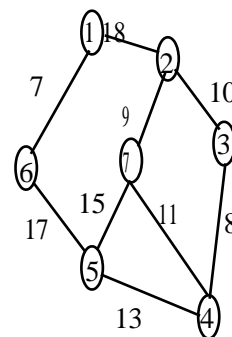
*T = T * {(u,v)}*

}

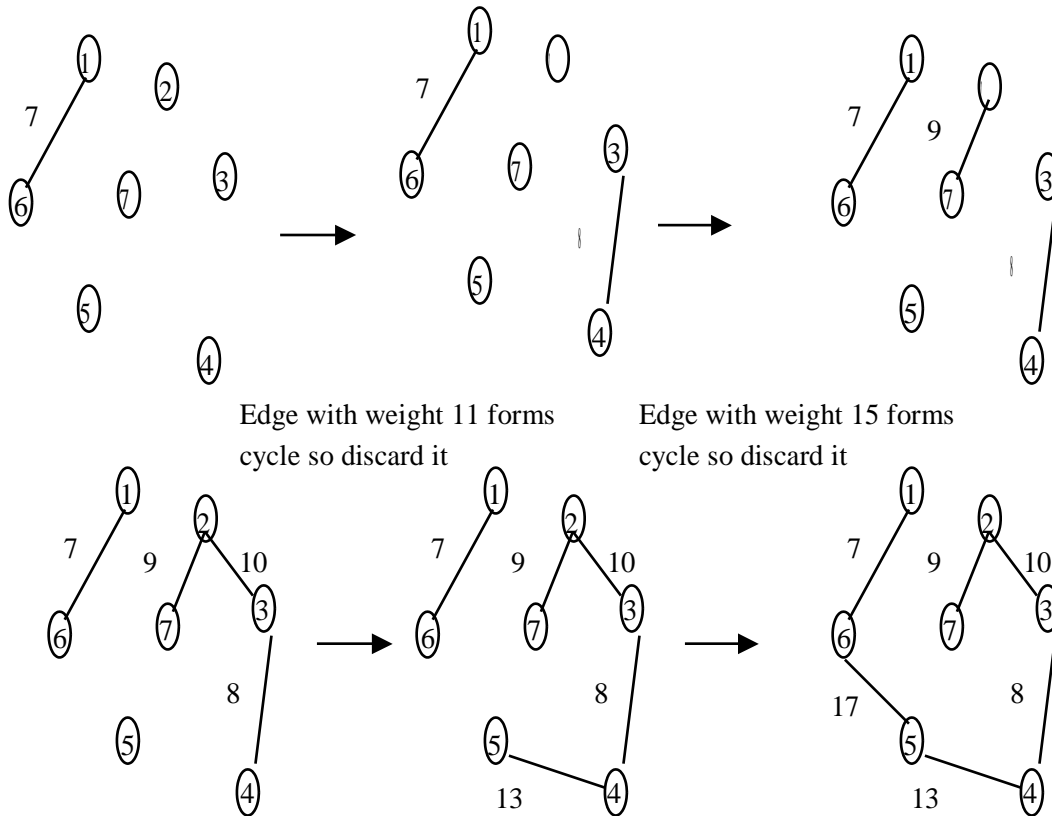
}

Example:

Find the MST and its weight of the graph.



Solution:



The total weight of MST is 64.

Analysis:

In the above algorithm the n tree forest at the beginning takes (V) time, the creation of set S takes $O(E \log E)$ time and while loop execute $O(n)$ times and the steps inside the loop take almost linear time (see disjoint set operations; find and union). So the total time taken is $O(E \log E)$ or asymptotically equivalently $O(E \log V)$!.

Theorem 1: Kruskal's algorithm generates a minimum spanning tree for every connected undirected weighted graph.

Proof:

Let $G = (V,E)$ be any connected undirected weighted graph. Let T be the spanning tree generated by Kruskal's algorithm. Let T' be the MST of the graph G . We have to show that both T and T' have same cost.

If G has n number of vertices then both T and T' must have $n-1$ edges. Say $E(T)$ is the set of edges in tree T and $E(T')$ is the set of edges in tree T' . If we have $E(T) = E(T')$ then

we can say that both tree T and T' have the same cost thus the cost is minimum since T' is MST. If $E(T) \neq E(T')$, then take an minimum cost edge $e \in E(T)$ and $e \notin E(T')$. Such an edge must exist. Now if we include the edge e in tree T' it will form a cycle. Let the cycle be e, e_1, \dots, e_k . Then it is clear that at least one of the edges from tree T' is not in $E(T)$ otherwise T also would contain cycle e, e_1, \dots, e_k . Take any edge e_j from the cycle such that $e_j \notin E(T)$. If e_j is of the lower cost than e then algorithm will consider e_j before e and include e_j in T . So we have cost of $e_j \leq$ cost of e since $e_j \notin E(T)$. Consider a graph with edge set $E(T') + \{e\}$, where there is a cycle e, e_1, \dots, e_k . If we remove any edge from the cycle another tree will be formed say T'' . If the removed edge is e_j , then it is guaranteed that the resulting tree T'' will have no more cost than that of T' , hence T'' is also MST. If we take as many as required edges from the T' that is not in T and remove the cycle as described above then the tree so formed after all transformations will be transformation from T' to T with the cost no more than T' .

Prim's Algorithm

This is another algorithm for finding MST. The idea behind this algorithm is just take any arbitrary vertex and choose the edge with minimum weight incident on the chosen vertex. Add the vertex and continue the above process taking all the vertices added. Remember the cycle must be avoided.

Algorithm:

PrimMST(G)

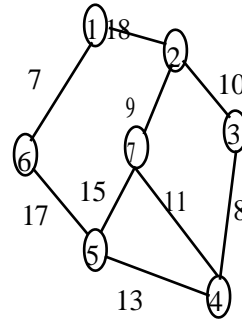
```

{ T = ∅; // T is a set of edges of MST
S = {s}; // s is randomly chosen vertex and S is set of vertices
while(S != V)
{
    e = (u,v) an edge of minimum weight incident to vertices in T and not forming a
    simple circuit in T if added to T i.e. u ∈ S and v ∉ V-S
    T = T * {(u,v)};
    S = S * {v};
}

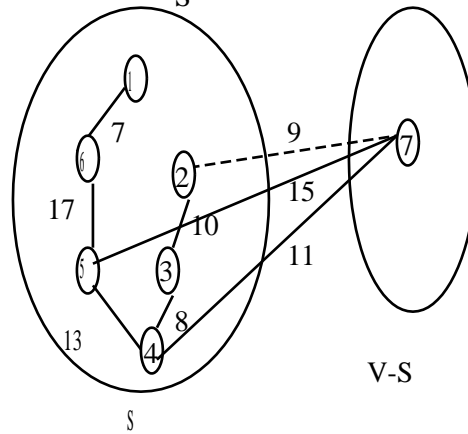
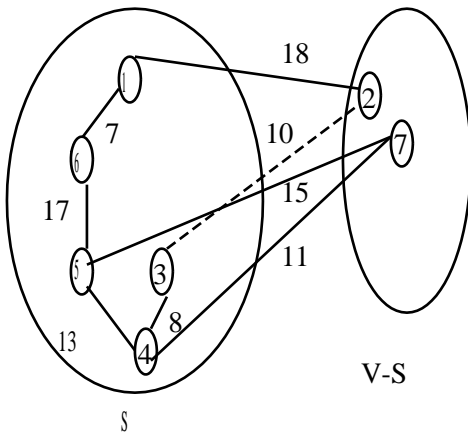
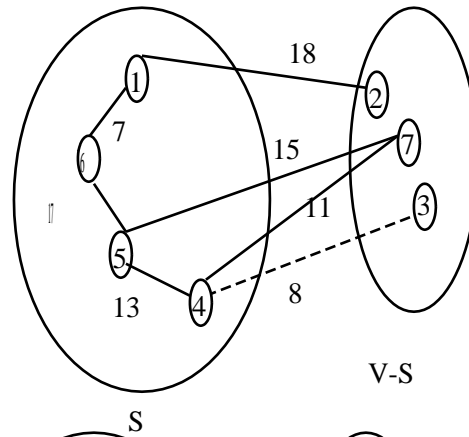
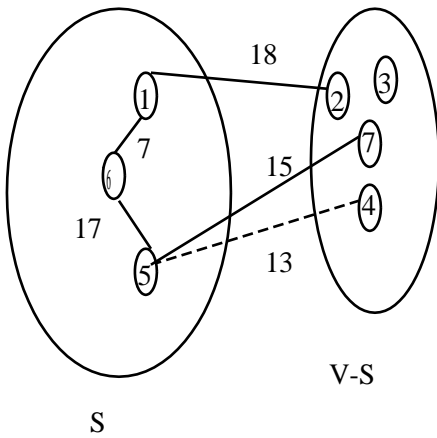
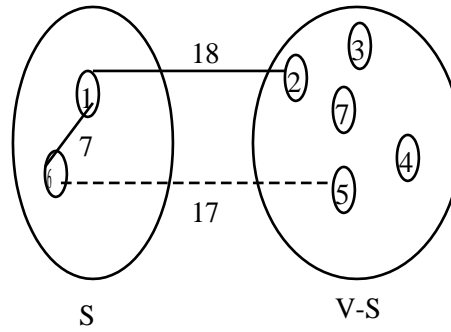
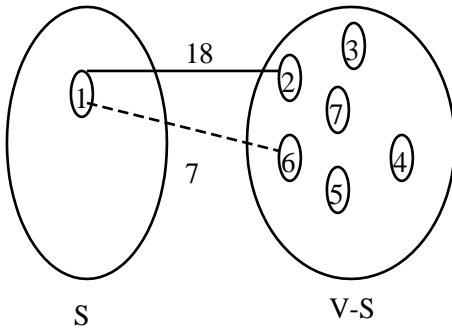
```

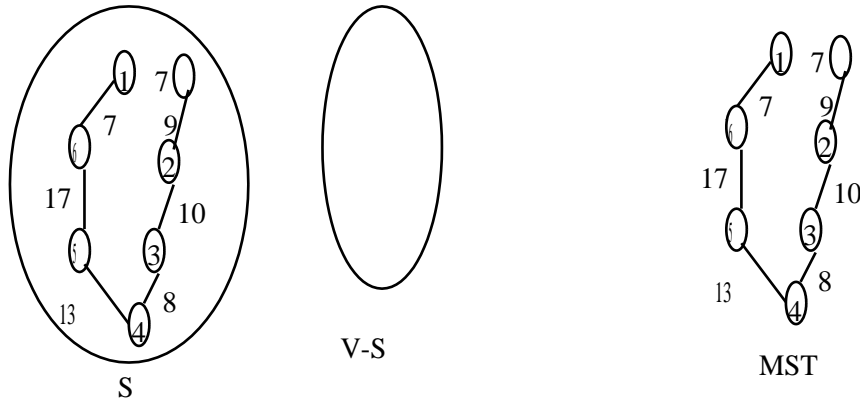
Example:

Find the minimum spanning tree of the following graph.



Solution: note: dotted edge is chosen.





The total weight of MST is 64.

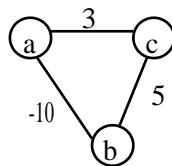
Analysis:

In the above algorithm while loop execute $O(V)$. The edge of minimum weight incident on a vertex can be found in $O(E)$, so the total time is $O(EV)$. We can improve the performance of the above algorithm by choosing better data structures as priority queue and normally it will be seen that the running time of prim's algorithm is $O(E \log V)$!

Correctness: See Assignment 4

Shortest Path Problem

Given a weighted graph $G = (V, E)$, then it has weight for every path $p = \langle v_0, v_1, \dots, v_k \rangle$ as $w(p) = w(v_0, v_1) + w(v_1, v_2) + \dots + w(v_{k-1}, v_k)$. A shortest path from u to v is the path from u to v with minimum weight. Shortest path from u to v is denoted by $TM(u, v)$. It is important to remember that the shortest path may exist in a graph or may not i.e. if there is negative weight cycle then there is no shortest path. For e.g the below graph has no shortest path from **a** to **c**. You can notice the negative weight cycle for path a to b.



As a matter of fact even the positive weight cycle doesn't constitute shortest path but there will be shortest path. Some of the variations of shortest path problem include:

Single Source: This type of problem asks us to find the shortest path from the given vertex (source) to all other vertices in a connected graph.

Single Destination: This type of problem asks us to find the shortest path to the given

vertex (destination) from all other vertices in a connected graph.

Single Pair: This type of problem asks us to find the shortest path from the given vertex (source) to another given vertex (destination).

All Pairs: This type of problem asks us to find the shortest path from the all vertices to all other vertices in a connected graph.

Optimal Substructure property for shortest path problem

Lemma 1: (subpaths of shortest paths are shortest paths)

Given a weighted, graph $G = (V, E)$ with weight function $w: E \rightarrow \mathbb{R}$, let $p = \langle v_1, v_2, \dots, v_k \rangle$ be a shortest path from a vertex v_1 to vertex v_k and, for any i and j such that $1 \leq i \leq j \leq k$, let $p_{ij} = \langle v_i, v_{i+1}, \dots, v_j \rangle$ be the subpath of p from vertex v_i to v_j . Then, p_{ij} is a shortest path from v_i to v_j .

Proof:

We have p as the path from v_1 to v_k . Lets break the path as $p_{1i} = \langle v_1, v_2, \dots, v_i \rangle$, $p_{ij} = \langle v_i, v_{i+1}, \dots, v_j \rangle$, and $p_{jk} = \langle v_j, v_{j+1}, \dots, v_k \rangle$. The we have weight of the path from v_1 to v_k as sum of paths from v_1 to v_i , v_i to v_j , and v_j to v_k i.e. $w(p) = w(p_{1i}) + w(p_{ij}) + w(p_{jk})$. If we have the path p'_{ij} that is shorter than p_{ij} then we will get the weight of the path $w(p_{1i}) + w(p'_{ij}) + w(p_{jk})$ shorter than $w(p)$, this is a contradiction that p is the shortest path, so we p_{ij} must be shortest path.

Single Source Problem

Relaxation: Relaxation of an edge (u, v) is a process of testing the total weight of the shortest path to v by going through u and if we get the weight less than the previous one then replacing the record of previous shortest path by new one.

Bellman Ford Algorithm

This is an algorithm that solves single source shortest path problem where we can have negative edges. This algorithm returns a Boolean value indicating whether or not there is a negative weight cycle that is reachable from the source. If the negative weight cycle is obtained then algorithms returns false otherwise it returns true. The main idea of this algorithm is relaxation. Distance from the source to every vertex is compared and relaxed until the shortest path is obtained.

Algorithm: $G = (V, E)$ is a weighted directed graph.

BellFordSP(G, w, s)

{

for each vertex $v \in V$

do $d[v] = \infty$ // $d[v]$ is shortest path estimate of vertex v from source.

$p[v] = Nil$ // $p[v]$ is the predecessor of v .

$d[s] = 0$

for $i = 1$ to $|V|-1$

do for each edge $(u, v) \in E$

do if $d[v] > d[u] + w(u, v)$

then $d[v] = d[u] + w(u, v)$

$p[v] = u$

Relaxation

for each edge $(u, v) \in E$

do if $d[v] > d[u] + w(u, v)$

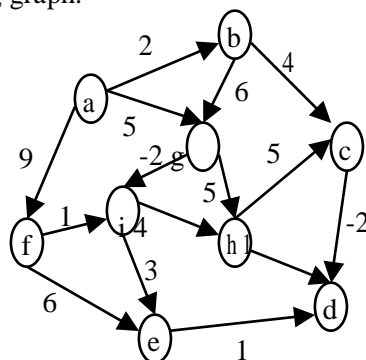
then return FALSE

return TRUE

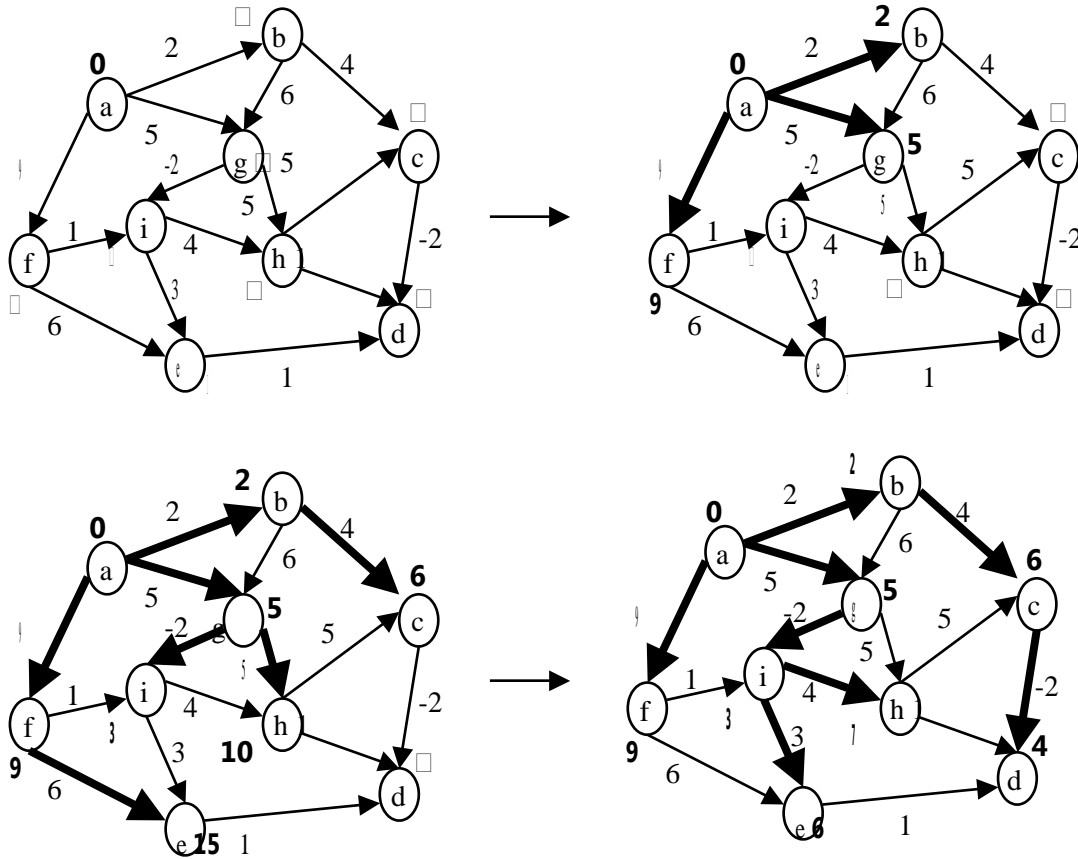
}

Example:

Find the shortest path using Bellman Ford algorithm, from the source a to all other vertices in the following graph.



Solution:

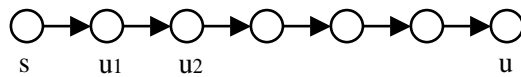


For other 5 iterations there are no changes. The returned value will be true for this example.

Analysis:

Execution of first for loop block takes $O(V)$ time. The execution of second for loop block takes $O(EV)$ time and the execution of third for loop block takes $O(E)$ time. So the total running time for above algorithm is (EV) .

Correctness:



Let u be any vertex. Assume any shortest path from s to u . All vertices on this path must be distinct otherwise we can get another shorter path so that the above assumption becomes invalid. When the first iteration of the second for loop in the above algorithm completes, we have $d[u_1] = \text{TM}(s, u_1)$, similarly after the second iteration we have $d[u_2] = \text{TM}(s, u_2)$, and so on. If we have the path from s to u containing all the vertices of the graph

G (in worst case), then $d[u]$ will be the value after $|V|-1$ iterations. So, if there are no negative weight cycles, array $d[]$ will contain stable values after the termination of the loop (when all vertices are visited) and the returned value will be TRUE. Conversely, if we get the correct output, i.e. the value returned is TRUE and there is a negative weight cycle, say $w_n = \langle v_0, v_1, \dots, v_k \rangle$ reachable from s and $v_0 = v_k$ then sum of weights of all the

edges in a cycle must be negative i.e. $w(v_{i-1}, v_i) < 0$. We have $d[v_i] \leq d[v_{i-1}] + w(v_{i-1},$

$v_i)$ (since we have return value as TRUE this condition must be true), summing up inequalities $d[v_i] \leq d[v_{i-1}] + w(v_{i-1}, v_i)$ around the cycle produces,

$$d[v_i] \leq d[v_{i-1}] + w(v_{i-1}, v_i) = d[v_{i-1}] + w(v_{i-1}, v_i) \dots \dots \dots (1)$$

Since we have $v_0 = v_k$ each vertex in c appears exactly once in each of the summations

$$d[v_i] \text{ and } d[v_{i-1}], \text{ hence } d[v_i] = d[v_{i-1}] \text{ so we have from (1)}$$

$0 \leq w(v_{i-1}, v_i) = w(c)$, this is a contradiction that there is a negative weight cycle.

This is the proof.

Directed Acyclic Graphs (Single Source Shortest paths)

Recall the definition of DAG, DAG is a directed graph $G = (V, E)$ without a cycle. The algorithm that finds the shortest paths in a DAG starts by topologically sorting the DAG for getting the linear ordering of the vertices. The next step is to relax the edges as usual.

Algorithm:

DagSP(G, w, s)

{

Topologically Sort the vertices of G

for each vertex $v \in V$

do $d[v] = \infty$

$p[v] = Nil$

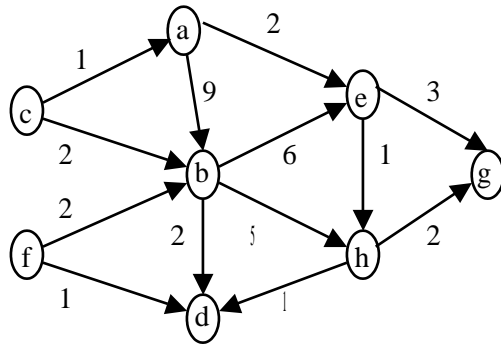
$d[s] = 0$


```

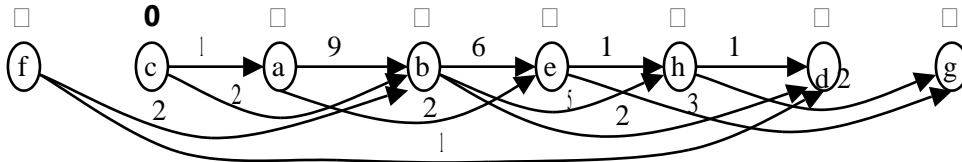
for each vertex u, taken in topologically sorted order
  do for each vertex v adjacent to u
    do if  $d[v] > d[u] + w(u,v)$ 
      then  $d[v] = d[u] + w(u,v)$ 
       $p[v] = u$ 
}
    
```

Example:

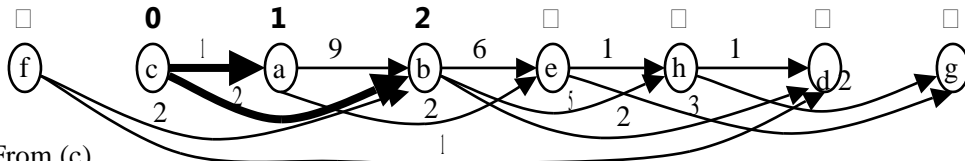
Find the shortest path from the vertex c to all other vertices in the following DAG.



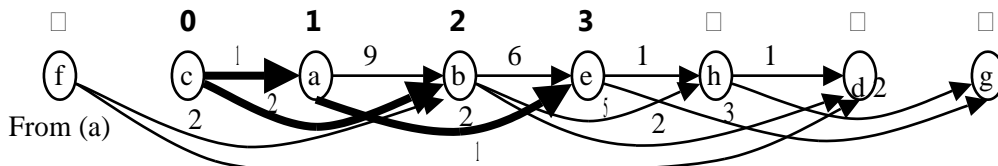
Solution:



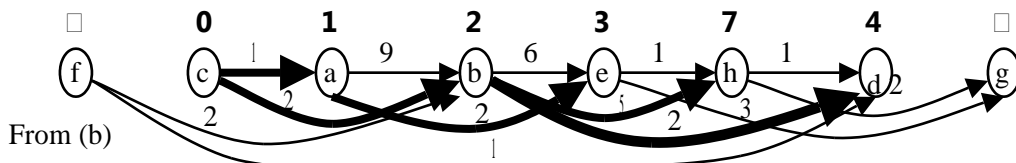
Topologically sorted and initialized.



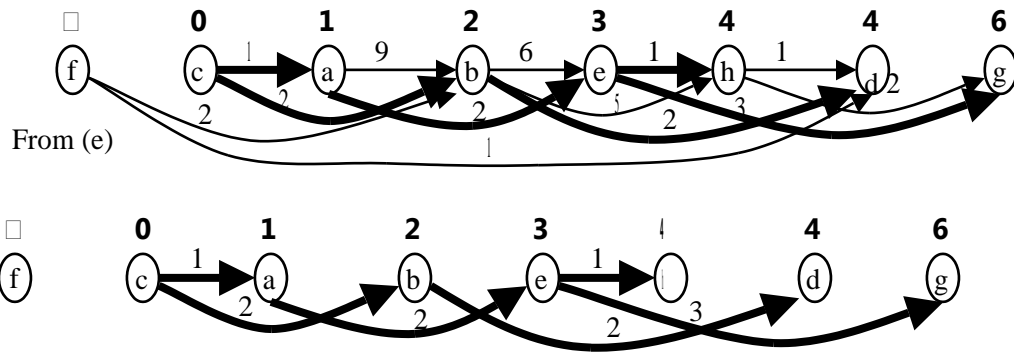
From (c)



From (a)



From (b)



From (h) (d) and (g) no change. So above is the shortest path tree.

Analysis:

In the above algorithm, the topological sort can be done in $O(V+E)$ time (Since this is similar to DFS! see book.). The first for loop block takes $O(V)$ time. In case of second for loop it executes in $O(V^2)$ Time so the total running time is $O(V^2)$. Aggregate analysis gives us the running time $O(E+V)$.

Correctness:

Lemma 2: (path relaxation property)

If $p = \langle v_0, v_1, \dots, v_k \rangle$ is a shortest path from $s = v_0$ to v_k , and the edges of p are relaxed in the order $(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)$, then $d[v_k] = TM(s, v_k)$.

Lemma 3: (Predecessor subgraph property)

Once $d[v] = TM(s,v)$ for all $v \in V$, the predecessor subgraph is a shortest paths tree rooted at s .

Theorem 2:

For a DAG $G = (V,E)$ with source vertex s , at the termination of DagSP algorithm, $d[v] = TM(s,v)$ for all vertices $v \in V$, and the predecessor subgraph G_\square is a shortest paths tree.

Proof:

If v is unreachable from s , then there is no path from s to v so we have $d[v] = TM(s,v) = \infty$.

Assume that v is reachable from s , then there is some shortest path $p = \langle v_0, v_1, \dots, v_k \rangle$, where $v_0 = s$ and $v_k = v$. Now from the above algorithm it is guaranteed that relaxation of edges on the path are done in a particular order as $(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)$ due to the topological ordering. So from the above lemmas 2 and 3 we complete the proof.

Dijkstra's Algorithm

This is another approach of getting single source shortest paths. In this algorithm it is assumed that there is no negative weight edge. Dijkstra's algorithm works using greedy approach, as we will see later.

Algorithm:

Dijkstra(G,w,s)

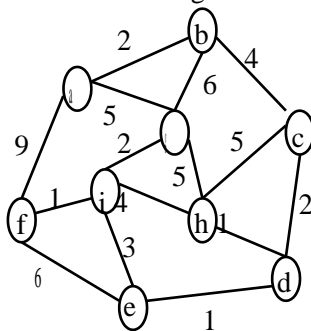
```

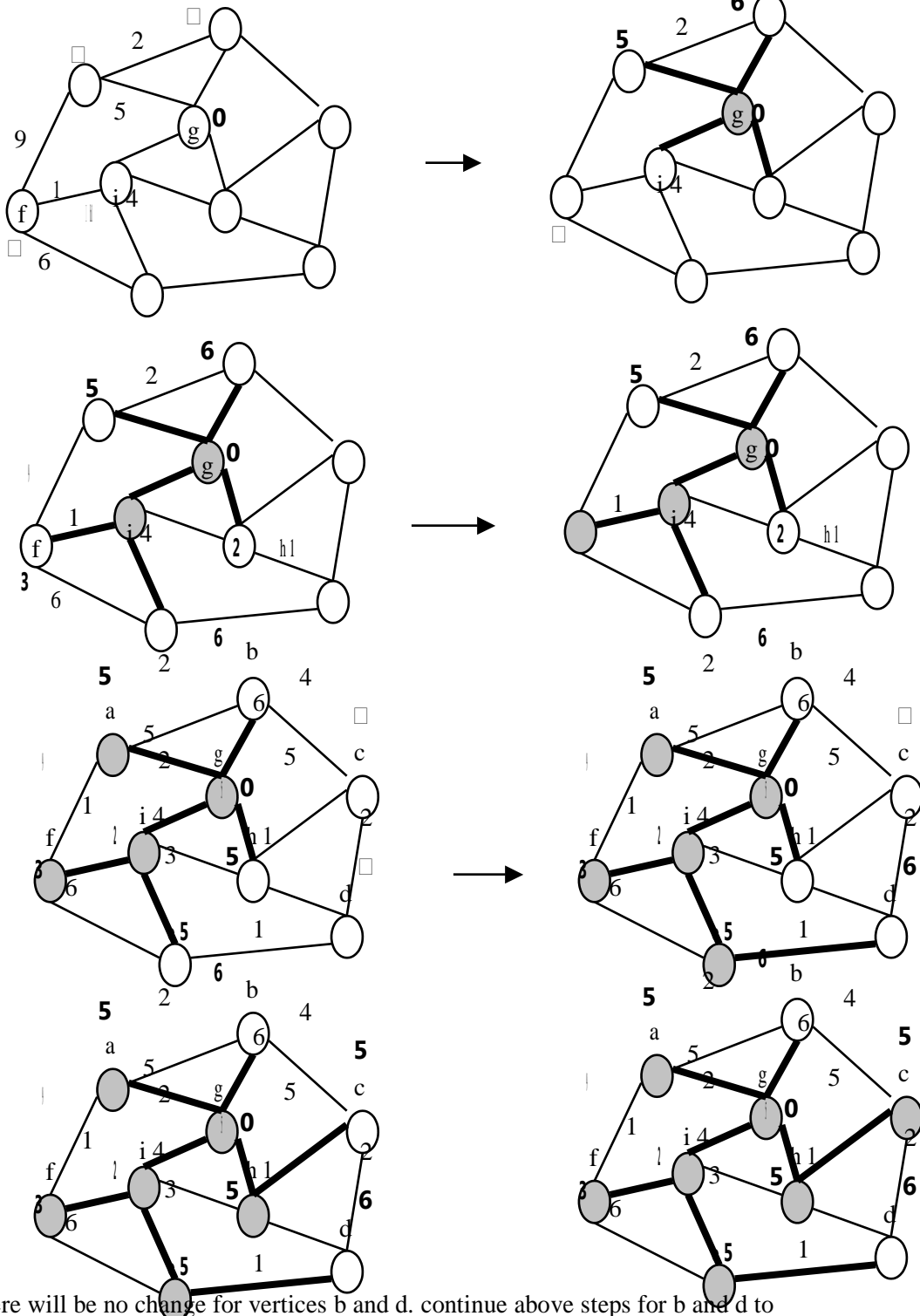
{
for each vertex  $v \in V$ 
    do  $d[v] = \infty$ 
     $p[v] = Nil$ 
 $d[s] = 0$ 
 $S = \emptyset$ 
 $Q = V$ 
While( $Q \neq \emptyset$ )
{
     $u =$  Take minimum from  $Q$  and delete.
     $S = S \cup \{u\}$ 
    for each vertex  $v$  adjacent to  $u$ 
        do if  $d[v] > d[u] + w(u,v)$ 
            then  $d[v] = d[u] + w(u,v)$ 
}
}

```

Example:

Find the shortest paths from the source g to all other vertices using Dijkstra's algorithm.





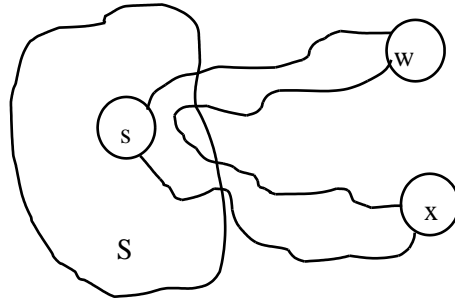
There will be no change for vertices b and d. continue above steps for b and d to complete. The tree is shown as dark connection.

Analysis:

In the above algorithm, the first for loop block takes $O(V)$ time. Initialization of priority queue Q takes $O(V)$ time. The while loop executes for $O(V)$, where for each execution the block inside the loop takes $O(V)$ times. Hence the total running time is $O(V^2)$.

Correctness:

Let S be the set of vertices to which the shortest paths from the source is already known.



Let w be the node that is not in the set S such that the length of the path connecting s to w and passing only through vertices in S is the shortest among all choices of w . Say this path as **special path**. Let us argue that any other path connecting s to w and passing through vertices outside of S cannot be shorter than the special path. Suppose that a path connecting s to w passes through a vertex x outside of S and is shorter than the special path. Then the length of the path from the s to x is shorter than the special path, this is a contradiction special path is shortest. Hence Dijkstra's algorithm is correct.

[**Remark:** You can prove for the correctness taking $d[v] = \text{TM}(s,v)$ for each vertex $v \in S$]

All Pairs Problem

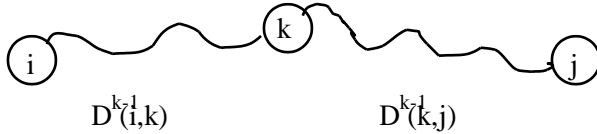
As defined in above sections, we can apply single source shortest path algorithms $|V|$ times to solve all pair shortest paths problem.

Floyd's Warshall Algorithm

The algorithm being discussed uses dynamic programming approach. The algorithm being presented here works even if some of the edges have negative weights. Consider a weighted graph $G = (V,E)$ and denote the weight of edge connecting vertices i and j by w_{ij} . Let W be the adjacency matrix for the given graph G . Let D_k denote an $n \times n$ matrix such that $D_k(i,j)$ is defined as the weight of the shortest path from the vertex i to vertex j

using only vertices from $1, 2, \dots, k$ as intermediate vertices in the path. If we consider shortest path with intermediate vertices as above then computing the path contains two cases. $D_k(i, j)$ does not contain k as intermediate vertex and $D_k(i, j)$ contains k as intermediate vertex. Then we have the following relations

$D_k(i, j) = D_{k-1}(i, j)$, when k is not an intermediate vertex, and



$D_k(i, j) = D_{k-1}(i, k) + D_{k-1}(k, j)$, when k is an intermediate vertex.

So from the above relations we obtain:

$$D_k(i, j) = \min\{D_{k-1}(i, j), D_{k-1}(i, k) + D_{k-1}(k, j)\}.$$

The above relation is used by Floyd's algorithm to compute all pairs shortest path in bottom up manner for finding D_1, D_2, \dots, D_n .

Algorithm:

FloydWarshalAPSP(W, D, n) // W is adjacency matrix of graph G .

{

for($i=1; i \leq n; i++$)

for($j=1; j \leq n; j++$)

$D[i][j] = W[i][j];$ // initially $D[][]$ is D_0 .

For($k=1; k \leq n; k++$)

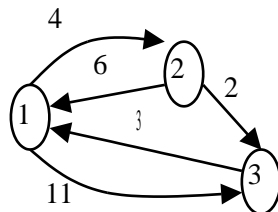
for($i=1; i \leq n; i++$)

for($j=1; j \leq n; j++$)

$D[i][j] = \min\{D[i][j], D[i][k] + D[k][j]\};$ // $D[][]$'s are D_k 's.

}

Example:



Solution:

Adjacency Matrix

W	1	2	3
1	0	4	11
2	6	0	2
3	3	□	0

D₁	1	2	3
1	0	4	11
2	6	0	2
3	3	7	0

D₂	1	2	3
1	0	4	6
2	6	0	2
3	3	7	0

D₃	1	2	3
1	0	4	6
2	5	0	2
3	3	7	0

Remember we are not showing $D_k(i,i)$, since there will be no change i.e. shortest path is zero.

$$\begin{aligned} D_1(1,2) &= \min\{D_0(1,2), D_0(1,1)+D_0(1,2)\} \\ &= \min\{4, 0+4\} = 4 \end{aligned}$$

$$\begin{aligned} D^1(1,3) &= \min\{D_0(1,3), D_0(1,1)+D_0(1,3)\} \\ &= \min\{11, 0+11\} = 11 \end{aligned}$$

$$\begin{aligned} D^1(2,1) &= \min\{D_0(2,1), D_0(2,1)+D_0(1,1)\} \\ &= \min\{6, 6+0\} = 6 \end{aligned}$$

$$\begin{aligned} \mathbf{D^1(2,3)} &= \mathbf{\min\{D_0(2,3), D_0(2,1)+D_0(1,3)\}} \\ &= \mathbf{\min\{2, 6+11\} = 2} \end{aligned}$$

$$\begin{aligned} D_1(3,1) &= \min\{D_0(3,1), D_0(3,1)+D_0(1,1)\} \\ &= \min\{3, 3+0\} = 3 \end{aligned}$$

$$\begin{aligned} \mathbf{D_1(3,2)} &= \mathbf{\min\{D_0(3,2), D_0(3,1)+D_0(1,2)\}} \\ &= \mathbf{\min\{\square, 3+4\} = 7} \end{aligned}$$

$$\begin{aligned} D_2(1,2) &= \min\{D_1(1,2), D_1(1,2)+D_1(2,2)\} \\ &= \min\{4, 4+0\} = 4 \end{aligned}$$

$$\begin{aligned} \mathbf{D^2(1,3)} &= \mathbf{\min\{D_1(1,3), D_1(1,2)+D_1(2,3)\}} \\ &= \mathbf{\min\{11, 4+2\} = 6} \end{aligned}$$

$$\begin{aligned} D^2(2,1) &= \min\{D_1(2,1), D_1(2,2)+D_1(2,1)\} \\ &= \min\{6, 0+6\} = 6 \end{aligned}$$

$$\begin{aligned} D^2(2,3) &= \min\{D_1(2,3), D_1(2,2)+D_1(2,3)\} \\ &= \min\{2, 0+2\} = 2 \end{aligned}$$

$$\begin{aligned} \mathbf{D^2(3,1)} &= \mathbf{\min\{D_1(3,1), D_1(3,2)+D_1(2,1)\}} \\ &= \mathbf{\min\{3, 7+6\} = 3} \end{aligned}$$

$$\begin{aligned} D^2(3,2) &= \min\{D_1(3,2), D_1(3,2)+D_1(2,2)\} \\ &= \min\{7, 7+0\} = 7 \end{aligned}$$

$$\begin{aligned} \mathbf{D_3(1,2)} &= \mathbf{\min\{D_2(1,2), D_2(1,3)+D_2(3,2)\}} \\ &= \mathbf{\min\{4, 6+7\} = 4} \end{aligned}$$

$$\begin{aligned} D^3(1,3) &= \min\{D_2(1,3), D_2(1,3)+D_2(3,3)\} \\ &= \min\{6, 6+0\} = 6 \end{aligned}$$

$$\begin{aligned} \mathbf{D^3(2,1)} &= \mathbf{\min\{D_2(2,1), D_2(2,3)+D_2(3,1)\}} \\ &= \mathbf{\min\{6, 2+3\} = 5} \end{aligned}$$

$$\begin{aligned} D^3(2,3) &= \min\{D_2(2,3), D_2(2,3)+D_2(3,3)\} \\ &= \min\{2, 2+0\} = 2 \end{aligned}$$

$$\begin{aligned} D_3(3,1) &= \min\{D_2(3,1), D_2(3,3)+D_2(3,1)\} \\ &= \min\{3, 0+3\} = 3 \end{aligned}$$

$$\begin{aligned} D_3(3,2) &= \min\{D_2(3,2), D_2(3,3)+D_2(3,2)\} \\ &= \min\{7, 0+7\} = 7 \end{aligned}$$

Analysis:

Clearly the above algorithm's running time is $O(n^3)$, where n is cardinality of set V of vertices.

Exercises

1. Write an algorithm for Topological sorting the directed graph.
2. Explore the applications of DFS and BFS, Describe the biconnected component and algorithm for its detection in a graph
3. Give an example graph where bellman ford algorithm returns FALSE, Justify for the falsity of the return value
4. Give an example graph where Dijkstra's algorithm fails to work. Why the found graph does not work, give reason?
5. Maximum Spanning Tree of a weighted Graph $G = (V,E)$ is a subgraph $T = (V,E')$ of G such that T is a tree and the sum of weights of all the edges of E' is maximum among all possible set of edges that would form a spanning tree.
Modify the prim's algorithm to solve for maximum spanning tree.

[Geometric Algorithms]

Design and Analysis of Algorithms (CSc 523)

Samujjwal Bhandari

Central Department of Computer Science and Information Technology (CDCSIT)

Tribhuvan University, Kirtipur,

Kathmandu, Nepal.

Computational Geometry

The field of computational geometry deals with the study of geometric problems. In our class we present few geometric problems for e.g. detecting the intersection between line segments, and try to solve them by using known algorithms. In this lecture, we discuss and present algorithms on context of 2-D.

Some Definitions

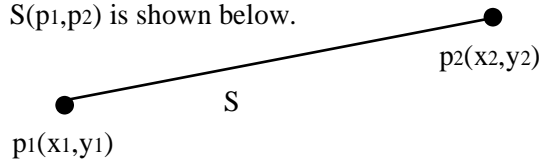
Point

A point is a pair of numbers. The numbers are real numbers, but in our usual calculation we concentrate on integers. For e.g. $p_1(x_1, y_1)$ and $p_2(x_2, y_2)$ are two points as shown below.



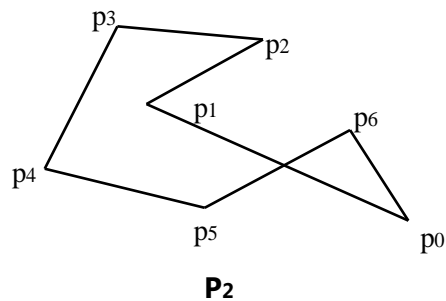
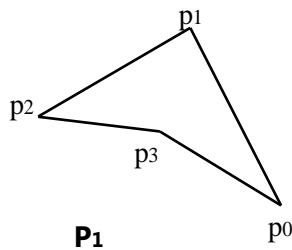
Line segment

A line segment is a pair of points p_1 and p_2 , where two points are end points of the segment. For e.g. $S(p_1, p_2)$ is shown below.



Polygon

A closed figure of n line segments (p_i, p_{i+1}) for $0 \leq i \leq n-1$ and (p_{n-1}, p_0) , where $n \geq 3$. The polygon P is represented by its vertices, usually in counterclockwise order of traversal of its boundary, $P = (p_0, p_1, \dots, p_{n-1})$ or the line segments that are ordered as $P = (S_0, S_1, \dots, S_{n-1})$ such that the end point of preceding line segment becomes starting point of the next line segment. Examples are shown below.

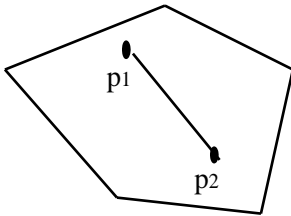


Simple Polygon

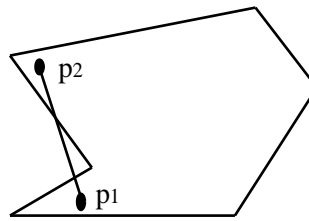
A Simple polygon is a polygon P with no two non-consecutive edges intersecting. There is a well-defined bounded interior and unbounded exterior for a simple polygon, where the interior is surrounded by edges. When referring to P , the convention is to include the interior of P . In the above figure of polygon P_1 is a simple polygon but P_2 is not.

Convex Polygon

A simple polygon P is convex if and only if for any pair of points x, y in P the line segment between x and y lies entirely in P . We can notice that if all the interior angle is less than 180° , then the simple polygon is a convex polygon.



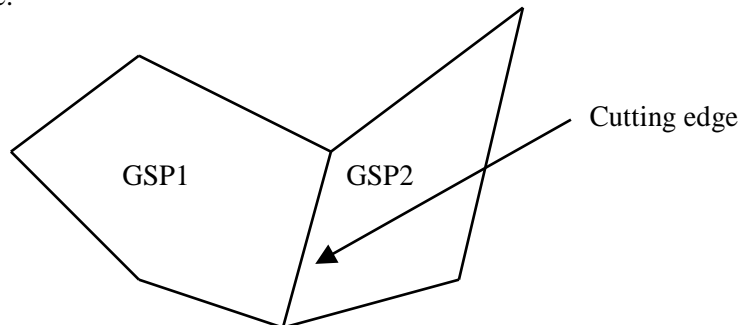
A convex polygon



A non-convex polygon

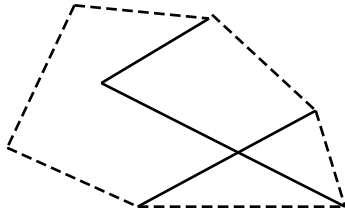
Good Sub-Polygon

A good sub-polygon of a simple polygon P , denoted by GSP, is a sub-polygon whose boundary differs from that of P by at most one edge. This edge, if it exists, is called the cutting edge.

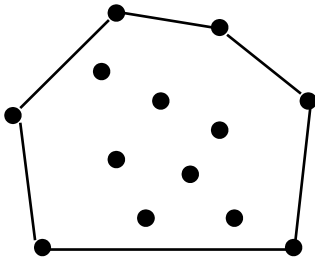
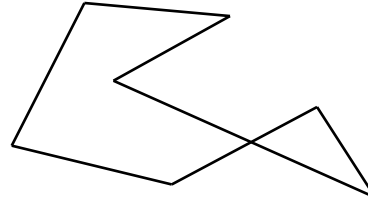


Convex Hull

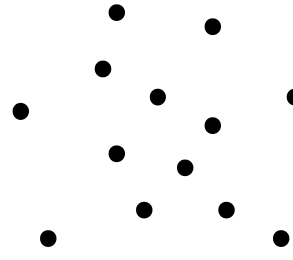
The convex hull of a polygon P is the smallest convex polygon that contains P . Similarly, we can define the convex hull of a set of points R as the smallest convex polygon containing R .



Polygon with dotted lines is the convex hull of a polygon given alongside

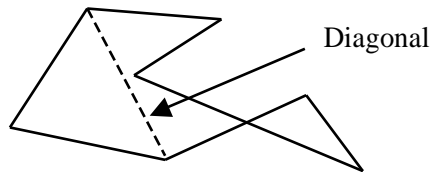


The convex hull of a set of points given alongside



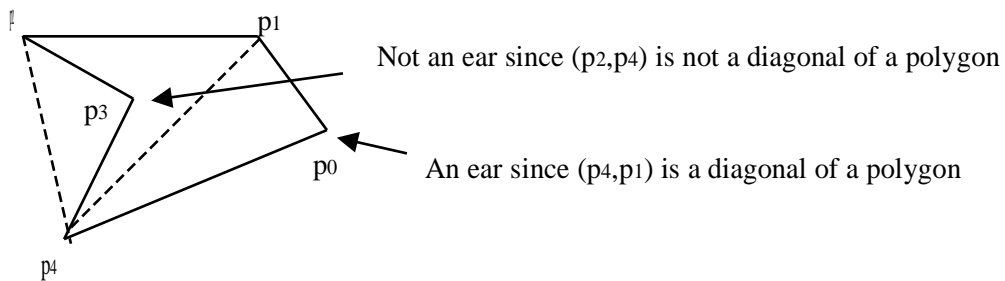
Diagonal

A line segment lying entirely inside polygon P and joining two non-consecutive vertices p_i and p_j



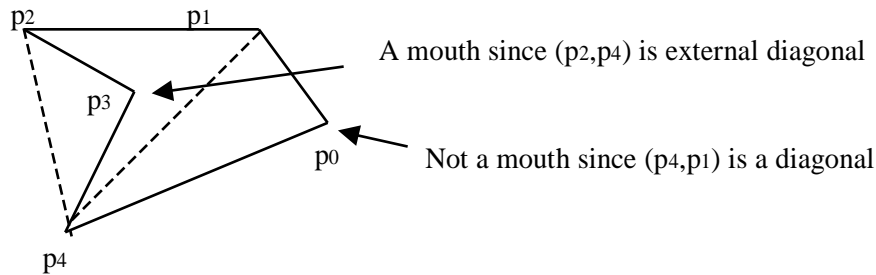
Ear

A vertex p_i of a simple polygon P is called an ear if for the consecutive vertices p_{i-1}, p_i, p_{i+1} (p_{i-1}, p_{i+1}) is a diagonal. We say that two ears p_i and p_j are non-overlapping if the interior of triangle (p_{i-1}, p_i, p_{i+1}) does not intersect the interior of triangle (p_{j-1}, p_j, p_{j+1}) .



Mouth

A vertex p_i of a simple polygon P is called a mouth if the diagonal (p_{i-1}, p_{i+1}) is an external diagonal, i.e., the interior of (p_{i-1}, p_{i+1}) lies in the exterior of P .



One-Mouth Theorem

Except for convex polygons, every simple polygon has at least one mouth.

Two-Ears Theorem

Except for triangles every simple polygon has at least two non-overlapping ears.

Jordan Curve Theorem

A simple closed curve C in the plane divides the plane into exactly two domains, an inside and an outside.

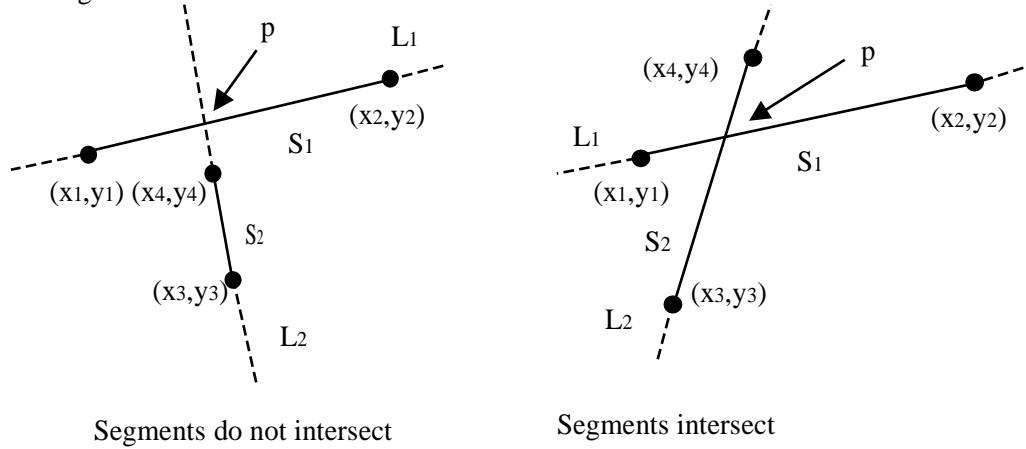
Computing point of intersection between two line segments

We can apply our coordinate geometry method for finding the point of intersection between two line segments. Let S_1 and S_2 be any two line segments. The following steps are used to calculate point of intersection between two line segments. We are not considering parallel line segments here in this discussion.

- Determine the equations of line through the line segment S_1 and S_2 . Say the equations are $L_1 = (y = m_1x + c_1)$ and $L_2 = (y = m_2x + c_2)$ respectively. We can find the equation of line L_1 using the formula of slope $(m_1) = (y_2 - y_1) / (x_2 - x_1)$, where (x_1, y_1) and (x_2, y_2) are two given end points of the line segment S_1 . Similarly we can find the m_2 for L_2 also. The values of c_i 's can be obtained by using the point of the line segment on the obtained equation after getting slope of the respective lines.
- Solve two equations of lines L_1 and L_2 , let the value obtained by solving be $p = (x_i, y_i)$. Here we confront with two cases. The first case is, if p is the intersection of two line segments then p lies on both S_1 and S_2 . The second case is if p is not an

intersection point then p does not lie on at least one of the line segments S1 and S2.

The figure below shows both the cases.



Segments do not intersect

Segments intersect

Detecting point of intersection

In straightforward manner we can compute the point of intersection (p) between the lines passing through S1 and S2 and see whether the line segments intersects or not as done in above discussion. However, the above method uses the division in the computation and we know that division is costly process. Here we try to detect the intersection without using division.

Left and Right Turn: Given points $p_0(x_0,y_0)$, $p_1(x_1,y_1)$, and $p_2(x_2,y_2)$. If we try to find whether the path $p_0p_1p_2$ make left or right turn, we check whether the vector **pop1** is clockwise or counterclockwise with respect to vector **pop2**. We compute the cross product of the vectors given by two line segments as

$(p_1 - p_0) \cdot (p_2 - p_0) = (x_1 - x_0, y_1 - y_0) \cdot (x_2 - x_0, y_2 - y_0) = (x_1 - x_0)(y_2 - y_0) - (y_1 - y_0)(x_2 - x_0)$, this can be represented as

- Here we have,
- $$\square = \begin{vmatrix} 1 & 1 & 1 \\ x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \end{vmatrix}$$
- If $\square = 0$ then p_0, p_1, p_2 are collinear
 - If $\square > 0$ then $p_0p_1p_2$ make left turn i.e. there is left turn at p_1 . (**pop1** is clockwise with respect to **pop2**).
 - If $\square < 0$ then $p_0p_1p_2$ make right turn i.e. there is right turn at p_1 . (**pop1** is anticlockwise with respect to **pop2**).

See figure below to have idea on left and right turn as well as direction of points. The cross product's geometric interpretation is also shown below.

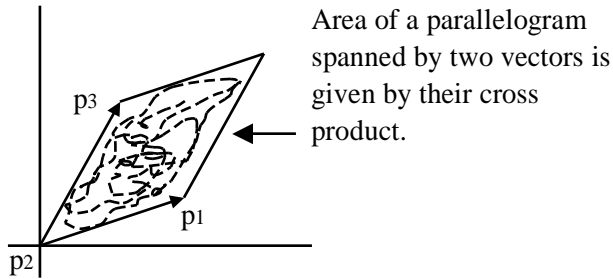


Fig: Cross product geometrical interpretation.
op

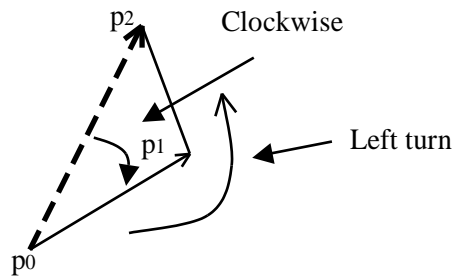


Fig: Left turn at p_1

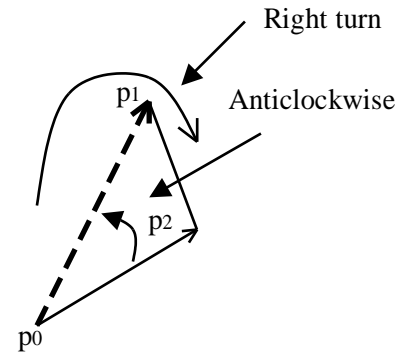


Fig: Right turn at p_1

Using the concept of left and right turn we can detect the intersection between the two line segments in very efficient manner. To detect the intersection we follow the following lemma.

Lemma 1:

Two segments $S_1 = (P, Q)$ and $S_2 = (R, S)$ do not intersect if PQR and PQS are of same turn type or RSP and RSQ are of same turn type. (See Assignment 4)

Graham's scan algorithm (Convex Hull)

We learned about convex hull. Graham's scan algorithm is an algorithm to obtain the convex hull from the given set of points say P . Graham's scan depends on angular sorting. The key idea is select the starting point p_0 , with minimum y-coordinate value (leftmost point in the set in case of the tie). After selection of the point p_0 sort the points of P by polar angle with respect to p_0 in anticlockwise direction. In case of the same angle remove the point with lower y-coordinate value. Push each point of set Q onto the stack once and the points that are not of convex hull of P are popped from the stack.

Algorithm:

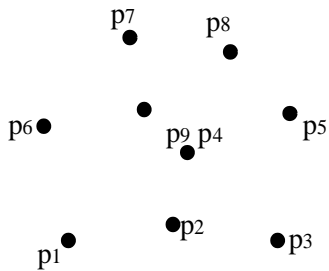
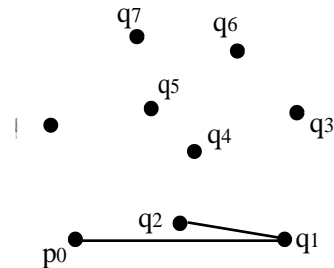
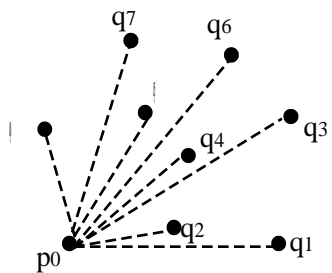
```

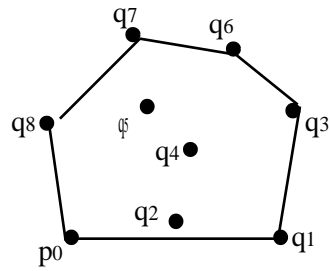
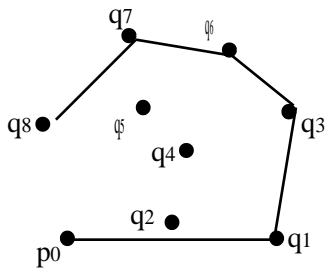
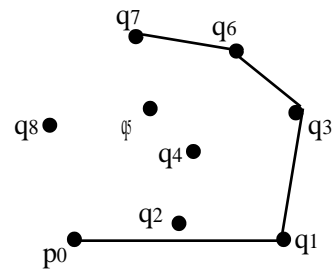
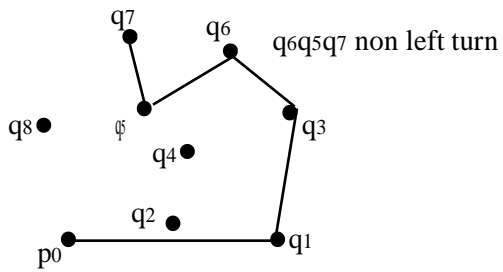
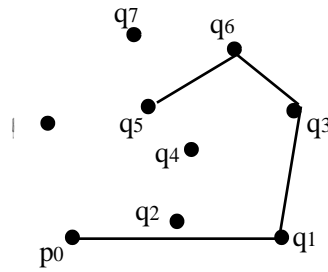
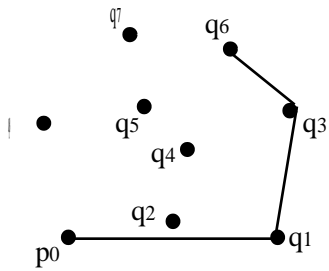
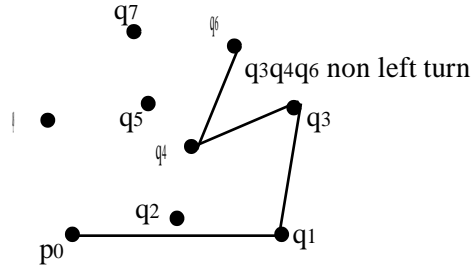
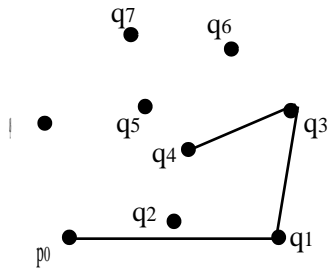
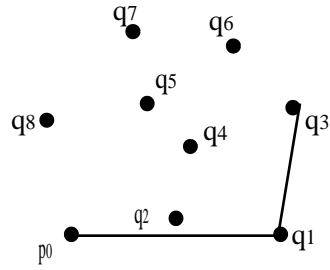
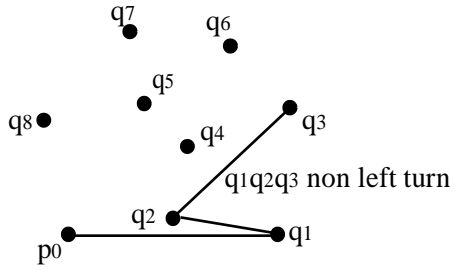
GrahamScan(P) // P = {p1, p2, ..., pn}
{
  p0 = point with lowest y-coordinate value.
  Angularly sort the other points with respect to p0. Let q = {q1, q2, ..., qm} be sorted points.
  Push(S, p0);           // S is a stack
  Push(S, q1);
  Push(S, q2);
  For(i=3; i < m; i++)
  {
    a = NexttoTop(S);
    b = Top(S);
    while (a, b, qi makes non left turn)
      Pop(S);
    Push(S, qi);
  }
  return S;
}

```

Example:

Find the convex hull of the set of points given below using graham's scan algorithm.

**Solution:**



Analysis:

It requires $O(n)$ time to find p_0 . Sorting of points require $O(n \log n)$ time,. Push operation takes constant time i.e., $O(1)$. We can understand that the pop operation inside the while loop is done at most $O(m)$ time infact, at most $m - 2$ pop operations are performed.

Therefore, the worst case for for loop takes $O(n)$ ($n > m$) because while loop is executed at most $O(m)$ times so if it is regarded with each iteration of for loop, the cost while loop is about constant time [$O(m)/m = O(1)$]. So, the worst case running time of the algorithm is $T(n) = O(n) + O(n \lg n) + O(1) + O(n) = O(n \lg n)$, where $n = |P|$.

Correctness:

The correctness of Graham's scan lies in two situations, non-left turns and left turns

Lemma 2:

Each point popped from the stack is not a vertex of convex hull of P and lies inside new convex hull of points in the stack S .

Proof:

Suppose that point q_j is popped from the stack because $q_k q_j q_i$ makes a non-left turn. Since points are ordered in increasing polar angle about the point p_0 , there exists a triangle $p_0 q_i q_k$, where p_0, q_i, q_k are all in the stack, with q_j either in the interior of the triangle (see figure 1 below) or on the line segment line $q_i q_k$. In either case, point q_j cannot be a vertex of convex hull of P . This completes the proof.

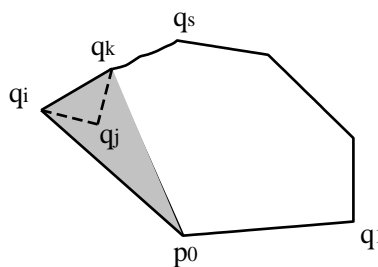


Figure 1

Lemma 3:

The points on stack always form the vertices of a convex polygon.

Proof:

The lemma holds after $p_0, q_1,$ and q_2 are pushed since $p_0, q_1,$ and q_2 form a convex polygon. Inside for loop stack changes when the points are popped or when the points are

pushed. If points are popped, we have the polygon having points of stack as convex because the geometric property says: If a vertex is removed from a convex polygon, the resulting polygon is convex. If points are pushed we claim that the points on stack forms convex polygon by using following lemma.

Lemma 4:

If a point q_i is pushed onto the stack, the resulting points form a convex polygon.

Proof:

By pushing the point q_i we have the points in the stack from the set $\{p_0, q_1, q_2, \dots, q_i\}$ in order from left to right in the set, where we may omit some points from the set other than p_0, q_1 , and q_i (we must have at least 3 elements in the stack otherwise in the next iteration we may pop the point if there is non left turn such that checking turn with point set will reduce to less than 3 points-impossible!). The points other than q_i from the stack form the convex polygon (since if convexity is not there we pop the point: see lemma 1). When q_i is pushed the point at the top of the stack will make the left turn towards q_i , such that it is in the region bounded by the boundary of the polygon that includes the points from the stack and the point, say q_j that will be considered later (see figure 2 below). This is true since we have the polar angle of q_j is greater than that of q_i . By the geometry property that, if we add point p_s in the bounded regions of the polygon P then the resulting polygon is convex, so we conclude that by pushing a point the resulting polygon so formed with the points of the stack will be convex.

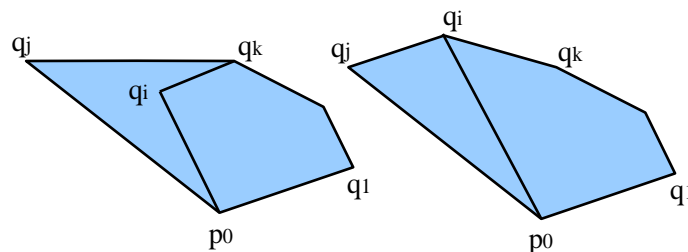


Figure 2

Conclusion

From lemma 2 and lemma 3 graham's scan algorithm is correct.

Closest Pair Problem

To solve the problem of finding closest pair of points among the given set P of n points we can use straightforward way by selecting 2 points from the set of n times. This approach takes $O(n^2)$ time. However we can use divide and conquer approach to lessen the running time of the algorithm that would solve the closest pair problem. The idea behind the algorithm under discussion is given below.

Divide: Set of points, P is divided into two parts say P_l and P_r with respect to some vertical line L such that the two parts are nearly equal i.e. number of points in both the divided sets differ by at most one point. All points on set P_l are on or to the left of L and all the points on P_r are on or to the right of L . We have two arrays X and Y consisting the points where X has the points with monotonically increasing x -coordinates whereas Y has the points with monotonically increasing y -coordinates. Divide both X and Y as X_l , X_r and Y_l , Y_r with the meaning of the set as defined for their original sets respectively.

Conquer: Recursively solve the problem with set of points, with left set of points and right set of points and find the closest pair between those two.

Combine: while combining there are things that are to be considered. The closest pair may be returned by either left part or the right part. Let the distance returned so be d . The second possibility is that the closest pair may be there such that one point lies on the left part and the other in the right part. To incorporate these possibilities we do the following.

Create an array $Y' \subseteq Y$ such that the points of Y beyond the distance d on either side of the line L are not in Y' . The points (actually only seven points) within the rectangle enclosed with the dimension d (vertical) and $2d$ (horizontal) are checked every point in that rectangle to see whether there is a distance between two points that is less than d if such a distance is found it is returned.

Algorithm:

Closest-Pair(P, X, Y)

{

if ($|P| \leq 3$) then

Compute closest pair;

Return closest pair;

```

    Divide points evenly along x axis at  $x = L$  into  $P_l, X_l, Y_l$  and  $P_r, X_r, Y_r$ .
     $d_l = \text{distance}(\text{Closest-Pair}(P_l, X_l, Y_l));$  // distance routine gives Euclidean distance
     $d_r = \text{distance}(\text{Closest-Pair}(P_r, X_r, Y_r));$ 
     $d = \min\{d_l, d_r\}$ 
    for each point  $p$  in  $(L - d) \delta \times \delta (L + d)$ 
        check 7 points  $p'$  closest to  $p$  by y-coordinate
         $d' = \text{distance}(p, p')$ 
        if  $(d' < d)$  then
            get new closest pair
    return closest pair
}

```

Analysis:

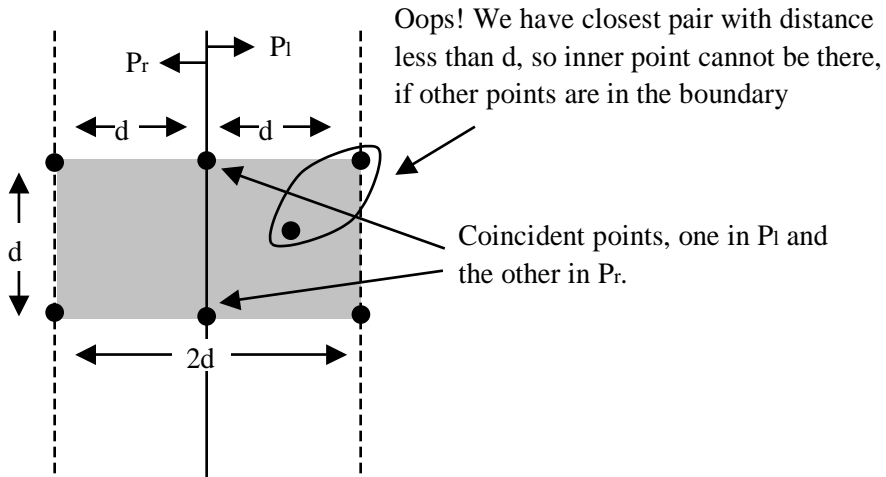
Sorting of the set of points takes $O(n \log n)$ time. The complexity of above algorithm is obtained from the recurrence relation below.

$T(n) = \begin{cases} 1 & \text{if } n \leq 3, \\ T(n/2) + \Theta(n) & \text{otherwise} \end{cases}$. (Here the cost of division of the arrays X and Y seem to be not achievable in $O(n)$ but if we use the reverse process of merge sort that the two subproblems that are merged are extracted we can do this in $O(n)$ time.). So complexity is $\Theta(n \log n)$!

Correctness:

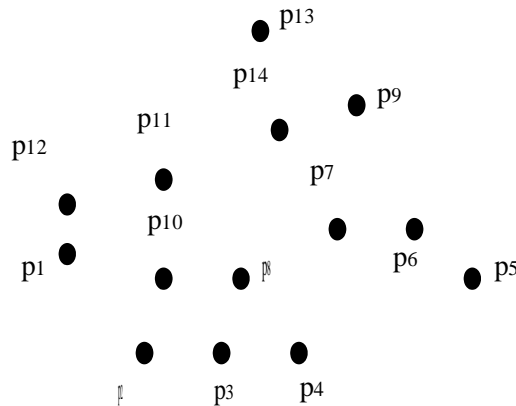
To prove the correctness of the closest pair algorithm there are two aspects to consider. At first if the number of points is less than or equal to 3 then we can easily calculate the possible distances and compare for the closest one (this will be terminating condition for the recurrence). We can notice the fact that we are not trying to solve the problem with only one point with our above terminating condition. We can use induction to show that for higher number of points algorithm does return closest pair. Secondly we can show that it is needed to check only 7 points from any points in the set Y' . Somewhere while running the algorithm we let some pair of points (p_l, p_r) is encountered where $p_l \in P_l$ and $p_r \in P_r$, then the distance between the pair (p_l, p_r) must be less than d say d' . We have p_r to the right of the line L and p_l to the left of the line L with distance from the line to the points less than d . so we can say that the pair (p_l, p_r) falls within the rectangle of the

dimension $d \cdot 2d$ with rectangle centered at line L. If we take the left or right half of the rectangle we can say that at most 4 points can reside on one half otherwise by including any other points within the half rectangle would create closest pair having distance less than d in the one side and this is not true (see figure below). So we have at most 8 points within the rectangle that are to be checked for and only 7 points need to be consider by each point on the set Y' . By this we can say that our algorithm is correct.



Exercises

1. Write down the algorithm to find the point of intersection of two line segments, if exists.
2. Use Graham's scan algorithm to find convex hull of the set of points below.



[NP Complete Problems & Approximation Algorithms]

Design and Analysis of Algorithms (CSc 523)

Samujjwal Bhandari

Central Department of Computer Science and Information Technology (CDCSIT)

Tribhuvan University, Kirtipur,
Kathmandu, Nepal.

Up to now we were considering on the problems that can be solved by algorithms in worst-case polynomial time. There are many problems and it is not necessary that all the problems have the apparent solution. This concept, somehow, can be applied in solving the problem using the computers. The computer can solve: some problems in limited time e.g. sorting, some problems requires unmanageable amount of time e.g. Hamiltonian cycles, and some problems cannot be solved e.g. Halting Problem. In this section we concentrate on the specific class of problems called NP complete problems (will be defined later).

Tractable and Intractable Problems

We call problems as tractable or easy, if the problem can be solved using polynomial time algorithms. The problems that cannot be solved in polynomial time but requires superpolynomial time algorithm are called intractable or hard problems. There are many problems for which no algorithm with running time better than exponential time is known some of them are, traveling salesman problem, Hamiltonian cycles, and circuit satisfiability, etc.

P and NP classes and NP completeness

The set of problems that can be solved using polynomial time algorithm is regarded as class P. The problems that are verifiable in polynomial time constitute the class NP. The class of NP complete problems consists of those problems that are NP as well as they are as hard as any problem in NP (more on this later). The main concern of studying NP completeness is to understand how hard the problem is. So if we can find some problem as NP complete then we try to solve the problem using methods like approximation, rather than searching for the faster algorithm for solving the problem exactly.

Problems

Abstract Problems:

Abstract problem A is binary relation on set I of problem instances, and the set S of problem solutions. For e.g. Minimum spanning tree of a graph G can be viewed as a pair of the given graph G and MST graph T.

Decision Problems:

Decision problem D is a problem that has an answer as either “true”, “yes”, “1” or “false”, “no”, “0”. For e.g. if we have the abstract shortest path with instances of the problem and the solution set as $\{0,1\}$, then we can transform that abstract problem by reformulating the problem as “Is there a path from u to v with at most k edges”. In this situation the answer is either yes or no.

Optimization Problems:

We encounter many problems where there are many feasible solutions and our aim is to find the feasible solution with the best value. This kind of problem is called optimization problem. For e.g. given the graph G , and the vertices u and v find the shortest path from u to v with minimum number of edges. The NP completeness does not directly deal with optimizations problems, however we can translate the optimization problem to the decision problem.

Encoding:

Encoding of a set S is a function e from S to the set of binary strings. With the help of encoding, we define **concrete problem** as a problem with problem instances as the set of binary strings i.e. if we encode the abstract problem, then the resulting encoded problem is concrete problem. So, encoding as a concrete problem assures that every encoded problem can be regarded as a language i.e. subset of $\{0,1\}^*$.

Complexity Class P

Complexity class **P** is the set of concrete decision problems that are polynomial time solvable by deterministic algorithm. If we have an abstract decision problem A with instance set I mapping the set $\{0,1\}$, an encoding $e: I \rightarrow \{0,1\}^*$ is used to denote the concrete decision problem $e(A)$. We have the solutions to both the abstract problem instance $i \in I$ and concrete problem instance $e(i) \in \{0,1\}^*$ as $A(i) \in \{0,1\}$. It is important to understand that the encoding mechanism does greatly vary the running time of the algorithm for e.g. take some algorithm that runs in $O(n)$ time, where the n is size of the input. Say if the input is just a natural number k , then its unary encoding makes the size of the input as k bits as k number of 1's and hence the order of the algorithm's running time is $O(k)$. In other situation if we encode the natural number k as binary encoding then

we can represent the number k with just **logk** bits (try to represent with 0 and 1 only) here the algorithm runs in $O(n)$ time. We can notice that if $n = \log k$ then $O(k)$ becomes $O(2^n)$ with unary encoding. However in our discussion we try to discard the encoding like unary such that there is not much difference in complexity.

We define **polynomial time computable** function $f: \{0,1\}^* \rightarrow \{0,1\}^*$ with respect to some polynomial time algorithm PA such that given any input $x \in \{0,1\}^*$, results in output $f(x)$.

For some set I of problem instances two encoding e_1 and e_2 are **polynomially related** if there are two polynomial time computable functions **f** and **g** such that for any $i \in I$, both $f(e_1(i)) = e_2(i)$ and $g(e_2(i)) = e_1(i)$ are true i.e. both the encoding should be computed from one encoding to another encoding in polynomial time by some algorithm.

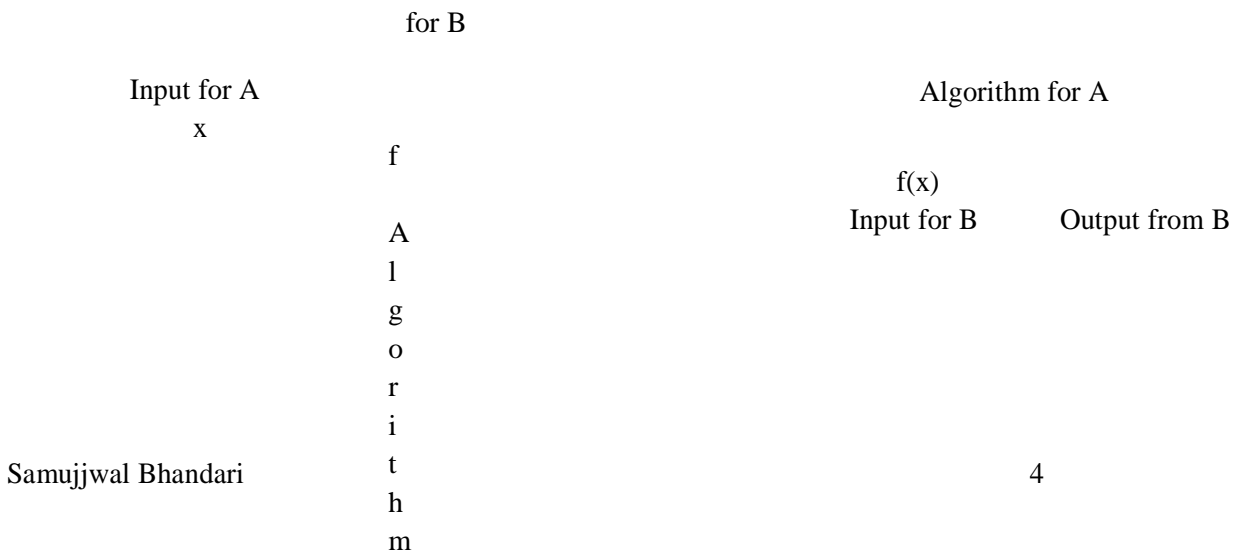
Lemma 1:

Let A be an abstract decision problem on an instance set I , and let e_1 and e_2 be polynomially related encodings on I . Then, $e_1(A) \in \mathbf{P}$ iff $e_2(A) \in \mathbf{P}$.

The above lemma says that if we have encodings that are polynomially related then the use of the encoding does not alter the result of the fact, whether the problem is polynomially solvable or not.

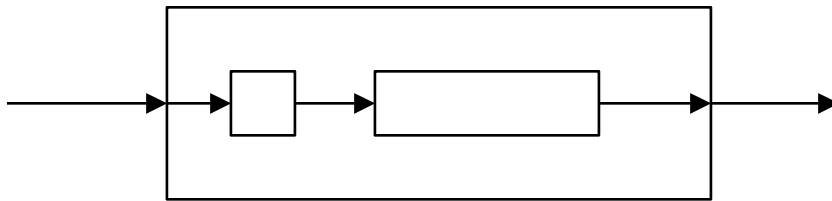
Polynomial time reduction

Given two decision problems A and B , a polynomial time reduction from A to B is a polynomial time function f that transforms the instances of A into instances of B such that the output of algorithm for the problem A on input instance x must be same as the output of the algorithm for the problem B on input instance $f(x)$ as shown in the figure below. If there is polynomial time computable function f such that it is possible to reduce A to B , then it is denoted as $A \delta_p B$. The function f described above is called reduction function and the algorithm for computing f is called reduction algorithm.



yes/no

Output from A



Lemma 2:

If $L_1, L_2 \subseteq \{0,1\}^*$ are languages¹ such that $L_1 \leq_p L_2$, then $L_2 \in \mathbf{P} \iff L_1 \in \mathbf{P}$.

Proof:

(Try to understand from the above figure: **see book for the detail proof**)

Fact1: If $L_1 \leq_p L_2$ and $L_2 \leq_p L_3$, then $L_1 \leq_p L_3$.

Complexity Class NP

NP is the set of decision problems solvable by nondeterministic algorithms in polynomial time. When we have a problem, it is generally much easier to verify that a given value is solution to the problem rather than calculating the solution of the problem. Using the above idea we say the problem is in class NP (nondeterministic polynomial time) if there is an algorithm for the problem that verifies the problem in polynomial time. V is the verification algorithm to the decision problem D if V takes input string x as an instance of the problem D and another binary string y , certificate, whose size is no more than the polynomial in the size of x . the algorithm V verifies an input x if there is a certificate y such that answer of D to the input x with certificate y is yes. *For e.g. Circuit satisfiability problem (SAT) is the question "Given a Boolean combinational circuit, is it satisfiable? i.e. does the circuit has assignment sequence of truth values that produces the output of the circuit as 1?"* Given the circuit satisfiability problem take a circuit x and a certificate y with the set of values that produce output 1, we can verify that whether the given certificate satisfies the circuit in polynomial time. So we can say that circuit satisfiability problem is NP.

We can always say $\mathbf{P} \subseteq \mathbf{NP}$, since if we have the problem for which the polynomial time algorithm exists to solve (decide: notice the difference between decide and accept) the problem, then we can always get the verification algorithm that neglects the certificate and accepts the output of the polynomial time algorithm. From the above fact we are clear that $\mathbf{P} \subseteq \mathbf{NP}$ but the question, whether $\mathbf{P} = \mathbf{NP}$ remains unsolved and is still the big question in theoretical computer science. Most of the computer scientists, however, believes that $\mathbf{P} \neq \mathbf{NP}$.

¹ Every encoded decision problem can be viewed as language over alphabet $\{0,1\}$.

NP-Completeness

NP complete problems are those problems that are hardest problems in class NP. We define some problem say A, is NP-complete if

1. $A \in \text{NP}$, and
2. $B \leq_p A$, for every $B \in \text{NP}$.

We call the problem (or language) A satisfying property 2 is called NP-hard.

Theorem 1:

If any NP-complete problem is polynomial time solvable, then $P = \text{NP}$. Equivalently, if any problem in NP is not polynomial time solvable, then no NP-complete problem is polynomial time solvable.

Proof:

Let a problem $A \in P$ and $A \in \text{NP-complete}$. If we have another problem $B \in \text{NP}$, then we have $B \leq_p A$. Again we have, from lemma 2, $B \in P$ so we can say $P = \text{NP}$. This is the proof. (Remember the statements in the above theorem are equivalent, where the first one is direct implication and the second one is its contrapositive. We know contrapositive and direct implication are logically equivalent.)

Cook's Theorem

Lemma 3:

SAT is NP-hard

Proof: (This is not actual proof as given by cook, this is just a sketch)

Take a problem $V \in \text{NP}$, let A be the algorithm that verifies V in polynomial time (this must be true since $V \in \text{NP}$). We can program A on a computer and therefore there exists a (huge) logical circuit whose input wires correspond to bits of the inputs x and y of A and which outputs 1 precisely when $A(x,y)$ returns yes.

For any instance x of V let A_x be the circuit obtained from A by setting the x-input wire values according to the specific string x. The construction of A_x from x is our reduction function. If x is a yes instance of V, then the certificate y for x gives satisfying assignments for A_x . Conversely, if A_x outputs 1 for some assignments to its input wires, that assignment translates into a certificate for x.

Theorem 2: (Cook's Theorem)

SAT is NP-complete

Proof:

To show that SAT is NP-complete we have to show two properties as given by the definition of NP-complete problems. The first property i.e. SAT is in NP we showed above (see pg 5 italicized part), so it is sufficient to show the second property holds for SAT. The proof for the second property i.e. SAT is NP-hard is from lemma 3. This completes the proof.

NP completeness proofs

Lemma 4:

If L is a language such that $L' \delta_p L$ for some $L' \in \text{NP-complete}$, then L is NP-hard and if $L \in \text{NP}$, then $L \in \text{NP-complete}$.

Proof:

We have L' as NP-complete so, for all $M \in \text{NP}$, $M \delta_p L'$. Since we have $L' \delta_p L$ using the fact 1 (above pg 5), we have $M \delta_p L$ i.e. L is NP-hard. It is clear that if $L \in \text{NP}$, then $L \in \text{NP-complete}$ from the definition of NP-completeness.

Steps for proving a problem (language) A to be NP-complete

1. Prove $A \in \text{NP}$.
2. Select a known NP-complete problem B.
3. Describe an algorithm that computes a function f mapping every instance x of B to an instance $f(x)$ of A.
4. Prove that the function satisfies $x \in B$ iff $f(x) \in A$ for all x .
5. Prove the algorithm for computing f to be polynomial time algorithm.

Example:

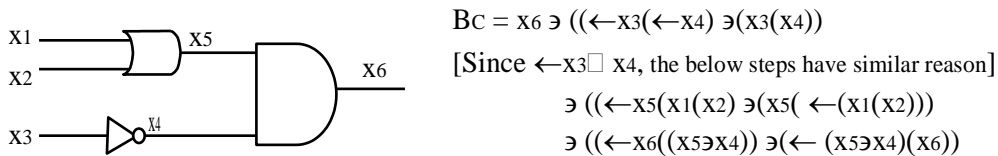
Satisfiability of Boolean formula, BSAT, is a problem that consist of of n Boolean variables x_1, x_2, \dots, x_n ; m Boolean operators as AND, OR, NOT, Implication, Biconditional; and non redundant parenthesis for precedence. Satisfying assignment of values of Boolean variables is the string of truth-values that produces output as 1. BSAT ask us to find whether the given formula is satisfiable? The naïve algorithm takes $O(2^n)$

time (how!). Now we show that BSAT is NP-complete.

We know that SAT is NP-complete. If we can show $SAT \delta_p BSAT$ and $BSAT \sqsubseteq NP$, then using the lemma 4 we prove that BSAT is NP-complete.

BSAT \sqsubseteq NP: In BSAT problem take a formula (with n Boolean variables) x and a certificate y with the set of truth values that produce output 1, we can verify the formula with the given certificate for satisfying the formula in polynomial time. So, $BSAT \sqsubseteq NP$.

SAT δ_p BSAT: Take a circuit C with only NOT, AND, and OR gates (All the circuits can be constructed using the mentioned operators, since set of operators {NOT, AND, OR} is functionally complete). For each wire x_i in the circuit C assign the variable x_i for the Boolean formula, now if each gate is replaced by its appropriate operator then we come up with the Boolean formula B_C that outputs exactly what the formula outputs. This reduction can be done in polynomial time in (circuit size) straightforward way (reducing each input wire by a variable). If C has satisfying assignment, then clearly the formula must be satisfied. Reduction is shown by the figure below. So we have $SAT \delta_p BSAT$.



Approximation Algorithms

An approximate algorithm is a way of dealing with NP-completeness for optimization problem. This technique does not guarantee the best solution. The goal of an approximation algorithm is to come as close as possible to the optimum value in a reasonable amount of time which is at most polynomial time. If we are dealing with optimization problem (maximization or minimization) with feasible solution having positive cost then it is worthy to look at approximate algorithm for near optimal solution. An algorithm has an **approximate ratio** of $\gamma(n)$ if, for any problem of input size n , the cost C of solution by an algorithm and the cost C^* of optimal solution have the relation as $\max(C/C^*, C^*/C) \delta \gamma(n)$. Such an algorithm is called **$\gamma(n)$ -approximation algorithm**. The relation applies for both maximization ($0 < C \delta C^*$) and minimization ($0 < C^* \delta C$)

problems. $\gamma(n)$ is always greater than or equal to 1. If solution produced by approximation algorithm is true optimal solution then clearly we have $\gamma(n) = 1$.

Vertex Cover Problem

A **vertex cover** of an undirected graph $G=(V,E)$ is a subset $V' \subseteq V$ such that for all edges $(u,v) \in E$ either $u \in V'$ or $v \in V'$ or u and $v \in V'$. The problem here is to find the vertex cover of minimum size in a given graph G . Optimal vertex-cover is the optimization version of an NP-complete problem but it is not too hard to find a vertex-cover that is near optimal.

Algorithm:

ApproxVertexCover (G){

$C = \{\}; E' = E$

while E' is not empty

do Let (u, v) be an arbitrary edge of E'

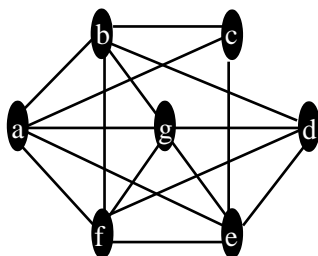
$C = C * \{u, v\}$

Remove from E' every edge incident on either u or v

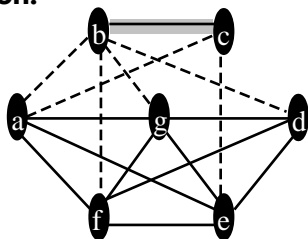
return C

}

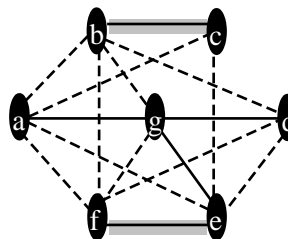
Example: (vertex cover running example for graph below)



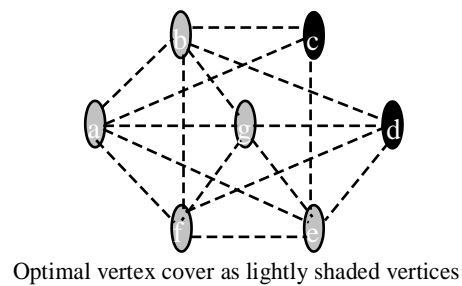
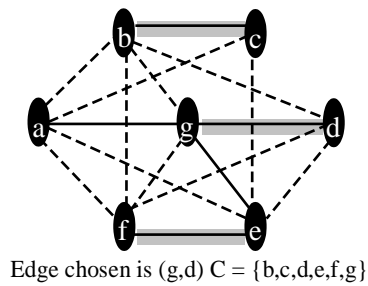
Solution:



Edge chosen is (b,c) $C = \{b,c\}$



Edge chosen is (f,e) $C = \{b,c,e,f\}$



Analysis:

If E' is represented using the adjacency lists the above algorithm takes $O(V+E)$ since each edge is processed only once and every vertex is processed only once throughout the whole operation.

Theorem 3: ApproxVertexCover is a polynomial-time 2-approximate algorithm.

Proof: We know that ApproxVertexCover is a polynomial-time algorithm (see above). Lets try to show that it is 2-approximate algorithm, let the set C^* and C be the sets output by OptimalVertexCover and ApproxVertexCover respectively. Let A be the set of edges selected by the first instruction after while loop in the above algorithm. Since, we have added vertices from the edge not already in C , we get $|C| = 2|A|$, the upper bound on the size of vertex cover. As we don not know what is the size of optimal solution we can argue that C^* must have atleast one vertex from the edge from the set A , and no two edges in A are covered by the same vertex due to the deletion of all the edges adjacent to the vertices added, so $|C^*| \leq |A|$ is the lower bound for optimal solution. From the above two facts we have $|C| \leq 2|C^*|$ i.e. $|C|/|C^*| \leq 2 = \gamma(n)$. Hence the ApproxVertexCover is a polynomial-time 2-approximate algorithm.

Exercises

1. Show that vertex cover problem is NP-complete.
2. Investigate the travelling salesman problem and present the approximation algorithm to deal with NP-complete travelling salesman problem.

[Mathematical Foundation]

Design and Analysis of Algorithms (CSc 523)

Samujjwal Bhandari

Central Department of Computer Science and Information Technology (CDCSIT)

Tribhuvan University, Kirtipur,

Kathmandu, Nepal.

Since mathematics can provide clear view of an algorithm. Understanding the concepts of mathematics aid in the design and analysis of good algorithms. Here we present some of the mathematical concepts that are helpful in our study.

Exponents

Some of the formulas that are helpful are :

$$X^a X^b = X^{a+b}$$

$$X^a / X^b = X^{a-b}$$

$$(X^a)^b = X^{ab}$$

$$X^n + X^n = 2X^n$$

$$2^n + 2^n = 2^{n+1}$$

Logarithms

Some of the formulas that are helpful are :

1. $\log_{ab} = \log_{cb} / \log_{ca} ; c > 0$
2. $\log ab = \log a + \log b$
3. $\log a/b = \log a - \log b$
4. $\log (a^b) = b \log a$
5. $\log x < x$ for all $x > 0$
6. $\log 1 = 0, \log 2 = 1, \log 1024 = 10.$
7. $a \log bn = n \log ba$

Series

$$1. \sum_{i=0}^n 2^i = 2^{n+1} - 1$$

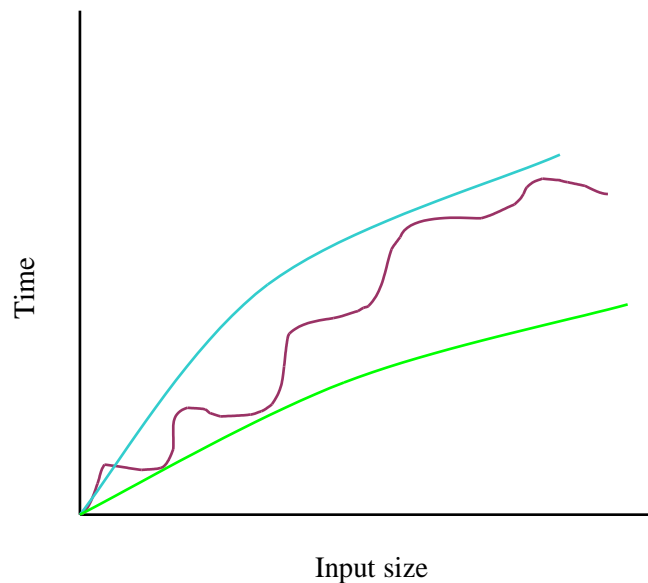
$$2. \sum_{i=0}^n a^i = \begin{cases} 1 / 1-a ; & \text{if } 0 < a < 1 \\ a^{n+1} - 1 / a-1 ; & \text{else} \end{cases}$$

3. $\sum_{i=1}^n i = n(n+1) / 2$
4. $\sum_{i=1}^n i^2 = n(n+1)(2n+1) / 6$
5. $\sum_{i=1}^n i^k = H_{n,k+1} / |k+1| ; k \neq -1$
6. $\sum_{i=1}^n 1/i = H_n \log e$

Asymptotic Notation

Complexity analysis of an algorithm is very hard if we try to analyze exact. we know that the complexity (worst, best, or average) of an algorithm is the mathematical function of the size of the input. So if we analyze the algorithm in terms of bound (upper and lower) then it would be easier. For this purpose we need the concept of asymptotic notations.

The figure below gives upper and lower bound concept.



Why we concentrate on worst case mostly ? Why not best case or average case?

Big Oh (O) notation

When we have only asymptotic upper bound then we use O notation. A function $f(x) = O(g(x))$ (read as $f(x)$ is big oh of $g(x)$) iff there exists two positive constants c and x_0 such that for all $x \geq x_0$,

$$0 \leq f(x) \leq c \cdot g(x)$$

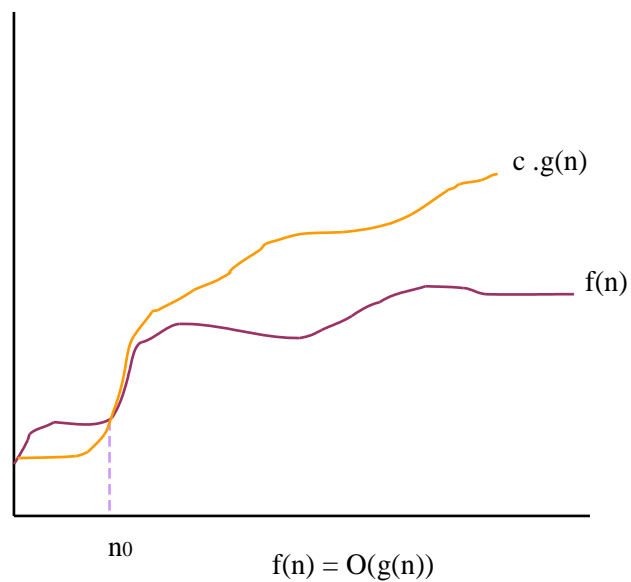
The above relation says that $g(x)$ is an upper bound of $f(x)$

some properties:

Transitivity : $f(x) = O(g(x))$ & $g(x) = O(h(x))$ $f(x) = O(h(x))$

Reflexivity: $f(x) = O(f(x))$

$O(1)$ is used to denote constants.



For all values of $n \geq n_0$, plot shows clearly that $f(n)$ lies below or on the curve of $c \cdot g(n)$

Examples

1. $f(n) = 3n^2 + 4n + 7$
 $g(n) = n^2$, then prove that $f(n) = O(g(n))$.

Proof: let us choose c and n_0 values as 14 and 1 respectively then we can have

$$f(n) \leq c \cdot g(n), n \geq n_0 \text{ as}$$

$$3n^2 + 4n + 7 \leq 14 \cdot n^2 \text{ for all } n \geq 1$$

the above inequality is trivially true

$$\text{hence } f(n) = O(g(n))$$

2. Prove that $n \log(n^3)$ is $O(n^3)$.

Proof: we have $n \log(n^3) = 3n \log n$

again, $n^3 = n \cdot n^2$,

if we can prove $\log n = O(n)$ then problem is solved

because $n \log n = n \cdot O(n)$ that gives the question again.

We can remember the fact that $\log_a n$ is $O(n^b)$ for all $a, b > 0$.

In our problem $a = 1$ and $b = \frac{1}{2}$,

$$\text{hence } \log n = O(n).$$

So by knowing $\log n = O(n)$ we proved that

$$n \log(n^3) = O(n^3).$$

3. Is $2^{n+1} = O(2^n)$?

$$\text{Is } 2^{2n} = O(2^n) ?$$

(See solution in the class)

Big Omega () notation

Big omega notation gives asymptotic lower bound. A function $f(x) = (g(x))$ (read as $f(x)$) is big omega of $g(x)$ iff there exists two positive constants c and x_0 such that for all $x \geq x_0$,

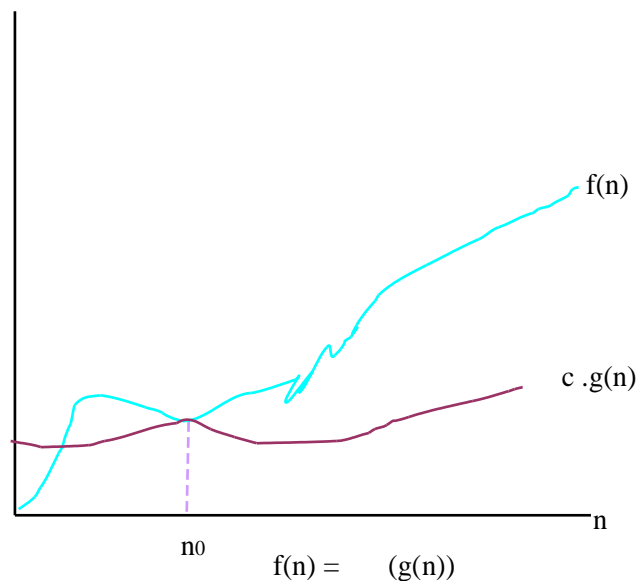
$$0 \leq c \cdot g(x) \leq f(x).$$

The above relation says that $g(x)$ is an lower bound of $f(x)$.

some properties:

Transitivity : $f(x) = O(g(x))$ & $g(x) = O(h(x))$ $f(x) = O(h(x))$

Reflexivity: $f(x) = O(f(x))$



For all values of $n \geq n_0$, plot shows clearly that $f(n)$ lies above or on the curve of $c \cdot g(n)$.

Examples

1. $f(n) = 3n^2 + 4n + 7$
 $g(n) = n^2$, then prove that $f(n) = \Theta(g(n))$.

Proof: let us choose c and n_0 values as 1 and 1, respectively then we can have

$$f(n) \geq c \cdot g(n), \quad n \geq n_0 \text{ as}$$

$$3n^2 + 4n + 7 \geq 1 \cdot n^2 \text{ for all } n \geq 1$$

the above inequality is trivially true

$$\text{hence } f(n) = \Theta(g(n))$$

Big Theta (Θ) notation

When we need asymptotically tight bound then we use Θ notation. A function $f(x) = \Theta(g(x))$ (read as $f(x)$ is big theta of $g(x)$) iff there exists three positive constants c_1 , c_2 and x_0 such that for all $x \geq x_0$,

$$0 < c_1 \cdot g(x) \leq f(x) \leq c_2 \cdot g(x)$$

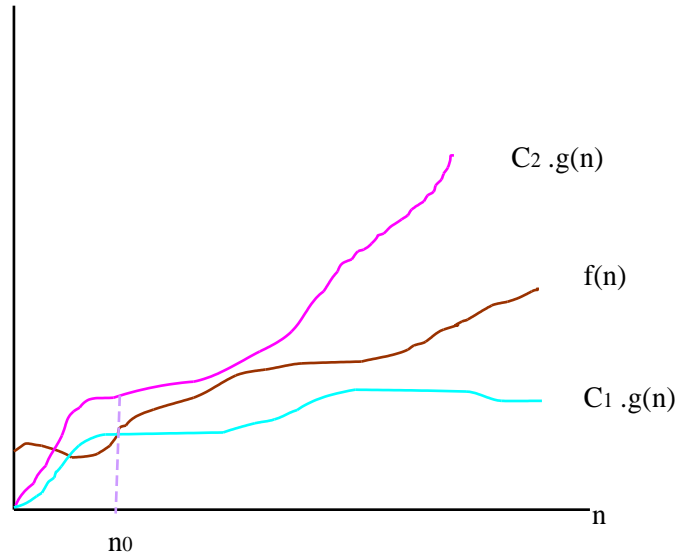
The above relation says that $f(x)$ is order of $g(x)$

some properties:

$$\text{Transitivity : } f(x) = \Theta(g(x)) \ \& \ g(x) = \Theta(h(x)) \quad f(x) = \Theta(h(x))$$

$$\text{Reflexivity: } f(x) = \Theta(f(x))$$

$$\text{Symmetry: } f(x) = \Theta(g(x)) \text{ iff } g(x) = \Theta(f(x))$$



$$f(n) = \cup (g(n))$$

For all values of $n \geq n_0$, plot shows clearly that $f(n)$ lies between $c_1 \cdot g(n)$ and $c_2 \cdot g(n)$.

Examples

1. $f(n) = 3n^2 + 4n + 7$
 $g(n) = n^2$, then prove that $f(n) = \cup (g(n))$.

Proof: let us choose c_1 , c_2 and n_0 values as 14, 1 and 1 respectively then we can have,

$$f(n) \leq c_1 \cdot g(n), n \geq n_0 \text{ as } 3n^2 + 4n + 7 \leq 14 \cdot n^2, \text{ and}$$

$$f(n) \geq c_2 \cdot g(n), n \geq n_0 \text{ as } 3n^2 + 4n + 7 \geq 1 \cdot n^2$$

for all $n \geq 1$ (in both cases).

So $c_2 \cdot g(n) \leq f(n) \leq c_1 \cdot g(n)$ is trivial.

$$\text{hence } f(n) = \cup (g(n)).$$

2. Show $(n + a)^b = \cup (n^b)$, for any real constants a and b , where $b > 0$.

Here, using Binomial theorem for expanding $(n + a)^b$, we get ,

$$C(b,0)n^b + C(b,1)n^{b-1}a + \dots + C(b,b-1)na^{b-1} + C(b,b)a^b$$

we can obtain some constants such that $(n+a)^b \leq c_1 n^b$, for all $n \geq n_0$

and

$$(n+a)^b \geq c_2 n^b, \text{ for all } n \geq n_0$$

here we may take $c_1 = 2^b$ $c_2 = 1$ $n_0 = |a|$,

since $n^b \leq (n+a)^b \leq 2^b n^b$.

Hence the problem is solved.

Why $c_1 = 2^b$? since $C(n,k) = 2^k$, where $k=0$ to n .

Little Oh (o) notation

Little oh (o) notation is used to denote the upper bound that is not asymptotically tight. A function $f(x) = o(g(x))$ (read as $f(x)$ is little oh of $g(x)$) iff for any positive constant c there exists positive constant x_0 such that for all $x \geq x_0$,

$$0 \leq f(x) < c \cdot g(x)$$

for example $4x^4$ is $O(x^4)$ but not $o(x^4)$.

Alternatively $f(x)$ is little oh of $g(x)$ if

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = 0$$

Note: transitivity is satisfied.

Little Omega () notation

Little omega (o) notation is used to denote the lower bound that is not asymptotically tight. A function $f(x) = \Omega(g(x))$ (read as $f(x)$ is little omega of $g(x)$) iff for any positive constant c there exists positive constant x_0 such that for all $x \geq x_0$,

$$0 \leq c \cdot g(x) < f(x) .$$

for example $x^3/7$ is (x^2) but not (x^3) .

Alternatively $f(x)$ is little omega of $g(x)$ if

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = \square$$

Note: transitivity is satisfied.

Complexity	10	20	30	40	50
n	0.00001 sec	0.00002 sec	0.00003 sec	0.00004 sec	0.00005 sec
	0.0001 sec	0.0004 sec	0.0009 sec	0.016 sec	0.025 sec
	0.001 sec	0.008 sec	0.027 sec	0.064 sec	0.125 sec
	0.001 sec	1.0 sec	17.9 min	12.7 days	35.7 years
	0.59 sec	58 min	6.5 years	3855 cent	cent

Some commonly used functions in algorithm analysis

$f(x) = O(1)$	constant
$f(x) = C \cdot \log x$	logarithmic
$f(x) = C \cdot x$	linear
$f(x) = C \cdot x \log x$	linearithmic
$f(x) = C \cdot x^2$	quadratic
$f(x) = C \cdot x^3$	cubic
$f(x) = C \cdot x^k$	polynomial in k
$f(x) = C \cdot k^x$	exponential in k

Order of growth

$$O(1) < C \cdot \log x < C \cdot x < C \cdot x \log x < C \cdot x^2 < C \cdot x^3 < C \cdot x^k < C \cdot kx$$

Value of k should be in increasing order.

Before we start: Is the following statement true or false? $f(n) + g(n) = O(\min(f(n), g(n)))$ Explain your answer.

Example Algorithm 1: Insertion Sort

Input: An array of numbers to be sorted A [] of length n.

Output: Sorted array A [].

Algorithm: InsertionSort (A [])

```

for (i=1; i<n; i++)
{
    x = A[i];
    //A[0]...A[i-1] is already sorted.
    j = i-1;
    while(j>=0 and A[j] > x)
    {
        A[j+1] = A[j];
        j = j-1;
    }
    A[j+1] = x;
}

```

Correctness:

The invariant of the outer loop that the array A[0] ... A[i-1] is sorted contains no elements other than of original one. Finding invariant for inner loop is very hard still we can see that the inner loop is for insertion and always insert correctly.

The algorithm terminates when arrays last element is read by outer loop.

Efficiency

Time Complexity

InsertionSort (A [])	cost	times
1. for (i=1; i<n; i++)	c1	n
2. {	e2	
3. x = A[i]	c3	n-1
4. //A[0]...A[i-1] is already sorted	e4	
5. j = i-1;	c5	n-1
6. while(j>=0 and A[j] > x)	c6	$\sum_{l=1}^n l$
7. {	e7	
8. A[j+1] = A[j];	c8	$\sum_{l=1}^n (l-1)$
9. j = j-1;	c9	$\sum_{l=1}^n (l-1)$
10. }	e10	
11. A[j+1] = x;	c11	n-1
12. }	e12	

Remember: $l = i-1$, cost and times are multiplied for each steps (not for space complexity). Strikethrough costs means the costs of no importance in our analysis.

$$\text{Time complexity } T(n) = c1*n + c3*(n-1) + c5*(n-1) + c6* \sum_{i=1}^n l + c8* \sum_{i=1}^n (l-1) + c9* \sum_{i=1}^n (l-1) + c11*(n-1)$$

Best Case

Best case is achieved when the inner loop is not executed, for already sorted array this happens so best case analysis gives,

$$T(n) = c_1 * n + c_3 * (n-1) + c_5 * (n-1) + c_6 * (n-1) + c_{11} * (n-1)$$

$$= O(n)$$

Note: how some instances of inputs change the general behavior of an algorithm.

Worst Case

Worst case occurs when each execution of inner loop moves every element of an array.

$$T(n) = c_1 * n + c_3 * (n-1) + c_5 * (n-1) + c_6 * \frac{n(n+1)}{2} + c_8 * \frac{n(n-1)}{2} + c_9 * \frac{n(n-1)}{2} + c_{11} * (n-1)$$

$$= O(n^2)$$

Average Case

For average case inner loop inserts element from an array in the middle of an array, this takes $O(i/2)$ time.

$$T_a(n) = O(i/2) = O(i) = O(n)$$

Space Complexity

Space complexity is $O(n)$ since space depends on the size of the input.

Example Algorithm 2: Merge Sort

Input: An array of numbers to be sorted A [] of length n.

Output: Sorted array A [].

Algorithm: MergeSort (A,l,h)

```

if(l < h)
{
    q = (l + h) / 2;
    MergeSort(A, l, q);
    MergeSort(A, q+1, h);
    Merge(A, l, q, h);
}

```

Algorithm: Merge (A,low,mid,high)

$l = m = \text{low}; k = \text{mid} + 1;$

while(($l \leq \text{mid}$) and $k \leq \text{high}$)

{

if($A[l] \leq A[k]$)

 {

$B[m] = A[l];$

$l = l + 1;$

 }

else

 {

$B[m] = A[k];$

$k = k + 1;$

 }

$m = m + 1;$

}

if($l > \text{mid}$)

for($i = k; i < \text{high}; i++$)

 {

$B[m] = A[i];$

$m = m + 1;$

 }

else

for($i = l; i < \text{mid}; i++$)

 {

$B[m] = A[i];$

$m = m + 1;$

 }

for($j = \text{low}; j < \text{high}; j++$)

$A[j] = B[j];$

Correctness:

Correctness of a **MergeSort** can be verified by induction if we can prove that **Merge** is correct and correctness of **Merge** means two ranges of sorted sequence when get merged the result should be sorted and the content of the original input must be same. See, second loop just copies the array so if we prove that first loop produces sorted array then we are done. The invariants for the first loop are

Either $m=1$ or $B[m]$ is sorted at every point in the algorithm, $B[m-1] \leq A[l]$ and $B[m-1] \leq A[k]$. This property is true at the beginning and ending of loop (requires a bit effort to verify with all four conditions).

Efficiency

Time Complexity

To calculate the time complexity of the above MergeSort algorithm, first for the simplicity assume that the size of the input is power of 2. Now we can obtain time complexity as,

$$T(n) = O(1)\{\text{for if}\} + O(1)\{\text{for 2nd statement}\} + T(n/2) + T(n/2) + T(\text{Merge}).$$

Here we need to calculate complexity of Merge algorithm also, complexity for this can be straightforwardly calculated as $O(n)$ {do yourself}

Now, if $n=1$ then there is no process involved other than if ($l < h$) so $T(n) = O(1)$

Floors and ceilings can be eliminated in asymptotic analysis. So for $n > 1$ we can write

$$T(n) = 2T(n/2) + O(n)$$

You can notice that we obtained valid recurrence relation with boundary condition and is expressed like

$$T(n) = O(1), \text{ when } n = 1$$

$$T(n) = 2T(n/2) + O(n), \text{ when } n > 1$$

We can write $n = 2^k$ (why? See our assumption) then,

$$\begin{aligned}
T(n) &= 2T(2^{k-1}) + \cup(2^k) \\
&= 2(2T(2^{k-2}) + \cup(2^{k-1})) + \cup(2^k) \\
&= 4T(2^{k-2}) + 2\cup(2^{k-1}) + \cup(2^k) \\
&\dots \\
&= 2^k\cup(1) + 2^{k-1}\cup(1) + \dots + 2\cup(2^{k-1}) + \cup(2^k) \\
&= k\cup(2^k) \\
&= \cup(k \log n) \\
&= \cup(n \log n)
\end{aligned}$$

What if n is not a power of 2. Then there exists k such that $2^k < n < 2^{k+1}$ and we have

$$\begin{aligned}
T(2^k) &\leq T(n) \leq T(2^{k+1}) \\
\cup(k \log 2^k) &\leq T(n) \leq \cup((k+1) \log 2^{k+1})
\end{aligned}$$

here $\cup((k+1) \log 2^{k+1}) < 4k \log 2^k$ so $\cup((k+1) \log 2^{k+1})$ is also $\cup(k \log 2^k)$. we have $k = \log n$ (not exactly but floors and ceilings can be omitted). So $T(n)$ is $\cup(n \log n)$.

Comparing performance

Comparing two sorting algorithms we can infer that MergeSort beats InsertionSort for larger enough data due to the fact that $\cup(n \log n)$ grows more slowly than $\cup(n^2)$. Since we are concerned with asymptotic analysis constants are neglected. However, to know at which point does merge sort beats insertion sort we need finer analysis. If we can do this then we can develop and design adaptive algorithms for e.g. By mixing both algorithms such that small data set uses insertion sort while the large data use merge sort.

After these materials all the detail analysis of most of the algorithms are avoided students can try for it (if interested).

Exercises

1. Show that 2^n is $O(n!)$.
2. Show that the time complexity of TOH is $O(2^n)$.
- 3.

2.1.5: Prove theorem 2.1.

Theorem 2.1: For any functions $f(n)$ and $g(n)$, $f(n) = O(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

2.1.6: Prove that the running time of an algorithm is $O(g(n))$ if and only if its worst case running time is $O(g(n))$ and its best case running time is $\Omega(g(n))$.

4. $f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$, where a_0, a_2, \dots, a_n are real numbers with $a_n \neq 0$. Then show that $f(x)$ is $O(x^n)$, $f(x)$ is $\Omega(x^n)$ and then show $f(x)$ is order of x^n .

[Recurrences]

Design and Analysis of Algorithms (CSc 523)

Samujjwal Bhandari

Central Department of Computer Science and Information Technology (CDCSIT)

Tribhuvan University, Kirtipur,
Kathmandu, Nepal.

Recurrence relations, in the computer algorithms, can model the complexity of the running time, particularly divide and conquer algorithms. Recurrence relation is an equation or an inequality that is defined in term of itself. Recursions are useful in computer science because it makes the expression of an algorithm easy and can be solved easily. There are many natural problems that are easily described through recursion.

Examples

- a. $a_n = a_{n-1} + 1, a_1 = 1$; (what function?)
- b. $a_n = 2a_{n-1}, a_1 = 1$; (what function?)
- c. $a_n = na_{n-1}, a_1 = 1$; (what function?)
 - a. $a_n = n$ (polynomial)
 - b. $a_n = 2^n$ (exponential)
 - c. $a_n = n!$ (factorial)

Solving Recurrence Relations

See an example below

$$\begin{aligned}
 T(n) &= 2T(n-1) + k, \text{ where } T(0) = 1 \\
 &= 2(2T(n-2) + k) + k \\
 &= 4(2T(n-3) + k) + 3k = 2^3T(n-3) + (1+2+4)k \\
 &= 2^4T(n-4) + (2^0+2^1+2^2+2^3)k \\
 &\dots \\
 &= 2^pT(n-p) + (2^0+2^1+ \dots +2^{p-2}+2^{p-1})k \\
 &\text{let } p = n \text{ then,} \\
 T(n) &= 2^n + (2^0+2^1+ \dots +2^{n-2}+2^{n-1})k \{ \sum_{i=0}^{n-1} 2^i = 2^n - 1, i = 0 \text{ to } n \} \\
 &= 2^n + 2^n - 1 = 2 \cdot 2^n - 1 = O(2^n).
 \end{aligned}$$

Above we solved the recurrence relation but what are the available methods to solve the relation is next.

Solving Methods

No general procedure for solving recurrence relations is known solving the recurrence relation is an art. However there are generally used method that can solve problems of specific type and characteristics.

Linear homogeneous recurrence relations of degree k with constant coefficients is a recurrence relation of the form

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k}$$

where c_1, c_2, \dots, c_k are real numbers, and $c_k \neq 0$.

A sequence satisfying a recurrence relation above uniquely defined by the recurrence relation and the k initial conditions:

$a_0 = c_0, a_1 = c_1, \dots, a_{k-1} = c_{k-1}$ This kind of linear homogenous recurrence relation can be solved by constructing characteristic equation and solving it for answer (interested student can follow books on recurrence relation for more detail).

Now we try to solve the recurrence relation through different methods.

Note: we do not take care of floors, ceilings and boundary conditions in solving the relations since we are solving for asymptotic value and those conditions generates extra constants only.

Substitution Method

Given a recurrence relation

$$T(n) = 1, \text{ when } n = 1$$

$$T(n) = 2T(n/2) + n, \text{ when } n > 1$$

In this method to obtain lower or upper bound

1. We guess the solution
2. Prove that solution to be true by mathematical induction.

The question comes over guessing i.e. how to guess for the solution?

For the above given relation we can guess the upper bound solution is $O(n \log n)$ [we have already come through such a relation so guessing is easy]. For the guess to be true

$T(n) \leq c * n \log n$ must be satisfied for some positive constant c .

Since we guessed solution as $O(n \log n)$ we can assume

$$T(n/2) \leq c * n/2 \log(n/2)$$

Then

$$\begin{aligned} T(n) &\leq 2(c * n/2 \log(n/2)) + n \\ &= c * n \log(n/2) + n \\ &= c * n (\log n - \log 2) + n \\ &= c * n \log n + (1-c) * n \\ &\leq c * n \log n \text{ [this step holds as long as } c \geq 1] \end{aligned}$$

Now we have to prove that the solution holds for boundary condition. For this mathematical induction is required.

we have, $T(1) = 1$,

$$\begin{aligned} T(1) &\leq c * 1 \log 1 \\ &\leq 0 \text{ (false)} \end{aligned}$$

But since we can have some $n \geq n_0$, we may choose boundary condition for $n=2, 3 \dots$

Now we can find c large enough to hold the above relation with the help of extended boundary conditions $T(2)$ and $T(3)$.

Why $n=2$ and 3 for boundary condition? Because after $n > 3$ relation doesn't depend on $T(1)$.

$$\begin{aligned} \text{We obtain } T(2) &= 2T(1) + 2 && = 4 \\ T(3) &= 2T(1) + 3 && = 5 \quad [3/2 = 1 \text{ as integer division}] \end{aligned}$$

If we put any $c \geq 2$ then,

$$\begin{aligned} T(2) &\leq c \cdot 2 \log 2 \\ &\leq 2c \quad (\text{true}) \\ T(3) &\leq c \cdot 3 \log 3 \\ &\leq 3c \log 3 \quad (\text{true}) \end{aligned}$$

How to guess

Relate with similar previously done problem for e.g.

$$T(n) = 2T(n/2 + 3) + 3 + n$$

Here the problem is similar to the example previously done so we guess solution as $O(n \log n)$.

Choose good lower bound and upper bound and converge the solution by reducing range. For e.g. for above relation since we see n on the relation we guess lower bound as $\Omega(n)$ and upper bound as $O(n^2)$. Then we can increase lower and decrease upper bound to get tighter bound $\cup (n \log n)$.

Problem: Lower order term may defeat mathematical induction of substitution method.

Another Example

Consider the relation, $T(n) = 2T(n/2) + 1$

Guessing $T(n) = O(n)$

To show $T(n) \leq c \cdot n$

We can assume $T(n/2) \leq c \cdot n/2$ then,

$$\begin{aligned} T(n) &\leq 2(c \cdot n/2) + 1 \\ &\leq c \cdot n + 1 \text{ is not } \leq c \cdot n \text{ for any } c \text{ and } n, \text{ since we cannot subtract the } 1. \end{aligned}$$

Lets try to subtract the lower order term then,

Guess $T(n) = O(n)$

To show $T(n) \leq c \cdot n - b$ [since $c \cdot n - b = O(n)$]

Assume $T(n/2) \leq c \cdot n/2 - b$ then,

$$\begin{aligned} T(n) &\leq 2(c \cdot n/2 - b) + 1 \\ &\leq c \cdot n - 2b + 1 \\ &\leq c \cdot n - b, \text{ for } b \geq 1 \end{aligned}$$

Some time little modification on the variable can make the problem similar to other. For

e.g. $T(n) = 2T(\lfloor n \rfloor) + 1$

Take $m = \log n$ then $2^m = n$, i.e. $n^{1/2} = 2^{m/2}$ so

$$T(2^m) = 2T(2^{m/2}) + 1$$

Now rename $T(2^m)$ as $S(m)$ so the above relation can be written as

$$S(m) = 2S(m/2) + 1$$

This relation is simpler and we know solution to this, however you verify the result, so solution is $S(m) = O(m)$ changing to $T(n)$ we get $T(\log n)$.

Example Factorial

Factorial(n)

if $n < 1$

then return 1

*else return $n * \text{Factorial}(n-1)$*

The above algorithm has recurrence relation as

$$T(n) = 1 \quad n=0$$

$$T(n) = T(n-1) + O(1) \quad n>0$$

Guess $O(n)$, Show $T(n) \leq c*n$ [proof left as an exercise]

Iteration Method

The iteration method does not require guessing but it needs more experience in solving algebra. Here we are concerned with following steps.

1. Expand the relation so that summation dependent on n is obtained.
2. Bound the summation.

Take $T(n) = 2T(n/2) + 1$, $T(1) = 1$

Then we can expand the relation as

$$\begin{aligned}
T(n) &= 2T(n/2) + 1 \\
&= 2(2T(n/4) + 1) + 1 = 2^2T(n/2^2) + (1+2) \\
&= 2^3T(n/2^3) + (1+2+4) \\
&\dots \text{ Up to } p^{\text{th}} \text{ term} \\
&= 2^pT(n/2^p) + (2^0 + 2^1 + \dots + 2^{p-1})
\end{aligned}$$

observe that $(2^0 + 2^1 + \dots + 2^{p-1}) = 2^p - 1$

let $2^p = n$ then expanded relation becomes

$$\begin{aligned}
T(n) &= nT(1) + n-1 \\
&= \cup(n)
\end{aligned}$$

Note : iterate until the boundary is met.

Example

Given the recurrence relation

$$T(1) = \cup(1) \quad \text{when } n = 1$$

$$T(n) = T(n/3) + \cup(n) \text{ when } n > 1, \text{ solve using iteration method.}$$

Here,

$$\begin{aligned}
T(n) &= T(n/3) + \cup(n) \\
&= T(n/9) + \cup(n) + \cup(n) \\
&= T(n/3^3) + \cup(n) + \cup(n) + \cup(n) \\
&\dots \text{ Up to } p^{\text{th}} \text{ term} \\
&= T(n/3^p) + p\cup(n)
\end{aligned}$$

let $n = 3^p$ then $p = \log_3 n$ so,

$$\begin{aligned} T(n) &= T(1) + \log_3 n \cup(n) \\ &= \cup(n \log_3 n) \end{aligned}$$

This is Wrong

Right method

$$\begin{aligned} T(n) &= T(n/3) + \cup(n) \\ &= T(n/9) + \cup(n) + \cup(n/3) \\ &= T(n/3^3) + \cup(n) + \cup(n/3) + \cup(n/9) \\ &\dots \text{Up to } p^{\text{th}} \text{ term} \\ &= T(n/3^p) + \sum_{i=0}^{p-1} \cup(n/3^i) \end{aligned}$$

let $n = 3^p$ then,

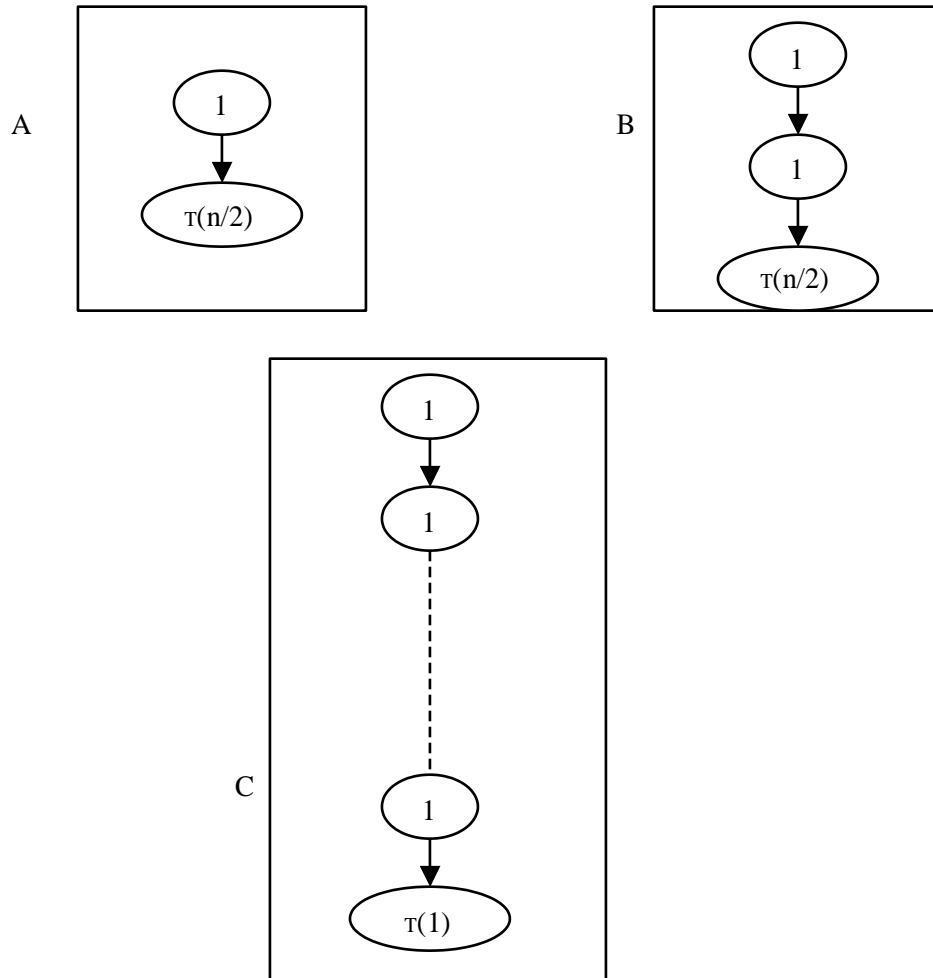
$$\begin{aligned} T(n) &= T(1) + \sum_{i=0}^{p-1} \cup(n/3^i) \\ &= 1 + \cup(n) \sum_{i=0}^{p-1} (1/3^i) \\ &= 1 + \cup(n) \left(\frac{1 - (1/3)^p}{1 - 1/3} \right) \\ &= \cup(3n/2) \\ &= \cup(n). \end{aligned} \quad \left\{ \sum_{i=0}^{p-1} a^i = \frac{1 - a^p}{1 - a} \text{ when } 0 < a < 1 \right\}$$

Recursion Tree Method

This method is just the modification to the iteration method for pictorial representation.

Given $T(n) = 1$

$$T(n) = T(n/2) + 1$$

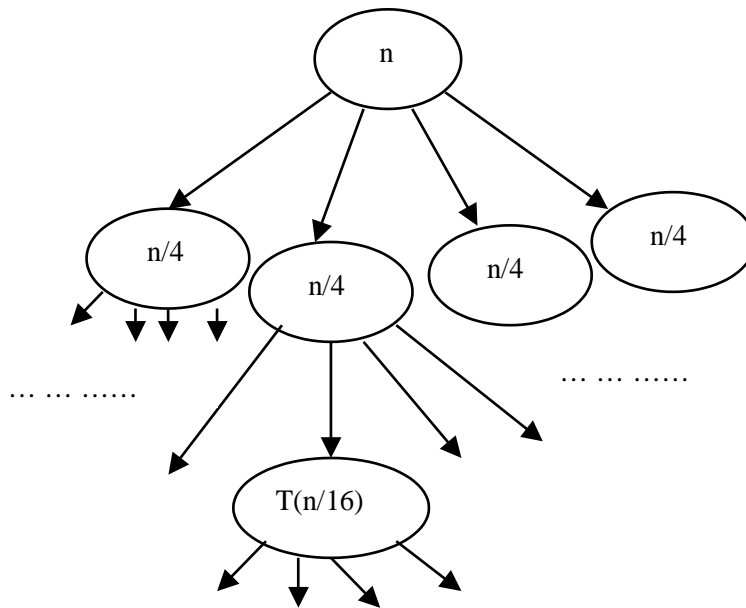
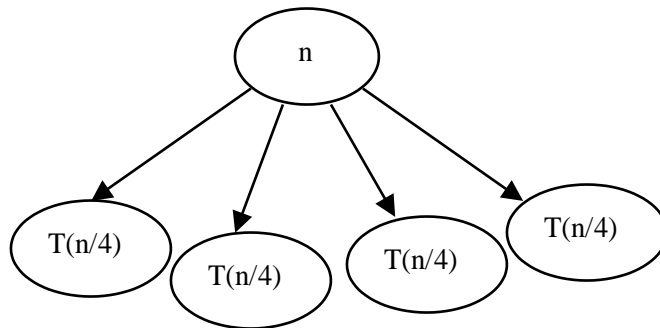


Here the height is $\log n$ so $T(n) = O(\log n)$

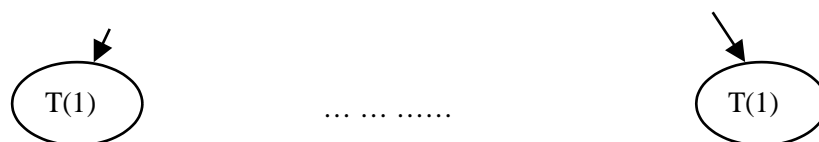
Example: Recursion Tree

Given $T(n) = 1$

$T(n) = 4T(n/4) + n$, solve by appealing to recurrence tree.



Continue like this



Add the terms at each level then you will obtain

$$n + n + n + \dots + n \text{ up to height of the tree (logn time)}$$

$$= n \log n$$

so $T(n) = O(n \log n)$

Master Method

Master method is used to solve the recurrences of the form

$$T(n) = aT(n/b) + f(n),$$

where $a \geq 1$ and $b > 1$ are constants and $f(n)$ is asymptotically positive function. This method is very handy because most of the recurrence relations we encounter in the analysis of algorithms are of the above kinds and it is very easy to use this method.

The above recurrence divides problem of size n into a sub problems each of size n/b where a and b are positive constants. The cost of dividing and combining the problem is given by $f(n)$.

Master Theorem: let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = aT(n/b) + f(n),$$

Where n/b is either n / b or $n / \lfloor b \rfloor$. Then $T(n)$ can be bounded asymptotically as follows.

1. If $f(n) = O(n^{\log_b a})$ for some constant > 0 , then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$.
3. If $f(n) = \Theta(n^{\log_b a})$ for some constant > 0 , and if $a f(n/b) \leq c f(n)$ for some constant $c < 1$ and all sufficiently large n , then $T(n) = \Theta(f(n))$.

In each of the three cases we are comparing $n^{\log_b a}$ and $f(n)$. Solution of the recurrence is derived from the larger of the two. The function that is smaller or greater must be polynomially smaller or greater by the factor of n for case 1 and 3. Additionally in case 3 is that regularity condition for polynomial $a f(n/b) \leq c f(n)$.

Examples

1. $T(n) = 9T(n/3) + n$

$$a=9, b=3, f(n) = n$$

$$n \log_3 9 = n^2$$

$$f(n) = n = O(n^{\log_3 9}), \text{ where } = 1, \text{ **Case 1**}$$

$$\text{So, } T(n) = \Theta(n^2)$$

2. $T(n) = 2T(n/2) + n$

$$a=2, b=2, f(n) = n$$

$$n \log_2 2 = n$$

$$f(n) = n = \Theta(n^{\log_2 2}), \text{ **Case 2**}$$

$$\text{So, } T(n) = \Theta(n \log n)$$

$$3. \quad T(n) = 3T(n/4) + n \log n$$

$$a=3, b=4, f(n) = n \log n$$

$$n \log_4 3 = O(n^{0.793})$$

$$f(n) = \Theta(n^{\log_4 3 + \epsilon}), \epsilon = (\text{nearly}) 0.2,$$

$$\text{For large } n, af(n/b) = 3(n/4) \log(n/4) \delta^{3/4} n \log n = cf(n), c=3/4, \text{ Case 3}$$

$$\text{So, } T(n) = \Theta(n \log n)$$

Exercises

1.

4.1.2: Show that the solution of $T(n) = 2T(n/2) + n$ is $\Theta(n \log n)$. Conclude that solution is $\Theta(n \log n)$.

4.1.5: Show that the solution to $T(n) = 2T(n/2 + 17) + n$ is $O(n \log n)$.

2. Write recursive Fibonacci number algorithm derive recurrence relation for it and solve by substitution method.

3.

4.2.2: Argue that the solution to the recurrence $T(n) = T(n/3) + T(2n/3) + n$ is $\Theta(n \log n)$ by appealing to a recursion tree.

4.2.4: Use iteration to solve the recurrence $T(n) = T(n-a) + T(a) + n$, where $a \geq 1$ is a constant.

4.

4.3.1: Use the master method to give tight asymptotic bounds for the following recurrences.

4.3.2: The running time of an algorithm A is described by the recurrence $T(n) = 7T(n/2) + n^2$. A competing algorithm A' has a running time of $T'(n) = aT'(n/4) + n^2$. What is the largest integer value for a such that A' is asymptotically faster than A?

[Recurrences]

Design and Analysis of Algorithms (CSc 523)

Samujjwal Bhandari

Central Department of Computer Science and Information Technology (CDCSIT)

Tribhuvan University, Kirtipur,
Kathmandu, Nepal.

Recurrence relations, in the computer algorithms, can model the complexity of the running time, particularly divide and conquer algorithms. Recurrence relation is an equation or an inequality that is defined in term of itself. Recursions are useful in computer science because it makes the expression of an algorithm easy and can be solved easily. There are many natural problems that are easily described through recursion.

Examples

- a. $a_n = a_{n-1} + 1, a_1 = 1$; (what function?)
- b. $a_n = 2a_{n-1}, a_1 = 1$; (what function?)
- c. $a_n = na_{n-1}, a_1 = 1$; (what function?)
 - a. $a_n = n$ (polynomial)
 - b. $a_n = 2^n$ (exponential)
 - c. $a_n = n!$ (factorial)

Solving Recurrence Relations

See an example below

$$\begin{aligned}
 T(n) &= 2T(n-1) + k, \text{ where } T(0) = 1 \\
 &= 2(2T(n-2) + k) + k \\
 &= 4(2T(n-3) + k) + 3k = 2^3T(n-3) + (1+2+4)k \\
 &= 2^4T(n-4) + (2^0+2^1+2^2+2^3)k \\
 &\dots \\
 &= 2^pT(n-p) + (2^0+2^1+ \dots +2^{p-2}+2^{p-1})k \\
 &\text{let } p = n \text{ then,} \\
 T(n) &= 2^n + (2^0+2^1+ \dots +2^{n-2}+2^{n-1})k \{ \sum_{i=0}^{n-1} 2^i = 2^n - 1, i = 0 \text{ to } n \} \\
 &= 2^n + 2^n - 1 = 2 \cdot 2^n - 1 = O(2^n).
 \end{aligned}$$

Above we solved the recurrence relation but what are the available methods to solve the relation is next.

Solving Methods

No general procedure for solving recurrence relations is known solving the recurrence relation is an art. However there are generally used method that can solve problems of specific type and characteristics.

Linear homogeneous recurrence relations of degree k with constant coefficients is a recurrence relation of the form

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k}$$

where c_1, c_2, \dots, c_k are real numbers, and $c_k \neq 0$.

A sequence satisfying a recurrence relation above uniquely defined by the recurrence relation and the k initial conditions:

$a_0 = c_0, a_1 = c_1, \dots, a_{k-1} = c_{k-1}$ This kind of linear homogenous recurrence relation can be solved by constructing characteristic equation and solving it for answer (interested student can follow books on recurrence relation for more detail).

Now we try to solve the recurrence relation through different methods.

Note: we do not take care of floors, ceilings and boundary conditions in solving the relations since we are solving for asymptotic value and those conditions generates extra constants only.

Substitution Method

Given a recurrence relation

$$T(n) = 1, \text{ when } n = 1$$

$$T(n) = 2T(n/2) + n, \text{ when } n > 1$$

In this method to obtain lower or upper bound

1. We guess the solution
2. Prove that solution to be true by mathematical induction.

The question comes over guessing i.e. how to guess for the solution?

For the above given relation we can guess the upper bound solution is $O(n \log n)$ [we have already come through such a relation so guessing is easy]. For the guess to be true

$T(n) \leq c * n \log n$ must be satisfied for some positive constant c .

Since we guessed solution as $O(n \log n)$ we can assume

$$T(n/2) \leq c * n/2 \log(n/2)$$

Then

$$\begin{aligned} T(n) &\leq 2(c * n/2 \log(n/2)) + n \\ &= c * n \log(n/2) + n \\ &= c * n (\log n - \log 2) + n \\ &= c * n \log n + (1-c) * n \\ &\leq c * n \log n \text{ [this step holds as long as } c \geq 1] \end{aligned}$$

Now we have to prove that the solution holds for boundary condition. For this mathematical induction is required.

we have , $T(1) = 1$,

$$\begin{aligned} T(1) &\leq c * 1 \log 1 \\ &\leq 0 \text{ (false)} \end{aligned}$$

But since we can have some $n \geq n_0$, we may choose boundary condition for $n=2, 3 \dots$

Now we can find c large enough to hold the above relation with the help of extended boundary conditions $T(2)$ and $T(3)$.

Why $n=2$ and 3 for boundary condition? Because after $n > 3$ relation doesn't depend on $T(1)$.

$$\begin{aligned} \text{We obtain } T(2) &= 2T(1) + 2 && = 4 \\ T(3) &= 2T(1) + 3 && = 5 \quad [3/2 = 1 \text{ as integer division}] \end{aligned}$$

If we put any $c \geq 2$ then,

$$\begin{aligned} T(2) &\leq c \cdot 2 \log 2 \\ &\leq 2c \quad (\text{true}) \\ T(3) &\leq c \cdot 3 \log 3 \\ &\leq 3c \log 3 \quad (\text{true}) \end{aligned}$$

How to guess

Relate with similar previously done problem for e.g.

$$T(n) = 2T(n/2 + 3) + 3 + n$$

Here the problem is similar to the example previously done so we guess solution as $O(n \log n)$.

Choose good lower bound and upper bound and converge the solution by reducing range. For e.g. for above relation since we see n on the relation we guess lower bound as $\Omega(n)$ and upper bound as $O(n^2)$. Then we can increase lower and decrease upper bound to get tighter bound $\cup (n \log n)$.

Problem: Lower order term may defeat mathematical induction of substitution method.

Another Example

Consider the relation, $T(n) = 2T(n/2) + 1$

Guessing $T(n) = O(n)$

To show $T(n) \leq c \cdot n$

We can assume $T(n/2) \leq c \cdot n/2$ then,

$$\begin{aligned} T(n) &\leq 2(c \cdot n/2) + 1 \\ &\leq c \cdot n + 1 \text{ is not } \leq c \cdot n \text{ for any } c \text{ and } n, \text{ since we cannot subtract the } 1. \end{aligned}$$

Lets try to subtract the lower order term then,

Guess $T(n) = O(n)$

To show $T(n) \leq c \cdot n - b$ [since $c \cdot n - b = O(n)$]

Assume $T(n/2) \leq c \cdot n/2 - b$ then,

$$\begin{aligned} T(n) &\leq 2(c \cdot n/2 - b) + 1 \\ &\leq c \cdot n - 2b + 1 \\ &\leq c \cdot n - b, \text{ for } b \geq 1 \end{aligned}$$

Some time little modification on the variable can make the problem similar to other. For

e.g. $T(n) = 2T(\lfloor n \rfloor) + 1$

Take $m = \lceil \log n \rceil$ then $2^m = n$, i.e. $n^{1/2} = 2^{m/2}$ so

$$T(2^m) = 2T(2^{m/2}) + 1$$

Now rename $T(2^m)$ as $S(m)$ so the above relation can be written as

$$S(m) = 2S(m/2) + 1$$

This relation is simpler and we know solution to this, however you verify the result, so solution is $S(m) = O(m)$ changing to $T(n)$ we get $T(\log n)$.

Example Factorial

Factorial(n)

if $n < 1$

then return 1

*else return $n * \text{Factorial}(n-1)$*

The above algorithm has recurrence relation as

$$T(n) = 1 \quad n=0$$

$$T(n) = T(n-1) + O(1) \quad n>0$$

Guess $O(n)$, Show $T(n) \leq c*n$ [proof left as an exercise]

Iteration Method

The iteration method does not require guessing but it needs more experience in solving algebra. Here we are concerned with following steps.

1. Expand the relation so that summation dependent on n is obtained.
2. Bound the summation.

Take $T(n) = 2T(n/2) + 1$, $T(1) = 1$

Then we can expand the relation as

$$\begin{aligned}
T(n) &= 2T(n/2) + 1 \\
&= 2(2T(n/4) + 1) + 1 = 2^2T(n/2^2) + (1+2) \\
&= 2^3T(n/2^3) + (1+2+4) \\
&\dots \text{ Up to } p^{\text{th}} \text{ term} \\
&= 2^pT(n/2^p) + (2^0 + 2^1 + \dots + 2^{p-1})
\end{aligned}$$

observe that $(2^0 + 2^1 + \dots + 2^{p-1}) = 2^p - 1$

let $2^p = n$ then expanded relation becomes

$$\begin{aligned}
T(n) &= nT(1) + n-1 \\
&= \cup(n)
\end{aligned}$$

Note : iterate until the boundary is met.

Example

Given the recurrence relation

$$T(1) = \cup(1) \quad \text{when } n = 1$$

$$T(n) = T(n/3) + \cup(n) \quad \text{when } n > 1, \text{ solve using iteration method.}$$

Here,

$$\begin{aligned}
T(n) &= T(n/3) + \cup(n) \\
&= T(n/9) + \cup(n) + \cup(n) \\
&= T(n/3^3) + \cup(n) + \cup(n) + \cup(n) \\
&\dots \text{ Up to } p^{\text{th}} \text{ term} \\
&= T(n/3^p) + p\cup(n)
\end{aligned}$$

let $n = 3^p$ then $p = \log_3 n$ so,

$$\begin{aligned} T(n) &= T(1) + \log_3 n \cup(n) \\ &= \cup(n \log_3 n) \end{aligned}$$

This is Wrong

Right method

$$\begin{aligned} T(n) &= T(n/3) + \cup(n) \\ &= T(n/9) + \cup(n) + \cup(n/3) \\ &= T(n/3^3) + \cup(n) + \cup(n/3) + \cup(n/9) \\ &\dots \text{Up to } p^{\text{th}} \text{ term} \\ &= T(n/3^p) + \sum_{i=0}^p \cup(n/3^i) \end{aligned}$$

let $n = 3^p$ then,

$$\begin{aligned} T(n) &= T(1) + \sum_{i=0}^p \cup(n/3^i) \\ &= 1 + \cup(n) \sum_{i=0}^p (1/3^i) \\ &= 1 + \cup(n) \left(\frac{1 - (1/3)^{p+1}}{1 - 1/3} \right) \\ &= \cup(n) \left(\frac{1 - (1/3)^{p+1}}{2/3} \right) \\ &= \cup(n) \left(\frac{3}{2} (1 - (1/3)^{p+1}) \right) \\ &= \cup(n) \left(\frac{3}{2} (1 - 1/3^{p+1}) \right) \\ &= \cup(n) \left(\frac{3}{2} (1 - 1/n) \right) \\ &= \cup(n) \left(\frac{3}{2} - \frac{3}{2n} \right) \\ &= \cup(n) \left(\frac{3}{2} \right) \\ &= \cup(n) \end{aligned}$$

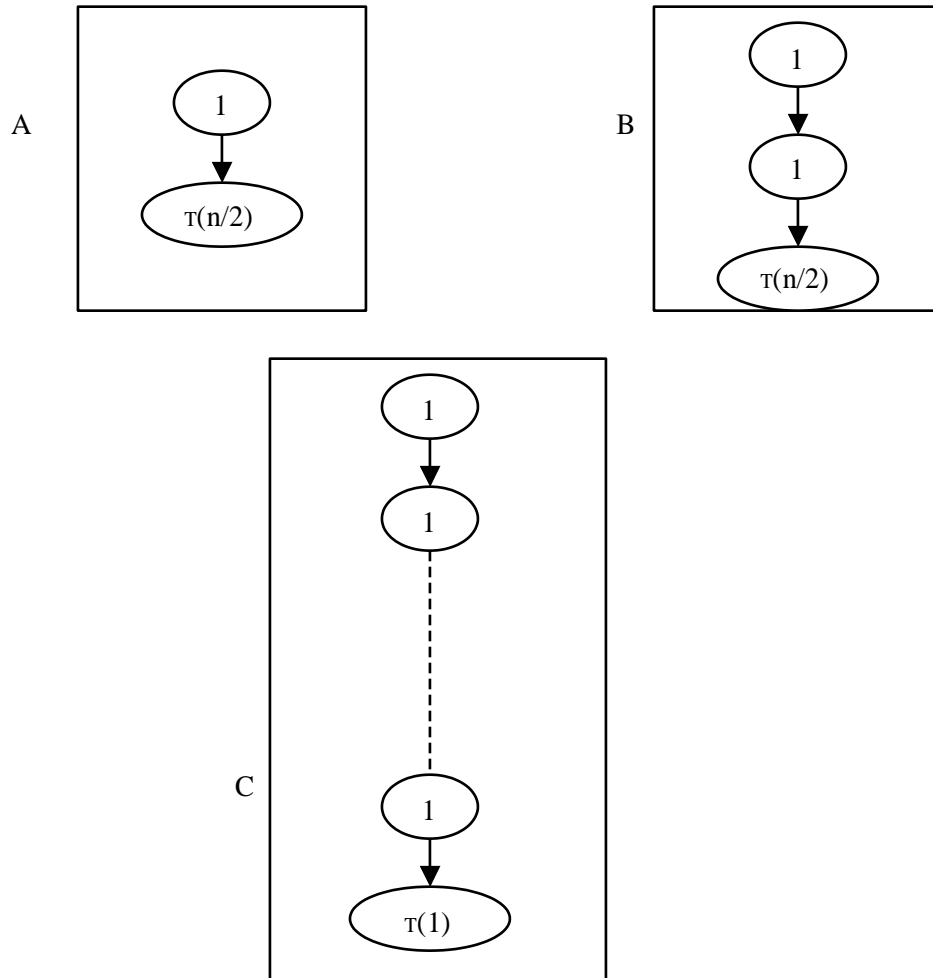
$$\left\{ \sum_{i=0}^p a^i \right\} \text{ H } 1/(1 - a) \text{ when } 0 < a < 1$$

Recursion Tree Method

This method is just the modification to the iteration method for pictorial representation.

Given $T(n) = 1$

$$T(n) = T(n/2) + 1$$

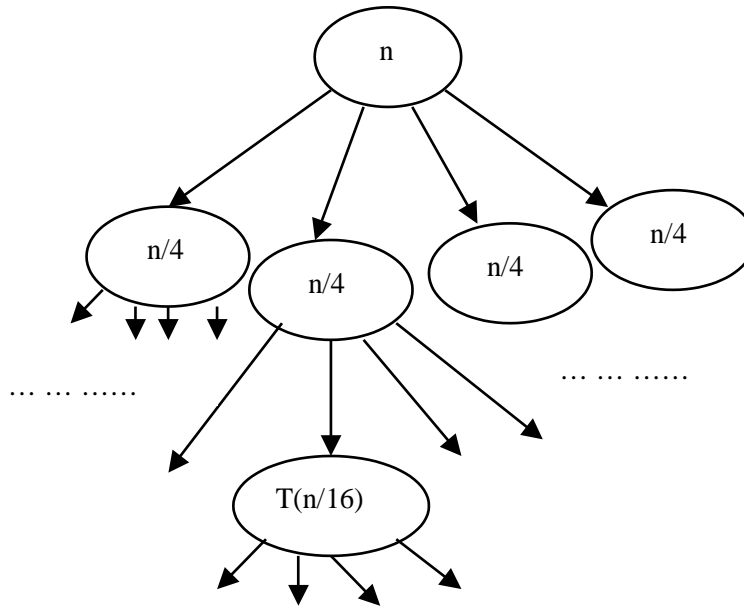
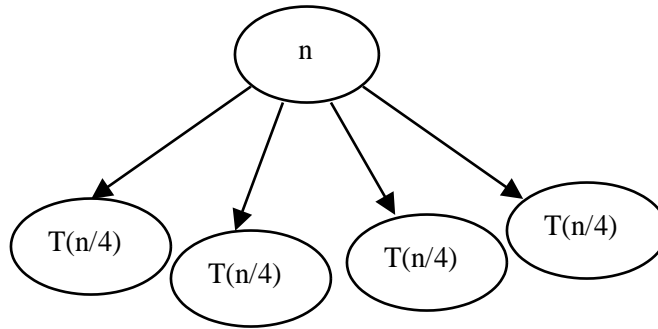


Here the height is $\log n$ so $T(n) = O(\log n)$

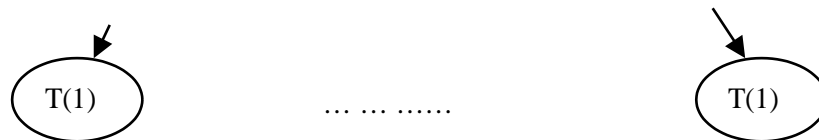
Example: Recursion Tree

Given $T(n) = 1$

$T(n) = 4T(n/4) + n$, solve by appealing to recurrence tree.



Continue like this



Add the terms at each level then you will obtain

$$n + n + n + \dots + n \text{ up to height of the tree (logn time)}$$

$$= n \log n$$

so $T(n) = O(n \log n)$

Master Method

Master method is used to solve the recurrences of the form

$$T(n) = aT(n/b) + f(n),$$

where $a \geq 1$ and $b > 1$ are constants and $f(n)$ is asymptotically positive function. This method is very handy because most of the recurrence relations we encounter in the analysis of algorithms are of the above kinds and it is very easy to use this method.

The above recurrence divides problem of size n into a sub problems each of size n/b where a and b are positive constants. The cost of dividing and combining the problem is given by $f(n)$.

Master Theorem: let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = aT(n/b) + f(n),$$

Where n/b is either n / b or n / b . Then $T(n)$ can be bounded asymptotically as follows.

1. If $f(n) = O(n^{\log_b a})$ for some constant > 0 , then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$.
3. If $f(n) = \Theta(n^{\log_b a})$ for some constant > 0 , and if $a f(n/b) \leq c f(n)$ for some constant $c < 1$ and all sufficiently large n , then $T(n) = \Theta(f(n))$.

In each of the three cases we are comparing $n^{\log_b a}$ and $f(n)$. Solution of the recurrence is derived from the larger of the two. The function that is smaller or greater must be polynomially smaller or greater by the factor of n for case 1 and 3. Additionally in case 3 is that regularity condition for polynomial $a f(n/b) \leq c f(n)$.

Examples

1. $T(n) = 9T(n/3) + n$

$$a=9, b=3, f(n) = n$$

$$n^{\log_3 9} = n^2$$

$$f(n) = n = O(n^{\log_3 9}), \text{ where } = 1, \text{ **Case 1**}$$

$$\text{So, } T(n) = \Theta(n^2)$$

2. $T(n) = 2T(n/2) + n$

$$a=2, b=2, f(n) = n$$

$$n^{\log_2 2} = n$$

$$f(n) = n = \Theta(n^{\log_2 2}), \text{ **Case 2**}$$

$$\text{So, } T(n) = \Theta(n \log n)$$

$$3. \quad T(n) = 3T(n/4) + n \log n$$

$$a=3, b=4, f(n) = n \log n$$

$$n \log_4 3 = O(n^{0.793})$$

$$f(n) = \Theta(n^{\log_4 3 + \epsilon}), \epsilon = (\text{nearly}) 0.2,$$

$$\text{For large } n, af(n/b) = 3(n/4) \log(n/4) \delta^{3/4} n \log n = cf(n), c=3/4, \text{ Case 3}$$

$$\text{So, } T(n) = \Theta(n \log n)$$

Exercises

1.

4.1.2: Show that the solution of $T(n) = 2T(n/2) + n$ is $\Theta(n \log n)$. Conclude that solution is $\Theta(n \log n)$.

4.1.5: Show that the solution to $T(n) = 2T(n/2 + 17) + n$ is $O(n \log n)$.

2. Write recursive Fibonacci number algorithm derive recurrence relation for it and solve by substitution method.

3.

4.2.2: Argue that the solution to the recurrence $T(n) = T(n/3) + T(2n/3) + n$ is $\Theta(n \log n)$ by appealing to a recursion tree.

4.2.4: Use iteration to solve the recurrence $T(n) = T(n-a) + T(a) + n$, where $a \geq 1$ is a constant.

4.

4.3.1: Use the master method to give tight asymptotic bounds for the following recurrences.

4.3.2: The running time of an algorithm A is described by the recurrence $T(n) = 7T(n/2) + n^2$. A competing algorithm A' has a running time of $T'(n) = aT'(n/4) + n^2$. What is the largest integer value for a such that A' is asymptotically faster than A?

[Data Structures Overviews]

Design and Analysis of Algorithms (CSc 523)

Samujjwal Bhandari

Central Department of Computer Science and Information Technology (CDCSIT)

Tribhuvan University, Kirtipur,

Kathmandu, Nepal.

Since the main aim of this part is to introduce some of the data structures if you want rigorous study you can always consult the book on Data Structures.

Simple Data structures

The basic structure to represent unit value types are bits, integers, floating numbers, etc. The collection of values of basic types can be represented by arrays, structure, etc. The access of the values are done in constant time for these kind of data structured.

Linear Data Structures

Linear data structures are widely used data structures we quickly go through the following linear data structures.

- Lists

- Stacks and queues

a. Lists

List is the simplest general-purpose data structure. They are of different variety. Most fundamental representation of a list is through an array representation. The other representation includes linked list. There are also variety of representations for lists as linked list like singly linked, doubly linked, circular, etc.

There is a mechanism to point to the first element. For this some pointer is used. To traverse there is a mechanism of pointing the next (also previous in doubly linked).

Lists require linear space to collect and store the elements where linearity is proportional to the number of items. For e.g. to store n items in an array nd space is require were d is size of data. Singly linked list takes $n(d + p)$, where p is size of pointer. Similarly for doubly linked list space requirement is $n(d + 2p)$.

Array representation(ordered list)

- Operations require simple implementations.
- Insert, delete, and search, require linear time, search can take $O(\log n)$!!
- Inefficient use of space

Singly linked representation (unordered)

- Insert and delete in $O(1)$ time, for deletion pointer must be given otherwise $O(n)$.
- Search in $O(n)$ time
- Memory overhead, but allocated only to entries that are present.

Doubly linked representation

- Insert and delete in $O(1)$ time !!

Operations on lists

create(): create an empty list

destroy():

IsEmpty()

Length():

Find(k): find k'th element

Search(x): find position of x

delete():

insert(x): insert x after the current element element
and plenty of others

b. Stacks and Queues

These types of data structures are special cases of lists. Stack also called LIFO (Last In First Out) list. In this structure items can be added or removed from only one end. Stacks are generally represented either in array or in singly linked list and in both cases insertion/deletion time is $O(1)$.

Operations on stacks

Create()	IsEmpty()	IsFull()
Top()	push(x)	pop()

The queues are also like stacks but they implement FIFO(First In First Out) policy. One end is for insertion and other is for deletion. They are represented mostly circularly in array for $O(1)$ insertion/deletion time. Circular singly linked representation takes $O(1)$ insertion time and $O(n)$ deletion time, or vice versa. Representing queues in doubly linked list have $O(1)$ insertion and deletion time.

Operations on queues

Create()	IsEmpty()	IsFull()
First()	Last()	Enqueue(x)
Dequeue()		

Tree Data Structures

Tree is a collection of nodes. If the collection is empty the tree is empty otherwise it contains a distinct node called root (r) and zero or more subtrees whose roots are directly connected to the node r by edges. The root of each tree is called child of r , and r the parent. Any node without a child is called leaf (you may find other definition from any data structure book). We can also call the tree as a connected graph without a cycle. So

there is a path from one node to any other nodes in the tree. The main concern with this data structure is due to the running time of most of the operation require $O(\log n)$. we can represent tree as an array or linked list.

Some of the definitions

- Level h of a full tree has d^{h-1} nodes.
- The first h levels of a full tree have

$$1 + d + d^2 + \dots + d^{h-1} = \frac{d^h - 1}{d - 1}$$

nodes.

- A tree of height h and degree d has at most $d^h - 1$ elements

Remember the following terminology tree, subtree, complete tree, root, leaf, parent, children, level, degree of node, degree of tree, height, tree traversal, etc.

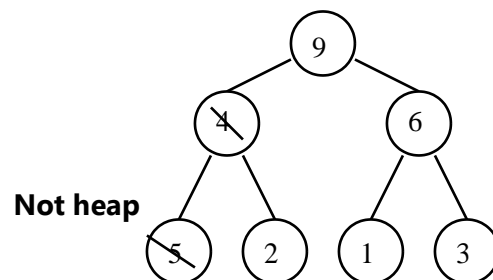
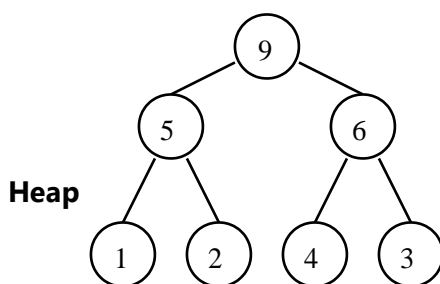
Priority Queue Trees

Heap

A heap is a complete tree with an ordering-relation R holding between each node and its descendant. *Note that the complete tree here means tree can miss only rightmost part of the bottom level.*

R can be smaller-than, bigger-than.

E.g. Heap with degree 2 and R is “bigger than”.



Applications

Priority Queue A dynamic set in which elements are deleted according to a given ordering-relation.

Heap Sort Build a heap from the given set ($O(n)$) time, then repeatedly remove the elements from the heap ($O(n \log n)$).

Implementation

An array

Insertion and deletion of an element takes $O(\log n)$ time. More on this later.

Search Trees

a. Binary Search Trees

BST has at most two children for each parent. In BST a key at each vertex must be greater than all the keys held by its left descendants and smaller or equal than all the keys held by its right descendants.

Searching and insertion both takes $O(h)$ worst case time, where h is height of tree and the relation between height and number of nodes n is given by $\log n < h+1 \leq n$. for e.g. height of binary tree with 16 nodes may be anywhere between 4 and 15.

When height is 4 and when height is 15?

So if we are sure that the tree is height balanced then we can say that search and insertion has $O(\log n)$ run time. Other wise we have to content with $O(n)$.

b. AVL Trees

Balanced tree named after Adelson, Velskii and Landis. AVL trees consist of a special case in which the subtrees of each node differ by at most 1 in their height (Read more on your own like rotations, insertions, deletions, etc.).

c. Red Black Trees

A binary search tree in which

- All nodes are either red or black.
- The root and leaves are colored black.
- All the paths from the root to the leaves agree on the number of black nodes
- No path from the root to a leaf may contain two consecutive nodes colored red
- Red nodes have only black children

Empty subtrees of a node are treated as subtrees with roots of black color.

The relation $n \geq 2^{h/2} - 1$ implies the bound $h \leq 2 \log_2(n + 1)$.

Insertion

First insert new node as in BST. If it is root color it black else color it red. When we color the non-root node as red then many cases arises some of them violating the red black tree properties.

Case 1: if the parent is black, it is ok

Case 2: If the parent and parent's sibling (better say uncle) are red. Change the color of parent, uncle and grandparent. Some node g (parent, uncle and grandparent) may be root if so do not change the color. This process will introduce property violation in upper part so continue the process.

Case 3: Red parent, Black uncle; the node is "outside". Rotate the parent about grandparent and recolor them.

Case 4: Red parent, Black uncle; the node is “inside”. Double rotation; recolor node and grandparent.

Note: normal insertion running time $O(h)$ i.e. $O(\log n)$ [$h \leq 2 \log_2(n + 1)$] and the rotation needs extra $O(1)$ time.

c. Multiway Trees

A m-way search tree is a tree in which

- The nodes hold between 1 to m-1 distinct keys
- The keys in each node are sorted
- A node with k values has k+1 subtrees, where the subtrees may be empty.
- The i th subtree of a node $[v_1, \dots, v_k]$, $0 < i < k$, may hold only values v in the range $v_i < v < v_{i+1}$ (v_0 is assumed to equal $-\infty$, and v_{k+1} is assumed to equal ∞).

A m-way tree of height h has between h and $m^h - 1$ keys.

Insertion

- Search the key going down the tree until reaching an empty subtree
- Insert the key to the parent of the empty subtree, if there is room in the node.
- Insert the key to the subtree, if there is no room in its parent.

Deletion

- If the key to be deleted is between two empty subtrees, delete it
- If the key is adjacent to a nonempty subtree, replace it with the largest key from the left subtree or the smallest key from the right subtree.

d. B Trees

A m-way tree in which

- The root has at least one key
- Non-root nodes have at least $\lceil m/2 \rceil$ subtrees (i.e., at least $\lceil (m-1)/2 \rceil$ keys)
- All the empty subtrees (i.e., external nodes) are at the same level

B-trees are especially useful for trees stored on disks, since their height, and hence also the number of disk accesses, can be kept small.

The growth and contraction of m-way search trees occur at the leaves. On the other hand, B-trees grow and contract at the root.

Insertion

- Insert the key to a leaf
- Overfilled nodes should send the middle key to their parent, and split into two at the location of the submitted key.

Deletion

- Key that is to be removed from a node with non-empty subtrees is being replaced with the largest key of the left subtree or the smallest key in the right subtree. (The replacement is guaranteed to come from a leaf.)
- If a node becomes under staffed, it looks for a sibling with an extra key. If such a sibling exists, the node takes a key from the parent, and the parent gets the extra key from the sibling.
- If a node becomes under staffed, and it can't receive a key from a sibling, the node is merged with a sibling and a key from the parent is moved down to the node.

e. Splay Trees

These trees do not keep extra balancing data so also known as self-adjusting trees. Splaying a node means moving it up to the root by the help of rotation. Most of the rotation are double and possibly single at the ends. Double rotations have two cases “zig-zig” and “zig-zag”.

Here let x is the non-root node on the access path at which we are rotating. If the parent of x is the root we just rotate x and the root otherwise x has both parent and grandparent, here we need to consider above-mentioned two cases. In the case of “zig-zag” x is a right child and p is the left child (or vice versa). Otherwise we will have “zig-zig” case where both x and p lies on the same side i.e. they are both, either left children, or right children.

Operations

Search: Find the node with a given key if not found splay key's predecessor or successor node.

Insertion: Insert new node as in other case and splay it.

Deletion: Find the node to be deleted, splay it and delete.

Complexity

Splay trees are not explicitly balanced but with each new operation it tends to be more balanced. The main idea behind splay tree is that even though some operation takes $O(n)$ times the but for m operation it takes $O(m \log n)$ time where $O(\log n)$ is amortized running time for each operation. Another concept is 90-10 rule i.e. in practice 90% of processes access 10% of data.

Suggested Exercises

1. The *level order* listing of the nodes of a tree first lists the root, then all the nodes of depth 1, then all the nodes of depth 2, and so on. Order of listing of nodes of the same level is left to right. Write an algorithm to list the nodes of a tree in *level order* (*writing program also is good*).
2. Give an algorithm to sort 5 elements with 7 comparisons.

[Sorting]

Design and Analysis of Algorithms (CSc 523)

Samujjwal Bhandari

Central Department of Computer Science and Information Technology (CDCSIT)

Tribhuvan University, Kirtipur,

Kathmandu, Nepal.

We talk about sorting very much and in every part of our study of computer science sorting is studied more than others. The reasons for this are sorting is the most fundamental operation that is done most by the computer, the sorting algorithms are widely studied problem and different varieties of algorithms are known. Most of the ideas for algorithm design and analysis can be obtained from sorting algorithms like divide and conquer, lower bounds etc.

Applications of Sorting:

Sorting being the fundamental operation in computer science has many applications.

Searching:

Searching speeds up by the use of sorted elements. Binary search is an example that takes sorted sequence of keys to search and the whole searching operation is done in $O(\log n)$ time.

Closest pair:

Given n numbers, we have to find the pair which are closest to each other. Once the numbers are sorted, the closest pair will be next to each other in sorted order, so an $O(n)$ linear scan completes the job.

Element uniqueness:

Given n elements, are there any duplicate values for all elements or they are unique. When the elements are sorted we can check every adjacent elements by linear searching.

Frequency Distribution:

Given n numbers of element, when the elements are sorted we can linearly search the elements to calculate the number of times the elements repeats. This can be applied for determining Huffman codes for each letters.

Median and selection:

Given n elements, to find k th largest or smallest element we sort the elements and find it in constant time (array is used).

Convex hulls:

Given n numbers of points find the smallest area polygon such that every points is within that polygon. When elements are sorted we can improve the efficiency.

Selection Sort

In selection sort the all the elements are examined to obtain smallest (greatest) element at one pass then it is placed into its position, this process continues until the whole array is sorted.

Algorithm:

Selection-sort(A)

```
{  
    for(  $i = 0; i < n ; i++$ )  
        for(  $j = i + 1; j < n ; j++$ )  
            if ( $A[j] < A[i]$ )  
                swap( $A[i], A[j]$ )  
}
```

From the above algorithm it is clear that time complexity is $O(n^2)$. We can also see that for every instances of input complexity is $O(n^2)$.

Bubble Sort

The bubble sort algorithm compares adjacent elements. If the first is greater than the second, swap them. This is done for every pair of adjacent elements starting from first two elements to last two elements. Here the last element is the greatest one. The whole process is repeated except for the last one so that at each pass the comparison becomes fewer.

Algorithm:

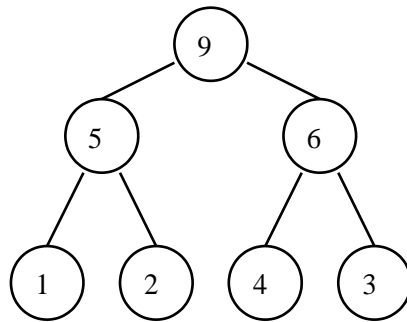
BubbleSort(A, n)

```
{
  for( $i = n - 1; i > 0; i--$ )
    for( $j = 0; j < i; j++$ )
      if( $A[j] > A[j+1]$ )
        {
          temp =  $A[j]$ ;
           $A[j] = A[j+1]$ ;
           $A[j+1] = temp$ ;
        }
}
```

The above algorithm for any instances of input runs in $O(n^2)$ time. This algorithm is similar to insertion sort algorithm studied earlier but operates in reverse order. However there is no best-case linear time complexity for this algorithm as of insertion sort.

Heap Sort

We studied about the heap data structure. Here we present the implementation of the heap in sorting called heap sort. We can consider heap as max heap or min heap. Consider the heap below:

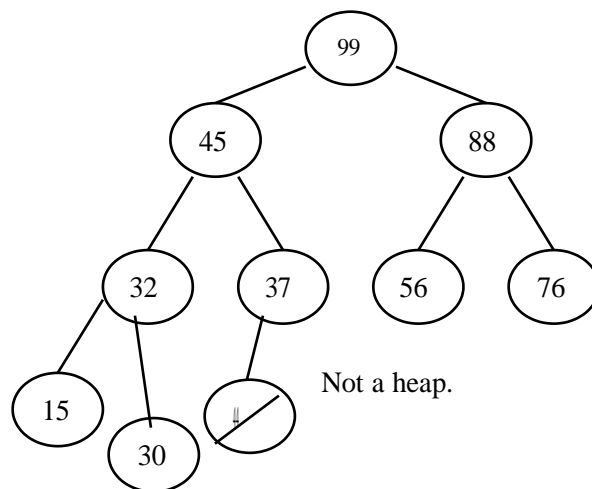


We represent the above max heap as an array like:

1	2	3	4	5	6	7
9	5	6	1	2	4	3

Remember if parent p is at $A[i]$ place then its left child is at $A[2i]$ and the right child is at $A[2i + 1]$

Is the sequence 99, 45, 88, 32, 37, 56, 76, 15, 30, 44 a max heap?



Constructing Heaps (max)

To create a heap, insert an element into the leftmost empty slot of an array if the inserted element is larger than its parent then swap the elements and recur. At every step we swap the element with parent to maintain the heap order.

All the levels but the last one are filled completely so height (h) of n elements is bounded as

$$\sum_{i=1}^h 2^i = 2^{h+1} - 1 \geq n, \text{ so } h = \log n$$

Doing n such insertion takes $O(n \log n)$.

Parent(i)

return i / 2

Left(i)

return 2i

Right(i)

return 2i+1

Building a Heap

The bottom up procedure for constructing heap is a good way but there is a process that uses merge method to create a heap called heapify. Here giving the two heaps and new element those two heaps are merged for single root.

Algorithm:

BuildHeap(A)

```
{
    heapsize[A] = length[A]
    for i = length[ A] / 2 to 1
        do Heapify(A,i)
}
```

In the above algorithm the first step takes constant time and the second and third steps run about $n/2$ so the time complexity is *Complexity of heapify* * $n/2$. see analysis below.

Algorithm:*Heapify(A,i)*

```

{
    l = left(i)
    r = Right(i)
    if l <= heapsize[A] and A[l] > A[i]
        max = l
    else
        max = i
    if r <= heapsize[A] and A[r] > A[max]
        max = r
    if max != i
    {
        swap(A[i],A[max])
        Heapify(A,max)
    }
}

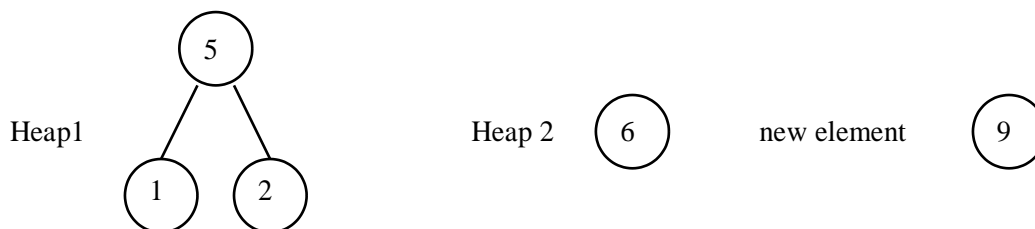
```

Analysis :

Time complexity for *Heapify(A,i)* for n node tree is given by

$$T(n) \leq T(2n/3) + O(1)$$

Here $2/3$ represents for worst case where the two heaps that are getting merged differ by level 1. i.e. the last level is exactly half filled.



What is the resulting heap after merging? See the number of elements in last level.

Use master method on the above recurrence relation where $a = 1$, $b = 3/2$, $f(n) = O(1)$ and $\log_{3/2} 1 = 0 = O(1)$.

Observe that it gives **case 2** so the solution is $O(\log n)$.

So the building of heap is $O(n \log n)$. This bound is asymptotic but not tight.

Exact Analysis:

In full binary tree of n nodes there are $n/2$ nodes that are leaves, $n/4$ nodes that have height 1, $n/8$ nodes of height 2, ... and so on so heapify acts on small heaps for many times.

So there are at most $n/2^{h+1}$ nodes at height h , so cost of building heap is

$$\sum_{h=0}^{\log n} \left(\frac{n}{2^{h+1}} \right) O(h) = O(n \sum_{h=0}^{\log n} \frac{h}{2^h})$$

see above that the series is not quite geometric. However the series converges so that there is some ending point.

Proving convergence:

We know

$$\sum_{i=0}^{\infty} x^i = 1/(1-x) \text{ for } 0 < x < 1$$

Taking derivatives of both sides

$$\sum_{i=0}^{\infty} i x^{i-1} = 1/(1-x)^2.$$

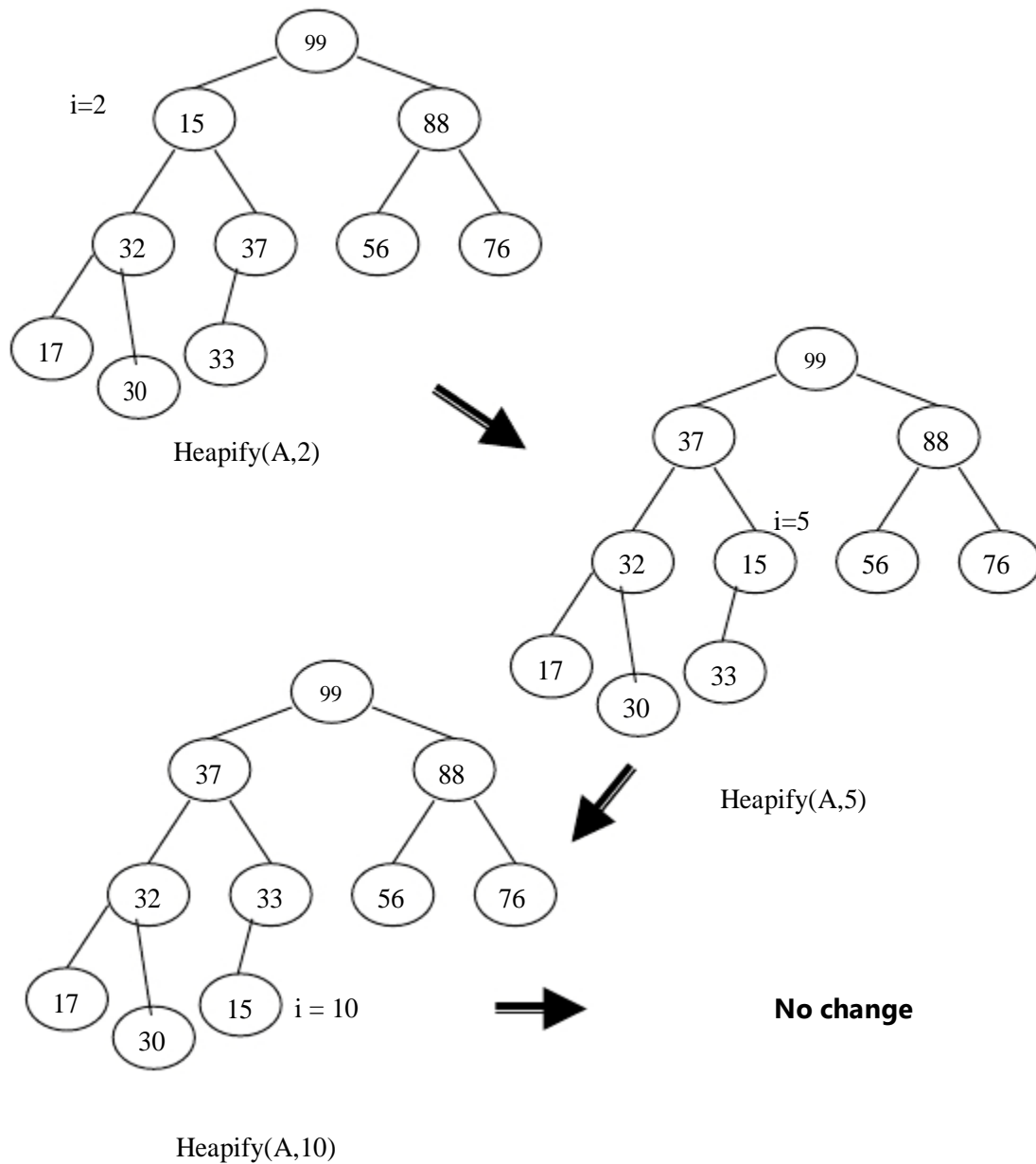
Multiplying by x on both sides

$$\sum_{i=0}^{\infty} i x^i = x/(1-x)^2.$$

Substituting $1/2$ for x we get summation as 2.

So BuildHeap(A) time complexity is $O(2n)$ i.e. linear.

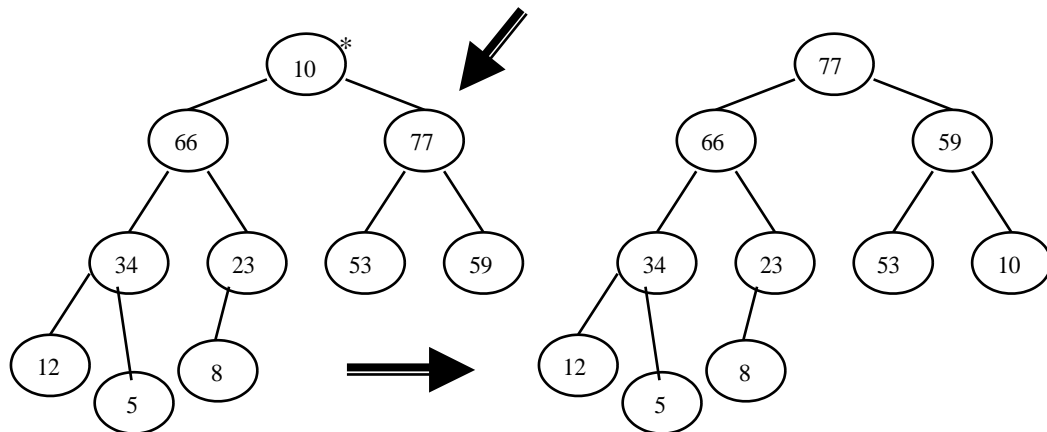
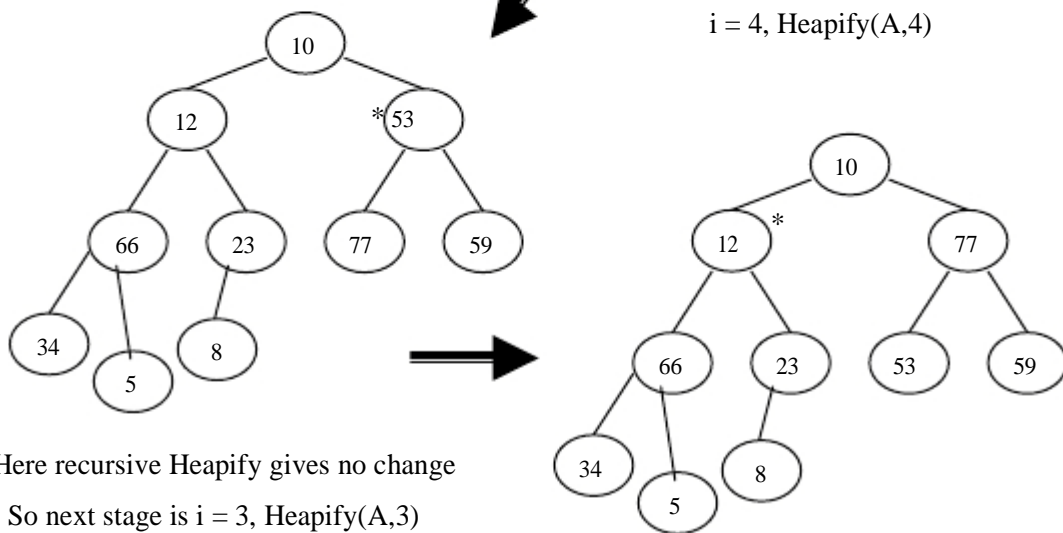
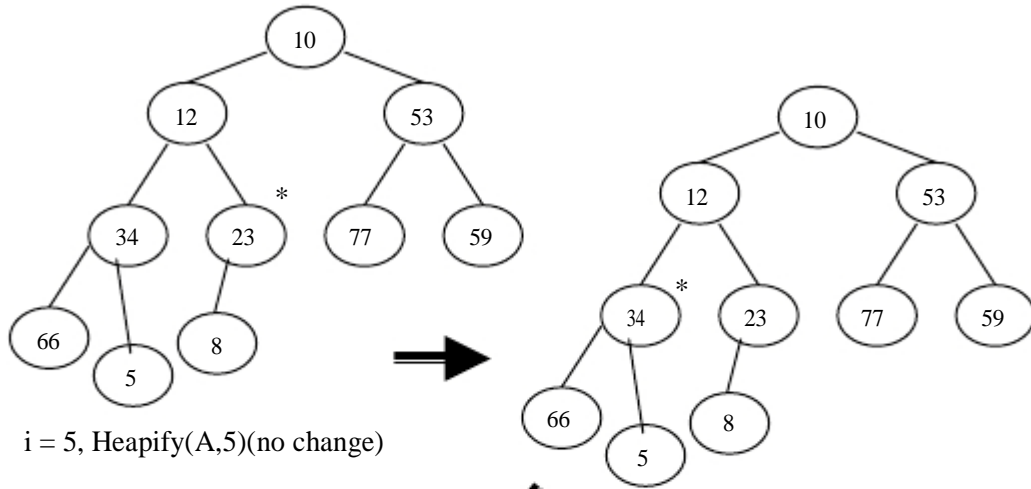
Lesson from above analysis: do not over estimate, analyze carefully to get tighter bound.

Running Example of Heapify for $\text{heapsize}[A] = 10$ 

In the above running example the action of `Heapify(A,2)` is shown. Where the recursive calls to `Heapify(A,5)` and then `Heapify(A,2)` are called. When `Heapify(A,2)` is called no changes occurs to the data structure.

Running Example of BuildHeap

$A[] = \{10, 12, 53, 34, 23, 77, 59, 66, 5, 8\}$



Algorithm:*HeapSort(A)*

```

{
    BuildHeap(A);           //into max heap
    m = length[A];
    for(i = m ; i >= 2; i--)
    {
        swap(A[1],A[m]);
        m = m-1;
        Heapify(A,1);
    }
}

```

Analysis:

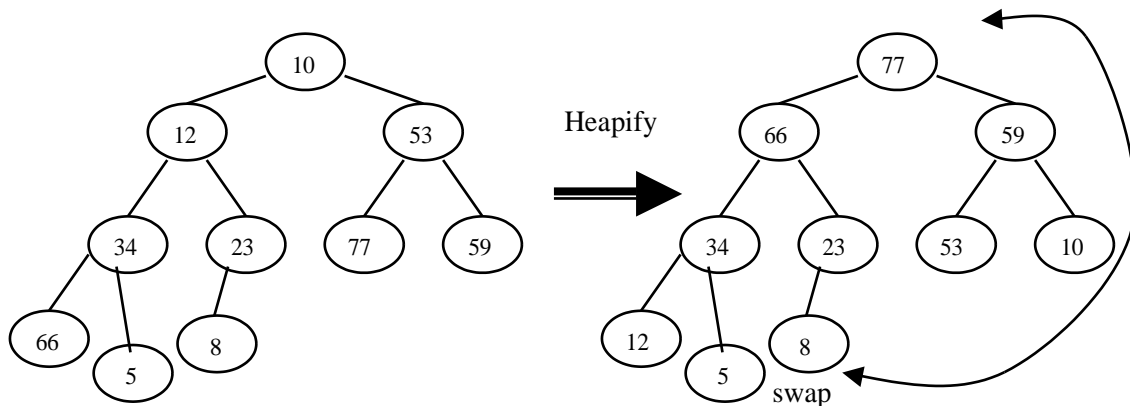
BuildHeap takes $O(n)$ times.

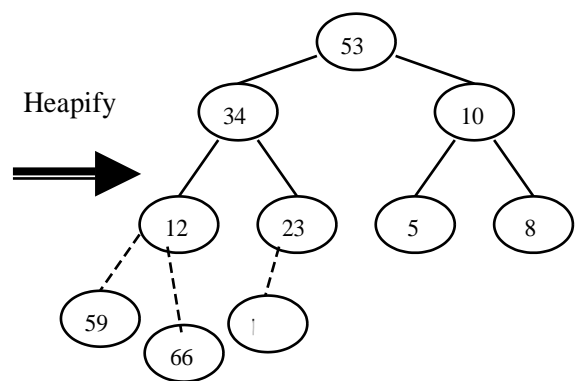
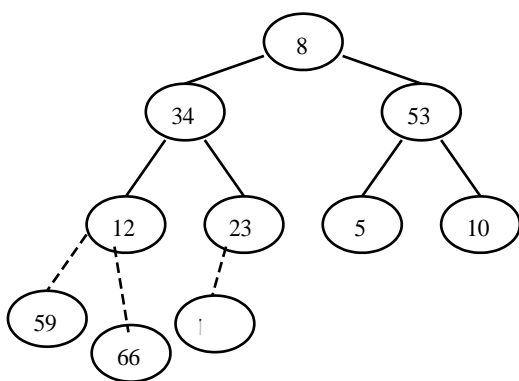
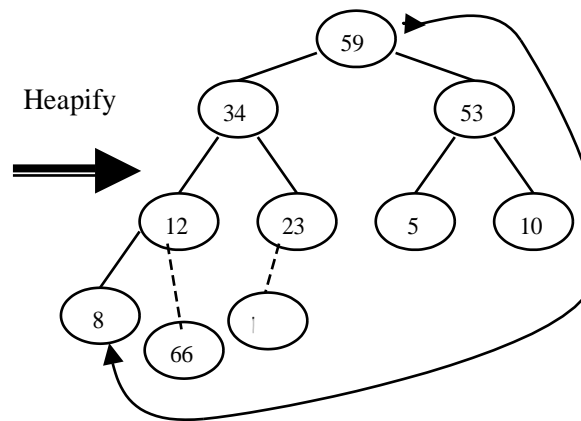
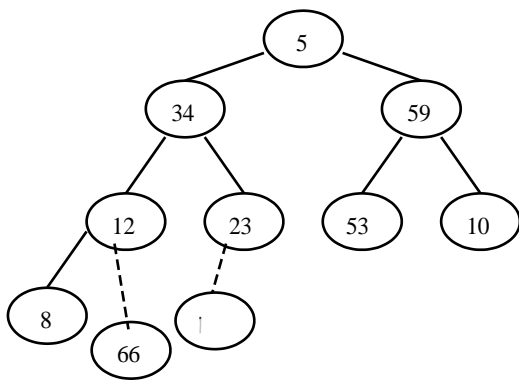
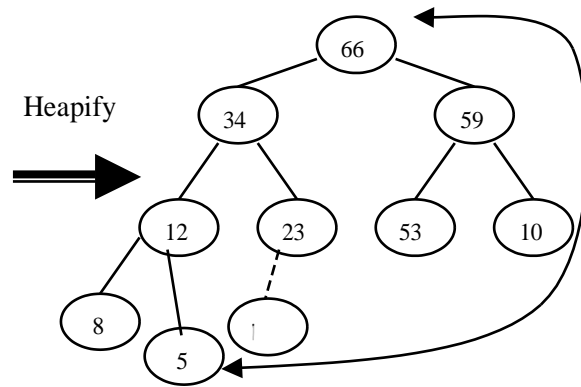
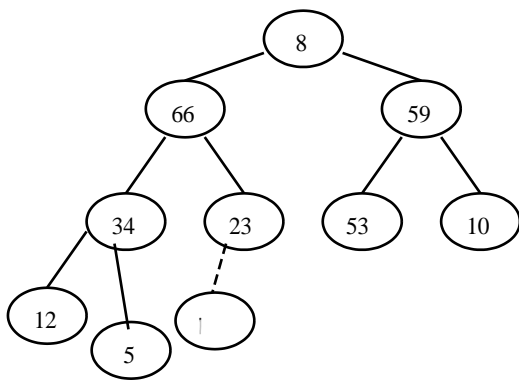
For loop executes for n times but each instruction inside loop executes for $n-1$ times where instruction inside loops takes $O(\log n)$ time.

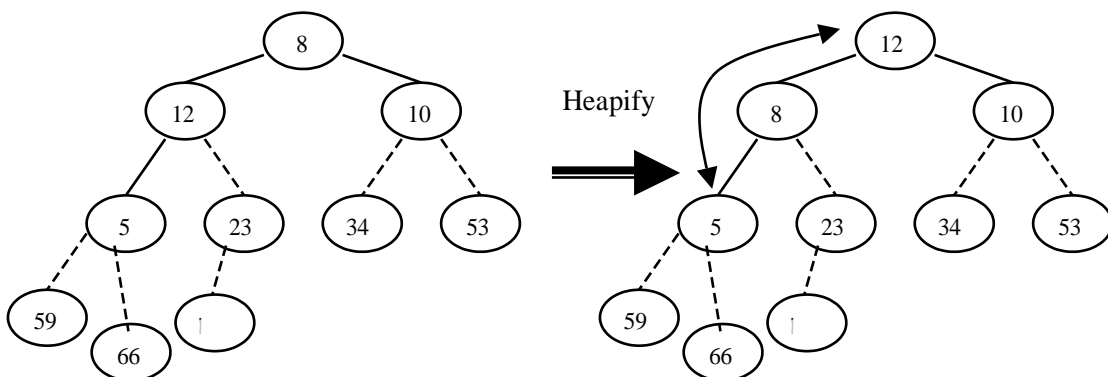
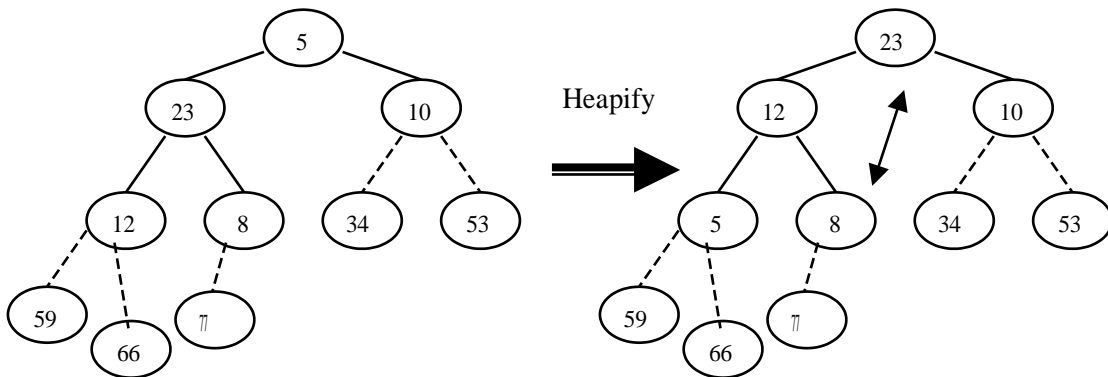
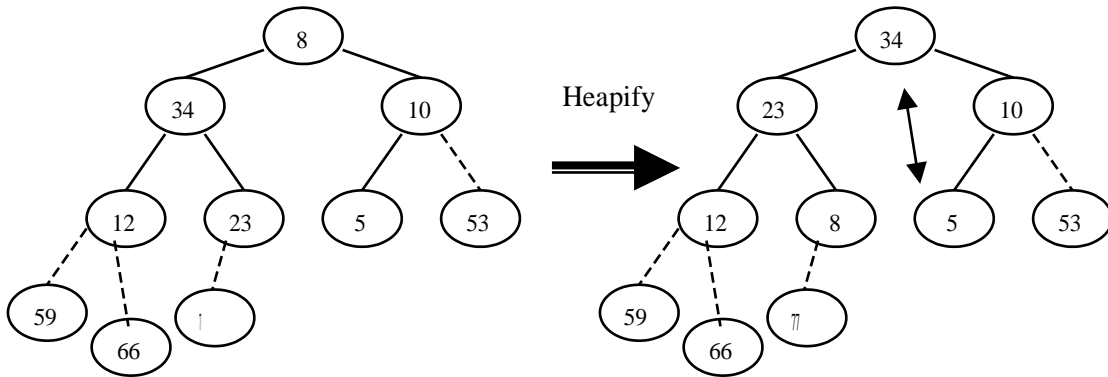
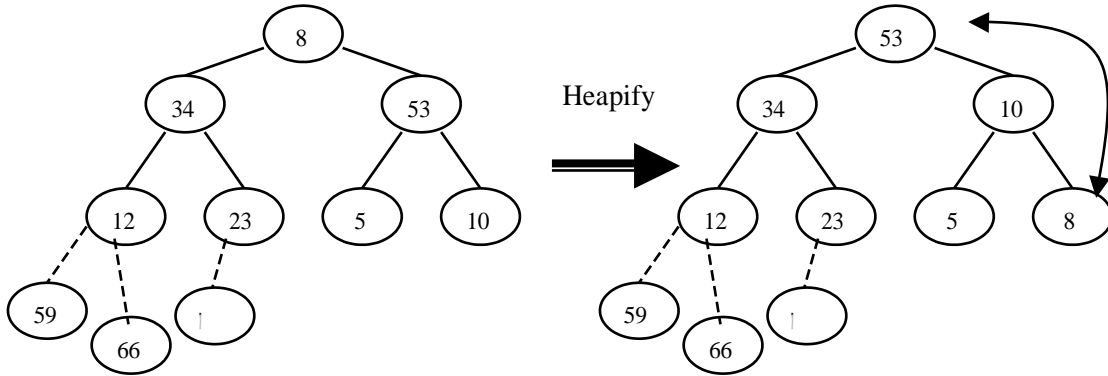
So total time $T(n) = O(n) + (n-1) O(\log n) = O(n \log n)$.

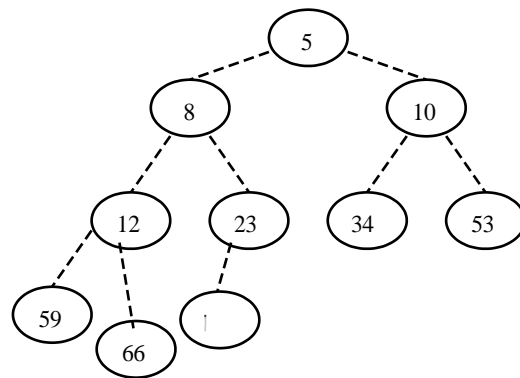
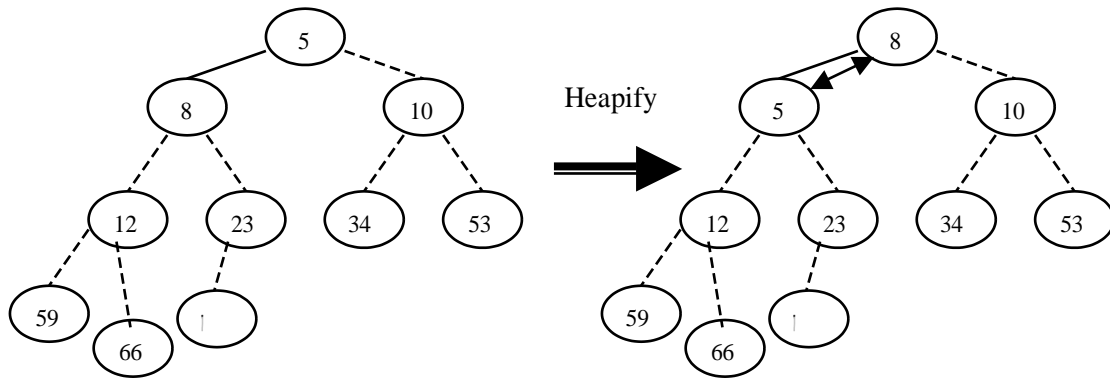
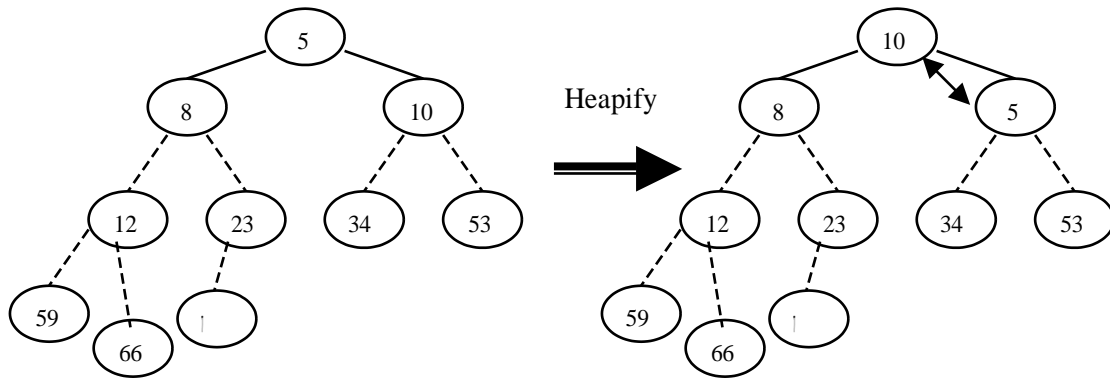
Running example of HeapSort:

$A[] = \{10, 12, 53, 34, 23, 77, 59, 66, 5, 8\}$









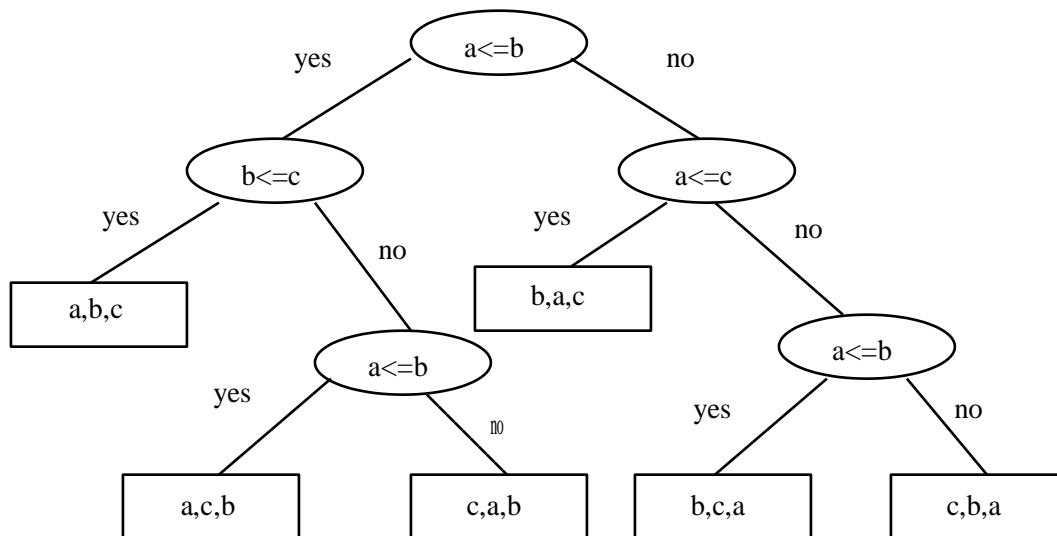
Sorted sequence

$A[] = \{5, 8, 10, 12, 23, 34, 53, 59, 66, 77\}$

Self study: priority queues

Lower Bound for Sorting

Comparison sorts algorithms compare pair of data for relation greater than or smaller than. They work in same manner for every kind of data. Consider all the possible comparison runs of sorting algorithm on input size n . When we unwind loops and recursive calls and focus on the point where there is comparison, we get binary decision tree. The tree describes all the possibilities for the algorithm execution on an input of n elements.



At the leaves, we have original input in the sorted order. By comparison we can generate all the possible permutations of the inputs are possible so there must be at least $n!$ leaves in the tree. A complete tree of height h has 2^h leaves, so our decision tree must have $h \geq \log_2(n!)$, but $\log_2(n!)$ is $\Theta(n \log n)$. Thus, the number of comparison in the worst case is $\Theta(n \log n)$. So we can say that runtime complexity of any comparison sort is $\Theta(n \log n)$.

Exercises

1. Understand what is stable sorting algorithm and identify the stable sorting algorithms we have studied.
2. Give the running example for Heapify, BuildHeap and Heapsort for the 14 elements input.
3. what is the running time of heap sort on array A of length n that is already sorted in increasing order? what about decreasing order?

[Divide and Conquer Paradigm]

Design and Analysis of Algorithms (CSc 523)

Samujjwal Bhandari

Central Department of Computer Science and Information Technology (CDCSIT)

Tribhuvan University, Kirtipur,

Kathmandu, Nepal.

In the ancient time Roman politicians used the concept of dividing the enemy unity somehow to conquer them. The method they used was not conceived for algorithms. While designing any algorithm if we try to break the problem of larger size into smaller then we adhere to the designing technique called “Divide and Conquer”. Most of the computational problems can be solved by using this technique. Analysis of such an algorithm developed using divide and conquer paradigm needs recurrence since divide and conquer algorithms use recursion.

The main elements of the Divide and Conquer Solutions are:

Divide: Divide the larger problem into the problem of smaller size.

Conquer: Solve each piece by applying recursive divide and conquer.

Combine: add up all the solved pieces to a larger solution.

Binary Search

Algorithm: An array of elements A[] has sorted elements (here in nondecreasing order)

```
BinarySearch(low, high, key){
    if(high == low){
        if(key == A[low]) then return low+1; //index starts from 0
        else return 0;
    }
    else{
        mid = (low + high) /2 ; //integer division
        if(key == A[mid]) then return mid+1;
        else if (key < A[mid]) then return BinarySearch(low, mid-1, key) ;
        else return BinarySearch(mid+1, high, key) ;
    }
}
```

We can see that the above algorithm terminates whenever key value is found or both the index for an array low and high equals. If result is found it is returned otherwise 0 is returned. (Students should verify the correctness in detail).

Efficiency:

From the above algorithm we can say that the running time of the algorithm is

$$\begin{aligned} T(n) &= T(n/2) + \cup(1) \\ &= \cup(\log n) \text{ (verify)}. \end{aligned}$$

In the best case output is obtained at one run i.e. $\cup(1)$ time if the key is at middle.

In the worst case the output is at the end of the array so running time is $\cup(\log n)$ time.

In the average case also running time is $\cup(\log n)$.

For unsuccessful search best, worst and average time complexity is $\cup(\log n)$.

Running example

Take input array $A[] = \{2, 5, 7, 9, 18, 45, 53, 59, 67, 72, 88, 95, 101, 104\}$

For key = 2

low	high	mid	
0	13	6	key < A[6]
0	5	2	key < A[2]
0	1	0	

Terminating condition, since $A[\text{mid}] = 2$, return 1(successful).

For key = 103

low	high	mid	
0	13	6	key > A[6]
7	13	10	key > A[10]
11	13	12	key > A[12]
13	13	-	

Terminating condition $high == low$, since $A[0] \neq 103$, return 0(unsuccessful).

For key = 67

low	high	mid	
0	13	6	key > A[6]
7	13	10	key < A[10]
7	9	8	

Terminating condition, since $A[mid] = 67$, return 9(successful).

Fast Exponentiation

Here our aim is to calculate the value of a_n . In simple way we can do this in by $n-1$ multiplication. However, we can improve the running time by using divide and conquer strategy. Here we can break n into two parts i and j such that we have $n = i + j$. So we can write $a_n = a_i * a_j$. if we use $i = n / 2$ then we can have, if n is even then $a_n = a_i * a_i$, and if n is odd $a_n = a_i * a_j * a$.

Algorithm:*Fastexp(a,n)*

```

{
    if(n == 1)
        return a;
    else
        x = Fastexp(a, n / 2);
        if(n is odd)
            return x*x*a;
        else
            return x*x;
}

```

Correctness:

In the above algorithm every time the value of x is a_i where $i = n / 2$, in both the conditions returned value will be a_n . hence by induction it follows.

Efficiency:

Observe the above algorithm then you will find that each time problem is divided into approximately half part and the cost of dividing and combining the problem constant. So we can write time complexity as

$$T(n) = T(n / 2) + O(1)$$

Solving this relation we get $O(\log n)$.

If we concern about space then the stacks are called upto $\log n$ time hence $O(\log n)$.

Max and Min Finding

Here our problem is to find the minimum and maximum items in a set of n elements. We see two methods here first one is iterative version and the next one uses divide and conquer strategy to solve the problem.

Algorithm:

IterMinMax(A,n)

```
{
    max = min = A[0];
    for(i = 1; i < n; i++)
    {
        if(A[i] > max)
            max = A[i];
        if(A[i] < min)
            min = A[i];
    }
}
```

The above algorithm requires $2(n-1)$ comparison in worst, best, and average cases Since the comparison $A[i] < min$ is needed only if $A[i] > max$ is not true. if we replace the content inside the for loop by

```
if(A[i] > max)
    max = A[i];
else if(A[i] < min)
    min = A[i];
```

Then the best case occurs when the elements are in increasing order with $(n-1)$ comparisons and worst case occurs when elements are in decreasing order with $2(n-1)$ comparisons. For the average case $A[i] > max$ is about half of the time so number of comparisons is $3n/2 - 1$.

We can clearly conclude that the time complexity is $O(n)$.

Now let us see the divide and conquer strategy to solve the problem.

Algorithm:

MinMax(i,j,max,min)

```

{
    if(i == j)
        max = min = A[i];
    else if(i = j-1)
    {
        if(A[i] < A[j])
        {
            max = A[j]; min = A[i];
        }
        else
        {
            max = A[i]; min = A[j];
        }
    }
    else
    {
        //Divide the problems
        mid = (i + j)/2;           //integer division
        //solve the subproblems
        MinMax(i,mid,max,min);
        MinMax(mid + 1,j,max1,min1);
        //Combine the solutions
        if(max1 > max)      max = max1;
        if(min1 < min)     min = min1;
    }
}

```


The above algorithm adopts the following idea; if the number of elements is 1 or 2 then max and min are obtained trivially. Otherwise split problem into approximately equal part and solved recursively.

Analysis:

Here we analyze in terms of number of comparisons as cost because elements may be polynomials, strings, real numbers, etc. Now we can give recurrence relation as below for MinMax algorithm in terms of number of comparisons.

$$T(n) = T(n/2) + T(n/2) + 2, \text{ if } n > 2$$

$$T(n) = 1, \text{ if } n = 2$$

$$T(n) = 0, \text{ if } n = 1$$

We can simplify above relation to

$$T(n) = 2T(n/2) + \cup(1).$$

Solving the recurrence by using master method complexity is $\cup(n)$.

Note: Both the algorithms we have studied above have $\cup(n)$ complexity but due to space overhead in divide and conquer approach we prefer iterative one.

Integer Multiplication

Here our problem definition gives

Inputs:

$$X = x_{n-1}, x_{n-2}, \dots, x_0.$$

$$Y = y_{n-1}, y_{n-2}, \dots, y_0. \text{ Here } X \text{ and } Y \text{ are } n \text{ digit positive numbers.}$$

Output:

$$Z = z_{2n-1}, z_{2n-2}, \dots, z_0. \text{ Here } Z \text{ is product } X * Y \text{ of digits } 2n.$$

Our school method solves this problem in $O(n^2)$ time (how?). But we use divide and conquer approach algorithm (by A.A. Karatsuba, in 1962) to solve the problem asymptotically faster. The number can be of any base but for simplicity assume decimal numbers. Now we can have

$$X = \sum_{i=0}^{n-1} (x_i) * 10^i \text{ and } Y = \sum_{i=0}^{n-1} (y_i) * 10^i$$

So that $Z = X * Y$ is

$$\sum_{i=0}^{2n-1} (z_i) * 10^i = \sum_{i=0}^{n-1} (x_i) * 10^i * \sum_{i=0}^{n-1} (y_i) * 10^i$$

For example

$$X = 141 = 1 * 10^2 + 4 * 10 + 1 * 10^0 \quad ; \quad Y = 123 = 1 * 10^2 + 2 * 10 + 3 * 10^0.$$

$$Z = 1 * 10^4 + 7 * 10^3 + 3 * 10^2 + 4 * 10 + 3 * 10^0. = 17343$$

Now if we suppose,

$$A = x_{n-1}, x_{n-2}, \dots, x_{n/2}.$$

$$B = x_{(n/2)-1}, x_{(n/2)-2}, \dots, x_0.$$

$$C = y_{n-1}, y_{n-2}, \dots, y_{n/2}.$$

$$D = y_{(n/2)-1}, y_{(n/2)-2}, \dots, y_0.$$

Then we have

$$X = A * 10^{(n/2)} + B$$

$$Y = C * 10^{(n/2)} + D, \text{ and}$$

$$\begin{aligned} Z &= (A * 10^{(n/2)} + B) * (C * 10^{(n/2)} + D) \\ &= A * C * 10^n + (A * D + B * C) * 10^{n/2} + B * D. \end{aligned}$$

For example if $X = 2345$, $Y = 5684$, $A = 23$, $B = 45$, $C = 56$, $D = 84$ then

$$X = 23 * 10^2 + 45$$

$$Y = 56 * 10^2 + 84 \text{ then}$$

$$Z = 23 * 56 * 10^4 + (23 * 84 + 45 * 56) * 10^2 + 45 * 84 = 13328980 = 2345 * 5684.$$

The terms $(A*C)$, $(A*D)$, $(B*C)$, and $(B*D)$ are each products of 2 $(n/2)$ -digit numbers. From the above facts we can generate an idea that can be applied to design an algorithm for integer multiplication. The main idea is

If the two numbers are of single digit can be multiplied immediately (Boundary).

If $n > 1$ then the product of 2 n digit numbers can be divided into 4 products of $n/2$ digit numbers (Divide and Conquer).

Calculating product of two numbers requires additions of 4 products that can be done in linear time and multiplication by the power of 10 that can also be done in linear time (Combine). (Devising algorithm as an exercise #1)

Our above discussion needs the representation of an algorithm for multiplying 2 n digit numbers by 4 products of $n/2$ digit number, however Karatsuba discovered how the product of 2 n -digit numbers could be expressed in terms of three products each of 2 $n/2$ digit numbers. This concept however increases the number of steps required in combine process though the complexity is $O(n)$.

Suppose

$$U = A*C$$

$$V = B*D$$

$$W = (A+B)*(C+D)$$

Then we have $A*D + B*C = W - U - V$ so

$$\begin{aligned} Z = X*Y &= A*C*10^n + (A*D + B*C)*10^{n/2} + B*D. \\ &= U*10^n + (W - U - V)*10^{n/2} + V. \end{aligned}$$

Algorithm: A, B, C, D, U, V, W are as defined in our discussion

ProdInt(X,Y,n)

{

if(n == 1)

*return X[0]*Y[0];*

```

else
{
    A = X[n-1] ...X[n/2];
    B = X[(n/2)-1] ... X[0];
    C = Y[n-1] ...Y[n/2];
    D = Y[(n/2)-1] ... Y[0];
    U = ProdInt(A,C,n/2);
    V = ProdInt(B,D,n/2);
    W = ProdInt((A+B),(C+D),n/2);
    return U*10n + (W - U - V)* 10n/2 + V;
}
}

```

Analysis:

From the above algorithm we can give recurrence relation as

$$T(n) = 3T(n/2) + O(n).$$

Solving this recurrence by using master method where $a = 3$, $b = 2$, we get

$$T(n) = O(n^{1.58}).$$

Quick Sort

Quick sort developed by C.A.R Hoare is an unstable sorting. In practice this is the fastest sorting method. This algorithm is based on the divide and conquer paradigm. The main idea behind this sorting is partitioning of the elements.

Steps for Quick Sort

Divide: partition the array into two nonempty sub arrays.

Conquer: two sub arrays are sorted recursively.

Combine: two sub arrays are already sorted in place so no need to combine.

Definition: Partitioning

An array $A[]$ is said to be partitioned about $A[t]$ if all the elements on the left of $A[t]$ are less than or equal to and the all the elements on the right are greater than or equal to $A[t]$.

Example: $A[] = \{ 3, 7, 6, 15, 14, 18, 25, 55, 32, 45, 37 \}$

Here the array $A[]$ is partitioned about $A[5]$ (index 0, 1, 2,).

Algorithm:

Partition(A,i,j)

```

{
    x = i - 1; y = j + 1; v = A[i];
    while(x < y)
    {
        do {
            x++;
        } while(A[x] <= v);
        do {
            y--;
        } while(A[y] >= v);
        if(x < y)
            swap(A[x], A[y]);
    }
    A[i] = A[y]; A[y] = v;
    return y;
}

```

Analysis:

In the above algorithm either left pointer or right pointer moves at a time and scans the array at most 1 time. You can note that at most n swaps are done hence the time complexity for above algorithm is $O(n)$.

Running Example

Take an array $A[] = \{ 16, 7, 6, 15, 14, 18, 25, 55, 32, 45, 37 \}$

(v) 16 7 6 15 14(y) (x) 18 25 55 32 45 37
 $A[i] = A[4]; A[4] = 16; i$ is 0 here

[14 7 6 15] 16 [18 25 55 32 45 37]
 Partitioned sub arrays.

Algorithm:

QuickSort(A,p,q)

```
{
    if(p < q)
    {
        r = Partition(A,p,q);
        QuickSort(A,p,r-1);
        QuickSort(A,r+1,q);
    }
}
```

Analysis:

The quick sort algorithm time complexity i.e. the time to sort an array $A[]$ can be written as the recurrence relation

$$T(n) = T(k-1) + T(n-k) + O(n).$$

Where k is the result of $\text{Partition}(A,1,n)$

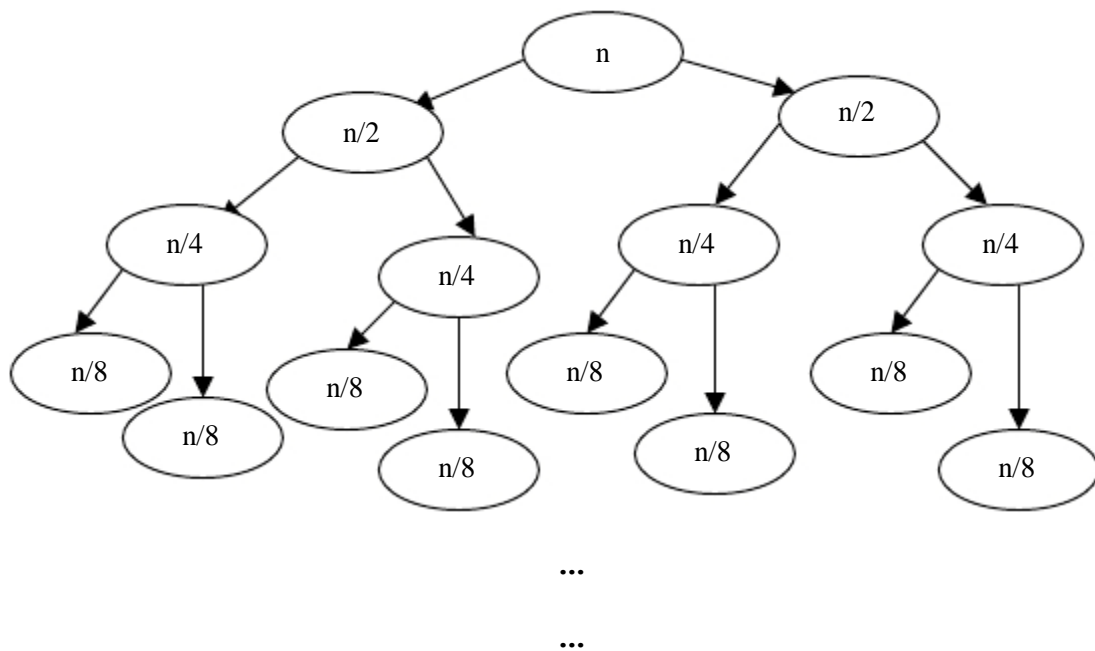
Best Case:

The best case for divide and conquer relation occurs when division is as balanced as possible. Here best case occurs if we can choose pivot as the middle part of the array at all time, so we have

$$T(n) = 2T(n/2) + O(n).$$

Solving this recurrence we get $T(n) = O(n \log n)$.

The tree for best case is like



Up to point when all are 1

This gives the total sum of the nodes as $O(n \log n)$ (verify!).

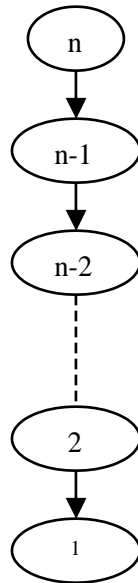
Worst Case:

Worst case occurs if the partition gives the pivot as first element (last element) all the time i.e. $k=1$ or $k=n$ this happens when the elements are completely sorted, so we have

$$T(n) = T(n-1) + O(n).$$

Solving this relation we get $O(n^2)$.

The tree for worst case is like



Adding all the levels we get

$$1+2+3+ \dots + n = O(n^2)(\text{verify!}).$$

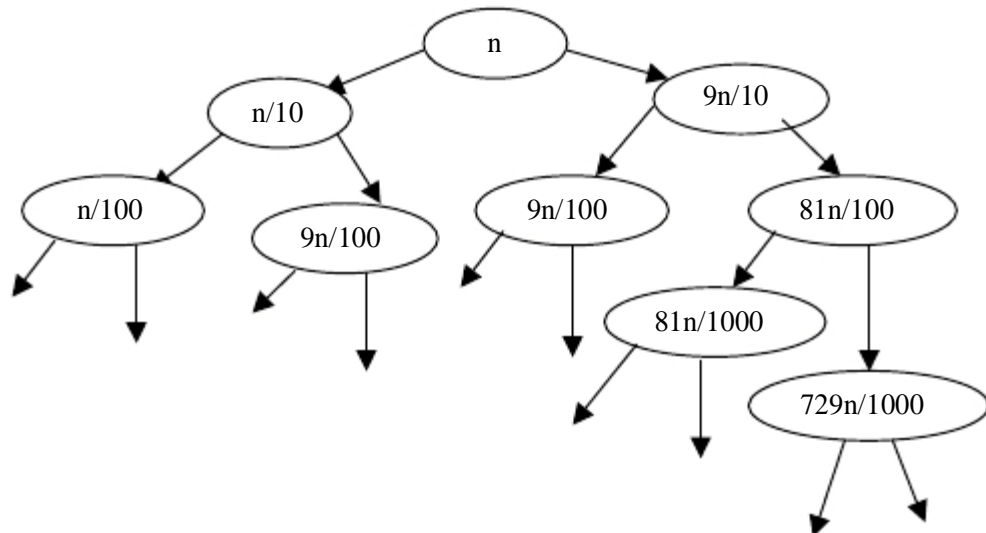
Case when the partition is balanced but not as of best-case partition

Lets suppose that partitioning algorithm always produce 9:1 split of the array. See here

9:1 seems unbalanced. For this situation we can write recurrence relation as

$$T(n) = T(9n/10) + T(n/10) + O(n).$$

Then we can show that this is $O(n \log n)$ from the below recurrence tree



Adding up all the levels we get

$$n + n + n + (\leq n) + (\leq n) + \dots + 1$$

Here when the height $\log_{10} n$ is reached number of nodes in a level is $\leq n$. The recursion terminates at the height $\log_{10} n$.

Average Case:

Assumptions:

All permutations of inputs are equally likely.

Partitions may not be in the same way at all levels.

Expect some partition be balance and some are not balanced.

When we adhere to the above assumptions then we get expected running time for quick sort as $O(n \log n)$.

Randomized Quick Sort

We assumed that all permutations of inputs are equally likely but this assumption is not always true (think of already sorted array). So we use random approach for selecting pivot to relax that assumption. The algorithm is called randomized if its behavior depends on input as well as random value generated by random number generator. The beauty of the randomized algorithm is that no particular input can produce worst-case behavior of an algorithm. The worst case only depends on random number generated.

Algorithm:

```
RandPartition(A,i,j){
k = random(i,j); //generates random number between i and j including both.
swap(A[l],A[k]);
return Partition(A,i,j);
}
```

Algorithm:

```

RandQuickSort(A,p,q)
{
    if(p < q)
    {
        r = RandPartition(A,p,q);
        RandQuickSort(A,p,r-1);
        RandQuickSort(A,r+1,q);
    }
}

```

Analysis:

The analysis here works for the both versions of quick sort we have discussed.

Worst Case Revisited:

We saw that the worst case occurs when the partition selects the pivot at first or the last every time. Now assume we do not know what the worst case partition is lets say q such that,

$$T(n) = \max_{1 \leq q \leq n-1} (T(q) + T(n-q)) + \cup(n).$$

Here q is from 1 to $n-1$ since partition algorithm produces two partitions where there is at least 1 element. Since we have seen the worst case bound lets use substitution method.

Guess $T(n) = O(n^2)$.

To show $T(n) \leq c n^2$.

Assume $T(q) \leq c q^2$ and $T(n-q) \leq c (n-q)^2$.

Now substituting this we get,

$$\begin{aligned} T(n) &\leq \max_{1 \leq q \leq n-1} (c q^2 + c (n-q)^2) + \cup(n). \\ &= c \max_{1 \leq q \leq n-1} (q^2 + (n-q)^2) + \cup(n). \end{aligned}$$

Taking first derivatives of $(q^2 + (n-q)^2)$ with respect to q we get,

$$2q - 2(n-q) = 2q - 2n + 2q = 4q - 2n$$

Second derivative with respect to q is 4 (positive).

We know if second derivative is positive and first derivative is zero then we have minima at q and that q is $n/2$. So there must be some other value of q less than (or greater than) $n/2$ such that the function has maxima at q . Take q at one end point (see at both end points value is same). So we can write,

$$\max_{1 \leq q \leq n-1} (q^2 + (n-q)^2) \leq (1^2 + (n-1)^2) + O(n), \quad q=1 \text{ (same for } n-1\text{)}$$

$$= n^2 - 2(n-1), \text{ so}$$

$$T(n) \leq cn^2 - 2c(n-1) + O(n).$$

$$\leq cn^2.$$

For constant c when $2c(n-1) \geq O(n)$.

Hence the worst case is $O(n^2)$.

We can eliminate the worst-case behavior of input instance by the following

Use the middle element of the subarray as pivot.

Use a *random* element of the array as the pivot.

Perhaps best of all, take the median of three elements (first, last, middle) as the pivot.

However, above care still gives the worst case running time as $O(n^2)$.

Average Case:

To analyze average case, assume that all the input elements are distinct for simplicity. If we are to take care of duplicate elements also the complexity bound is same but it needs more intricate analysis. Consider the probability of choosing pivot from n elements is equally likely i.e. $1/n$.

Now we give recurrence relation for the algorithm as

$$T(n) = \frac{1}{n} \sum_{q=1}^{n-1} (T(q) + T(n-q)) + O(n).$$

[In randomized version extra $O(1)$ is added but the final relation would be same as above]

Now for some $j = 1, 2, \dots, n-1$

$T(q) + T(n-q)$ is repeated two times so we can write the relation as,

$$T(n) = 2 \sum_{j=1}^{n-1} T(j) + O(n).$$

Or, $nT(n) = 2 \sum_{j=1}^{n-1} T(j) + n^2$. ---- (I) [taking $O(n) = n$ for simplicity]

For $n = n-1$ we have,

$$(n-1)T(n-1) = 2 \sum_{j=1}^{n-2} T(j) + (n-1)^2$$
. ---- (II)

(I) – (II) gives,

$$nT(n) - (n+1)T(n-1) = 2n - 1.$$

Or, $T(n)/(n+1) = T(n-1)/n + 2n - 1 / n(n+1)$.

Put $T(n)/(n+1)$ as A_n then we have above relation as

$$A_n = A_{n-1} + 2n - 1 / n(n+1).$$

The above relation is written as,

$$A_n = \sum_{i=1}^n \frac{2i - 1}{i(i + 1)} \quad [2i - 1 = \frac{1}{i} - \frac{1}{i+1}]$$

$$= \sum_{i=1}^n \left(\frac{1}{i} - \frac{1}{i+1} \right) \quad [Harmonic series H_n = \sum_{i=1}^n \frac{1}{i} \approx \ln n]$$

$$= H_{2 \log n}$$

But we have $T(n) = (n+1) A_n \approx H_{2(n+1)} \log n$.

So,

$$T(n) = O(n \log n).$$

Final Remark on Quick Sort

Practical implementations have shown that the quick sort is the faster sorting algorithm of all we have studied. Though worst time complexity for heap sort and merge sort are better than that of quick sort due to the large constant overhead quick sort runs faster. Another important factor is extra space also.

Sorting Comparison

Sort	Worst Case	Average Case	Best Case	Comments
Insertion Sort	$\cup(n^2)$	$\cup(n^2)$	$\cup(n)$	
Selection Sort	$\cup(n^2)$	$\cup(n^2)$	$\cup(n^2)$	(*Unstable)
Bubble Sort	$\cup(n^2)$	$\cup(n^2)$	$\cup(n^2)$	
Merge Sort	$\cup(n \log n)$	$\cup(n \log n)$	$\cup(n \log n)$	Requires Memory
Heap Sort	$\cup(n \log n)$	$\cup(n \log n)$	$\cup(n \log n)$	*Large constants
Quick Sort	$\cup(n^2)$	$\cup(n \log n)$	$\cup(n \log n)$	*Small constants

Median Order Statistics (Selection Problems)

i th order statistic of a set of elements gives i th largest(smallest) element. In general lets think of i th order statistic gives i th smallest. Then minimum is first order statistic and the maximum is last order statistic. Similarly a median is given by i th order statistic where $i = (n+1)/2$ for odd n and $i = n/2$ and $n/2 + 1$ for even n . This kind of problem commonly called selection problem. We can specify selection problem as:

Input: A set of n distinct elements and number i , with $1 \leq i \leq n$.

Output: The element from the set of elements, that is larger than exactly $i-1$ other elements of the given set.

This problem can be solved in $\cup(n \log n)$ in a very straightforward way. First sort the elements in $\cup(n \log n)$ time and then pick up the i th item from the array in constant time. What about the linear time algorithm for this problem? The next is answer to this.

General Selection

This problem is solved by using the “divide and conquer” method. The main idea for this problem solving is to partition the element set as in Quick Sort where partition is randomized one.

Algorithm:

```

RandSelect(A,p,r,i)
{
    if(p = r)
        return A[p];
    q = RandPartition(A,p,r);
    k = q - p + 1;
    if(i <= k)
        return RandSelect(A,p,q,i);
    else
        return RandSelect(A,q+1,r,i - k);
}

```

Analysis:

Since our algorithm is randomized algorithm no particular input is responsible for worst case however the worst case running time of this algorithm is $O(n^2)$. This happens if every time unfortunately the pivot chosen is always the largest one (if we are finding minimum element).

Assume that the probability of selecting pivot is equal to all the elements i.e $1/n$ then we have the recurrence relation

$$T(n) = \frac{1}{n} \left(T(\max(1, n-1)) + \sum_{j=1}^{n-1} T(\max(j, n-j)) \right) + O(n).$$

Where $T(\max(1, n-1))$ is either $T(1)$ or $T(n-1)$ is due to partition must at least partition a set with one element at a side, and

$\sum_{j=1}^{n-1} T(\max(j, n-j))$ is for a instance of recursive call for all except nth partition (see above is the nth partition) where,

$$\max(j, n-j) = j, \text{ if } j \geq n/2$$

$$\text{and } \max(j, n-j) = n-j, \text{ otherwise.}$$

Observe that every $T(j)$ or $T(n-j)$ will repeat twice for both odd and even value of n (one may not be repeated) one time from 1 to $n/2$ and second time for $n/2$ to $n-1$, so we can write,

$$T(n) \leq 1/n(T(n-1) + 2 \sum_{j=n/2}^{n-1} T(j)) + O(n).$$

In the worst case $T(n-1) = O(n^2)$. We have,

$$T(n) \leq 2/n \sum_{j=n/2}^{n-1} T(j) + O(n).$$

Using substitution method,

Guess $T(n) = O(n)$

To show $T(n) \leq cn$

Assume $T(j) \leq cj$

Substituting on the relation

$$\begin{aligned} T(n) &\leq 2/n \sum_{j=n/2}^{n-1} ck + O(n). \\ &\leq 2c/n \left(\sum_{j=n/2}^{n-1} k - \sum_{j=1}^{n/2-1} k \right) + O(n). \\ &= 2c/n \left(n(n-1)/2 - n/2 (n/2 - 1)/2 \right) + O(n). \\ &\leq c(n-1) - cn/4 + c/2 + O(n) \\ &= c(3n/4 + 1/2) + O(n). \\ &\leq cn. \end{aligned}$$

For c large enough such that $c(3n/4 + 1/2)$ dominates $O(n)$.

Hence any order statistic is $O(n)$.

Worst-case linear time Selection

Here we study the algorithm that guarantees the running time of the selection problem is $O(n)$ at the worst case. In this algorithm like in RandSelect mentioned previously the input set is recursively partitioned. If we can guarantee that there will be good split of the input by choosing some pivot then we may come up with the solution. For this purpose we use median of medians as pivot point. To find median of medians we divide n input elements into n/r groups of r elements each and $n - n/r$ elements are not used.

When median of medians is used as a pivot we have each medians as $r/2$ smallest element so at least $n/r/2$ of the medians are less than or equal to median of medians and at least $n/r - n/r/2 + 1 \geq n/r/2$ of the medians are greater than or equal to median of medians. That means at least $r/2$ $n/r/2$ elements are less (greater) than or equal to median of medians.

Algorithm:

1. Select (A, k, l, u)
2. {
3. $n = u - l + 1;$
4. if($n \leq r$) {
 - $\text{sort}(A, l, u);$ *return k th element*
5. }
6. $\text{divide}(A, l, u, r);$ //ignore excess elements
7. Get medians from all the subsets and put on array m ;
8. $v = \text{select}(m, n/r/2, 1, n/r);$
9. Partition A using v as pivot.
10. If($k = (j - l + 1)$) *return v ; //j is the position of v*
11. else if($k < (j - l + 1)$) *return $\text{select}(A, k, l, j-1)$;*
12. else *return $\text{select}(A, k - (j - l + 1), j+1, u)$;*
13. }

First consider $r = 5$ and the elements in the array are distinct. $r/2$ is at least $1.5n/5$ then we can say at least $1.5n/5$ elements are greater than or equal to v this implies there are at most $n - 1.5n/5 \leq .7n + a$ ($a > 0$) [Analysis shows a as 1.2, see text book] elements smaller than or equal to v . Here we are considering that the median finding is done in $O(1)$ since there are few number to be sorted. Hence the steps other than the recursive ones take at most $O(n)$ time. Running time for line 8 is $T(n/5)$ since $r = 5$. Again $.7n + 1.2$ is no more than $3n/4$ for $n \geq 24$. Now we can write recurrence relation as

$$T(n) = T(n/5) + T(3n/4) + O(n).$$

Guess $T(n) = O(n)$ then we have to prove $T(n) \leq cn - b$. Assuming guess is true for $T(n/5)$ and $T(3n/4)$ we get,

$$\begin{aligned} T(n) &\leq cn/5 - cb + 3cn/4 - cb + O(n). \\ &= 19c/20n - 2cb + O(n). \\ &\leq cn \end{aligned}$$

For large c and some b such that, $19c/20n - 2cb \geq O(n)$.

Hence $T(n) = O(n)$.

Matrix Multiplication

Given two A and B n -by- n matrices our aim is to find the product of A and B as C that is also n -by- n matrix. We can find this by using the relation

$$C(i,j) = \sum_{k=1}^n A(i,k)B(k,j)$$

Using the above formula we need $O(n)$ time to get $C(i,j)$. There are n^2 elements in C hence the time required for matrix multiplication is $O(n^3)$. We can improve the above complexity by using divide and conquer strategy.

Strassen's Matrix Multiplication Algorithm

Strassen was the first person to give better algorithm for matrix multiplication. The idea behind his algorithm is divide the problem into four sub problems of dimension $n/2$ by $n/2$ and solve it by recursion. The basic calculation is done for 2 by 2 matrix. The 2 by 2 matrix is solved as

$$\begin{matrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{matrix} = \begin{matrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{matrix} \times \begin{matrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{matrix}$$

Where,

$$P = (A_{11} + A_{22})(B_{11} + B_{22}).$$

$$Q = (A_{21} + A_{22})B_{11}.$$

$$R = A_{11}(B_{12} - B_{22}).$$

$$S = A_{22}(B_{21} - B_{11}).$$

$$T = (A_{11} + A_{12})B_{22}.$$

$$U = (A_{21} - A_{11})(B_{11} + B_{12}).$$

$$V = (A_{12} - A_{22})(B_{21} + B_{22}).$$

And

$$C_{11} = P + S - T + V.$$

$$C_{12} = R + T.$$

$$C_{21} = Q + S.$$

$$C_{22} = P + R - Q + U.$$

See above there are total 7 multiplications and 18 additions.

We can have recurrence relation for this algorithm as

$$T(n) = 7T(n/2) + O(n^2).$$

Solving above relation by master method we get case 1 so, $T(n) = O(n^{2.81})$.

The fastest algorithm known for this problem is by Coppersmith and Winograd that runs in $O(n^{2.376})$ but not used due to large constant overhead.

Exercises

1. Write an algorithm for integer multiplication (divide and conquer) discussed in this material but not done.
2. Trace out the steps of Quick Sort for the following input array
 $A[] = \{13, 19, 9, 5, 12, 8, 7, 4, 11, 2, 6, 21\}$
3. Write an algorithm and analyze it that multiplies two n by n matrices and has complexity $O(n^3)$ (brute force way).
4. 10.3.5 (pg 192)
Given “black-box” worst-case linear time median subroutine, give a simple, linear time algorithm that solves the selection problem for an arbitrary order statistic.

[Sorting Revisited]

Design and Analysis of Algorithms (CSc 523)

Samujjwal Bhandari

Central Department of Computer Science and Information Technology (CDCSIT)

Tribhuvan University, Kirtipur,
Kathmandu, Nepal.

We saw previously that every comparison sort has lower bound of $\Theta(n \log n)$. If we want to improve running time then we can not use just only comparison to sort. Here we present few sorting algorithms that sort in linear time.

Counting Sort

The basic idea behind the counting sort is like this:

Assume that the elements to be sorted are in between 1 to k, where k is some integer.

Then try to find how many numbers are smaller than some value x this information gives the actual position of the element in an array.

There are three arrays array A is of inputs, B is an array for sorted output and C is an temporary array to hold the information about the element. Size of an array C is k.

Algorithm:

```

CountingSort(A,B,k)
{
  //all the initial elements in C are 0
  for(i = 1; i <= n; i++)
    C[A[i]] += 1; //number of elements count
  for(j = 2; j <= k; j++)
    C[j] += C[j-1]; //cumulative totaling
  for(l = n; l >= 1; l--)
  {
    B[C[A[l]]] = A[l];
    C[A[l]] -= 1;
  }
}

```

Analysis:

In the above algorithm first loop executes for $O(n)$ time, second loop executes for $O(k)$ time and the last loop executes for $O(n)$ time so as a whole total running time of an

algorithm is $O(n)$. Note that the counting sort is stable.

What happened to our $\Theta(n \log n)$ lower bound!

Radix sort

In this method we write the to be sorted as d digit numbers. Then we use some stable sorting algorithm to sort them by last digit, then sorting by second last digit and so on. Here if we use counting sort then the running time becomes $O(nd) = O(n)$.

Bucket Sort

If we want to sort n elements that are evenly distributed over the interval $[0, m]$. we split the interval $[0, m]$ into n equal buckets. $[0, d], [d, 2d], \dots, [(n-1)d, nd]$. ($d = m/n$)

Here we sort data by distributing it to appropriate buckets, then sort each bucket, then concatenate the result.

If we use an array of buckets, each item gets mapped to the right bucket in $O(1)$ time. With uniformly distributed keys, the expected number of items per bucket is 1. Thus sorting each bucket takes $O(1)$ time! The total effort of bucketing, sorting buckets, and concatenating the sorted buckets together is $O(n)$.

In this kind of sorting the great problem arises if wrong distribution is assumed where key distribution takes linear time but still a bucket may have full of the inputs to be sorted.

Exercises

1. Write an algorithm for bucket sorting and analyze it.

[Greedy Paradigm]

Design and Analysis of Algorithms (CSc 523)

Samujjwal Bhandari

Central Department of Computer Science and Information Technology (CDCSIT)

Tribhuvan University, Kirtipur,

Kathmandu, Nepal.

Greedy method is the simple straightforward way of algorithm design. The general class of problems solved by greedy approach is optimization problems. In this approach the input elements are exposed to some constraints to get feasible solution and the *feasible solution* that meets some objective function best among all the solutions is called *optimal solution*.

Greedy algorithms always makes optimal choice that is local to generate globally optimal solution however, it is not guaranteed that all greedy algorithms yield optimal solution.

We generally cannot tell whether the given optimization problem is solved by using greedy method or not, but most of the problems that can be solved using greedy approach have two parts:

Greedy choice property

Globally optimal solution can be obtained by making locally optimal choice and the choice at present cannot reflect possible choices at future.

Optimal substructure

Optimal substructure is exhibited by a problem if an optimal solution to the problem contains optimal solutions to the subproblems within it.

To prove that a greedy algorithm is optimal we must show the above two parts are exhibited. For this purpose first take globally optimal solution; then show that the greedy choice at the first step generates the same but the smaller problem, here greedy choice must be made at first and it should be the part of an optimal solution; at last we should be able to use induction to prove that the greedy choice at each step is best at each step, this is optimal substructure.

Fractional Knapsack Problem

Statement: A thief has a bag or knapsack that can contain maximum weight W of his loot. There are n items and the weight of i th item is w_i and it worth v_i . Any amount of item can be put into the bag i.e. x_i fraction of item can be collected, where $0 \leq x_i \leq 1$. Here the objective is to collect the items that maximize the total profit earned.

We can formally state this problem as, maximize $\sum_{i=1}^n x_i v_i$ using constraint $\sum_{i=1}^n x_i w_i \leq W$.

Here in this problem it is clear that any optimal solution must fill the knapsack completely otherwise there will be partial space left that can be filled by some part of

some items. i.e. $\sum_{i=1}^n x_i w_i = W$.

Algorithm:

Take as much of the item with the highest value per weight (v_i/w_i) as you can. If the item is finished then move on to next item that has highest (v_i/w_i), continue this until the knapsack is full.

$v[1 \dots n]$ and $w[1 \dots n]$ contain the values and weights respectively of the n objects sorted in non increasing order of $v[i]/w[i]$. W is the capacity of the knapsack, $x[1 \dots n]$ is the solution vector that includes fractional amount of items and n is the number of items.

GreedyFracKnapsack(W,n)

```
{
    for(i=1; i <= n; i++)
        x[i] = 0.0;
    tempW = W;
    for(i=1; i <= n; i++)
    {
        if(w[i] > tempW) then break;
        x[i] = 1.0;
        tempW -= w[i];
    }
    if(i <= n) x[i] = tempW/w[i];
}
```

Analysis:

We can see that the above algorithm just contain a single loop i.e. no nested loops the running time for above algorithm is $O(n)$. However our requirement is that $v[1 \dots n]$ and $w[1 \dots n]$ are sorted, so we can use sorting method to sort it in $O(n \log n)$ time such that the complexity of the algorithm above including sorting becomes $O(n \log n)$.

To prove the algorithm's correctness we prove that the strategy chosen for greedy approach gives optimal solution.

Lemma 1: when the sum of all weights is $\leq W$, then $x_i = 1, 1 \leq i \leq n$ is an optimal solution.

Lemma 2: All optimal solutions will fill the knapsack exactly

Proof of correctness:

Let v_h/w_h be the maximum value to weight ratio. Then we can say that $v_h/w_h \geq v/w$ for any pair of (v,w) . Now if the solution does not contain full w_h then by replacing some amount of w from other highest ratio value will improve the solution. This is greedy choice property. When the above process is continued then knapsack is filled completely giving the optimal solution. Say A be the optimal solution to the problem S then we can always find that $A-a$ is an optimal solution for $S-s$, where a is an item that is picked as the greedy choice and $S-s$ is the subproblem after the first greedy choice is made. This is optimal substructure.

Activity Selection (Scheduling) Problem

There is a set $S = \{1,2,3, \dots, n\}$ of n activities that are competing for resource which can be used by only one job at a time.

Each activity i is associated with start time s_i and finishing time f_i .

Selected activity i takes place during interval $[s_i, f_i)$.

Two activities i and j are compatible if they do not overlap i.e. $s_i \geq f_j$ or $s_j \geq f_i$.

Problem: To select the maximum size set of mutually compatible activities.

Algorithm:

Sort the input activities in order of increasing finishing time i.e. $f_1 \leq f_2 \leq \dots \leq f_n$.

This step can be done in $O(n \log n)$ time.

```

GreedyAS(s,f)
{
  n = length(s);
  A = {1};
  j = 1;
  for(i = 2 ; i <= n; i++)
    if(s_i >= f_j)
    {
      A += {i};
      j = i;
    }
  return A;
}

```

Example:

Let $S = \{1,2,3,4,5,6,7,8,9,10,11\}$ and the respective start and finish time are

$(1, 4), (3, 5), (2, 7), (5, 7), (3, 8), (5, 9), (6, 10), (8, 11), (8, 12), (2, 13)$ and $(12, 14)$.

Considering the above algorithm

$A = \{1\}$ Initialization

$A = \{1, 4\}$ 1st execution of if inside for loop

$A = \{1, 4, 8\}$ 2nd 1st execution of if inside for loop

$A = \{1, 4, 8, 11\}$ 3rd 1st execution of if inside for loop

Out of the for loop and Return $A = \{1, 4, 8, 11\}$

Analysis:

We can easily see that above algorithm has time complexity $O(n)$ if the input is assumed to be sorted. If we are applying the algorithm without sorting then there is a need of sorting step also thus the complexity becomes $O(n \log n)$.

Correctness:

To prove the correctness of the *GreedyAS* algorithm if we prove for its optimality then it is done so let's state the following theorem and prove it.

Theorem 1: Algorithm *GreedyAS* produces solution of maximum size for the activity selection problem.

Proof:

Idea behind the proof of the theorem is show that the problem satisfies

- i. Greedy choice property.
- ii. Optimal substructure.

Now let's apply the above idea

- i. Let $S = \{1, 2, \dots, n\}$ be the set of activities such that the order of activities are sorted in terms of the increasing finishing time so that it is guaranteed that the first activity that will be selected is 1. suppose A is a subset of S such that A is an optimal solution. Here activities in A are sorted by finishing time. Let first activity in A be a .

If $a = 1$ then the algorithm starts with greedy choice and we are done.

If a is not 1 then we can show that there is another solution B such that it starts with activity 1 (greedy choice).

Let $B = A - \{a\} + \{1\}$ since $f_1 \leq f_k$ activity 1 is still compatible with A . then we have $|B| = |A|$, so B is optimal that starts with greedy choice 1.

- ii. Once we made the greedy choice we must be able to show $S' = \{i \in S : s_i \geq f_1\}$ with optimal solution $A' = A - \{1\}$. The above solution must be optimal since if there is B' that solves S' with more number of activities in B' then adding activity 1 in B' must be the solution for S contradicting the optimality of A because since B is also an optimal solution with more number of activities than A .

Hence the Proof:

Huffman Codes

Huffman codes are used to compress data by representing each alphabet by unique binary codes in an optimal way. As an example consider the file of 100,000 characters with the following frequency distribution assuming that there are only 7 characters

$f(a) = 40,000$, $f(b) = 20,000$, $f(c) = 15,000$, $f(d) = 12,000$, $f(e) = 8,000$, $f(f) = 3,000$, $f(g) = 2,000$.

Here fixed length code for 7 characters we need 3 bits to represent all characters like $a = 000$, $b = 001$, $c = 010$, $d = 011$, $e = 100$, $f = 101$, $g = 110$.

Total number of bits required due to fixed length code is 300,000.

Now consider variable length character so that character with highest frequency is given smaller codes like

$a = 0$, $b = 10$, $c = 110$, $d = 1110$, $e = 11111$, $f = 111101$, $g = 111100$

Total number of bits required due to variable length code is

$40,000 * 1 + 20,000 * 2 + 15,000 * 3 + 12,000 * 4 + 8,000 * 5 + 3,000 * 6 + 2,000 * 6$.

i.e. 243,000 bits

Here we saved approximately 19% of the space.

Prefix Codes

Prefix codes are those that are not prefix of other codewords. We desire prefix codes because it is simple to encode and decode using prefix codes. For example using above derived variable length codes (they are prefix codes) we can encode word

fag as $111101.0.111100 = 1111010111100$

Above . is used for concatenation.

Since no code is prefix of other codes the codefile is unambiguous. So to decode the encoded file we just parse the codefile so as to break like $111101.0.111100$ that is *fag*.

The whole decoding process is represented by binary tree where all leaves are characters and path from the root to the character is code. We assume 0 as go to left child and 1 as go to right child.

Remember: Optimal code for file is always represented by full binary tree.

Theorem 2: A binary tree that is not full cannot represent optimal prefix codes.

Proof:

Consider a tree T of binary prefix codes that is not full then there exists an internal node, say x that has only one child, say y . Consider another Tree T' that represent same binary prefix codes with same number of leaves as of T and has same depth as T except for leaves from the subtree rooted at y . These leaves will have depth T' implies that T cannot corresponds to optimal prefix codes. The question here is how to get such a T' . To get T' , merge x and y into a single node, say z . z is a child of parent of x (if a parent exists) and z is a parent to any children of y . Then T' has the desired properties: it corresponds to a code on the same alphabet as the code which are obtained, in the subtree rooted at y in T have depth in T' strictly less (by one) than their depth in T .

Hence, the proof.

Some points to be noted

If C is a set of unique characters in a file then optimal prefix codes tree T will have exactly $|C|$ numbers of leaves and $|C|-1$ numbers of internal nodes.

$f(c)$ denotes the frequency of character c in a file. $dr(c)$ is depth of c 's leaf node in T .

Number of bits required to encode a file is

$$B(T) = \sum_{c \in C} f(c)dr(c), \text{ where } B(T) \text{ is cost of the tree } T.$$

Algorithm:

A greedy algorithm can construct Huffman code that is optimal prefix codes. A tree corresponding to optimal codes is constructed in a bottom up manner starting from the $|C|$ leaves and $|C|-1$ merging operations.

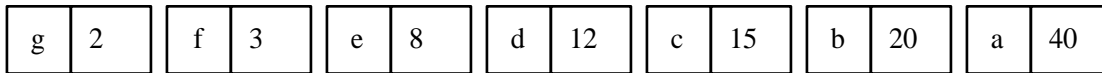
Use priority queue Q to keep nodes ordered by frequency. Here the priority queue we consider is binary heap.

```
HuffmanAlgo(C)
{
  n = |C|;
  Q = C;
  For(i=1; i<=n-1; i++)
  {
    z = Allocate-Node();
    x = Extract-Min(Q);
    y = Extract-Min(Q);
    left(z) = x;
    right(z) = y;
    f(z) = f(x) + f(y);
    Insert(Q,z);
  }
  return Extract-Min(Q)
}
```

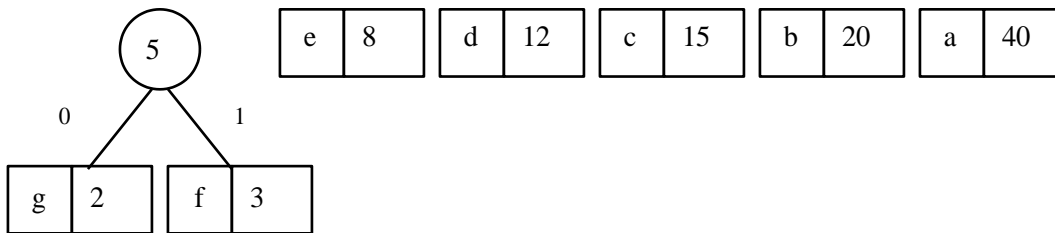
Example:

$C = \{a, b, c, d, e, f, g\}$; $f(c) = 40, 20, 15, 12, 8, 3, 2$; $n = 7$

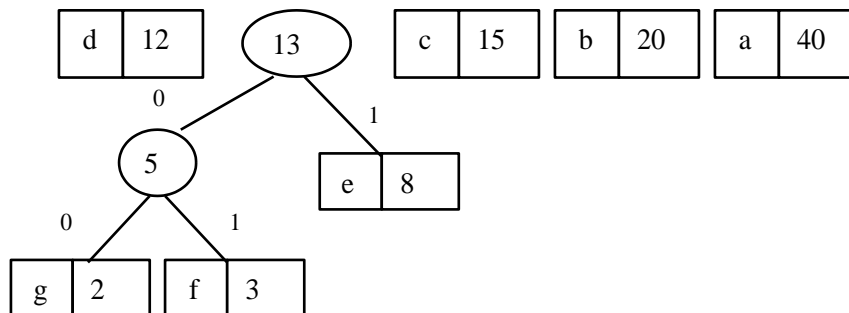
Initial priority queue is



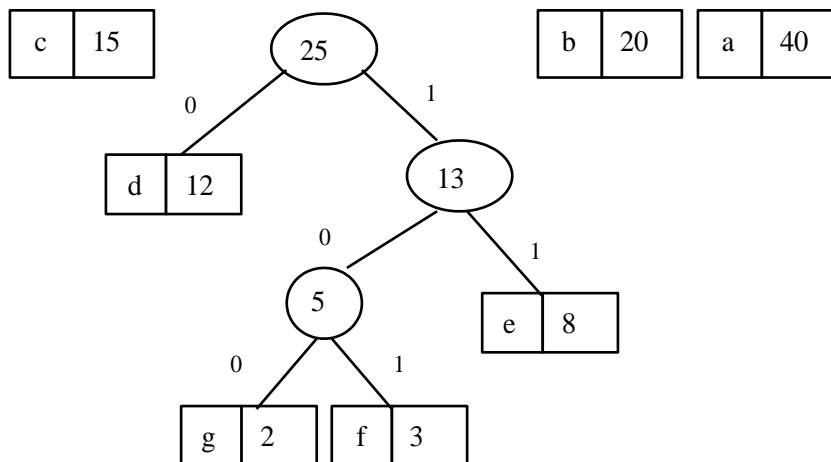
i=1

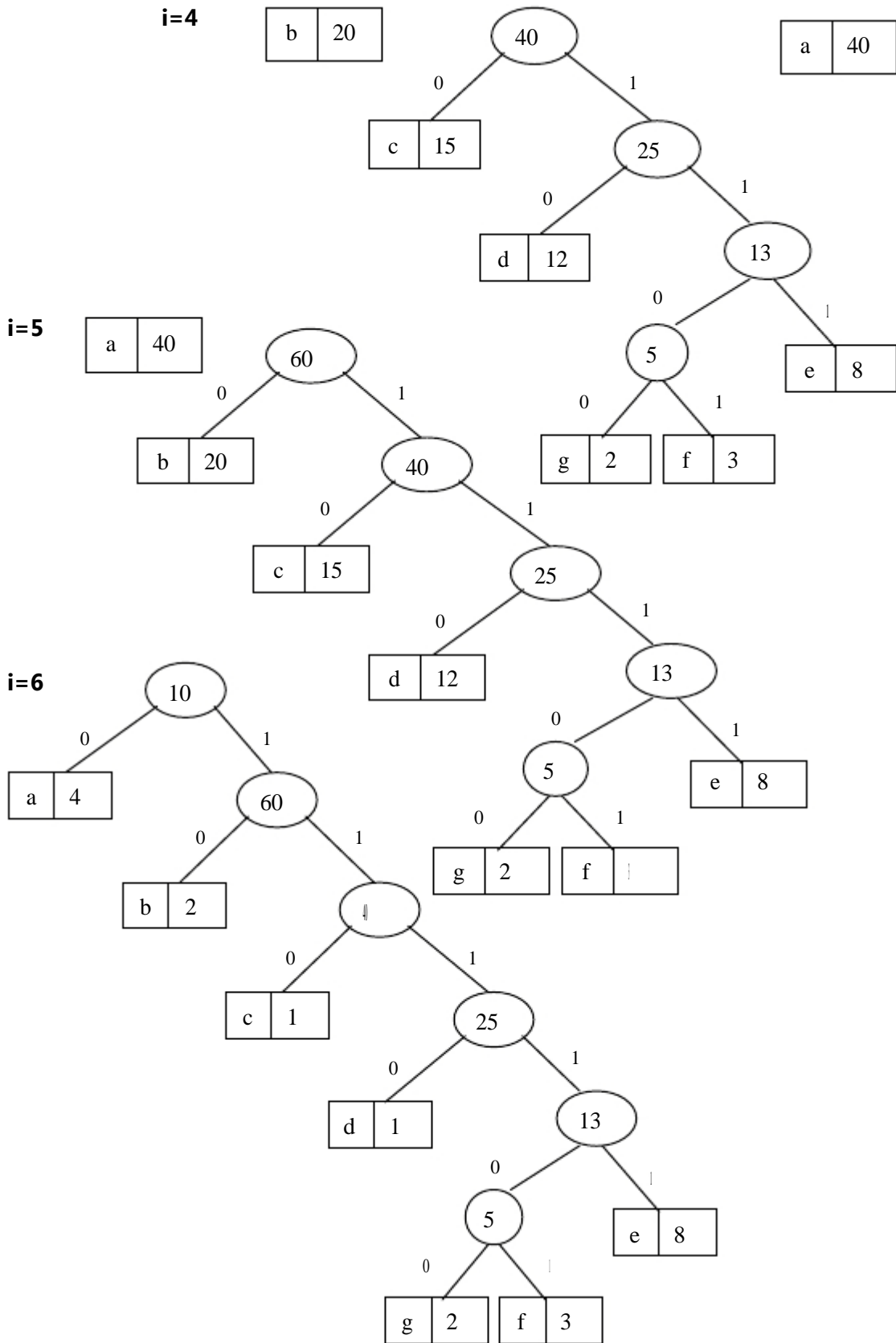


i=2



i=3





Analysis

We can use $BuildHeap(C)$ {see notes on sorting} to create a priority queue that takes $O(n)$ time. Inside the for loop the expensive operations can be done in $O(\log n)$ time. Since operations inside for loop executes for $n-1$ time total running time of *HuffmanAlgo* is $O(n \log n)$.

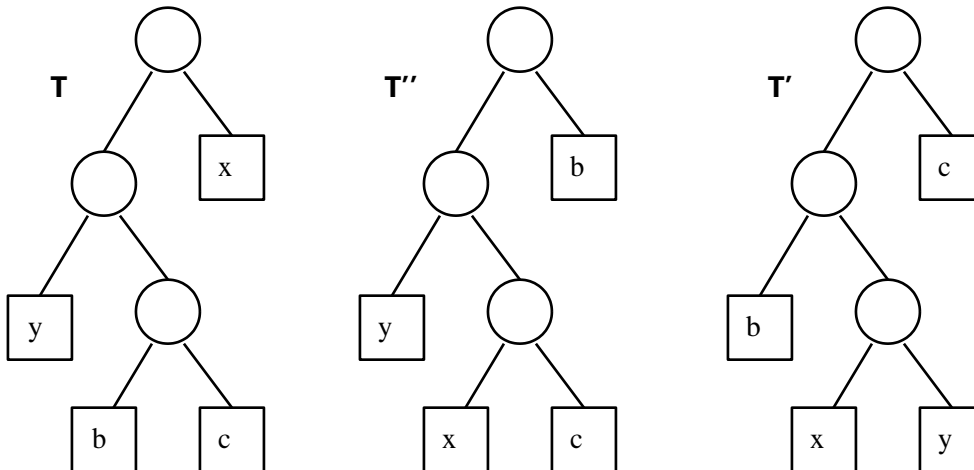
Now we need to prove for the correctness of the algorithm for this as usual lets try to prove that the problem shows greedy choice property and optimal substructure.

Lemma 3: Greedy Choice Property: Let C be an alphabet in which each character $c \in C$ has frequency $f(c)$. Let x and y be two characters in C having the lowest frequencies.

Then there exists an optimal prefix code for C in which the codewords for x and y have the same length and differ only in the last bit.

Proof:

Let T be arbitrary optimal prefix code generating tree. Take T' such that it also generates an optimal prefix code for the same set of alphabets as of T and has x and y as sibling nodes of maximum depth. If we can transform T into T' with changing the prefix code representation we are done since sibling nodes differ by last bit only with same length.



Let b and c are sibling leaves of maximum depth in T . we can assume that $f(b) \leq f(c)$ and $f(x) \leq f(y)$ since x and y are of lowest frequency and b and c are of arbitrary frequency. Now swap the position of x and b to get figure T'' then we have

$$\begin{aligned}
 B(T) - B(T'') &= \sum_{c \in C} f(c)d_T(c) - \sum_{c \in C} f(c)d_{T''}(c) \\
 &= f(x)d_T(x) + f(b)d_T(b) - f(x)d_{T''}(x) - f(b)d_{T''}(b) \\
 &= f(x)d_T(x) + f(b)d_T(b) - f(x)d_T(b) - f(b)d_T(x) \\
 &= (f(b) - f(x))(d_T(b) - d_T(x)) \\
 &\geq 0
 \end{aligned}$$

Here $f(b) - f(x)$ is positive since x is of lowest frequency and $d_T(b) - d_T(x)$ is also positive as b is leaf of maximum depth of T .

Similarly, we can swap y and c without increasing the cost i.e $B(T'') - B(T') \geq 0$ (as above). So we can say conclude that $B(T') \leq B(T)$. But T represents optimal solution we have $B(T) \leq B(T')$, so $B(T) = B(T')$.

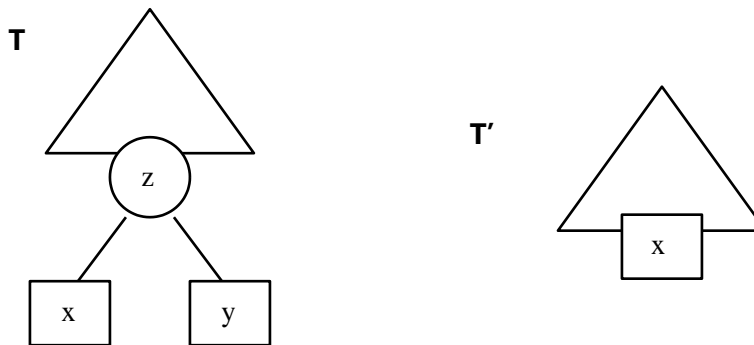
Hence T' is an optimal tree in which x and y are sibling leaves of maximum depth.

This lemma suggest that we can start merging by choosing two lowest frequency characters to get an optimal solution so this is greedy choice property.

Hence, the proof.

Lemma 4:Optimal substructure: Let T be full binary tree representing an optimal prefix code over an alphabet C , where frequency $f(c)$ is defined for each character $c \in C$. Consider any two characters x and y that appears as sibling leaves in T , and let z be their parent. Then, considering z as a character with frequency $f(z) = f(x) + f(y)$, the tree $T' = T - \{x, y\}$ represents an optimal prefix code for the alphabet $C' = C - \{x, y\} + \{z\}$.

Proof:



Since T represents optimal prefix code for C x and y are sibling leaves of lowest frequencies. We can represent cost $B(T)$ of a tree T in terms of cost $B(T')$. Here we have for each $c \in C - \{x, y\}$, we have $d_T(c) = d_{T'}(c)$

So, $B(T) = \sum_{c \in C} f(c)d_T(c)$ and $B(T') = \sum_{c \in C \cup \{x, y\}} f(c)d_{T'}(c)$ differ only for x and y

Then, $f(x)d_T(x) + f(y)d_T(y) = (f(x) + f(y))(d_{T'}(z) + 1)$ [since $d_T(x) = d_T(y) = d_T(z) + 1$]
 $= f(z)d_{T'}(z) + f(x) + f(y)$ [since $f(x) + f(y) = f(z)$]

So we can write $B(T) = B(T') + f(x) + f(y)$

If T' does not represent optimal prefix code for alphabets in C' , then we can always find some other tree T'' that is optimal for alphabets in C' i.e. $B(T'') < B(T')$ since z a character in C' it is a leaf node in T'' . if we add x and y on the node z then we have $B(T'') + f(x) + f(y) < B(T)$. This is contradiction since T is optimal. Thus T' must be optimal for C' .

Hence, the proof.

Theorem 3: Procedure HuffmanAlgo produces an optimal prefix code.

Proof: Using Lemma 3 and Lemma 4

Exercises

1. 17.3.2(pg 344)

What is an optimal Huffman code for the following set of frequencies, based on the first 8 Fibonacci numbers?

a:1 b:1 c:2 d:3 e:5 f:8 g:13 h:21

Can you generalize your answer to find the optimal code when the frequencies are the first n Fibonacci numbers?

[Dynamic Programming]

Design and Analysis of Algorithms (CSc 523)

Samujjwal Bhandari

Central Department of Computer Science and Information Technology (CDCSIT)

Tribhuvan University, Kirtipur,
Kathmandu, Nepal.

Dynamic Programming is a powerful way of designing the algorithm. Dynamic programming is widely used to solve the optimization problems. Here a set of choices is made to come up with the optimal solution later. Most of the problems dealt with dynamic programming technique consist of the subproblems recurring more than one time while calculating the solution to the problem. Here if we store the result of the solution of the subproblem then we do not have to recalculate the solution of the subproblem later and can be used when necessary. Unlike divide and conquer paradigm dynamic programming works in bottom up approach. To attack the problem using dynamic programming approach we must find the structure of an optimal solution, find the value of an optimal solution probably recursively, and the value of an optimal solution is calculated in bottom up fashion. If needed optimal solution should be obtained along with its value.

The optimization problems that are solved using dynamic programming have two properties namely optimal substructure property (this also applies for greedy method) and overlapping subproblems property.

Optimal Substructure

Optimal substructure is exhibited by a problem if an optimal solution to the problem contains optimal solutions to the subproblems within it.

Overlapping Subproblems

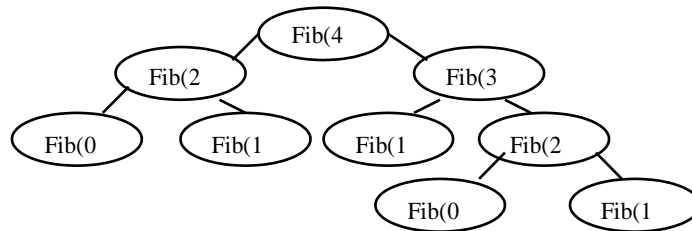
The given problem has overlapping subproblems if an algorithm calculates the solution to the subproblem more than once in order to use its result for problem of size greater than the solved subproblem.

There is another concept called **memoization**. This is the variation of dynamic programming. Using this approach we calculate the value of each subproblem and then store it in some table to access it later. The fashion of computation here is top down due to the control that helps store the solution to the subproblem is like recursive algorithm. Fibonacci numbers are calculated using this concept here in this chapter.

Fibonacci numbers

Definition: Fibonacci numbers are those numbers that are obtained by summing up two previous Fibonacci numbers. So here we have recursive definition as $f_n = f_{n-2} + f_{n-1}$.

Recursive fibonacci revisited: (see chapter Introduction pg. 7) In recursive version of an algorithm for finding Fibonacci number we can notice that for each calculation of the fibonacci number of the larger number we have to calculate the fibonacci number of the two previous number regardless of the computation of the Fibonacci number that has already be done. So there are many redundancies in calculating the Fibonacci number for a particular number. Lets try to calculate the fibonacci number of 4. The representation shown below shows the repetition in the calculation.



In the above tree we saw that calculations of fib(0) is done two times, fib(1) is done 3 times, fib(2) is done 2 times, and so on. So if we some how eliminate those repetitions we will save the running time. Now we try to work upon eliminating those repetitions.

Algorithm:

Input: A number n. **Output:** nth Fibonacci number.

Algorithm:

```

DynaFibo(n) {
    A[0] = 0, A[1] = 1;
    for(i = 2; i <= n; i++)
        A[i] = A[i-2] + A[i-1];
    return A[n];
}
  
```

Analyzing the above algorithm (see chapter Introduction pg. 7 this version has slight changes) we found that there are no repetition of calculation of the subproblems already solved and the running time decreased from $O(2^n)$ to $O(n)$. This reduction was possible due to the remembrance of the subproblem that is already solved to solve the problem of higher size. What is space complexity? Can we improve it?

Principle of Optimality: In optimal sequence of choices, each subsequence must also be optimal.

Matrix Chain Multiplication

Let us consider the problem of multiplying matrices. If we take three matrices say A of 40 by 30, B of 30 by 40 and C of 40 by 20, then we can calculate $A \cdot B \cdot C$ as 40 by 20 matrix as AB taking $40 \cdot 30 \cdot 40 = 48000$ multiplications, $(AB)C$ taking $40 \cdot 40 \cdot 20 = 32000$ multiplications totaling 80000 multiplications. The quest here is, since multiplication is expensive job, can we decrease the number of multiplication? The answer is it is possible if it can be. We know that the matrix multiplication is associative we can change the parenthesis to get the matrix multiplied; however we cannot change the permutation of the matrices, since matrix multiplication is not commutative. Lets take above example and see if we can reduce the number of multiplications. First compute BC this takes $30 \cdot 40 \cdot 20 = 24000$ multiplications and $A(BC)$ takes $40 \cdot 30 \cdot 20 = 24000$, totaling 48000 multiplications. Seeing the above fact we can say that it is possible to reduce the number of expensive multiplication by just changing the parenthesis.

Statement: Given a chain $\{A_1, A_2, \dots, A_n\}$ of n matrices where for $i = 1, 2, \dots, n$, matrix $p_{i-1} \cdot p_i$, fully parenthesize the product $A_1 A_2 \dots A_n$ in a way that minimizes the number of scalar multiplications.

The stated problem can be solved by exhaustively checking all the possible parenthesizations. By using this method the total time depends upon the number of ways of possible parenthesizations. Let $P(n)$ be the total number of parenthesizations for the sequence of n matrices. Then we can define number of parenthesizations by the recurrence relation as

$$P(n) = \begin{cases} 1 & \text{if } n = 1, \\ \sum_{j=1}^{n-1} P(j) P(n-j) & \text{if } n \geq 2. \end{cases}$$

The above recurrence shows that if we have single matrix there is only one possible way of parenthesizations. Otherwise the fully parenthesized matrix product is product of two fully parenthesized subproducts and for subproducts we can split the matrix sequence in anywhere for $j = 1$ to n . The above recurrence yields the similar sequence of Catalan numbers that grows as $\Theta(4^n/n^{3/2})$. This shows that the number of solution is exponential in n thus it is a poor strategy.

Algorithm and Idea:

Before presenting the algorithm let us see how matrix chain multiplication can be performed using dynamic programming approach. The basic steps for designing dynamic programming algorithm and their workouts for matrix chain multiplication are given below:

Characterizing structure to obtain optimal substructure: Given a chain $A_{i..j}$, if $i < j$ then any parenthesizations of the chain split product at some k , $i \leq k < j$. So for some k total cost is sum of costs of computing $A_{i..k}$, $A_{k+1..j}$, and multiplying $A_{i..k}$ and $A_{k+1..j}$. Suppose the optimal parenthesizations splits between A_k , and A_{k+1} , then parenthesizations of A_k , and A_{k+1} must be optimal, otherwise we can find another parenthesizations that is optimal.

Recursive definition of optimal solution: let $m[j,j]$ denotes minimum number of scalar multiplications needed to compute $A_{i..j}$. To solve the whole problem we have $m[1,n]$.

Now we can define the solution recursively as:

$$m[i,j] = \begin{cases} 0 & \text{if } i = j \text{ (since no multiplications needed),} \\ \min_{i \leq k < j} (m[i,k] + m[k+1,j] + p_{i-1} \cdot p_k \cdot p_j) & \text{if } i < j \text{ (assuming optimal split at } k) \end{cases}$$

Also keep track of $s[i,j] = k$ for optimal split.

Computation of optimal cost in bottom up manner: Recursive computation of above recurrence would take exponential time (see CLRS book for detail) but good thing is that the subproblems are computed over and over again, so it is showing us overlapping subproblems hence we can use the concept of dynamic programming to compute the cost in bottom up manner. From the above recurrence we saw that the cost $m[i,j]$ of $j-i+1$ matrices depends only on smaller subproblems where $A_{i..k}$ is product of $k-i+1 < j-i+1$ matrices and $A_{k+1..j}$ is product of $j-k < j-i+1$ matrices, for $k = i, i+1, \dots, j-1$. So if our algorithm fills table m using increasing length of chains we can use the result of subproblems.

Algorithm: The input to the algorithm is array p of the orders $\{p_0, p_1, \dots, p_n\}$

MatrixChainOrder(p)

$\{ n = \text{length}[p]-1;$

$\text{for}(i=1; i \leq n; i++)$

$m[i,i] = 0;$

$\text{for}(l = 2; l \leq n; l++)$

$\text{for}(i=1; i \leq n-l+1; i++)$

$\{ \quad j = i + l - 1;$

$m[i,j] = \infty;$

$\text{for}(k=i; k \leq j-1; k++)$

$\{ \quad q = m[i,k] + m[k+1,j] + p[i-1] \cdot p[k] \cdot p[j];$

$\text{if}(q < m[i,j]) \{ m[i,j] = q; s[i,j] = k; \}$

$\}$

$\}$

$\}$

Example:

Take $A_1(20 \cdot 15)$, $A_2(15 \cdot 30)$, $A_3(30 \cdot 5)$, $A_4(5 \cdot 25)$, $A_5(25 \cdot 10)$, $A_6(10 \cdot 5)$ matrices.

Then we have $p[] = \{20, 15, 30, 5, 25, 10, 5\}$ here considering index starts from zero. The

below table shows the steps followed in algorithm.

	j=1	2		6	S[i,j]		1	1	3	3	1	
0	9000	3750	6250	6000	5625								
	0	2250	4125	4250	4125					2	3	3	3
		0	3750	2750	2250						3	3	3
			0	1250	1500							4	5
				0	1250								5
					0								

Analysis:

The above algorithm can be easily analyzed for running time as $O(n^3)$, due to three nested loops.

The space complexity is $O(n^2)$ (how?).

Construction of optimal solution:

The optimal solution can be obtained from the table given by $s[i,j]$. Since each $s[i,j]$ holds value k for the optimal split the result for the sequence $A_{i..j}$ is $(A_{i..k})(A_{k+1..j})$. So the recovering sequence can be viewed as

$$S[1,n] = (A_{1..s[1,n]})(A_{s[1,n]+1..n})$$

$S[1,s[1,n]] = (A_{1..s[1,s[1,n]]})(A_{s[1,s[1,n]]+1..n})$ and so on. This can be written as recursive algorithm like

```

ParenthesizeChain(s,i,j)
{ if(i==j) print "Ai";
  else{
    print "(";
    ParenthesizeChain(s,i,s[i,j]);
    ParenthesizeChain(s,s[i,j]+1,j);
    print ")";
  }
}

```

Example: For our above example we have $s[1,6] = 1$ so $(A_1)(A_2 A_3 A_4 A_5 A_6)$; $s[2,6] = 3$ so $(A_1)((A_2 A_3)(A_4 A_5 A_6))$; $s[4,6] = 5$ so $(A_1)((A_2 A_3)((A_4 A_5)(A_6)))$ is the final parenthesizations. Here the intermediate steps are not shown.

What is the complexity? $O(n)!$

Longest Common Subsequence (LCS)

Given a sequence $X = \langle x_1, x_2, \dots, x_l \rangle$ and a sequence $Z = \langle z_1, z_2, \dots, z_m \rangle$, we say that Z is a subsequence of X if there exists a strictly increasing sequence $\langle i_1, i_2, \dots, i_k \rangle$ of indices of X such that for all $j = 1, 2, \dots, k$ $x_{i_j} = z_j$.

Example:

$X = \langle a, a, b, b, a, a, a, b, a, a, b, b, a, b \rangle$

$Y = \langle a, b, b, a, b, a, b, b \rangle$ (subsequence of X)

$Z = \langle a, b, a, b, a, b, a, b, a \rangle$ (not a subsequence of X)

For Given sequences X , Y and Z we say Z is a **common subsequence** of X and Y if Z is subsequence of both X and Y .

LCS problem statement: Given two sequences $X = \langle x_1, x_2, \dots, x_l \rangle$ and $Y = \langle y_1, y_2, \dots, y_m \rangle$, find the common subsequence of X and Y with maximum length.

Example: for $X = \langle a, b, b, a, a, b \rangle$ and $Y = \langle a, a, b, b, a, b, b \rangle$, $\langle a, b, b, a, b \rangle$ is the common subsequence (you can verify that it is the longest one by observation).

Algorithm and Idea:

The basic steps for designing dynamic programming algorithm and their workouts for LCS are given below:

Characterizing structure to obtain optimal substructure: Given a sequence $X = \langle x_1, x_2, \dots, x_m \rangle$, $X_i = \langle x_1, x_2, \dots, x_i \rangle$ is called i th prefix of X , here we have X_0 as empty sequence.

Theorem 1: (Optimal substructure of an LCS)

Let $X = \langle x_1, x_2, \dots, x_m \rangle$ and $Y = \langle y_1, y_2, \dots, y_n \rangle$ be sequences and $Z = \langle z_1, z_2, \dots, z_k \rangle$ be any LCS of X and Y .

1. if $x_m = y_n$, then $z_k = x_m = y_n$, and Z_{k-1} is an LCS of X_{m-1} and Y_{n-1} .
2. if $x_m \neq y_n$, then $z_k \neq x_m$ implies Z is an LCS of X_{m-1} and Y .
3. if $x_m \neq y_n$, then $z_k \neq y_n$ implies Z is an LCS of X and Y_{n-1} .

Proof:

1. if $z_k \neq x_m$ then we can add $x_m = y_n$ to Z to get the LCS of length $k+1$, this is a contradiction that Z is an LCS, so $z_k = x_m = y_n$ must be true. Now Z_{k-1} is LCS of X_{m-1} and Y_{n-1} , otherwise we can get common sequence of both X_{m-1} and Y_{n-1} , say L such that it has length greater than $k-1$ so by adding $x_m = y_n$ we get a common sequence of X and Y with length greater than k , contradiction that Z is an LCS of length k .
2. if $z_k \neq x_m$ then Z is the common sequence of X_{m-1} and Y . If there were common sequence L of X_{m-1} and Y with length greater than k , then L would also be the common sequence of X and Y , contradicting that Z is an LCS of X and Y .
3. Similar as proof of 2.

Recursive definition of optimal solution: let $c[i,j]$ denotes length of an LCS of sequences X_i and Y_j . We can define the solution, using above theorem, recursively as:

$$c[i, j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0, \\ c[i-1, j-1] + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j, \\ \max(c[i, j-1], c[i-1, j]) & \text{if } i, j > 0 \text{ and } x_i \neq y_j. \end{cases}$$

Computation of optimal cost in bottom up manner: Recursive computation of above recurrence would take exponential time however we can notice that there are $O(mn)$ distinct subproblems. So if we can compute these distinct subproblems only once then we save the running time. The below is an algorithm that uses dynamic programming approach to solve LCS problem.

Algorithm: The inputs to the algorithm are sequences X and Y and outputs two arrays $c[1 \dots m, 1 \dots n]$ and $b[1 \dots m, 1 \dots n]$, here b array is for easier computation of optimal solution.

```

LCS(X, Y)
{
  m = length[X];
  n = length[Y];
  for(i=1; i<=m; i++) c[i, 0] = 0;
  for(j=0; j<=n; j++) c[0, j] = 0;
  for(i = 1; i<=m; i++)
    for(j=1; j<=n; j++)
      {
        if(X[i]==Y[j]) { c[i][j] = c[i-1][j-1]+1;          b[i][j] = "upleft"; }
        else if(c[i-1][j]>= c[i][j-1]) { c[i][j] = c[i-1][j]; b[i][j] = "up"; }
        else { c[i][j] = c[i][j-1]; b[i][j] = "left"; }
      }
  return b and c;
}

```

Example:

Determining an LCS of $\langle A, B, B, A, B, A, B, A \rangle$ and $\langle B, A, B, A, A, B, A, A, B \rangle$

	0	A 1	B 2	B 3	A 4	B 5	A 6	B 7	A 8
0	0	0	0	0	0	0	0	0	0
B 1	0	0 u	1 ul	1 ul	1 l	1 ul	1 l	1 ul	1 l
A 2	0	1 ul	1 u	1 u	2 ul	2 l	2 ul	2 l	2 ul
B 3	0	1 u	2 ul	2 ul	2 u	3 ul	3 l	3 ul	3 l
A 4	0	1 ul	2 u	2 u	3 ul	3 u	4 ul	4 l	4 ul
A 5	0	1 ul	2 u	2 u	3 ul	3 u	4 ul	4 u	5 ul
B 6	0	1 u	2 ul	3 ul	3 u	4 ul	4 u	5 ul	5 u
A 7	0	1 ul	2 u	3 u	4 ul	4 u	5 ul	5 u	6 ul
A 8	0	1 ul	2 u	3 u	4 ul	4 u	5 ul	5 u	6 ul
B 9	0	1 u	2 ul	3 ul	4 u	5 ul	5 u	6 ul	6 u

Analysis:

The above algorithm can be easily analyzed for running time as $O(mn)$, due to two nested loops.

The space complexity is $O(mn)$.

Construction of optimal solution:

The optimal solution can be obtained from the table given by $b[i, j]$. First we look at $b[m, n]$ and trace through table following the words in it i.e. whether to left or up or upleft. The following algorithm prints the LCS.

```

PrintLCS(b, X, i, j)
{
  if((i==0)||j==0) return;
  if(b[i][j] = "upleft") {PrintLCS(b, X, i-1, j-1); print X[i];}
  else if(b[i][j] = "up") PrintLCS(b, X, i-1, j);
  else PrintLCS(b, X, i, j-1);
}

```

Example: For our above example we have BABABA as an LCS. (Remember the LCS may not be unique but the length is unique).

What is the complexity? $O(m+n)$!

0/1 Knapsack Problem

Statement: A thief has a bag or knapsack that can contain maximum weight W of his loot. There are n items and the weight of i th item is w_i and it worth v_i . An amount of item can be put into the bag is 0 or 1 i.e. x_i is 0 or 1. Here the objective is to collect the items that maximize the total profit earned.

We can formally state this problem as, maximize $\sum_{i=1}^n x_i v_i$ using constraint $\sum_{i=1}^n x_i w_i \leq W$.

Algorithm:

The algorithm takes as input maximum weight W , the number of items n , two arrays $v[]$ for values of items and $w[]$ for weight of items. Let us assume that the table $c[i,w]$ is the value of solution for items 1 to i and maximum weight w . Then we can define recurrence relation for 0/1 knapsack problem as

$$c[i, w] = \begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0 \\ c[i-1, w] & \text{if } w_i > w \\ \max(v_i + c[i-1, w-w_i], c[i-1, w]) & \text{if } i > 0 \text{ and } w \geq w_i \end{cases}$$

You can think the above relation like: if we have got no item or the maximum weight is zero then the solution is 0, if the weight of the object that is being added exceeds the total weight remaining then the solution is the solution up to the previous item addition and if the total weight remaining is greater than or equal to the weight of the item being added, then the solution consists of the value that is maximum between the summed up value up to previous addition of item or the value after addition of the last item.

DynaKnapsack(W, n, v, w)

```
{
    for(w=0; w<=W; w++) c[0,w] = 0;
    for(i=1; i<=n; i++){ c[i,0] = 0;
        for(w=1; w<=W;w++)
            if(w[i]<W) then if v[i] + c[i-1,w-w[i]] > c[i-1,w] then
                c[i,w] = v[i] + c[i-1,w-w[i]];
            else c[i,w] = c[i-1,w];
        else c[i,w] = c[i-1,w]; }}
```

Example:

Let the problem instance be with 9 items where $v[] = \{2,3,3,4,4,5,7,8,8\}$ and $w[] = \{3,5,7,4,3,9,2,11,5\}$ and $W = 15$.

W	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i=0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
i=1	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2
i=2	0	0	0	2	2	3	3	3	5	5	5	5	5	5	5	5
i=3	0	0	0	2	2	3	3	3	5	5	5	5	6	6	6	8
i=4	0	0	0	2	4	4	4	6	6	7	7	7	9	9	9	9
i=5	0	0	0	4	4	4	6	8	8	8	10	10	11	11	11	13
i=6	0	0	0	4	4	4	6	8	8	8	10	10	11	11	11	13
i=7	0	0	7	7	7	11	11	11	13	15	15	15	17	17	18	18
i=8	0	0	7	7	7	11	11	11	13	15	15	15	17	17	18	18
i=9	0	0	7	7	7	11	11	15	15	15	19	19	19	21	23	23

Analysis:

For run time analysis examining the above algorithm the overall run time of the algorithm is $O(nW)$.

Exercises

1. Disprove that all the instances of 0/1 knapsack problem can be solved using greedy approach.
2. Determine LCS for sequences $\langle 1,0,0,1,0,1,0,1 \rangle$ and $\langle 0,1,0,1,1,0,1,1,0 \rangle$
3. Determine the optimal parenthesizations for sequence of matrices ABCDEF with p array as $\{40,5,35,45,10,20,10\}$

WORLDLINK Convergence

WORLDLINK Convergence is a real-time, integrated call accounting, billing and customer care software solution that enables converged billing of voice products and services. It offers a comprehensive, highly customizable, reliable and scalable platform that can integrate into any voice network infrastructure. With WORLDLINK Convergence, voice service providers can achieve consolidated billing and single-point customer care for multiple voice service types, using RADIUS and other middleware that allow data aggregation and call detail input from disparate voice gateways and switches.

Highlights

- Unlimited, extensive and highly customizable products and account management
- Real-time authentication, authorization, accounting (AAA) and CDR generation
- Real-time charge accrual, billing, ratings and statistics
- Corporate accounts with group billing
- Highly customizable reseller management
- Web-based management, customer care and self-provisioning interface
- Online monitoring and statistics
- Call-to-call profit/loss analysis
- Highly customizable reports
- Hosted billing services
- Built-in data replication, redundancy and failover
- Open architecture and integration ready

Unlimited, Extensive and Highly Customizable Products and Account Management

WORLDLINK Convergence offers the flexibility and needed provision to support unlimited number of customized products and ratings, enabling service providers to tailor their services according to the market. Currently supported product types include:

- Prepaid
- Rechargeable Prepaid
- Post Paid
- Limited Credit Post Paid
- Corporate (Group)

Thanks to the open extensible architecture of WORLDLINK Convergence, adding any new product type can be done swiftly. Customizations of products can be broadly applied in five areas. These are:

- Product Type
- Tariffs
- Rating Rules
- IVR Integration
- Invoicing
- Currency
- Surcharges & Taxes

Flexible tariff management includes the ability to define and instantly activate new tariffs or promotional offers. Tariff schemes allow the service provider to factor in product type, client location (originating node) time of day, day of week, holidays and promotional offers. New tariffs can be activated instantly or scheduled for a future date for a specific period or duration without affecting service. With support for multiple currencies and time zones, service providers can deploy customized products for any part of the world using one central system. New products can be created within minutes and offered online to customers. Service providers can even create customized products for individual customers.



Reseller Management

Real-time Authentication, Authorization, Accounting (AAA) and CDR Generation

WORLDLINK Convergence includes a full-featured advanced RADIUS module that works with all RADIUS-compliant gateways such as Cisco and Quintum voice gateways. This high performance multi-threaded server fully supports the standard as well as Vendor Specific Attributes (VSAs). The power of this module lies in the fact that it is highly customizable to enable a multitude of rich product features such as flexible Interactive Voice Response (IVR) scripts.

Real-time Charge Accrual, Billing, Ratings and Statistics

WORLDLINK Convergence performs high throughput charge application, billing, rating and statistical analysis in real-time. Billing is done for each billable CDR based on set product definitions and ratings. Real-time billing allows the service provider to modify and fine-tune various rating parameters and view the results instantaneously. WORLDLINK Convergence tracks service usage and payments in real-time. When service usage reaches a pre-defined limit, the system automatically sends a low balance alert via SMS and email. A voice prompt can also be played to the customer through the IVR, if configured for the product. Once usage reaches the credit limit defined for an account, additional calls are not authorized and existing calls are dynamically disconnected.

Corporate Accounts with Group Billing

WORLDLINK Convergence supports corporate customers with multiple end-user accounts under a single group account. This account can control and add end-user accounts under it while affording full and independent end-user account management through the online self-provisioning interface. A corporate account gets a pool of credit (prepaid deposit or postpaid credit limit) that can be distributed among its end-user accounts or shared between the accounts as a single pool. Individual as well as consolidated invoices are automatically generated and sent at predefined billing intervals.

Highly Flexible Reseller Management

Resellers are customers that market existing or customized products to other end users on a revenue sharing model. WORLDLINK Convergence provides resellers the capability to independently manage all aspects of end-user accounts under them through the online self-provisioning interface. For the service provider, it provides real-time monitoring and management of resellers' activities and receivables. For the prepaid industry, when batches of accounts are provided to the reseller on credit, WORLDLINK Convergence keeps track of the Failover utilization of accounts (first use and depletion) per batch and alerts when it crosses predefined limits for unpaid batches.

Web-based Management, Customer Care and Self-Provisioning Interface

WORLDLINK Convergence includes an intuitive and comprehensive Self-Provisioning Interface that provides management features for all aspects of the system. All configurations, product creation and definitions, modifications or enhancements, reporting, security and customer service are carried out through this interface. Secure access is provided for all users including end-users, resellers, corporate customers, customer service operators and administrators based on predefined roles and access privileges. Customers can service themselves thereby lowering operational costs for the service provider. The interface easily integrates into the service provider's web site to instantly provide a storefront for online product purchases. For end-users, it includes easy to use features to reliably and securely:

- Purchase any number of products using the online storefront.
- View remaining credit and account balances
- Replenish accounts using secure connections
- Modify existing services that have been purchased
- View or download invoices and call detail records
- Dispute call charges (Call Dispute Management) depending on the service provider

Online Monitoring and Statistics

WORLDLINK Convergence provides a host of real-time tools that monitor and graph critical performance indicators such as quality, utilization and profitability. Vital quality metrics such as ASR, ACD and PDD can be monitored on the basis of a destination, a breakout within the destination, a carrier, a product, carriers within a product, destinations

within a product, and any other combination of the above. Additionally, quality alarms can be configured using highly flexible rules to send an alert (email and/or SMS) when any combination of quality parameters degrade below a preset level for any product, carrier, destination, or combination of these three. Alternatively, the alarms can be fed into a trouble-ticketing application.

Similarly, utilization and profitability can be monitored on the basis of a destination, a breakout within the destination, a carrier, a product, carriers within a product, destinations within a product, as well as any other combination of the above. These tools enable quick response to quality degradation and losses. It allows monitoring of peak resource utilization, and thus assists in capacity planning. It also allows help desk operators to manage customer service operations effectively.

Call-to-Call Profit/Loss Analysis

WORLDLINK Convergence provides profit/loss analysis on a call-to-call basis. The service provider can zoom in on the exact calls and destinations that are profitable. More importantly, unprofitable calls and destinations can be instantly revealed so that the service provider can take immediate corrective action. In real time, the effects of these changes can be viewed, and through fine-tuning, achieve optimal profits.

Highly Customizable Reports

WORLDLINK Convergence provides service providers with comprehensive reports including:

- Billing and Accounting Reports
- Revenue & Expenses Reports
- Call reports for End-users, Resellers, Corporate accounts and carriers
- Quality and performance reports

No	Name	Description	Nature	Amount	Frequency	Type	Tax	Date	Assign	Edit	Delete
1	Recurring Charge	Recurring Charge	<Charge	.05	Recurring	Percentage	0	12-09-2004	<input type="checkbox"/>		
2	Activation Fee	Activation Fee	<Charge	5	Activation	Fixed	0	12-09-2004	<input type="checkbox"/>		
3	Setup Fee	Setup Fee	<Charge	10	One Time	Fixed	0	12-09-2004	<input type="checkbox"/>		
4	Recurring Charge	Recurring Charge	<Charge	1	Recurring	Percentage	0	12-09-2004	<input type="checkbox"/>		
5	Maintenance Fee	Maintenance Fee	<Charge	5	Monthly	Fixed	0	12-09-2004	<input type="checkbox"/>		
6	Connecting Fee	Connection Fee	<Charge	.05	Recurring	Percentage	0	12-09-2004	<input type="checkbox"/>		
7	Connection Fee	Connection Fee	<Charge	.05	Recurring	Fixed	0	12-09-2004	<input type="checkbox"/>		

Product Charges

ID	Country	Interval	Rate	Status	Edit	Delete
1	FRANCE	30	1	●		
2	FRANCE	60	.4	●		
3	FRANCE	300	2	●		

ID	Country	Start Date	End Date	Start Time	End Time	Day	Calling Number	IMSI	Busy Call	Status	EOP	Delete
1	FRANCE	01-JAN-04	31-DEC-05	00:00:00	23:59:00	All				●		
2	FRANCE	01-JAN-04	31-JAN-05	00:00:00	00:00:00	Sat				●		
3	FRANCE MOBILE	01-JAN-04	31-JAN-05	00:00:00	23:59:00	All				●		
4	FRANCE FWR.13	12-SEP-04	12-SEP-04	00:00:00	00:00:00	All				●		
5	FRANCE MOBILE	01-JAN-04	31-DEC-04	00:00:00	00:00:00	Sat				●		
6	FRANCE FWR.13	01-JAN-04	31-DEC-04	00:00:00	23:59:00	All				●		
7	FRANCE	01-JAN-04	31-DEC-05	00:00:00	00:00:00	All	3367229474		✓	●		
8	FRANCE	01-JAN-04	31-DEC-04	01:00:00	00:00:00	All		33143228497	✓	●		

Adjusted Product Rates

These reports are easily exported into a variety of formats including PDF, Microsoft Excel, Microsoft Word, HTML or CSV. Leveraging the power of WORLDLINK Convergence’s relational database, service providers can generate almost any report in real-time with up-to-date information using third party reporting tools such as Crystal Reports and Oracle Reports.

Hosted Billing Services

WORLDLINK Convergence can be used to offer call billing services to third parties on an application service provider (ASP) model. These third parties, known as hosted service providers, can independently manage their voice products and services, carriers and customers. This partitioning is made possible by WORLDLINK Convergence’s four-tier billing model.

Built-in Data Replication, Redundancy and

WORLDLINK Convergence exists as a server-side application running on Oracle. The tight integration with Oracle using native Oracle functions makes WORLDLINK Convergence extremely fast and robust with powerful capabilities. WORLDLINK Convergence features a multi-server architecture with built in redundancy at multiple

levels. Real-time database replication with disparate servers for billing and administration ensures high processing speed and complete data integrity. Seamless and real-time failover with automatic recovery on proven industrial strength Oracle RDBMS guarantees telco-grade reliability. A full-featured version of WORLDLINK Convergence is also available for open source database, PostgreSQL.

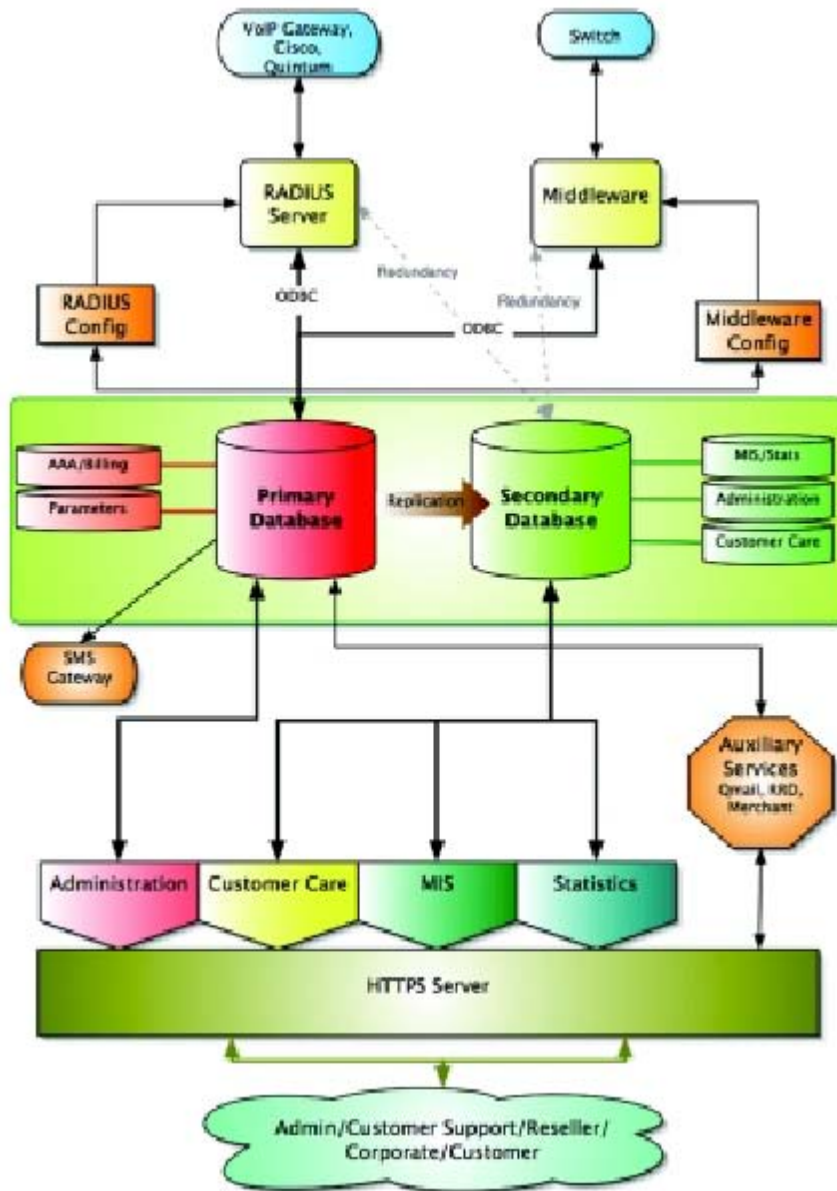
Open Architecture and Integration Ready

WORLDLINK Convergence has been integrated with voice gateways from Cisco and Quintum through open standards based RADIUS. It can seamlessly integrate with any other voice gateway or switch through customized middleware available through Fractalcom.

ID	Country	Interval	Rate	Status	Edit	Delete
1	FRANCE	30	1	●		
2	FRANCE	60	.4	●		
3	FRANCE	300	2	●		

Product Rates

System Architecture



System Components

Hardware

Two Linux Servers. Recommended configuration: Intel Pentium Xeon 2.8 GHz, 1 GB RAM, 100 GB of RAID storage
Any RADIUS-Compliant Voice Gateway (Cisco, Quintum) or Switch

System Software

WORLDLINK Convergence
Apache Web Server with modules
Oracle SE or PostgreSQL Database Server

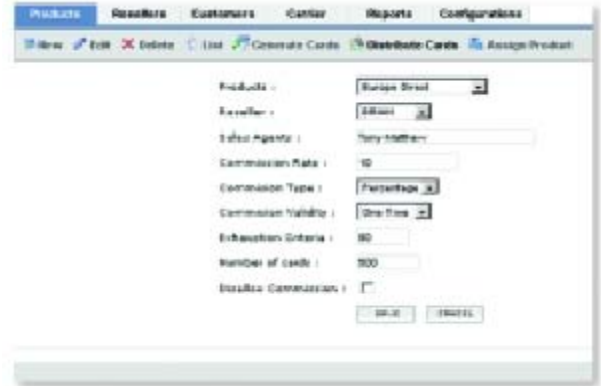
Client Software

Browser: Internet Explorer 5.5 or higher, Netscape 6.2 or higher, Mozilla 1.1 or higher

Key Features

Products & Services

- Corporate and regular product/service type
- Comprehensive product/service types
- Unlimited number of products/services
- Fine-grained customization of products/services
- Integration of IVR
- Cross product/service discount



Product Distribution

Multiple Authentication Options

- Account Number/Card Number
- ANI
- PIN
- Tech-Prefix
- DNIS

Multiple Billing Cycles

- Weekly
- Fortnightly
- Monthly
- Customizable by service provider

Dial Options

- ANI Authorization
- Local Access Number
- Toll Free Dialing
- Abbreviated Dial (10-10)
- Speed Dial

Flexible and Extensive Rating Rules

- Time of Day, Day of Week and/or Month of Year
- Vacation Rate
- Peak Time
- Flag fall
- Deny Call
- Based on ANI, DNIS
- Progressive
- Minute Fraction (Dual Credit Time)
- Adjusted Incremental/Declining Rates

IVR

- Credit Time or Balance Announcement
- Customizable through Product Configuration
- Multi-language IVR Support
- Customizable IVR Scripts
- Seamless integration of new IVRs

WORLDLINK Convergence is not tied to any single billing model. Instead, the most flexible approach possible has been taken, allowing complete flexibility to implement the business rules required to support virtually any billing model. This provides the service provider the ability to launch new products and services quickly in full support of market demands and regulatory requirements.

Multiple Surcharges

- Flat or Percentage Based
- Activation
- Periodic
- Recurring
- One-time

No	Name	User Name	Used Rate	Limit	Balance	Call Duration	Cost	Last Recharged
1	Jamy Gage (JENNY)	008281013074	12-SEP-2004 1.00	12	38	38		
2	Jonathan Edward (JONATHAN)	008413443755	15-SEP-2004 1.00	-87	104	107		
3	Jaykus W&O (MING)	008777048565	12-SEP-2004 1.00	10	98	90		
4	Maggie Smith (MARGIE)	007603009676	12-SEP-2004 1.00	30	37	50		
5	Mang Patel (MANGU)	002637692118	12-SEP-2004 1.00	-4	83	108		
6	Tony Parvathi (TONY)	004753148810	12-SEP-2004 1.00	74	11	4		

Product wise list of Customers

Fraud Management

Simultaneous calls are not allowed for end-user accounts, unless configured for post-paid customers.
 Attempts to authorize PINs belonging to batches that have not been sold indicate PIN leaks.
 Within a predefined interval, if more than a predefined number of different PIN authorization attempts are made from a single ANI, all further calls from that ANI are blocked.

Call Matching, Reconciliation, Settlements & Dispute Management

Reconcile bills of interconnecting partners by matching billing records
 Generate and send statements to carriers and interconnecting partners
 Settle accounts of interconnecting partners through reconciliation and settlement adjustments for disputed items

Rate Plans Analysis, Reports & Audits

Ability to model new rate plans and analyze the impact on costs and revenues
 Carry out mass rate updates
 User customizable and predefined traffic and management reports
 User definable roles and access privileges the system access and restriction
 Audit history of user activities & transactions

Carrier Account Management and Billing

Create and manage multiple carrier accounts
 Create and manage destination codes
 Define buying and selling rates per destination code per carrier
 Define multiple billing cycles
 Rates for time of day, day of week, holidays and exceptions
 Define usage-based, fixed and recurring charges

Name	Type	Generated	Remaining	Scratched	Exhausted	Card Value	No. Cards	Expiry Period	Status
CALL FRANCE	RPA	100	74	24	22	250	1	365	●
Call Roma	RPA	800	408	149	137	50	1	365	●
Call Italy	RPA	400	208	109	77	50	1	90	●
Europa Direct	RPA	100	74	25	25	100	1	90	●
First Connection	PRE	400	188	171	156	100	1	365	●
Hello France	PRE	500	408	101	34	20	1	90	●
Hello Italy	POST ANI	0	0	0	NA	20	1	90	●
Hello United Kingdom	PRE	1000	208	171	132	50	1	90	●
Instant Connection	RPD	100	95	5	5	250	1	365	●
Jupiter	POST One-Shot	4	0	0	NA	0	1	365	●
Talk United Kingdom	PRE	500	408	171	109	20	1	90	●
World Connection	POST ANI	0	0	0	NA	100	1	90	●

Product List

Product Types

- Regular – 1 Card (also called PINS) per Product when purchased online through credit-card; Any number of cards when distributed through a Reseller
- Corporate – Always more than 1 Card per Product

Card Types

- Prepaid
- Rechargeable Prepaid
- ANI based rechargeable prepaid
- ANI based rechargeable prepaid with Abbreviated Dial
- Post Paid
- ANI based postpaid
- ANI based postpaid with Abbreviated Dial
- Limited credit postpaid
- ANI based limited credit postpaid
- ANI based limited credit postpaid with Abbreviated Dial



Product Property

Product Attributes

- Product Name
- Product Code
- Detailed Product Description
- Product Nick-Name
- Product Card Type (Prepaid, Postpaid, ANI, Rechargeable, etc.)
- Card length – number of digits in each card
- Initial Credit Amount for Prepaid card types
- Credit Limit for Postpaid Limited card types
- Product type – Regular or Corporate
- Default Product – Product that defines Rates for all Country Codes
- Product Activation date
- Billing Method – Prepaid or Postpaid
- Expiration date – The date card expires
- Expiration offset – Number of days, months after which the card expires after first use
- Number of card(s) per Product
- Billing frequency – Intervals of time at which to generate invoice for the card
- Product currency
- Low credit warning
- Grace Period for invoice receipt in days
- IVR announcement of remaining balance
- IVR announcement of remaining time
- Ignore Duration – Calls with duration less than this are not billed
- Minimum Duration – Calls less than this duration are billed for this duration
- Restrict calls for Product(s) only through a particular PSTN Access Number
- Maximum call duration for postpaid accounts
- Rounding Schemes for call duration
- PIN authentication

Charges

Charges can be defined to be either Fixed or Percentage and applied to Products. The following charges are supported.

- Activation - When the card is first activated
- Periodic – Daily, Weekly, Fortnightly, Monthly, Bi-Monthly, Quarterly, Semi-Annual, Annual
- Recurring - Each time the card is used
- One-time - When the card is used for the first time or when the card is activated

Tax

Tax is defined in terms of Percentage and can be applied to card(s) under product.

Product Country Rates

Rate is defined for each Country Code under the Product. Rate is based on increments. For example, for call up to 1 minute a defined rate can be applied, for call duration more than 1 minute but less than 3 minutes, a new rate can be applied, and so on. There is no restriction to the number of intervals that can be defined. A default Product should have Rates defined for all Country Codes.

Rating Rules

Rating Rules is perhaps the most effective mechanism to maximize profit. A Rating Rule can be defined to select the desired Country Rate for a given call when the call originates. Moreover, a Rating Rule can be defined to pick an apt Country Rate for a given call when it terminates, at the time of real-time billing. These adjusted Rates override the Product Country Rates for the Country Code in contention. There is no limit to the number of Rules that can be defined. When a Rule qualifies for a call, if the Deny Call flag is set then the call will not be authorized. The following criteria determine whether a call qualifies for a Rule or not –

- Start Date/Time, End Date/Time
- Day of Month, Date of Month, Every Day
- Day of Week
- Time of Day
- Calling Number
- Called Number

The following components comprise Rating Rules.

Minute Fraction can be used to adjust number of seconds per minute for talk-time determination as well as for call duration.

Adjusted Incremental Rates are the Rates that override the Product Country Rates. For example, a rate of \$0.30 can be defined for the first 30 seconds, then a different rate of \$0.35 for calls between 30 and 60 seconds, and a different rate for different interval and so on. There is no limitation as to the number of intervals that can be defined.

Invoice

Invoicing is done automatically as defined for the Product. The following invoicing schemes are supported for postpaid and credit limited postpaid Products -

- Weekly
- Fortnightly
- Monthly

Product Management

- Create Products
- Customize Product Attributes
- Create Product type (Regular or Corporate)
- Create Charges
- Associate Charges with Products
- Create Taxes
- Associate Taxes with Products
- Define Default Product Country Rates
- Define Country Rates
- Define Rating Rules
- Define Incremental Rates
- Define Minute Fraction
- Generate Cards

- Distribute Batch of Cards to a Reseller or Distributor
- Define Sales Agents
- Disallow Commission for the Reseller
- Define Commission Rate for the Reseller
- Define Commission Type for the Reseller – Percentage or Flat
- Define Commission Validity – Adhoc or Life-long
- Define Exhaustion Criteria for Products distributed to a Reseller to better facilitate the invoice collection process
- Define Number of Cards assigned to the Reseller

Customer Attributes

- Customer First, Middle, Last Names
- Description
- Nick-Name
- Customer Type – Regular, Corporate, Trivial, End-User
- Login Code – E-mail Address is used as the login name
- Password to Login
- Creation Date
- Disabled Date
- Secondary E-mail Address
- Phone Number
- Mobile Number
- Fax Number
- Long Address
- PO Box
- State, Zip Address
- Corporate Customer Company Name
- Corporate Customer URL
- Customer Billing Frequency
- Speak Amount – Corporate Customers can use this flag to configure whether to announce Remaining Balance to Cards under the Customer
- Speak Time – Corporate Customers can use this flag to configure whether to announce Remaining Time to Cards under the Customer
- Credit Card Number
- Credit Card Expiry Date
- Credit Card Verification Code
- Credit Card Type



Product User Status

Customer Management

- Create Corporate Customers
- Create Trivial Customers
- Create Corporate End Users – Account holders under a Corporate Customer

Customer Self-Care

- Edit Profile
- Purchase new Products
- Register new ANIs for ANI-Based Products
- View Call Detail Information
- View Historical Call Details

Reseller Attributes

- First, Middle, Last Names
- Description
- Nick-Name
- Login Code – E-mail Address is used as the login name

- Password to Login
- Commission Rate
- Sales Agent
- Creation Date
- Disabled Date
- Secondary E-mail Address
- Phone Number
- Reseller Mobile Number
- Reseller Fax Number
- Reseller Long Address
- Reseller PO Box
- Reseller State, Zip Address
- Reseller Company Name
- Reseller URL
- Customer Billing Frequency

No.	Product Name	Card Type	Card Rate	Total Cards	Scratched	Exhausted
1.	Call 400	MSL	80	400	400	0
2.	Screen Guard	MSL	105	50	50	0
3.	Mobile Phone	PHS	20	200	200	0
4.	Mobile Data	POST PAID	30	200	0	0
5.	Mobile Linker Singapore	PHS	24	100	100	0
6.	Talk United Kingdom	PHS	20	400	400	0
7.	World Convergence	POST PAID	200	200	0	0

Reseller wise list of Products

Reseller Management

- Create Resellers
- Distribute Cards to Resellers
- Assign Commission Rate
- Assign Commission Type

Reseller Self-Care

- Edit Profile
- Purchase new Products
- Register new ANIs for ANI-Based Products
- View Call Detail Information
- View Historical Call Details

Reports

- Call Detail Information per Card (or PIN) – Card, Timestamp, DNIS, Call Duration, Duration Cost, Charges, Taxes
- Scratched Cards
- Percentage Scratched
- Exhausted Cards
- Percentage Exhausted
- Card used per Product
- Calls Completed
- Calls Failed
- Calls not Answered
- Fraudulent Uses
- Call Success Rate (CSR)
- Average Call Duration (ACD)
- Maximum Call Duration
- Minimum Call Duration
- Total Call Traffic in minutes, hours, week, month
- Analysis of terminated calls
- Real-time Remaining Balance for Prepaid and Rechargeable Prepaid Cards
- Real-time Remaining Credit for Credit Limited Postpaid Cards
- Reseller vs. Reseller Performance Analysis
- Country Code Analysis

No.	Date	Time	Call Duration	Country Code	Calling	Called Number	Duration
1	04-08-2004	11:08:01	00:00:00	000	0000000000	0000000000	00:00
2	04-08-2004	11:08:02	00:00:00	000	0000000000	0000000000	00:00
3	04-08-2004	11:08:03	00:00:00	000	0000000000	0000000000	00:00
4	04-08-2004	11:08:04	00:00:00	000	0000000000	0000000000	00:00
5	04-08-2004	11:08:05	00:00:00	000	0000000000	0000000000	00:00
6	04-08-2004	11:08:06	00:00:00	000	0000000000	0000000000	00:00
7	04-08-2004	11:08:07	00:00:00	000	0000000000	0000000000	00:00
8	04-08-2004	11:08:08	00:00:00	000	0000000000	0000000000	00:00
9	04-08-2004	11:08:09	00:00:00	000	0000000000	0000000000	00:00
10	04-08-2004	11:08:10	00:00:00	000	0000000000	0000000000	00:00
TOTAL							00:00

Call Records

Customer Service Operators

Depending upon the privileges, a Customer Service Operator can assist the Customer or Reseller in the following manner.

Search for Customers by card number (PIN), card distribution number, serial number, card type, product type, product, etc.
Search for Resellers by card number (PIN), card distribution number, serial number, card type, product type, product, distribution date, sales agent, commission type, commission validity, etc.
View or Edit Customer Account Information
View or Edit Reseller Account Information
View Call Detail Information for a Customer
View Customer Account Transactions
Credit Card Transaction
Add disputed balance
Log complaints

Customers

Customers are classified into – (i) Corporate Customers that own at least one Corporate Product (ii) Trivial Customers that own Regular Product(s) (iii) Corporate End Users are Customers under a Corporate Customer

A Corporate Customer can –

- Purchase any number of Products using the online web-shop
- View Remaining Balances of Cards under Products
- View Credit Remaining in case of Credit Limited Products
- View Current Balance of Postpaid Products
- Recharge Cards
- View Call Detail Information
- View Historical Call Details
- View or Download Invoices
- Transfer Balance from the Credit Pool to Cards or vice-versa
- Set Credit Limits on individual cards
- Opt for a secret PIN number authentication for a Card and set PIN number
- Add new ANIs for ANI-Based Products
- Override Product flag to not speak Remaining Balance
- Override Product flag to not speak Remaining Time
- Edit Personal Profile and Contact Information

Trivial Customers and Corporate End Users have the privileges to –

- Purchase any number of Products using the online web-shop
- View Remaining Balances of Cards under Products
- View Credit Remaining in case of Credit Limited Products
- View Current Balance of Postpaid Products
- Recharge Cards
- View Call Detail Information
- View Historical Call Details
- View or Download Invoices
- Opt for a secret PIN number authentication for a Card and set PIN number
- Edit Personal Profile and Contact Information

Web Storefront

Service providers can integrate their own merchant gateway services. All transactions are secured through Secure Socket Layer Protocol. The storefront is tightly integrated with billing and reflects real-time balances.

A Customer can:

- View product details like rates and charges
- Add purchased product into own account
- Provide payment via credit card



Intrusion Detection and Reaction in VoIP Networks

An internship report
submitted by

MORES AIRES Daniel
MSc

Responsible: CUPPENS Frédéric
CUPPENS Nora

June 2008
Département Réseaux et Services Multimédias (RSM)
Telecom Bretagne
2, Rue de la Chétaigneraie 35510 SESSON SEVIGNE - FRANCE

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my project guides Frédéric and Nora Cuppens, Enseignant-chercheur, Département Réseaux et Services Multimédias (RSM) Telecom Bretagne, for their constant guidance, encouragement and constructive criticism during the whole course of the internship.

I also wish to acknowledge the indispensable moral support extended by Yacine Bouzida, Département Réseaux et Services Multimédias (RSM), Telecom Bretagne, who has been painstakingly helped for the timely completion of the project.

I would like to place on record the deep gratitude towards Fabien Autrel for his help towards my odd work with CRIM and François Wang for his gratefully help on language coding.

I also sincerely acknowledge the help of all the people who directly or indirectly helped me in my project work and constantly encouraged me.

Daniel MORES AIRES
MSc 2A
Rennes, june 2008

Contents

Abstract	5
Introduction	6
1 VoIP Attacks: Detection, Correlation and Reaction Models	7
1.1 VoIP Introduction	7
1.1.1 Classic VoIP architecture	8
1.2 Introduction to SIP	9
1.2.1 SIP Architecture	9
1.2.2 SIP Methods	10
1.2.3 SIP Call Example	10
1.3 SIP: Different Vulnerabilities	11
1.3.1 Elementary Attacks	12
1.3.2 SIP Attacks by Special Crafted Messages	16
1.3.3 SIP Specific Vulnerabilities	16
1.4 Protection and Detection Mechanisms	19
1.4.1 Basic counter measures	19
1.4.2 Intrusion Detection Mechanisms	19
1.5 Reaction Techniques	21
1.6 CRIM as a Correlation and Reaction Model	22
1.7 Conclusion	24
2 Experimentation	25
2.1 Introduction	25
2.2 Implemented Attacks	25
2.2.1 ARP Cache Poisoning and MAC Spoofing	26
2.2.2 DNS Cache Poisoning: Transaction ID prediction	28
2.2.3 SIP Attack: Bye Message	30
2.2.4 Denial of Service: Identity Hijacking	33

2.2.5	SIP Fuzzing	33
2.3	Attack Scenario	34
2.3.1	Including Snort Attack Signatures	34
2.3.2	Experimentation: generating IDMEF alarms	36
2.4	Correlation and Reaction Models	37
2.4.1	Elaborating Lambda Action Models	38
2.4.2	Correlating Action Models	39
2.4.3	Correlating Attack Models with the Intrusion Objective	40
Results		43
Conclusion		45
Bibliography		46

Abstract

Packet-based services such as Voice over IP systems are being adopted globally as an advantageous way of communication. Cost savings in hardware by merging voice, video and data applications can bring telecom carriers and organizations tremendous benefits, increasing productivity and generating new advantage services. However, how to conciliate an adequate Quality of Service (QoS) with the same safety levels of the Plain Old Telephony Systems (POTS) is still a great challenge nowadays.

Authentication and coding schemes adopted by those companies and service suppliers may be efficient, but with vulnerabilities in the network infrastructure, the whole system will be seriously exposed to attacks. Generally, hackers explore vulnerabilities by pretending to be legitimate servers and by connecting to the target network, promoting some kind of transaction and causing damage to the vulnerable system.

The goal of this internship is to firstly experiment some of the most usual attacks targeting VoIP networks. Secondly, The reaction models are studied and combined to quickly and efficiently detect and react to VoIP-based attacks. More specifically, the correlation and reaction tool CRIM (Cooperation and Recognition of Malevolent Intentions) developed by Telecom Bretagne will be used to generate adapted models capable of detecting and reacting to the intrusions presented in this report.

Key words

VoIP attacks, intrusion detection, correlation, anti-correlation and reaction systems.

Introduction

There are two approaches commonly used to avoid attacks. The first approach attempts to prevent intrusions by using access control technique, such as configuring firewalls with best fitted access-lists, user-access control policies and so on. The other method is called the intrusion detection approach and is focused on detecting attacks and providing special counter measures to obstruct attacks before they cause damage. The advantage of the second approach is the capacity to adjust itself to new attacks, thus providing specific reaction models to each attack detected. In these circumstances, two different variations of this last method are applied to detect intrusions: the scenario-based approach which uses a collection of previous attacks as input to generate alarms, and the behaviour-based approach which focuses on abnormal user activity to generate alarms. On the other hand, both techniques have weaknesses when employed independently. In fact they compensate mutual lacks, because a weakness of one is a strength of the other.

Different reaction techniques can be found in the literature. Some of them are compensation actions that either apply end-to-end encryption to the traffic or insert new firewall policies on the system. Other reactions are based on vulnerabilities due to bad code implementation and thus could be corrected by applying software updates. The reaction scheme used in this document is based on the anti-correlation method, proposed by CRIM.

This report is organized in two chapters. Chapter 1 describes VoIP-based architectures (section 1.1); the utilisation of SIP protocol (section 1.2); several vulnerabilities that are specific to VoIP and SIP (section 1.3), and finally detection methods and reaction models (sections 1.4 and 1.5), specifically the approach used by CRIM (section 1.6). Chapter 2 firstly presents the implementation of VoIP attacks (section 2.2). Secondly, section 2.3 presents an attack scenario, to which test two reaction models are tested. Section 2.4.3 presents the results of the experimentations, that validate the reaction techniques that are experimented.

Chapter 1

VoIP Attacks: Detection, Correlation and Reaction Models

1.1 VoIP Introduction

Solutions based on Voice over IP (VoIP) offer telephony services across networks using the Internet Protocols. They are the potential evolution of the classic telephone system (PSTN). VoIP technology consists of signaling and transmission protocols. Signaling protocols establish, control, and terminate telephone calls. Transmission protocols are used to efficiently deliver digitalized voice signals between servers in the IP network.

Refers to the figure 1.1 that illustrates the different protocols used on a classical VoIP system. This figure takes the example of a basic call flow between two entities. The first step is converting analog voice signals to digital pulses, using an analog-digital converter. Since digitized voice requires a large number of bits, a compression algorithm can be used to reduce the volume of data to be transmitted (G.711, G.723.1, G.726, G.729, and GSM). The next step is to insert voice samples into data packets to be transported on the Internet. The protocol used to encapsulate voice packets is typically the Real-time Transport Protocol [24]. RTP packets have special headers to keep correlation among them and to be correctly re-assembled in the destination. RTP streams are carried as data by UDP datagrams, which can then be processed by an ordinary network such as the Internet.

At the destination, the process is reversed: packets are disassembled and delivered to the de-jittering buffer which temporally stores packets so that they can be played out in the correct order. Generally, packets are dropped if they arrive too late to be reassembled. If a frame is lost, the decoder interpolates this "gap" with the last and the previous frames. Finally, the digitized voice is extracted from the frames and uncompressed, processed by a digital-to-analog converter and sent the analog signal to the handset speaker.

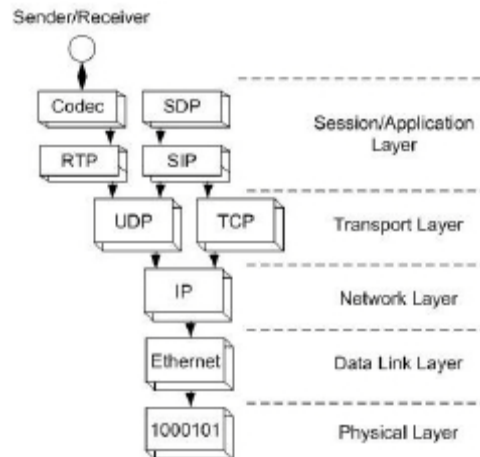


Fig. 1.1 : VoIP Protocols and Layers.

1.1.1 Classic VoIP architecture

Before utilizing VoIP as the infrastructure of protocols to transport voice over data networks, companies and service suppliers maintained two independent networks. These networks were controlled by two distinct signaling systems as well. The voice system had a centralized circuit switched network composed by Carrier Classes Switches (CCS) or Private Branch eXchange (PBX) and the packet-based network being composed by structured servers and gateways.

As depicted in figure 1.2, basically there are two ways to implement VoIP architectures. The simplest way is a distributed solution, also called peer-to-peer topology. In this approach, there is no single centralized point, each peer controls, initiates or terminates VoIP sessions. Another solution is a client/server mode. Clients are any kind of equipment that are set up or teardown connections, such as IP phones, softphones, etc. The server controls and routes incoming and outgoing calls to PSTN. Special requisites and functioning related to the SIP architecture will be further detailed in section 1.2.

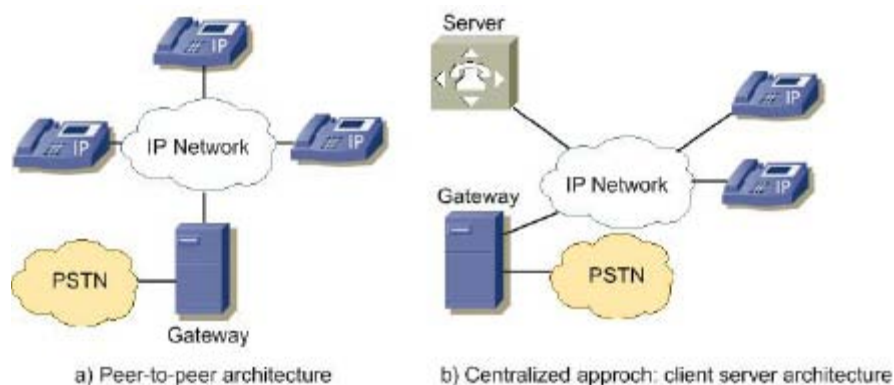


Fig. 1.2 : VoIP Architecture.

1.2 Introduction to SIP

There are several standardized protocols capable of executing call control on packet-based networks. The first protocol used to start and teardown voice sessions was H.323. This protocol is based on a set of PSTN protocols defined by ITU Telecommunication Standardization Sector (ITU-T). On the other hand, the Session Initiation Protocol (SIP), as defined by the Internet Engineering Task Force (IETF) [3], is a protocol used to create, modify and terminate media streams between servers. The experimentations described in Chapter 2 were conducted using SIP. The reasons for that choice are:

- SIP is being adopted by telecom carriers as a standard, mainly because it uses the same syntax and scripting technologies already used on Internet, thus facilitating its integration with already existing data systems.
- Most vendors of VoIP solutions are based on the SIP protocol.
- SIP makes possible to combine existing voice services with new multimedia services.
- The Third Generation Partnership Project (3GPP) accepted SIP as an element of the IP Multimedia Subsystem (IMS) to deliver IP based-services to mobile users.

1.2.1 SIP Architecture

SIP works in a client/server approach. The client part is called a "user agent". It originates and terminates voice sessions. The server part is normally called a "SIP server" and works as a registrar, proxy, redirect, location and presence server to their clients.

The user agent is divided in two basic components: the (UAC) User Agent Client from which a call is originated, and the (UAS) User Agent Server, that receives the call. The SIP peer-to-peer architecture presented in figure 1.3 has the benefit of scalability and reliability, as there are no single point of failure. In this architecture, each peer is a UAC and UAS at the same time.

SIP servers execute name resolution and user location. Therefore, a SIP client/server topology is mostly employed because of its facility to add and route clients. SIP servers are also divided into the following components:

- Proxy: contains both UAC and UAS functions: forward call requests to other servers.
- Redirect server: replies to clients to contact an alternative server. It is mainly used to exploit some voice features. It does not initiate sessions.
- Location server: sends replies to clients with the actual redirect and proxy location. Used to allow the growth of an intelligent network.

SIP uses requests or methods for communication between peers and server. These methods are included in the next section.

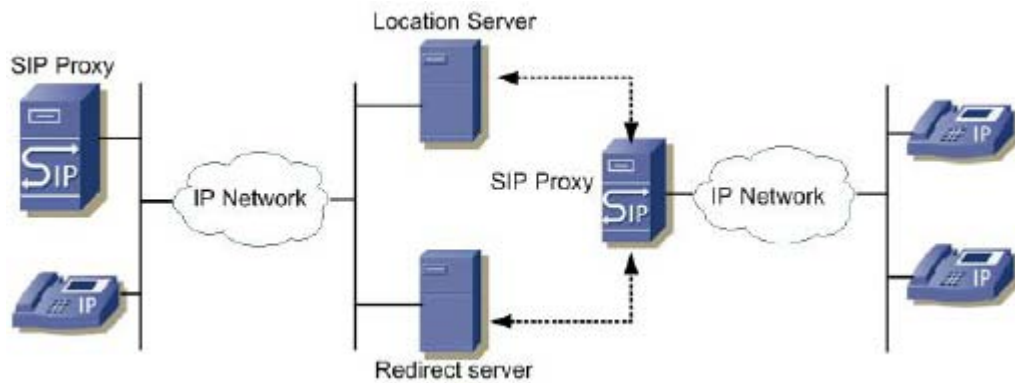


Fig. 1.3 : SIP Architecture.

1.2.2 SIP Methods

According to SIP's RFC standard [3], there are actually fourteen request types. They are used to establish signaling control between the elements.

INVITE: used to "invite" a user.

ACK: used to guarantee a reliable message exchange on the INVITE method.

BYE: used to teardown an established media session or to refuse a new call.

CANCEL: terminates a previous request.

OPTIONS: solicits information about a user's capability.

REGISTER: used to register a client in a SIP Registrar.

INFO: used to carry DTMF signals over an open RTP session.

SUBSCRIBE: requests the current state for a remote node.

MESSAGE: defines instant messaging capabilities between nodes in real-time.

NOTIFY: used to notify a node that an event which has been requested by an earlier SUBSCRIBE method has changed.

PRACK: has the same function as the ACK method but for provisional responses.

PUBLISH: used to publish presence state information.

REFER: used to enable many applications, such as call transfer.

UPDATE: allows the user to update parameters of a media session such as the voice codecs.

1.2.3 SIP Call Example

In the example of figure 1.4, two geographically distinct SIP Proxies are considered to mediate call signaling between Alice and Bob.

Alice initiates a call to Bob by sending an INVITE request to her proxy server. That INVITE contains a Session Description Protocol (SDP) which is used to negotiate media details between clients, such as the proposed voice codec, UDP port, etc. Assuming that Bob is available to talk to Alice, he will send back to his proxy server a "180 Ringing" and a "200 OK" message. The "200

OK " message contains Bob's media preferences in the SDP format. Once the ACK message is received, the media exchange starts through the RTP/RTCP ports previously negotiated. Notice that all the signaling was transported through one port in a simple text format (as this call does not use any encryption protocol such as IPSec¹ or SRTP²).

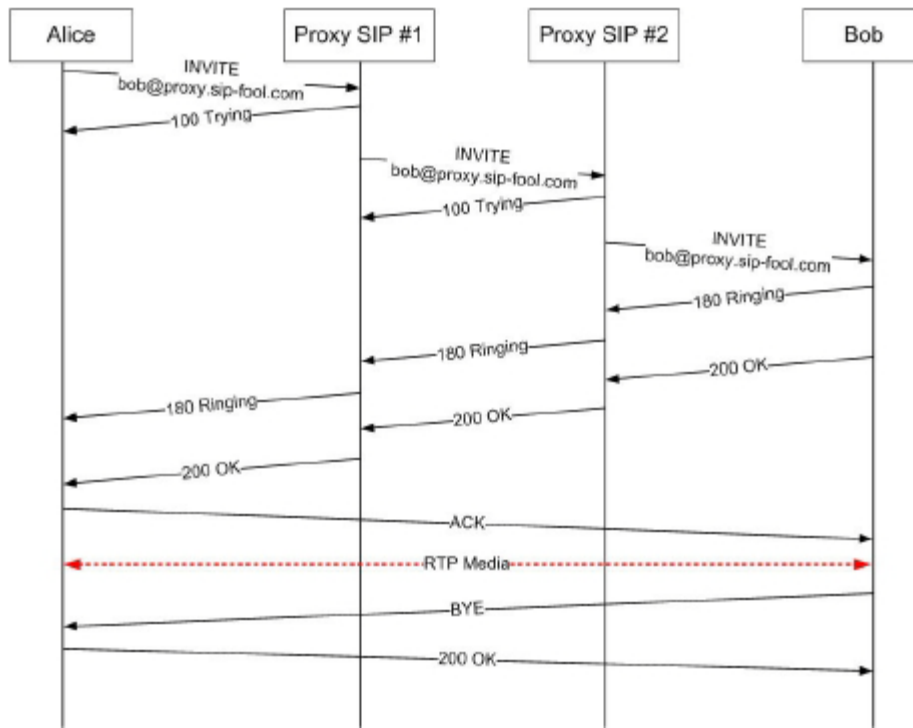


Fig. 1.4 : SIP call example.

1.3 SIP: Different Vulnerabilities

Ways to improve security of voice over IP networks is a very large subject. The list of possibilities to break a network security is tremendous. Nevertheless, it is always possible to classify those vulnerabilities into three major areas: integrity, availability and confidentiality. Intuitively, lack of integrity on communications results on illicit user's personification. Availability attacks are focused on denial of services or equipment crashes and confidentiality problems results the attacker to eavesdrop packets from users.

A list of some vulnerabilities explored in this chapter is described in Figure 1.5. The very first four vulnerabilities can be evaluated as elementary attacks, due to the fact that they could

¹IPSec (Internet Protocol Security) is a set of recommendations (OSI layer 3) that uses certain algorithms allowing the transportation of privacy data along a network.

²Secure Real-time Transport Protocol (SRTP) adds an encryption, authentication and integrity methods to RTP (Real-time Transport Protocol). SRTP was proposed by Cisco and Ericsson and is standardized by IETF under the RFC 3711.

be considered as pre-requisites to achieve VoIP attacks. For the rest of the table, some of the vulnerabilities are associated to exposures of the SIP protocol, in a similar way that message replaying results in identity hijacking, for instance. However, other evasions are related to SIP programming errors, resulting in dysfunctional actions, and causing the attacker to break through the system and corrupting it.

Layer	Vulnerability	Details and application	Confidentiality	Integrity	Availability
Data Link	ARP Cache Poisoning	Can be just applied on LAN networks	x	x	x
	DNS Cache Poisoning	Can be applied on WAN networks	x	x	x
Application	VoIP Attacks				
	SIP teardown attacks (cancel and bye messages)	Can be applied on WAN networks. Depends on a precedent ARP or DNS poisoning attack.			x
	SIP registration hijacking	Can be applied on WAN networks. Depends on a precedent ARP or DNS poisoning attack.	x	x	x
	Eavesdropping (RTP redirection)	Can be applied on WAN networks. Depends on a precedent ARP or DNS poisoning attack.	x	x	
	SPIT (Spam over IP Telephony)	Can be applied on WAN networks		x	
	Breaking Nonce by rainbow tables	Can be applied on WAN networks. Depends on a precedent ARP or DNS poisoning attack.		x	
	Directory scanning	Can be applied on WAN networks		x	x
	Nonce variation determining	Can be applied on WAN networks		x	x
	SQL code Injection	Can be applied on WAN networks. Depends on a precedent ARP or DNS poisoning attack.		x	x
	SIP Fuzzing Denial of Service (DoS)	Can be applied on WAN networks			x

Fig. 1.5 : Some VoIP attacks and vulnerabilities.

1.3.1 Elementary Attacks

The elementary attacks described in this section constitute a sort of a list of needed requisites to obtain access to a client's network and then engage other attacks against a VoIP operation. However, they do not exploit security breaches of the SIP; they are just required attacks to execute a variety of VoIP attacks, such as forging identities, redirecting RTP flows, and so on.

1. Attacks based on ARP Cache Poisoning APR (ARP Poison Routing) is a technique based on the legitimate use of the ARP Protocol [2]. Its aim is to modify traffic termination allowing attackers to sniff data frames in the same broadcast domain. It consists of sending spoofed ARP messages over the LAN network to associate the attacker's MAC address with the IP address of another machine. Hence, as the ARP table in the LAN switch is changed, all traffic originally sent to Alice (see figure 1.7), will be forwarded to the attacker's machine. The intruder may decide whether to re-transmit the packets to Alice's machine (passive capture) or to modify the traffic before sending it (man-in-the-middle attacks).

Figure 1.6 represents the normal case, where ARP tables were not poisoned. IP addresses from Alice and Bob were successfully associated to the MAC address from the switch ports to which they are connected. Thus, the traffic passes just between these corresponding ports; no other switch interface can observe the traffic.

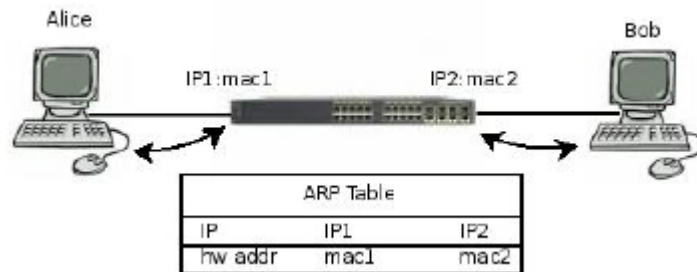


Fig. 1.6 : No ARP poisoning.

In the example of figure 1.7, the attacker replies to ARP messages sent by the switch with his own MAC address. This fact, induces the switch to associate Alice's IP address with the attacker's MAC address. Therefore, the original destination of the traffic (Alice) is transferred to the attacker.

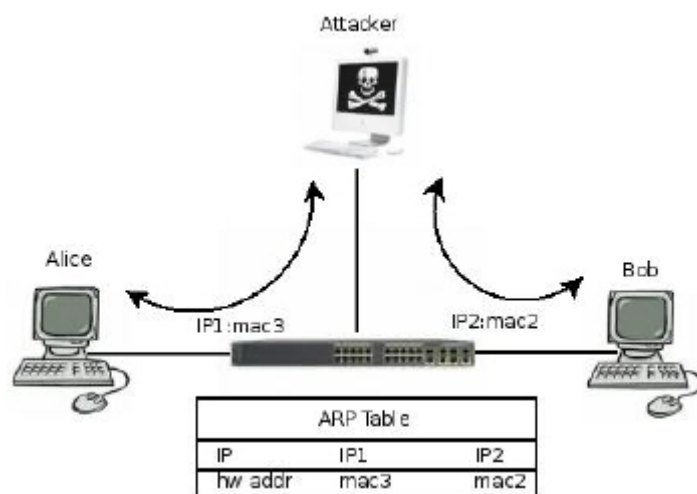


Fig. 1.7 : ARP poisoning in action.

2. DNS Cache Poisoning. There are numerous ways to poison a DNS cache. All of them conduct to the same objective: to associate an illegitimate IP address with a certain domain. Some variants of this attack will be presented below. The last method was studied and implemented during the internship at Telecom-Bretagne and is based on a prediction transaction ID number. The reasons for such choice will be further discussed in section 2.2.2.

(a) Redirecting name servers: attacker's domain to the target's domain

This attack is initiated when Alice sends a DNS query to the server, asking the IP address of attacker.com (see figure 1.8). To generate this first query, for example, Alice may have received an email to visit the attacker's web site. If the DNS server hasn't already cached this IP address, it will ask on the Internet who knows the authoritative nameserver responsible for attacker.com. At this point, the attacker's DNS server will reply this query with a redirection to the target domain (proxy.sip-fool.com) with an entry pointing to its own IP address. The attacked DNS server, after receiving this response, will add this new entry on its table that associates the attacker's IP address to proxy.sip-fool.com. Briefly speaking, it consists of changing the nameserver of the attacker's domain to the name server of the victim and then adding an additional A-record (Additional-record) pointing to the attacker's IP address.

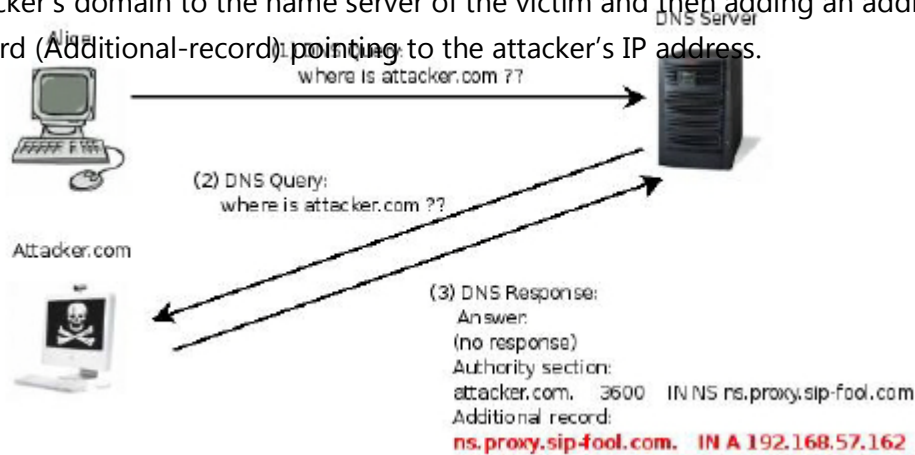


Fig. 1.8 : DNS poisoning: attacker's domain to the target's domain.

(b) Redirecting nameservers: target's domain to attacker's domain

This attack is very similar to the previous item. The difference is that the attacker redirects the target domain to his own domain and adds an additional A-record pointing to his IP address (see figure 1.9). This method is more difficult to implement because one needs to intercept the real response from the real domain and forge a fake message with his own IP address.

(c) Predicting DNS queries

For each DNS query, a random number is generated to identify this query and to correlate it to a specific response. This number must be unique for the tuple (query, reply). A DNS server will accept the first reply with the same ID expected that reaches it. It means that in the presence of two identical replies, the server will only take into account the first packet that arrives on it and thus inserting a new cache entry with the supplied IP address. However, if a reply with a different transaction ID is received, this packet will be automatically discarded.

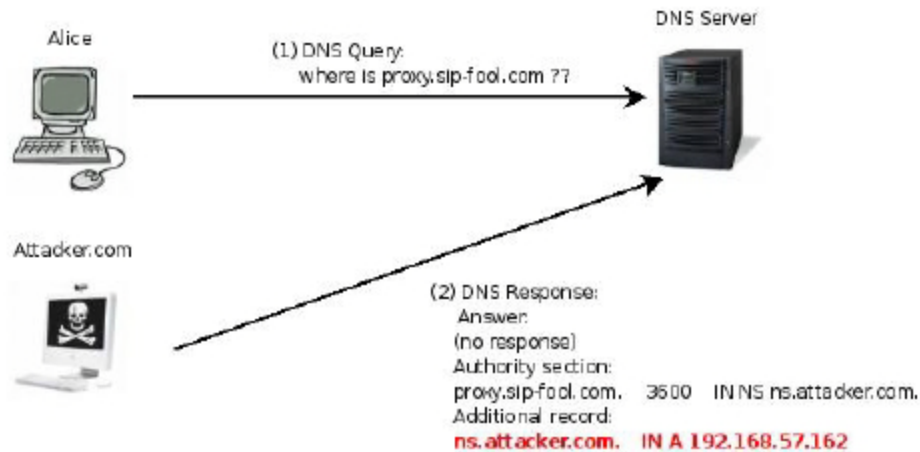


Fig. 1.9 : DNS poisoning: target's domain to attacker's domain.

Hence, if someone is able to predict the next generated value, he can forge faked replies pretending to be a legitimate DNS server and send those messages to the target DNS before the real answer arrives. As a result, the attacked DNS will include on its cache a fake IP address pointing to the requested domain. This information is cached in a DNS entry table and remains on it for a while, until the expiration of the value contained in the field time to live (TTL) present on every DNS response. This poisoning method was firstly described by A. Klein on [7] who details how those IDs can be predicted to generate fake DNS replies and poisoning vulnerable servers.

Concisely, this prevision mechanism works when the actual transaction ID (received by the attacker in some DNS query) has its Least Significant Bit (LSB) equal to 0. Afterwards, according to the technique described by A. Klein, an algorithm is used to predict 10 possible transaction IDs. It means that the next DNS query will be one of those predicted values. Thereby, when the DNS server sends its next query, the attacker prepares 10 DNS responses and sends all of them as quickly as possible, to avoid that the real DNS response be considered.

One possible way to force the DNS server to generate consecutive DNS queries is based on CNAMEs records³. For each CNAME reply, the attacked DNS will send a new query with a new transaction ID. This process is repeated until the attacker can find a transaction ID that can be predicted.

The diagram of this attack is depicted in figure 1.10. Alice asks for the resolution of attacker.com (message 1). As the DNS server could not solve this query because there is no entry related to this domain in its cache, it will send one new query directly to the attacker's nameserver (message 2). The attacker receives this query and verifies if the

³A CNAME record or canonical name record is a DNS response pointing to another domain. Generally it is used to provide many entries related to different applications running on the same IP address. For example, the domains ftp.company.com and http.company.com run on 192.168.60.70

Least Significant Bit (LSB) of the transaction ID is equal to 0. In this case 10 possible predictions are generated. They will be used to create 10 DNS replies. If not, the attacker sends one DNS CNAME response pointing attacker.com to www1.attacker.com (message 3). This will cause the DNS server send to the attacker a new query asking for the location of www1.attacker.com (message 4). This process is repeated until the LSB from the query receive is 0 (transaction ID is 81fc) (message 6). Consequently the attacker sends a last CNAME response pointing to the target domain proxy.sip-fool.com (message 7) and also calculates 10 possible transaction IDs (40fe c0fe 2032 2007 6032 6007 a032 a007 e032 e007). As the DNS server does not have the entry for proxy.sip-fool.com (or it expired), it will then send a query directly to the target's nameserver (message 8). At this point, the attacker sends responses as quickly as possible with all the predicted transaction IDs (those results are ordered in such a way that the first two values have a greater likelihood than the others to be exact). If succeeded, one fake entry that associates proxy.sip-fool.com with the attacker's IP address will be created (message 10). In this example, the cache was poisoned after receiving message 9. Note that the real answer from the real name server was just received at the end of the flow and by default, DNS servers discard duplicated messages. Alternatively, this attack can be also understood as a "message flow routine" (figure 1.11). This message flow graph was used as base to implement one algorithm that executes this attack, during the internship. Refers to section 2.2.2 to find the simulation results.

1.3.2 SIP Attacks by Special Crafted Messages

SIP Fuzzing Denial of Service DoS

This vulnerability was extensively exploited in PROTOS Test-Suite c07-sip [11]. Basically, PROTOS sends illegal characters, buffer overflows or malformed fields in SIP INVITE messages. The idea is to verify if softphones, IP phones or even SIP Proxies are immune to these kinds of attacks. Section 2.2.5 presents an experiment using PROTOS to provoke a crash of a specific softphone.

1.3.3 SIP Specific Vulnerabilities

The attacks presented in this section are based on security flaws in the SIP protocol. Some of them need pre-requisites (i.e., one elementary attack from section 1.3.1); others are successfully applied by their own.

1. SIP Registration Hijacking. This attack is mostly employed to make someone else believe that the attacker is a legitimate user. Thereby one can sniff SIP signaling and extract essential fields used to identify users or to distinguish calls. Those fields could be the Call-ID and the user's SIP phone user: URI (Uniform Resource Identifiers). Some attacks using those fields were described in sections 2.2.3 and 2.2.4.

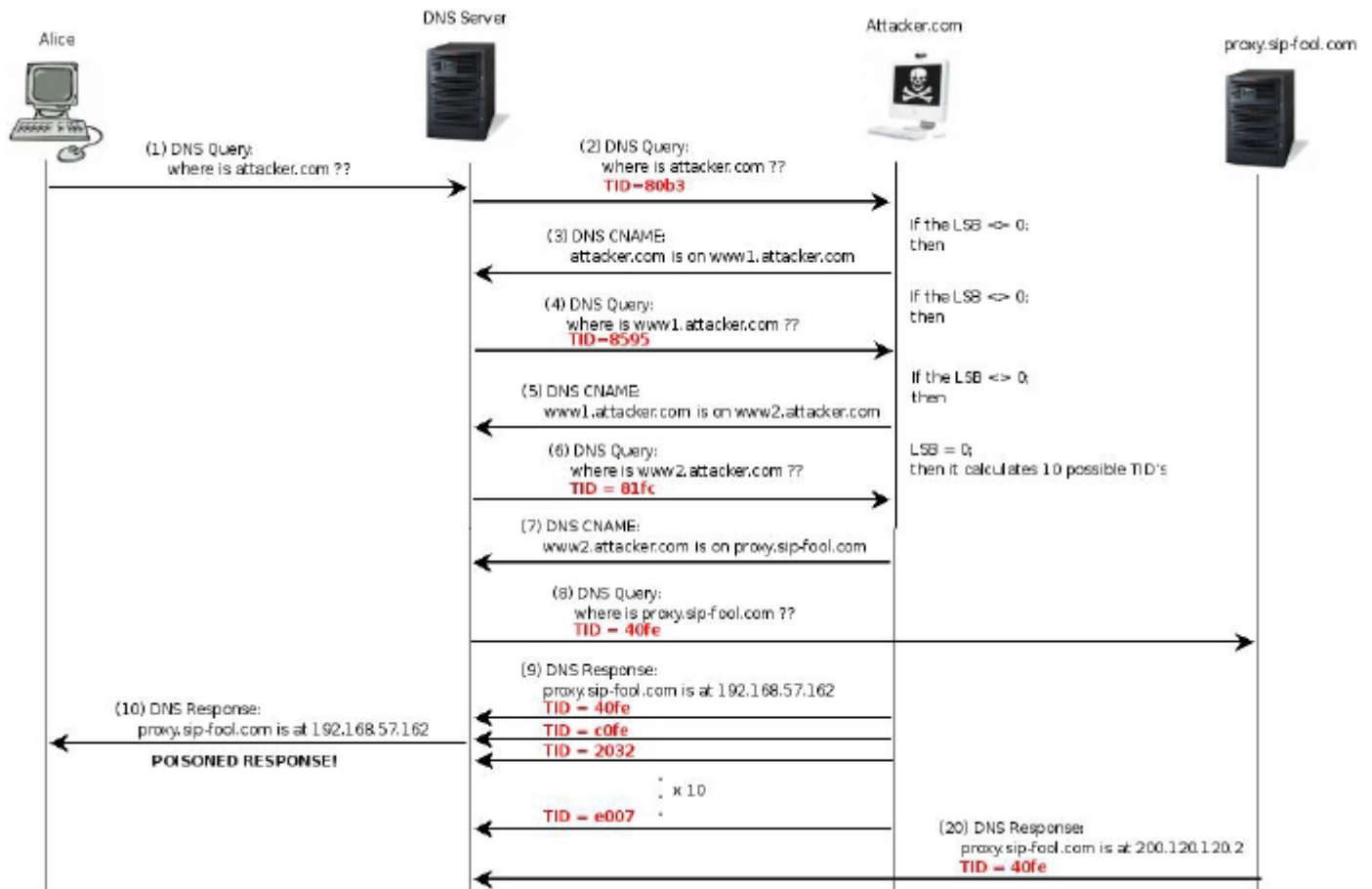


Fig. 1.10 : DNS poisoning by prediction the transaction ID.

To execute this attack, one has to intercept SIP packets. There are two alternatives for that. The first is to place the attacker in the LAN network and perform an ARP poisoning attack (described in section 1.3.1). The other alternative is to use a DNS poisoning attack. This last technique is more efficient, because generally, the attacker is located elsewhere.

2. SIP Teardown Attacks. SIP teardown messages use BYE and CANCEL messages. The attacker may use those messages to terminate calls, causing a denial of service to the legitimate user.

What is important with this attack is that depending of the softphone client, capturing only the Call-ID is not sufficient to teardown the call. Thus, the attacker has to use also the user's URI, destination UDP port, etc to force calls to terminate.

3. Eavesdropping. The aim of this attack is to redirect or to inject illegal RTP flows in normal conversations. It is employed mostly as a man-in-the-middle attack and there are numerous tools that exploit this failure [11, 12].

To perform this attack, similar to SIP Hijacking, the attacker needs to sniff the network (through a LAN connectivity) or to perform a DNS poisoning attack (intercepting packets

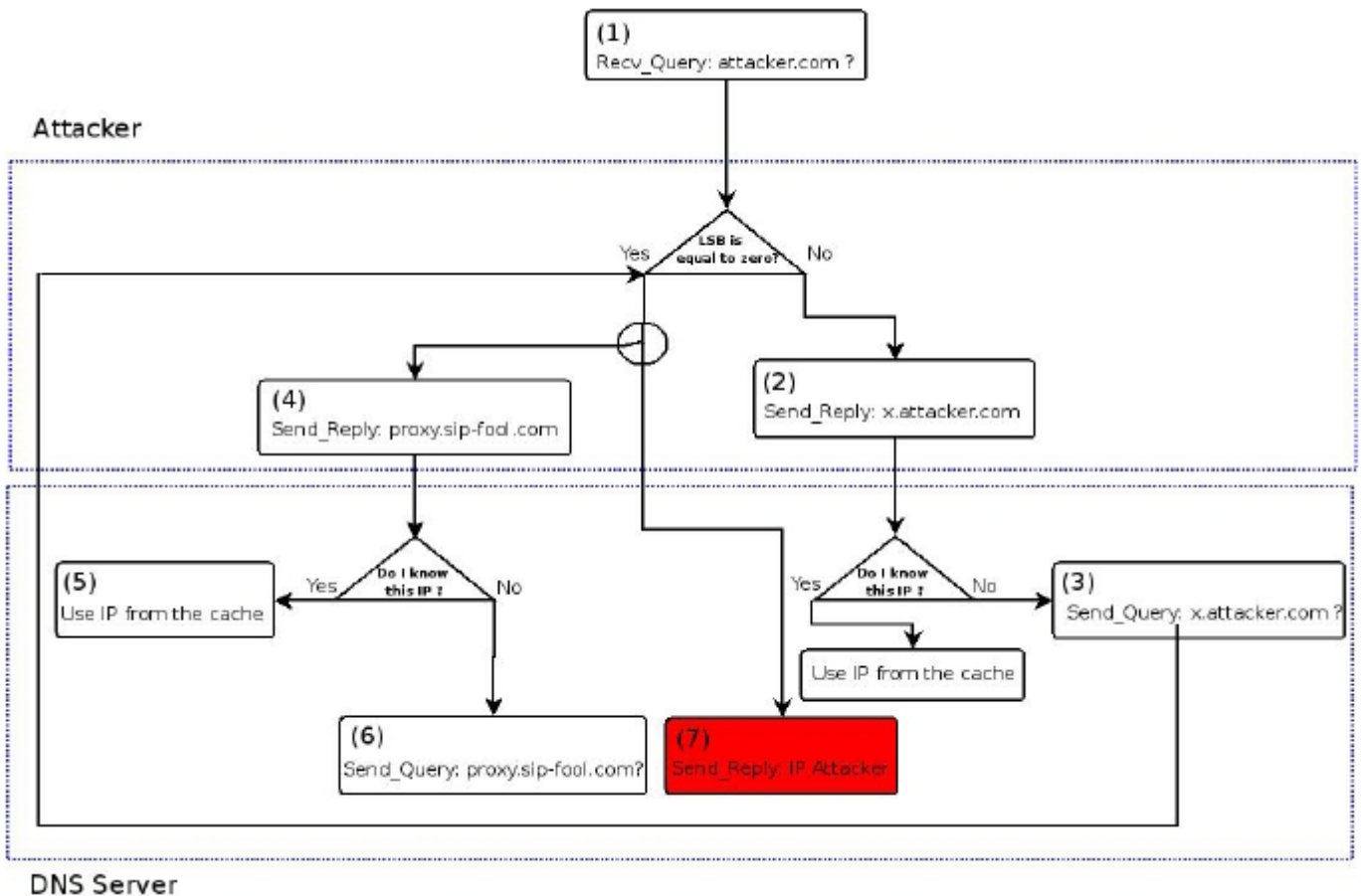


Fig. 1.11 : DNS poisoning by transaction ID prediction.

on the Internet).

4. Breaking Challenge Responses by Using Rainbow Tables. The SIP user's authentication described in RFC 3261 uses a challenge/response method (section 1.2.2. For each demand for authentication, SIP Proxies generate one credential (nonce number). That random generated number together, along with other parameters such as Alice's URI, SIP method used on authentication, realm, username and password, are hashed by an MD-5 function to compose the challenge response.

The idea is to use a method called rainbow tables [18] to crack the challenge response. However, it is necessary to generate the tables beforehand, which are formed by finite sequences of successive plaintexts that are passed through hash functions and then through reduction functions (producing other plaintexts) and again through hash functions. The article [18] provides more detailed information about the rainbow tables schema.

5. Directory Scanning. The idea behind this attack is to gather valid user SIP phone identification (URI) directly from the VoIP carrier operator. As described in [1], this vulnerability is based on different SIP responses from valid/invalid users when the server is asked for a REG-

ISTER or INVITE. After collecting valid URIs, the attacker can perform other elementary attacks such as a brute force or dictionary attacks to find the user's password.

6. Nonce Variation. This attack was described in [1] as a security weakness during the "nonce" generation of the SER SIP server [35]. This SIP server generates the same nonce if two requests for authentication arrive almost together. In such conditions, replay attacks can be executed.

1.4 Protection and Detection Mechanisms

1.4.1 Basic counter measures

There is a set of basic counter-measures that could be employed on a normal SIP network. They are based on basic security rules that could avoid many inconveniences with intruders.

- Security on the physical infrastructure (switches, routers, VoIP gateways, etc). Ideally, voice and data traffic should be segregated into different VLANs to enforce the security on the network. Therefore, even if one of those VLANs is exposed to an attack, the operation of the other subnetwork will not be corrupted.
- System and user authentication. To centralize access management of each network element and providing AAA (Authentication, Authorization and Accounting). Systems administrators can use RADIUS networking [30] protocol or its successor, DIAMETER [31] to provide better AAA. Using those methods, we can guarantee that users are authenticated in a trusted server.
- Voice and signaling confidentiality. User authentication and encryption, such as IPSec and SRTP, to guarantee confidentiality of flows.

1.4.2 Intrusion Detection Mechanisms

Classical Approaches

As most of the described attacks need another elementary attack to be executed, a detection system able to detect those anomalies and producing efficient alarms is a good approach to maximize the security level of the network. Those detection systems may be translated by IDSs probes (Intrusion Detection Systems) that are localized in strategic points in the network. When an illegal situation is detected, an alarm is generated and sent to the network administrator.

In the other hand, one of the issues faced by IDSs is the problem of inappropriately generating alarms. If an alarm was generated due to a false evasion, then it is called a false positive. If the attack is a success and no alarm was generated, then it is called a false negative. What causes those malfunctions are generally inefficient rules or a rule that does not correspond to an evasion.

There are two classic approaches that helps avoiding massive alert generation or no single alarm intrusion. They are described in the literature as behavior-based approach and scenario-based approach. By contrast, both of them have pros and counters. The table 1.12 depicts a comparison between these two approaches.

Approaches	Definitions	Pros	Counts
Behavior-based	Actions based on abnormal user behaviors. The long term behavior from the user is compared with his actual behavior.	Detection of new attacks	Not accurate in diagnosis
			False positives generation. Because changing the behavior does not reflects in one attack.
			Alerts not accurate. They need a deeper analysis by the system administrator.
Scenario-based	Actions based on similar attacks stored in a database.	Accurate diagnosis	Attacks/signatures not included in the database are not detected.
			Each attack must be written in details, to prevent mistakes in analysis.
			False negatives generation (undetected attacks). Mainly due to variation in attacks not included in the database.

Fig. 1.12 : Behavior and scenario-based approaches.

Note that both of the approaches are complementary in the sense that the weakness of one is a strength employed by the other. Besides that, there are some scenarios in which traditional IDS probes do not produce desired results when trying to detect some VoIP attacks. This is mainly due to the fact that some of those attacks (SIP identity hijacking for example) have a regular flow exchange not distinguishable from normal utilization.

Some good articles [1, 13, 14] introduce different approaches specifically to detect VoIP attacks in an efficiently way. But until the present day, there is no practical implementation of a VoIP-based IDS. This could penalize the whole applicability of my work in detecting and reacting to those anomalies. However, to perform some categories of VoIP attacks, the attacker must listen the traffic from legitimate users. One could use this argument to detect those signature-based attacks and then correlate alerts to finally react to the global intrusion attempt (i.e., to block ports or the attacker's IP address). The approach used by CRIM (Cooperation and Recognition of Malevolent Intentions) [15] is extremely valid on this scenario.

Different Intrusion Detection Systems

Table 1.13 shows a comparison between some of most popular intrusion detection systems. Snort IDS was used during my experiments. The reason for this choice is based in the following argu-

ments:

- This IDS is easily to configure and it is possible to download optimized rules to improve the detection of new attacks;
- Several snort plugins are available to execute special tasks. As an example, some plugins can convert the alarm file generated by Snort to a IDMEF (Intrusion Detection Message Exchange Format) data file [18], readable by CRIM. Other plugins are used to implement counter-measures to attacks based on Snort rules: Snort Inline⁴[16] and SnortSams⁵[17].

Tool	Description	Pros	Counts
Snort	An open source IDS largely used and recommended.	Rules can be created in two ways: 1) analyzing packet by packet; 2) analyzing TCP streaming, to detect floodings, etc.	Difficulty to detect VoIP based signatures, due to a "normal" behavior of packets.
		Rules for new attacks available on Internet	
Ossec Hids	An Open Source Host-based Intrusion Detection System	Creates and analyses log files to perform integrity checking, rootkit detection, time-based alerting and active response.	Difficulty to detect specific VoIP based signatures, such as hijacking or eavesdropping.
Bro IDS	An Open Source Host-based Intrusion Detection System	It has some special features to improve active response to attacks, such as the ability to terminate TCP connections and an ability to update a firewall access list to block packets/hosts.	Difficulty to detect specific VoIP based signatures, such as hijacking or eavesdropping.

Fig. 1.13 : IDS comparison.

1.5 Reaction Techniques

The main goal of reaction techniques is to implement best fitted counter-measures for attacks. Different reaction approaches can be used. Some examples are described below:

1. Software update. The administrator is conducted to apply patches that solves the specific vulnerability that might cause the attack.
2. The system administrator inserts new firewall policies. When the attacker is located in the outsider's network, his action can be blocked by restricting certain traffic on certain UDP/TCP port.

⁴Snort Inline: Snort plugin that configures IP tables whether the packet should be dropped, rejected, modified, or allowed to pass based on a snort rule.

⁵SnortSam: plugin allows for automated blocking of IP addresses on several firewalls available on the market.

3. Using Intrusion Detection Systems (IDS) with plugins to execute countermeasures. Attacks detection done by Snort IDS are based on signatures that best describes how attacks are executed. Those signatures (or traces) are used to differentiate normal user actions from illegitimate behaviors. Normally traces can be identified either in each packet launched by the attacker or in a dialog between two clients. In this last situation, if one client sends continuously ping packets to other client, this action could be identified as an attempt of intrusion, because replying pings could reveal some information about the network. When Snort recognizes an attack signature (activated by a rule), one alarm is generated and stored in a directory. At the same time, Snort triggers the reaction plugin that request the firewall to insert a new rule and blocking the intrusion.
4. Use the anti correlation approach used by CRIM. The approach proposed by [34] is based in a reaction library with different kinds of countermeasures. As the administrator cannot chose directly the most efficient countermeasure for one specific attack, the correlation and reaction model proposed by CRIM uses the alarms generated by IDSs as a base to chose the appropriated counter-measure. More specifically, CRIM uses the technique of alarm correlation to identify the attack scenario and the anti-correlation technique to propose counter-measures. Correlating attacks means to colligate different alarms previously generated to produce a model that describes the attack scenario. Consequently, the anti correlation technique tries to find the best reaction (with different counter-measures) that must be applied to stop the attack. For both techiques, CRIM uses a formalism called Lambda language. Refers to section 1.6 to find more information about CRIM and the Lambda language.

1.6 CRIM as a Correlation and Reaction Model

CRIM is a tool capable of correlating IDS-based alarms, analyze if they could become an attack and in such event, propose actions to react against the intrusion. As discussed before, the Behavior and Scenario models generates false positives and negatives alarms, respectively. One of the objectives of CRIM is to complement both approaches to minimize false negatives and mostly eliminate false positives.

To accomplish its goal, the attacker executes successfully steps called elementary attacks. Those atomic attacks are detected by IDS probes, hence generating an alarm for each action. Afterwards, CRIM combines those alarms in a cooperation schema to obtain a precise network diagnosis. To globally understand how this tool works, each one of its six modules is explained in details:

1. Management of alarms. Collects all the alerts generated by the probes and saves them in a database. Those alarms are described in the IDMEF standard. The main goal of using this class hierarchy data format is to facilitate communication between IDSs and the CRIM's reaction model.

2. Regrouping of alarms. This module determines if there is a link between the detected alarms and the alarms stored in a database. This function can reduce the number of alarms which will be analyzed by the next module.
3. Fusion of alarms. Creates a new alarm (global alarm) that synthesizes all the information included in each alarm collected by IDS probes.
4. Correlation of alarms. Analysis of the global alarm in such a way that it is possible to find the attacker's strategy by correlating his actions. This function returns a set of possible attacks for each global alarm. The correlation used by this module is based on a logical description, using the Lambda language [34]. This language defines four attributes to correctly identify an attack:
 - Pre condition: condition that must be fulfilled for the accomplishment of the attack. Pre conditions are described using special predicates that define actions taken by the attacker when the attack is a success.
 - Post condition: specify all the symptoms for this attack if successfully achieved. Post conditions are written using predicates to describe what the attacker knows or controls after the attack.
 - Detection: correspondence between one Lambda model and one alarm. This field is used to create a connection between pre and post conditions with the alert attributes, represented in the IDMEF data format.
 - Verification: this field is used to verify if the attack was accomplished. This verification will not be used in my experiments, because I suppose that the implemented attacks are successfully executed.
5. Intention recognition. The intention recognition function is the module responsible to determine the intentions of the attacker. For example, the attacker's objective is to provoke the crash of one softphone by sending malformed SIP packets. CRIM uses logical conditions to formulate intention recognition. The idea is to extend the same formalism adopted to correlate alarms using also the Lambda language. We can say that one attack (A) and the intrusion objective (I) are correlated if $\text{post}(A)$ and $\text{obj}(I)$ have at least one predicate in common, thus they can be unified.

The correlation process creates unified alarms called candidate plans. Each of these plans are verified by the intention recognition module if they could conduct to an intrusion objective. In case positive we can say that this specific candidate plan is a success attack scenario and this scenario is sent to the reaction function (item 6). Otherwise, we consider that this candidate plan did not accomplish its goal yet. In this case, the intention recognition module can extrapolates the next possible actions that the attacker will perform, and generate different scenarios of intrusion. Moreover, this module associates a coefficient of correlation to each step of the scenario. Thus, classifying by order each one of those potential attack scenarios.

6. Reaction based on anti-correlation. Determines the best fit counter-measures that could stop the attack progression. Generally those counter measures try to break the chain between one or two successive elementary attacks. CRIM uses a library also described in terms of the Lambda language to block the global attack.

The diagram 1.14 shows the CRIM architecture with each one of its modules. The relationship between these modules and the database is also included.

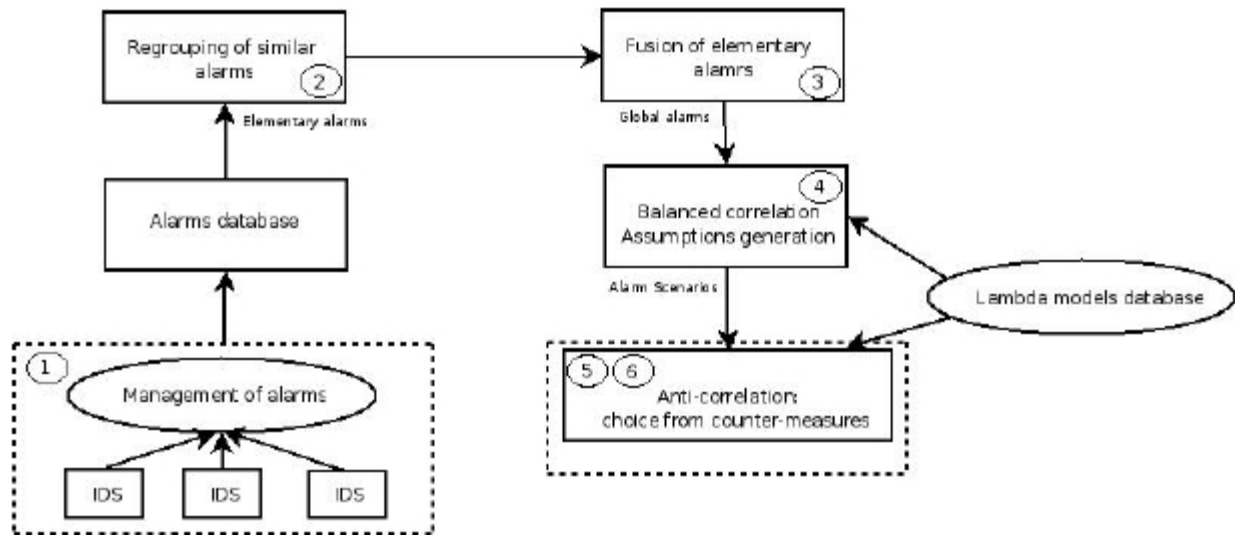


Fig. 1.14 : CRIM architecture [15].

1.7 Conclusion

In this chapter I introduce several VoIP attacks and vulnerabilities related to the SIP protocol. Secondly, I studied two approaches to avoid VoIP-based attacks. The first approach establish security policies to users and to the network infrastructure to prevent attacks. The other approach is a correlation and reaction model. CRIM is used to correlate alerts and thus identifying the attack scenario to which specific counter-measures are proposed to block the attack plan.

The next chapter includes different experiments that were proposed to test the correlation and reaction model used by CRIM as well as to examine another reaction model, proposed by Snort and SnortSam plugin.

Chapter 2

Experimentation

2.1 Introduction

This chapter introduces different experiments conducted entirely on the Security Lab/Telecom Bretagne. The experimentation is divided in 3 parts. Firstly, in (section [2.2](#)) I implement some of the attacks previously described in section [1.3](#) and verify the vulnerabilities caused by them. Secondly, section [2.3](#) presents an attack scenario, (chosen from the implemented attacks in the precedent section) to which I will test two reaction models: the correlation and reaction model proposed by CRIM (section [2.4.3](#)) and the second reaction model using SnortSam plugin (section [2.4.3](#)).

2.2 Implemented Attacks

ARP and DNS poisoning were called as basic/elementary attacks because they are needed to accomplish a more global attack. By using them, the attacker can listen or redirect IP traffic to a malicious machine. Afterwards, a VoIP attack can be executed against clients or the SIP proxy server. However there is another category of VoIP attacks that does not depend of ARP/DNS poisoning attacks. They explore some SIP vulnerabilities to achieve their goals: DoS against users, system crashes, etc. To demonstrate this last case, a SIP fuzzing attack was implemented and will be showed at the end of this session.

As the physical environment was reduced, more than one function was supported by each PC. The table [2.3](#) shows the functions/software installed and running on each computer.

The idea with these experiments was to develop six physical entities: (1) Alice and (2) Bob as SIP softphones; (3) the SIP proxy on which legitimate clients are registered; (4) the attacker, (5) the IDS Snort probe and finally (6) CRIM as a correlation and reaction tool.

The tool SIPp traffic generator tool [[22](#)] was used to create and modify SIP packets. In fact, originally it is used as a traffic generator for the SIP protocol, but in this experiment, I created useful scripts to execute SIP DoS attacks and identity hijacking. Its role will be further detailed in the next sessions.

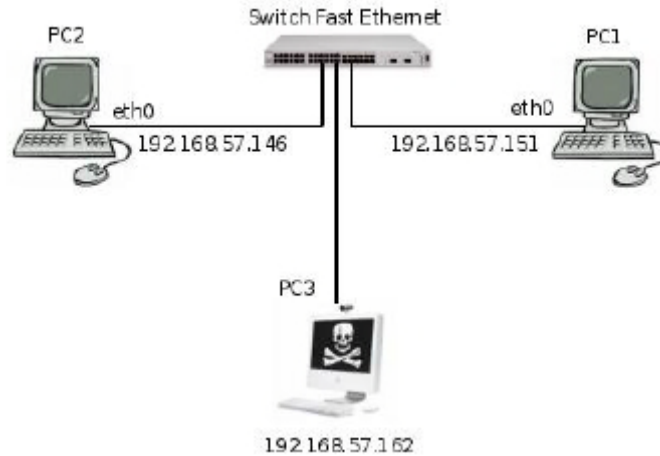


Fig. 2.1 : Architecture 1 - LAN topology.

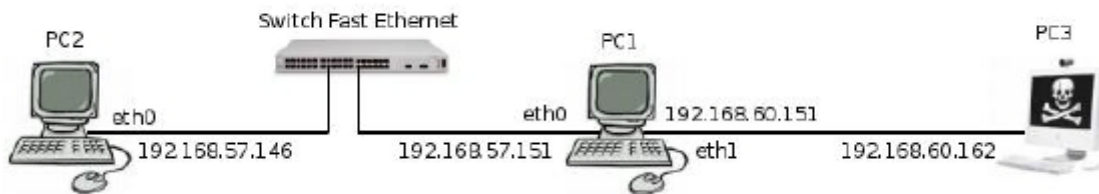


Fig. 2.2 : Architecture 2 - Internet topology.

The main objective of the PROTOS test case [23] in this scenario is to generate SIP packet fuzzing. PROTOS injects invalid characters on SIP INVITE messages to verify if the target is vulnerable to malformed packets that might either crash the software or allow the execution of arbitrary code as described in [10].

All the experimentations included in this Chapter were conducted under two scenario-based situations: figure 2.1 depicts the case where the attacker is located in the LAN (same network as Alice, Bob, SIP proxy) and he is able to sniff the target user, thus provoking several VoIP attacks. Figure 2.2 shows the attacker can be in a public domain (Internet) that uses a DNS poison attack to redirect the target flow to itself and then execute VoIP attacks as well.

2.2.1 ARP Cache Poisoning and MAC Spoofing

This attack can be reproduced with the first topology: attacker in the LAN - figure 2.1. The first constraint to this attack regards its applicability. This attack can only be applied in the same LAN segment where the target is localized. This resides on the fact that ARP protocol is used to associate layer 3 protocols (i.e., IP address) to the layer 2 MAC address, enabling communication between peers in the same LAN segment. Section 1.3.1 item 1 gives more detailed information

	Acting as	Softwares
PC1	Alice	Twinkle Softphone
		Ekiga Softphone
		X-Lite Softphone
	Firewall	IP Tables
	IDS probe	Snort v2.2.0
	Execute Iptables commands based on Snort alarms	SnortSam plugin
	Convert Snort alarms to IDMEF data format	1.2.6alpha2.2.0
DNS Server	BIND 9.4.1	
PC2	Bob	Twinkle Softphone
		Ekiga Softphone
		X-Lite Softphone
	Proxy SIP to register Alice and Bob	Asterisk v1.4.18
	Correlation and reaction model	CRIM v3.0
PC3	Attacker – Identity hijacking, DoS	SIPp v3.1
	Attacker – SIP Fuzzing	PROTOS SIP test case
	Attacker – DNS poisoning	DNS Poisoning
	Attacker – ARP poisoning	Cain & Abel v4.9.14

Fig. 2.3 : Tools used in the experiments.

about ARP protocol and associated attacks.

There is a huge quantity of software available on the Internet to produce such ARP poisoning attacks. The chosen tool used on this experiment was Cain & Abel [19]. This tool explores the APR technique (ARP Poison Routing) to achieve different results, such as password recovery, cracking encrypted passwords using dictionary, brute-force and cryptanalysis attacks, recording VoIP conversations, etc.

Nevertheless, this tool was used only to achieve our objective, which is to poison the ARP table on switches. As a result of this attack, the intruder is able to listen to all the traffic from the victim and then execute some VoIP attack against it.

The ARP poisoning tool - Cain & Abel - was running on PC3 (see figure 2.4). Its configuration is simple: it is sufficient to provide the target IP address at which the attacker wants to listen to the traffic. In the configuration experimented in figure 2.4, the attacker redirects and sniffs the traffic originally sent to Alice (PC1 with IP address 192.168.57.151) by sending ARP replies over the LAN network to associate the attacker's MAC address to Alice's IP address.

Afterwards, all traffic meant to Alice is forwarded mistakenly to the attacker. Note that this tool does not allow changing the spoofed MAC address. It is possible to associate two IP addresses to the same MAC address; however, Ethernet switches may have malfunctions in the presence of two identical MAC addresses sharing the same IP address on the same LAN.

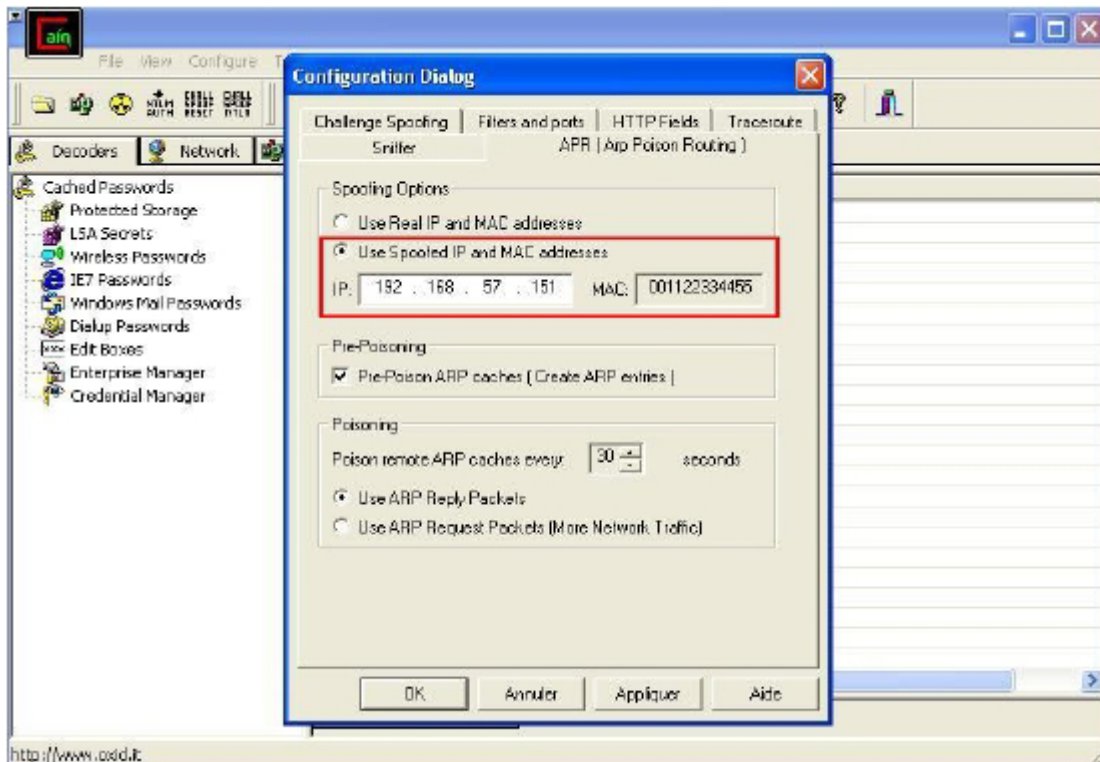


Fig. 2.4 : Cain & Abel configuration dialog.

2.2.2 DNS Cache Poisoning: Transaction ID prediction

There are several possibilities to accomplish a DNS cache poisoning attack as explained in section 1.3.1 item 2). The method chosen to be experimented, exploits a vulnerability on BIND9 servers with versions inferior to 9.5.0a5 [8]: transaction ID prediction. This attack was chosen because of its innovation, and because it can be explored on several DNS servers running on the Internet, as they use BIND9 as a DNS server solution. The document presented by A. Klein[7] shows one code written in Perl language to predict transaction ID numbers on BIND servers with versions under 9.5.0a5. Besides, a proof of concept (PoC) [21] written in Python language implements the same prediction algorithm and includes a basic DNS server.

As part of the work developed on the internship, the PoC was used as a base to develop a complete algorithm to explore this attack. The code was also written in Python and is able to interact with the target DNS server, creating CNAME replies, predicting the next transaction ID and finally forging DNS replies that causes the cache poisoning. Section 1.3.1, item 2c presents in details how this vulnerability is applied and also the reader can find a graph 1.11 with all the steps done by the attacker to poison the DNS server.

Conditions for a successful attack

Some constraints must be followed to maximize the likelihood of success for this attack. Some of those conditions are also defined in [7] and others were observed during the experimentation of our

algorithm:

- DNS Bind version inferior from 9.5.0a5;
- Cache entry for the target domain must be empty;
- The target DNS server must accept CNAME records [26];
- The 10 DNS forged responses must arrive sooner than the real response in order to guarantee effective poisoning. The prediction algorithm uses a Python script and the forged responses takes 3-4 milliseconds running on an Intel Core 2 CPU at 2.13 GHz. According to A. Klein, this delay could be reduced by implementing the code in a language that compiles to binary, such as C/C++.
- The UDP destination port used by the DNS must be static. Considering all the Bind versions when the process is started, one destination UDP port is randomly chosen and kept until the next restart.

Execution of the DNS poisoning attack

The scenario used to test this attack was the same described in figure 2.2 - Internet topology. The tools used in this test are showed in the table 2.5. In this experiment, the attacker is PC3, located in the outsider's network and the target DNS server is running on PC1. PC2 is the client that originally sends the first DNS query: attacker.com. This query can be transmitted in several ways, for example as an email with a link to that domain.

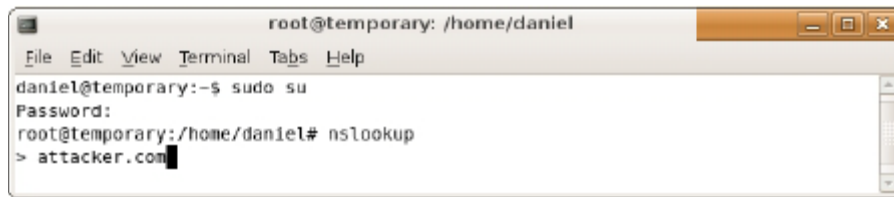
	Acting as	Softwares	Addresses
PC1	DNS Server	BIND 9.5.0a5	192.168.57.151 and 192.168.60.151
PC2	Bob	Linux Client	192.168.57.146
PC3	Attacker – DNS poisoning	DNS Poisoning	192.168.60.162

Fig. 2.5 : DNS poisoning experimentation.

The attack begins when the client (PC2) asks its DNS server to resolve attacker.com. To simulate this condition, the nslookup application was used on the client's machine:

This query is sent to PC1 running the vulnerable BIND9 DNS server. As this domain was not yet cached, the server will interrogate the authoritative server responsible for that domain: PC3. That causes our algorithm to verify on the received query if the next transaction ID can be predicted. If not, the algorithm will send back a CNAME record with the domain x.x.attacker.com and receive a new query demanding the attacker's server to solve x.x.attacker.com. This process

is repeated until the transaction ID can be predicted. The following screenshots show the message exchange between PC1 (target DNS server) and PC3 (attacker). What is important to verify is that



```

root@temporary: /home/daniel
File Edit View Terminal Tabs Help
daniel@temporary:~$ sudo su
Password:
root@temporary:/home/daniel# nslookup
> attacker.com

```

Fig. 2.6 : Nslookup configuration shell: client sends a DNS query attacker.com to the vulnerable server.

when PC1 (192.168.57.151) sends to PC3 (192.168.60.162) the query asking for proxy.sip-fool.com (figure 2.7), the very first attacker's response was able to predict the transaction ID number (7bf7) (figure 2.8).

No.	Time	Source	Destination	Protocol	Info.
4	5.002220	192.168.60.162	192.168.60.151	DNS	Standard query response A 192.168.60.162
5	5.037358	192.168.60.151	192.168.60.162	DNS	Standard query A attacker.com
6	5.039691	192.168.60.162	192.168.60.151	DNS	Standard query response CNAME x.attacker.com
7	5.039841	192.168.60.151	192.168.60.162	DNS	Standard query A x.attacker.com
8	5.041707	192.168.60.162	192.168.60.151	DNS	Standard query response CNAME x.x.attacker.com
9	5.099916	192.168.60.151	192.168.60.162	DNS	Standard query A x.x.attacker.com
10	5.118168	192.168.60.162	192.168.60.151	DNS	Standard query response CNAME proxy.sip-fool.com
11	5.118910	192.168.60.151	192.168.60.162	DNS	Standard query A proxy.sip-fool.com
12	5.148348	192.168.60.162	192.168.60.151	DNS	Standard query response A 192.168.60.162
13	5.169239	192.168.60.162	192.168.60.151	DNS	Standard query response A 192.168.60.162
14	5.169903	192.168.60.162	192.168.60.151	DNS	Standard query response A 192.168.60.162
15	5.170288	192.168.60.162	192.168.60.151	DNS	Standard query response A 192.168.60.162
16	5.170606	192.168.60.162	192.168.60.151	DNS	Standard query response A 192.168.60.162
17	5.170886	192.168.60.162	192.168.60.151	DNS	Standard query response A 192.168.60.162
18	5.171163	192.168.60.162	192.168.60.151	DNS	Standard query response A 192.168.60.162

▶ Internet Protocol, Src: 192.168.60.162 (192.168.60.162), Dst: 192.168.60.151 (192.168.60.151)
 ▶ User Datagram Protocol, Src Port: domain (53), Dst Port: 34757 (34757)
 ▼ Domain Name System [response]
 ↳ Request In: 111
 ↳ [Time: 0.028038000 seconds]
 ↳ Transaction ID: 0x7bf7
 ↳ Flags: 0x8410 (Standard query response, No error)

Fig. 2.7 : Wireshark capture: query from the vulnerable DNS server.

Finally this attack will result in the poisoning of the entry proxy.sip-fool.com with the attacker's IP address (192.168.60.162). From now on, every time a host interrogates this DNS server about proxy.sip-fool.com, the server will respond with a fake IP address (see figure 2.9). In this experiment, the TTL field was set up to 1 second, but in a real attack scenario, it is preferred to increase this value to be maintained as long as possible in the cache entry. Looking at the nslookup program started in the client's machine (PC2), the final result will be:

2.2.3 SIP Attack: Bye Message

This is one of the simplest attacks using Denial of Service that clearly exposes a vulnerability on the SIP protocol: to transport in clear text of the Call ID field. This field is used to uniquely identify each call attempt in one SIP proxy server. Both the server and the client have the ability to generate this number.

By sniffing SIP packets, it is possible to capture the Call ID used to establish a SIP call. The attacker can use this same value to forge a BYE message and tearing down that session. Basically,

No.	Time	Source	Destination	Protocol	Info.
4	0.002220	192.168.60.162	192.168.60.151	UDP	Standard query response A 192.168.60.162
5	5.037359	192.168.60.151	192.168.60.162	DNS	Standard query A attacker.com
6	5.039091	192.168.60.162	192.168.60.151	DNS	Standard query response CNAME x.attacker.com
7	5.039841	192.168.60.151	192.168.60.162	DNS	Standard query A x.attacker.com
8	5.041707	192.168.60.162	192.168.60.151	DNS	Standard query response CNAME x.x.attacker.com
9	5.099916	192.168.60.151	192.168.60.162	DNS	Standard query A x.x.attacker.com
10	5.118168	192.168.60.162	192.168.60.151	DNS	Standard query response CNAME proxy.sip-fool.com
11	5.118310	192.168.60.151	192.168.60.162	DNS	Standard query A proxy.sip-fool.com
12	5.148348	192.168.60.162	192.168.60.151	DNS	Standard query response A 192.168.60.162
13	5.168233	192.168.60.162	192.168.60.151	DNS	Standard query response A 192.168.60.162
14	5.168963	192.168.60.162	192.168.60.151	DNS	Standard query response A 192.168.60.162
15	5.170288	192.168.60.162	192.168.60.151	DNS	Standard query response A 192.168.60.162
16	5.170606	192.168.60.162	192.168.60.151	DNS	Standard query response A 192.168.60.162
17	5.170686	192.168.60.162	192.168.60.151	DNS	Standard query response A 192.168.60.162
18	5.171159	192.168.60.162	192.168.60.151	DNS	Standard query response A 192.168.60.162

▶ Internet Protocol, Src: 192.168.60.151 (192.168.60.151), Dst: 192.168.60.162 (192.168.60.162)
 ▶ User Datagram Protocol, Src Port: 54757 (54757), Dst Port: domain (53)
 Domain Name System (query)
 ↳ Response In: 22
 Transaction ID: 0x7bf7
 Flags: 0x0010 (Standard query)

Fig. 2.8 : Wireshark capture: response from the attacker (with the predicted transaction ID) to the vulnerable DNS server.

```

root@temporary: /home/daniel
File Edit View Terminal Tabs Help
daniel@temporary:~$ sudo su
Password:
root@temporary:~/home/daniel# nslookup
> attacker.com
Server:          192.168.57.151
Address:         192.168.57.151#53

Non-authoritative answer:
attacker.com     canonical name = x.attacker.com.
x.attacker.com  canonical name = x.x.attacker.com.
x.x.attacker.com canonical name = proxy.sip-fool.com.
Name:   proxy.sip-fool.com
Address: 192.168.60.162
>
  
```

Fig. 2.9 : Nslookup configuration shell: poisoned response from the attacked DNS server.

that was the procedure used in this experimentation to create this attack. Using an ARP or DNS poisoning, the attacker captures SIP packets, finds the actual Call ID and send SIP forged messages to the target.

Executing a SIP BYE attack

As mentioned before either an ARP poisoning or DNS poisoning needs firstly to be performed. This experiment adopted the topology of figure 2.1 - Architecture 1 - LAN topology and the used tools is presented in the figure 2.10.

At a first glance, Alice and Bob are initially registered on Asterisk SIP proxy. This SIP proxy acts as a classic PBX, receiving signal and media packets from one client and forwarding them to the other client. Therefore, in this communication two Call IDs are being used, one between

	Acting as	Softwares	Addresses
PC1	Alice	Ekiga Softphone	192.168.57.151
PC2	Bob	Ekiga Softphone	192.168.57.146
	Proxy SIP to register Alice and Bob	Asterisk v1.4.18	
PC3	Attacker – Identity hijacking, DoS	SIPp v3.1	192.168.60.162
	Attacker – ARP poisoning	Cain & Abel v4.9.14	

Fig. 2.10 : SIP BYE message experimentation.

Alice and Asterisk and another between Asterisk and Bob. As the attacker intends to drop down Alice, he will execute an ARP poisoning attack first, to associate his MAC address with Alice's IP address: 192.168.57.151. In the end, the intruder receives packets from Asterisk and forwards them to Alice. In this replay scenario, their Call ID is exposed and afterwards a BYE message is generated with the help of the SIPp tool.

The attacker finally sends this forged packet to Alice. Alice checks if the Call ID is the same one used to communicate with Asterisk and executes the bye routine, tearing down the session. Note that the field "from " includes the attacker's URI but Ekiga softphone does not use it to verify the integrity of this message.

This same experiment was performed with X-Lite and Twinkle softphones. With both of them, just re-injecting the Call-ID was not enough to bring down the connection. However, using the "branch " and "tag " identifiers from "via " and "from " fields and the source UDP port from the capture a new BYE message was forged and finally re-injected in one new BYE message, causing Alice to disconnect the call. As Bob is never warned about the disconnection, its session with Asterisk remains active until one sort of timer is exceeded, bringing down the connection also as RTP packets are no longer being received.

Conditions for a successful attack

During the experimentation, some remarks were noticed as important factors to achieve the success of this attack:

- One of the elementary attacks: ARP poisoning or DNS poisoning has to be performed first;
- All the fake BYE messages were executed using the same destination port used by Alice: SIP UDP 5070;
- Ekiga softphone only needs the Call ID to teardown a session;
- X-Lite and Twinkle softphones need additional fields to provoke disconnection: tag, branch and source SIP UDP port.

2.2.4 Denial of Service: Identity Hijacking

The attack experimented in the figure 2.11 originally intercepts a SIP INVITE message from Alice to Bob and uses it to invite another client. It was assumed that a DNS poisoning took effect earlier. First, the DNS cache entry for the domain `proxy.sip-fool.com` is poisoned with the attacker's IP address. The attack begins when Alice sends an INVITE to Bob towards the attacker (1). This INVITE is modified by the attacker, changing its destination from Bob to Charlie and then replaying the modified packet to the true SIP proxy (2). After receiving it, the proxy server sends an authentication SIP response with a random nonce number (407 proxy Authentication Required) to the attacker (4) that replays it to Alice (6), who is the only one able to solve that challenge. Alice sends a new INVITE with a response to that challenge¹ to the attacker (8). The attacker changes the destination of the new INVITE from Bob to Charlie and replays it with the response given by Alice to the SIP proxy (9). As the SIP proxy is able to calculate that response (because it knows Alice's password), it compares the response arrived with the one calculated. If both are equal, the authentication is accepted and the proxy server contacts Charlie (11) and an RTP flow is closed between Alice and the proxy server (18). In the end, Alice receives a BYE message (19), provoking a DoS against her.

SIPp tool was used to perform this attack. Since an attacker needs to exchange messages with two entities at the same time (Alice and the SIP proxy), there are in actually two XML scripts mutually synchronized running on the attacker's machine.

Note that the attacker only could contact Charlie because in the replayed INVITE message, the user's URI was modified from `bob@proxy.sip-fool.com` to `charlie@proxy.sip-fool.com`. Furthermore, Asterisk SIP proxy did not match the SIP URI included in the challenge response (`bob@proxy.sip-fool.com`) with the user's URI on top of the INVITE message (`charlie@proxy.sip-fool.com`).

2.2.5 SIP Fuzzing

The attack presented in figure 2.12 explores vulnerabilities in SIP Proxies or softphones. In our test scenario, one security flaw provoking a denial of service was identified on the twinkle softphone. The tool used to identify this vulnerability was the PROTOS sip test case [23]. This test case generates malformed and invalid fields, and injects them in a SIP INVITE message. Furthermore, for each modified packet, PROTOS sends a well-formed INVITE to verify if the subject under test is still responding to messages.

The vulnerability tested on the Twinkle softphone is in the CSeq field. PROTOS generates a buffer overflow by inserting the "a" character on that field. In the example below, just one malformed INVITE packet was necessary to crash Bob's softphone.

¹This response was generated by a 32-bit hash function that took as input the following parameters: nonce received, Alice's URI, SIP method, realm, username and password. All those parameters were transmitted in clear text, except Alice's password. Note that it would be very time consuming for the attacker to guess Alice's password from the hashed response.

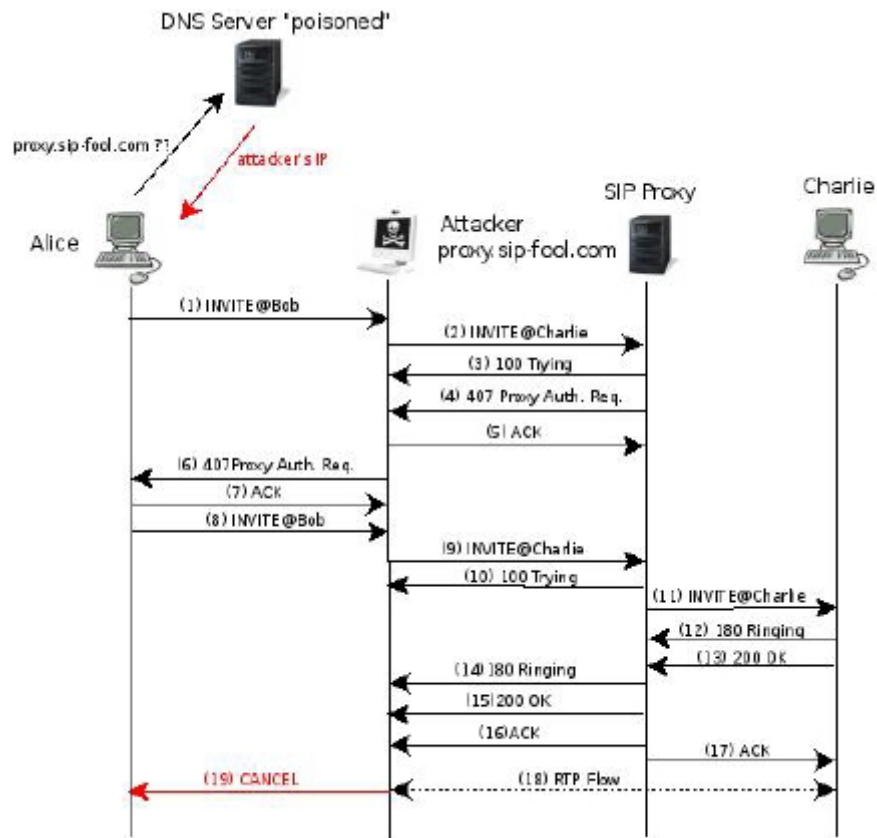


Fig. 2.11 : Denial of Service: Identity Hijacking.

2.3 Attack Scenario

The attacks described below were chosen to compose an attack scenario. I chose a DNS poisoning attack because of its strength to cause damages in real scenarios and because it can be used as an atomic attack to execute others attacks. The second attack was chosen mainly because it usurps from an user's identity to make illicit calls. The third and last attack does not need neither of previously described attacks to be executed. Therefore one solution adopted in this experiment was to detect this attack with a Snort rule and use SnortSam to configure IPTables to block this intrusion.

1. DNS Poisoning using transaction ID prediction
2. Denial of Service - Identity Hijacking
3. SIP Fuzzing

2.3.1 Including Snort Attack Signatures

Snort sniffs the network traffic and searches specific fields or contents that might have anomalies or indicates a suspicious behavior.

No.	Time	Source	Destination	Protocol	Info.
3	0.000845	192.168.60.162	192.168.57.146	SIP	Request: INVITE sip:bob@192.168.57.146[Malformed Packet]
<ul style="list-style-type: none"> ↳ Frame 3 (534 bytes on wire, 534 bytes captured) ↳ Ethernet II, Src: Vmware_04:2b:c8 (00:0c:29:64:2b:c8), Dst: D-Link_03:4e:01 (00:1b:11:03:4e:01) ↳ Internet Protocol, Src: 192.168.60.162 (192.168.60.162), Dst: 192.168.57.146 (192.168.57.146) ↳ User Datagram Protocol, Src Port: sip (5060), Dst Port: 5069 (5069) ↳ Session Initiation Protocol <ul style="list-style-type: none"> ↳ Request-Line: INVITE sip:bob@192.168.57.146 SIP/2.0 ↳ Message Header <ul style="list-style-type: none"> ↳ Via: SIP/2.0/UDP jars-desktop:5060;branch=z9hG4bK000019230 ↳ From: 1923 <sip:user@jars-desktop>;tag=1923 ↳ To: Receiver <sip:bob@192.168.57.146> ↳ Call-ID: 00jars-desktop ↳ CSeq: 1 aaaaaaaaaaaaaaaaaa ↳ CSeq String too big: 17 bytes 					

Fig. 2.12 : SIP fuzzing in action.

However, one inconvenient using Intrusion Detection Systems such as Snort is related to the detection of VoIP attacks, such as man-in-the-middle (MiD) attacks and session hijacking. As showed in the figure 2.11 of section 2.2.4 there is no irregularities in the message exchange in those attacks, when compared to an authentic SIP flow. To detect a possible MiD attack or a session hijacking, it would be necessary to employ a VoIP IDS [1], which would be capable of interpreting VoIP sessions and distinguish possible attacks from genuine communication attempts.

Despite this intrinsic characteristic, it is still possible to employ Snort to detect VoIP attacks. Section 2.2.4 presents that elementary attacks need to be performed to execute these VoIP attacks. As ARP and DNS poisoning have a detectable intrusion signature, it is possible to use a correlation model and associate those elementary attacks in a more global attack and finally define a reaction scheme to block those wrongful conducts.

As commented before, the second attack (Identity Hijacking) of my attack scenario, by itself, can not be detected using Snort, as there are no traces (signatures) on the message exchange that could express this attack using Snort rules set. However this attack can be modeled using the Lambda language [15] and by using CRIM, a correlation can be established with the first attack (DNS Poisoning). It means that to execute the identity hijacking, the attacker needs first to poison the DNS cache entry.

DNS Poisoning Signature

As also explained earlier, this poisoning attack starts by sending CNAME records to the vulnerable DNS. The attack detection is based exactly on that constraint. One Snort rule was created to detect each CNAME message sent to the target DNS and generate an alarm. The reason to base the attack detection on those canonical messages, is that their usages are discouraged by the respective RFC standard [26] to avoid problems with message integrity and cache poisoning.

The rule of figure 2.13 searches for CNAME fields on UDP source port 53 to any destination UDP port. Snort jumps 28 bits (offset parameter) and than starts searching for binary field "00 05 ", which by DNS definition means the type of DNS answer: CNAME (Canonical name for an

alias). The field "classtype " will be used by CRIM to correlate this elementary alarm with a global intrusion attempted. The field "idmef " generates the alarm file using the file format defined by IDMEF (Intrusion Detection Exchange Format) [27]. The objective of this data format is to define a standard between IDS probes and reaction modules from different manufacturers.

```
alert udp any 53 -> any any (msg:"CNAME chain attempt"; reference:DOS_POISON; content:"|00 05|";
sid:20000; offset:28; classtype:denial-of-service; idmef: default;)
```

Fig. 2.13 : Snort rule to detect CNAME fields in DNS messages.

SIP Fuzzing

Snort can also produce a detectable signature for the third attack (SIP fuzzing). The signature rule is presented in figure 2.14.

```
alert udp any any -> any 5060:5070 (msg:"VOIP-SIP CSeq buffer overflow attempt"; content:"CSeq|3A|";
nocase; isdataat:13,relative; content:!"|0A|"; within:13; reference:bugtraq,18906; reference:cve,2005-4050;
reference:url,www.ietf.org/rfc/rfc3261.txt; sid:11971; rev:2; idmef: default;)
```

Fig. 2.14 : Snort rule to detect the SIP fuzzing attack.

Snort sniffs UDP packets from any IP and UDP sources to any destination with UDP ports from 5060 ~ 5070. After that, by using the "isdataat " parameter, it verifies if the payload has data at a specified location (13 bits), looking for data relative to the end of the previous content match (the character sequence "CSeq "). If it finds this sequence, by using "within " this rule verifies if there are at most 13 bits until the end of the line. The fields "reference " just tag this alarm with the IETF attack classification. As in the previous rule, the parameter "idmef " generates an alarm using the IDMEF data format. The last field "fwsam " is used to trigger the reaction rule to this attack and will be further detailed on section 2.4.3.

2.3.2 Experimentation: generating IDMEF alarms

The rules described in the precedent section were added to the Snort rule's directory (/etc/snort/rules) under the file "voip.rules " and a new rule description was inserted in the Snort configuration file (/etc/snort/snort.conf). The topology used in this experiment is presented on figure 2.2. The tools used are presented in figure 2.15.

DNS Poisoning alarm

Firstly, Snort was start up on the machine PC1 to listen to all traffic on ethernet interface 1 (192.168.60.0/24). Secondly, the DNS poisoning tool was launched in the attacker's machine to poison the domain proxy.sip-fool.com with the attacker's IP address 192.168.60.162.

	Acting as	Softwares	Addresses
PC1	Alice	Twinkle Softphone	192.168.57.151 and 192.168.60.151
	IDS probe	Snort v2.2.0	
PC2	Bob	Twinkle Softphone	192.168.57.146
	Proxy SIP to register Alice and Bob	Asterisk v1.4.18	
PC3	Attacker – Identity hijacking, DoS	SIPp v3.1	192.168.60.162
	Attacker – SIP Fuzzing	PROTOS SIP test case	
	Attacker – DNS poisoning	DNS Poisoning	

Fig. 2.15 : Tools used to generate alarms.

After Snort had detected the CNAME messages exchanged with PC2 (DNS vulnerable), the IDMEF alert was created and stored on the file (/snort-idmef/idmef-messages.log). Part of the IDMEF alert is presented in figure 2.16.

```
<?xml version="1.0"?>
<!DOCTYPE IDMEF-Message PUBLIC "-//IETF//DTD RFC XXXX IDMEF v1.0//EN" "/home/daniel/tpids/snort-idmef/idmef-message.dtd">
<IDMEF-Message version="1.0">
  <Alert ident="23">
    <Analyzer analyzerid="IDS1" model="snort" version="2.2.0" ostype="Linux" osversion="2.6.22-14-generic"/>
    .
    .
    .
    <Classification origin="vendor-specific">
      <name>msg=CNAME chain attempt</name>
      <url>none</url>
    </IDMEF-Message>
```

Fig. 2.16 : IDMEF alert to the "CNAME chain attempt".

SIP Fuzzing alarm

PROTOS tool was started up on the attacker's machine (PC3) to generate a buffer overflow in the CSeq field in an INVITE message.

Consequently, the IDMEF alert referent to the SIP fuzzing was also created and stored on the same file (/snort-idmef/idmef-messages.log). Part of the IDMEF alert is presented in figure 2.17.

2.4 Correlation and Reaction Models

The prototype CRIM implements a cooperation model with alarm correlation, using the Lambda language [29]. This language is a technique to derivate links between alarms to produce a global attack scenario. This logical language is also used to describe possible counter-measures used by the system's administrator to block this attack scenario.

The subsections presented below describes the correlation, anti-correlation, intention recognition and finally the reaction models used by CRIM against DNS poisoning and the identity hijacking. The last subsection shows the reaction against SIP fuzzing using SnortSam plugin.

```

<?xml version="1.0"?>
<!DOCTYPE IDMEF-Message PUBLIC "-//IETF//DTD RFC XXXX IDMEF v1.0//EN" "/home/daniel/tpids/snort-idmef/idmef-message.dtd">
<IDMEF-Message version="1.0">
  <Alert ident="38">
    <Analyzer analyzerid="IDS1" model="snort" version="2.2.0" ostype="Linux" osversion="2.6.22-14-generic"/>
    .
    .
    .
    <Classification origin="vendor-specific">
      <name>msg=VOIP-SIP CSeq buffer overflow attempt</name>
      <url>none</url>
    </IDMEF-Message>

```

Fig. 2.17 : IDMEF alert to the SIP fuzzing attack.

2.4.1 Elaborating Lambda Action Models

Lambda language has four attributes to define attacks (pre conditions, post conditions, detection and verification). Pre and post conditions are created using predicates. Recall that pre conditions are conditions that must be fulfilled by the system so that the attack is feasible, and post-conditions are the effects on the system induced by the attack. If some actions done by the attack cannot be written using a composition of those predicates, new actions can be easily created.

The list of the predicates used to create the Lambda models of our attack scenario (DNS poisoning, identity hijacking and SIP fuzzing) is presented bellow:

- is on(Dns server): DNS server is up and running.
- run program(Dns server, bind 9, version 9.5.0a5, root, V1): Bind9 installed in the DNS server.
- vulnerable(Dns server, cve-2007-2926): DNS server is vulnerable to the transaction ID prediction.
- corrupted(Dns server, cve-2007-2926): DNS server is corrupted (poisoned).
- corrupted(Alice, cve-2005-4050): Alice's softphone is corrupted (softphone crashed).
- authorized(Attacker, cname reply, Dns server): Attacker is authorized to send CNAME queries to the DNS vulnerable.
- knows(Attacker, uri(Alice, URI)): Attacker knows Alice's URI.
- knows(Attacker, sip port(Alice, Port)): Attacker knows Alice's source UDP port.
- user access(Attacker, Asterisk, session, root): Attacker using Alice's identity, is registered on Asterisk.

Using the predicates above, for each attack, Lambda action models were written using pre and post conditions, as presented in figure 2.18.

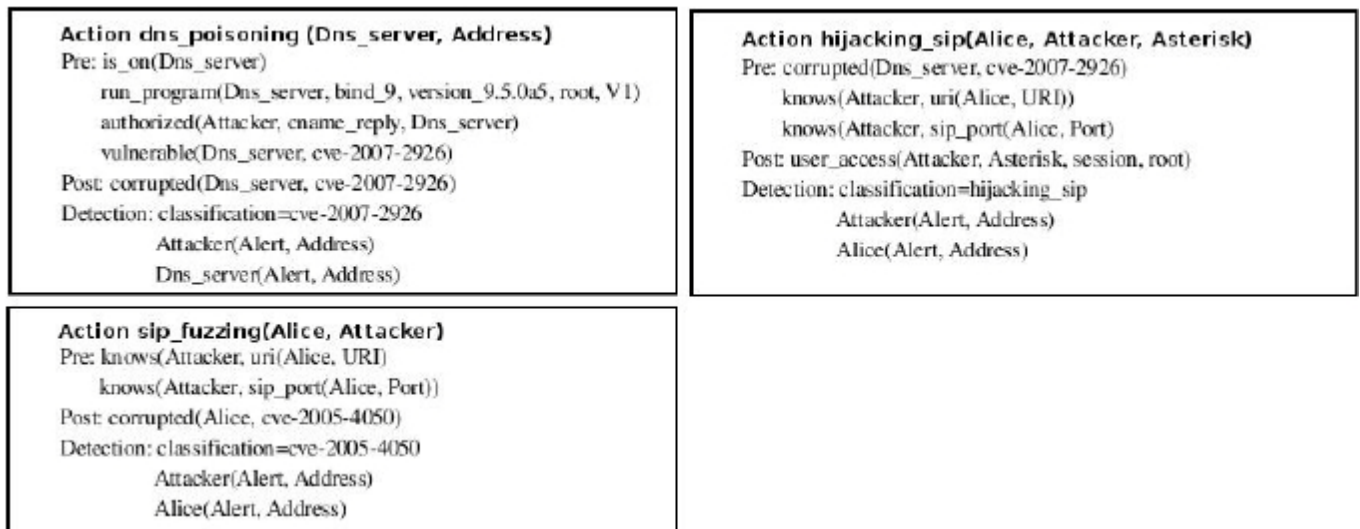


Fig. 2.18 : (a) DNS poisoning model; (b) SIP identity hijacking model; (c) SIP fuzzing model.

2.4.2 Correlating Action Models

Two action models can be correlated if they have at least one predicate in common. For example, the same predicate present in a pre condition of one action model and also present in the post condition of the other action model. By equivalence, this affirmation is also true when we have the same predicate appearing in a post condition of one action model and in a pre condition of the other action model.

When we compare the DNS poisoning action model with the SIP identity hijacking model, we can verify that both share the predicate (DNS server is corrupted, poisoned). This is the same conclusion presented by CRIM when these action models are loaded and correlated. The figure 2.19 shows the predicate that allowed these models to be correlated.

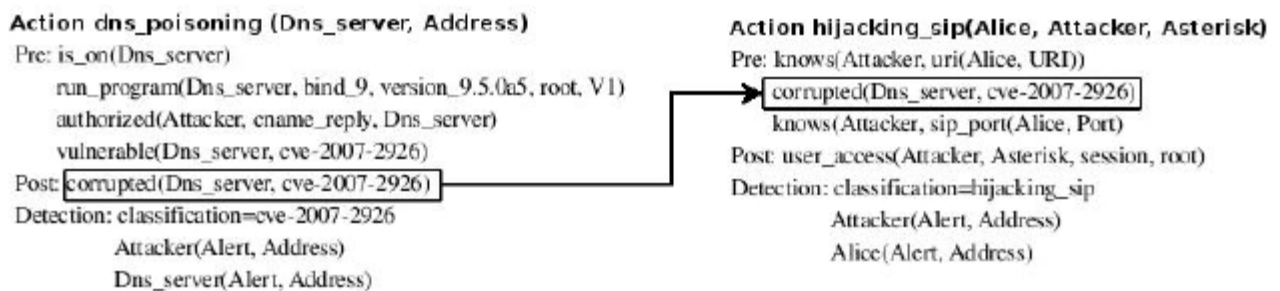


Fig. 2.19 : Correlating Action Models.

Comparing also the SIP fuzzing model with the others action models, we can verify that there is no predicate that could allow a correlation among the SIP fuzzing and the others two attacks. This result was already anticipated, as the attacker does not need to perform neither a DNS poisoning

nor an identity hijacking attack to accomplish a SIP fuzzing. As a matter of fact, the SIP fuzzing attack occurs in parallel with both DNS poisoning and SIP identity hijacking.

2.4.3 Correlating Attack Models with the Intrusion Objective

The intention recognition function is the CRIM's module responsible to determine the intentions of the attacker. In our case, the attacker's intention is to hijack Alice's identity and to provoke a DoS against Alice (softphone crash). We can express this intrusion objective using the Lambda language and correlate it with our action models. For example, if a post condition of an action model (A) and an objective intrusion (I) have at least one predicate in common, thus they can be unified.

Figure 2.20 shows the intrusion objective and the common predicates that permitted the correlation with our action models.

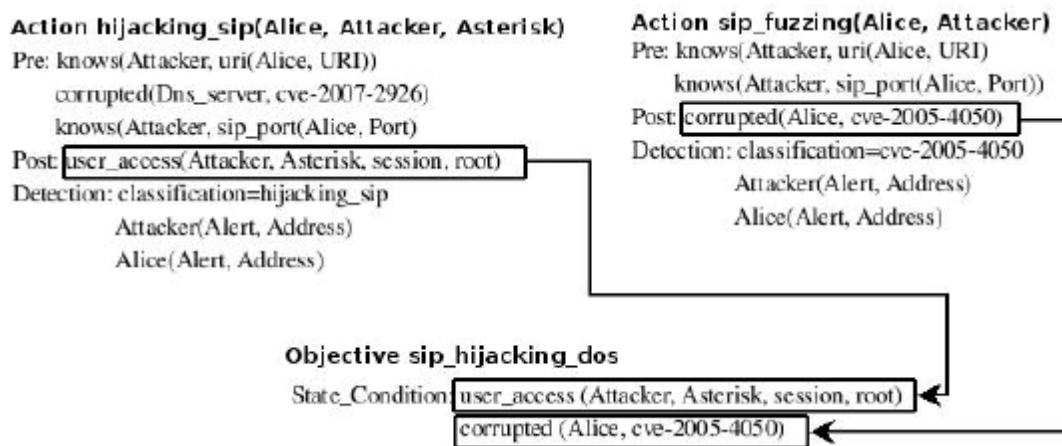


Fig. 2.20 : Correlating Attack Models with the Intrusion Objective.

CRIM: reaction by anti-correlation

Reaction models are written using the same formalism used to create action models: pre and post conditions. A pre condition for a reaction is defined as a logical condition in the system state that allows the counter-measure to be executed. A post condition is a logical condition that defines the effect of the counter-measure application.

The goal of the anti-correlation is to provide possible counter-measures that block the intrusion. Similarly to the correlation method, a counter-measure (C) is anti-correlated with the intrusion objective (I) if one predicate in the post condition (post(C)) and the negation of this same predicate appears in (obj(I)).

The common predicate that correlates the DNS poisoning with the identity hijacking means that the attacker needs firstly to poison the DNS cache entry to finally hijack Alice's identity. Therefore, the counter-measures for this attack are to block the DNS poisoning by applying: a

patch installation that corrects the Bind9 vulnerability (prediction of the transaction ID); or to block/reroute CNAME responses from the attacker at the firewall level. Both of these counter-measures can be expressed using the Lambda language. Figure 2.21, shows these reaction models and the anti-correlation done with the actions models.

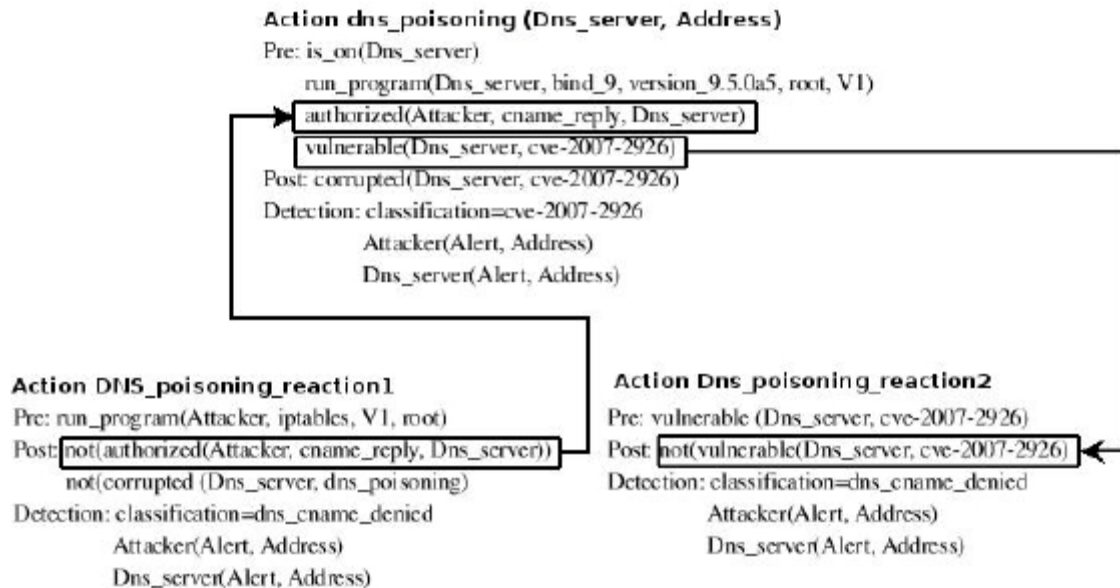


Fig. 2.21 : Reaction by anti-correlation based in the intrusion objective.

SnortSam: reaction against the SIP Fuzzing

As the SIP fuzzing cannot be correlated neither with DNS poisoning nor with SIP hijacking, it is impossible to CRIM applies the same counter-measures. Therefore an alternative solution was used. When an alarm corresponding to this attack is generated, the SnortSam plugin [17] is used to automatically insert a firewall rule and stop this intrusion attempt by blocking the UDP SIP port from the attacker's IP address.

In the other hand, similarly to the anti-correlation done in the precedent section, CRIM could be used to elaborate an anti-correlation rule by inserting another elementary attack (refers to section 1.3.1) that allows the correlation with this particular attack.

SnortSam is composed of two parts: one output plugin that interacts with Snort and an agent that runs on the firewall. It was decided to use the Linux firewall IPTables for its simplicity of configuration. To apply SnortSam plugin, it is necessary to re-compile Snort and insert the line above in the Snort configuration file ("snort.conf"). This command informs the localization of the SnortSam agent and the password it must use to close the TCP encrypted:

```
output alert fwsam: 192.168.57.151/12345
```

and the parameter "fwsam: src, 15 minutes " was also included on the Snort rule (2.22):

```
alert udp any any -> any 5060:5070 (msg:"VOIP-SIP CSeq buffer overflow attempt";
content:"CSeq|3A|"; nocase; isdataat:13,relative; content:"\|0A|";
within:13; reference:bugtraq,18906; reference:cve,2005-4050; reference:url,www.ietf.org/rfc/rfc3261.txt;
sid:11971; rev:2; fwsam: src,15 minutes)
```

Fig. 2.22 : Snort rule to detect SIP fuzzing and triggering SnortSam plugin.

The SnortSam configuration file was created accordingly to the figure 2.23. This file means that SnortSam will accept connections from Snort, localized on 192.168.57.151 with a password 12345. It also indicates the IP address where the IPTables firewall is running and the ethernet interface where that rule must be applied. A log file directory was optionally included.

```
#IP and password of your Snort sensor.
accept 192.168.57.151
defaultkey 12345
fwsam 192.168.57.151
iptables eth1
logfile /var/log/snortsam.log
loglevel 3
```

Fig. 2.23 : SnortSam configuration file.

After those configurations, the SIP fuzzing attack was launched on the attacker's machine (PC3). When the correspondent rule is triggered, Snort sends an encrypted TCP packet to the SnortSam agent that requests the firewall to insert a new rule to block the attacker's IP address:

```
iptables -I FORWARD -i eth1 -s 192.168.60.162 -j REJECT
```

To test this configuration, a new attack was launched. One ICMP packet was received in the attacker's machine, proving that all the traffic from the attacker's machine is blocked up. After the stipulated 15 minutes, SnortSam unblocks the traffic from the 192.168.560.162, by removing that IPTables rule.

```
iptables -D FORWARD -i eth1 -s 192.168.60.162 -j REJECT
```

Alternatively, the traffic coming from the attacker's machine could be permanently blocked. However, this temporarily blocking is useful, as SIP fuzzing attacks are intermittent, differently from DoS attacks.

Results

Using both scenarios described in section 2.2, figures 2.1 and 2.2, different experiments were elaborated over the proposed attack scenarios: 1) DNS Poisoning using transaction ID prediction; 2) Denial of Service - Identity Hijacking; and 3) SIP Fuzzing. The first two attacks are called “elementary attacks” because the attacker executes successive steps in order to achieve his final result. The third one, by itself is completely independent from the precedent attacks. In fact, to perform this attack, it is sufficient to know the target IP address, SIP UDP port and the user’s URI SIP. Those information could come from another elementary attack, such as a dictionary attack, as explained on [1] or by another illicit methods.

Detection rules were inserted in the Snort Intrusion Detection System which were able to detect the first and the last attack, generating IDMEF standards alerts for them. However, for the second attack, Snort was not used to detect it due to a lack of a signature on packets, that could denounce it. IDS generally operates either with rules based on certain exceptions or how a packet flow arrives at the target. But the behavior of this hijacking attack is exactly the same as a normal user registration which invalidates its detection by a normal IDS probe.

For the three attacks, Lambda models were created, specifying pre and post-conditions to describe each attack. Finally CRIM was used to merge those elementary attacks and try out to correlate them as an attempt to perform a more global attack. As presumed, CRIM was able to correlate the first and the second attack, hence generating counter-measures to block this “more global” attack. But CRIM could not correlate the third attack with the precedents due to the fact that to accomplish its task, the attacker does not need to poison the DNS server to fuzzing a SIP client. The counter-measure proposed to stop this attack was triggered directly by Snort, using a plugin that insert IPTables rules, thus blocking the attacker’s IP address.

Table 2.24 resumes the experimentations executed and includes the counter-measures to the attacks and by whom they were produced.

I recall that the SIP fuzzing attack could be used with a previous attack, such as a dictionary scanning. Therefore in this case, CRIM can be used to correlate the alerts and propose an appropriate counter-measure. However, this condition was not implemented because one of the

Atomic attacks	Detection	Correlation	Reaction
DNS Poisoning	Snort rule based on CNAME messages detection.	Correlation with DoS – Identity Hijacking done by CRIM.	Two counter-measures proposed by CRIM: apply patch upgrading Bind9 version or insert IPTables rule and reroute CNAME messages to another destination.
DoS - Identity Hijacking	No valid signature that could be used by Snort to generate an alarm.	Correlation with DNS Poisoning done by CRIM.	
SIP Fuzzing	Snort rule based on the length field of the CSeq parameter that causes a buffer overflow and crashes the softphone.	No correlation executed by CRIM (based on the loaded models).	One counter-measure applied by Snort plugin: Insert IPTables rule to block the attacker's IP address.

Fig. 2.24 : Attack scenario with counter-measures applied.

purposes of this experimentation was also to verify the usability of other reaction models, such as using IPTables triggered by the SnortSam plugin.

Conclusion

In this document, it was exhibited several VoIP vulnerabilities and proposed CRIM as a method to correlate and react to attacks beginning with ARP or DNS poisoning and finishing by VoIP attacks that do not have detectable IDS signatures. In the other hand, for VoIP attacks that may leave a signature trace, the IDS Snort with the reaction plugging SnortSam provided a good and best fitted solution as well.

Future works for this study considers to include VoIP-based predicates on CRIM to extend its database of attacks and to study the applicability of a VoIP IDS to detect attacks that cannot be identified on regular IDS available on the market.

Bibliography

1. Y. Bouzida, C. Mangin. A framework for detecting anomalies in VoIP networks. Ares 2008, march 2008
2. Y. Volobuev. ARP and ICMP redirection games. September 1997. Available at: <http://insecure.org/sploits/arp.games.html>. Accessed on april 2008
3. RFC 3261 SIP: Session Initiation Protocol. Available at: <http://www.ietf.org/rfc/rfc3261.txt>. Accessed on february 2008
4. RFC 1035 Domain Names: Implementation and Specification. Available at: <http://www.ietf.org/rfc/rfc1035.txt>. Accessed on march and april 2008
5. Birthday Attack. Available at: http://en.wikipedia.org/wiki/Birthday_attack. Accessed on february 2008
6. J. Stewart. DNS Cache Poisoning: The Next Generation. Available at: <http://www.lurhq.com/dnscache.pdf>. Accessed on march and april 2008
7. A. Klein. BIND 9 DNS Cache Poisoning. March-june 2007. Available at: <http://www.trusteer.com/docs/bind9dns.html>. Accessed on march and april 2008
8. BIND Predictable DNS Query IDs Vulnerability. Available at: <http://secunia.com/advisories/26152/>
9. Microsoft Windows DNS Client Predictable Transaction ID. Available at: <http://secunia.com/advisories/29696/>
10. D. Geneiatakis, G. Kambourakis, C. Lambrinouidakis, T. Dagiuklas, S. Gritzalis. SIP Message Tampering: The SQL Code Injection attack.
11. VOMIT: Voice over Misconfigured Internet Telephone. Available at: <http://vomit.xtdnet.nl/>. Accessed on march 2008
12. RTP InsertSound and RTP MixSound. Available at: http://www.hackingvoip.com/sec_tools.html/. Accessed on march 2008

13. Y.S. Wu, S. Bagchi, S. Garg, N. Singh, and T. K. Tsai. SCIDIVE: A Stateful and Cross Protocol Intrusion Detection Architecture for Voice-over-IP Environments. In Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN'04), Florence, Italy, July 2004.
14. H. Sengar, D. Wijesekera, H. Wang, and S. Jajodia. VoIP Intrusion Detection Through Interacting Protocol State Machines. In Proceedings of the 2006 International Conference on Dependable Systems and Networks (DSN'06), Pennsylvania, USA, June 2006.
15. Fabien Autrel and Frédéric Cuppens. CRIM - Cooperation and Recognition of Malevolent Intentions. CRIM : an alert correlation and reaction module. *Annales des Télécommunications*, 2006.
16. Snort Inline. Available at: <http://snort-inline.sourceforge.net/>. Accessed on may 2008
17. SnortSam. Available at: <http://www.snortsam.net/>. Accessed on may 2008
18. IDMEF - Intrusion Detection Message Exchange Requirements. IETF draft document: draft-ietf-idwg-requirements-02.txt. Accessed on may 2008
19. Caim & Abel software. Available at: <http://www.oxid.it/index.html>. Accessed on february and march 2008
20. BIND DNS Server. Available at: <http://www.bind9.net>. Accessed on march and april 2008
21. Bind9 ID Prediction - Proof of Concept. Available at: <http://www.coromputer.net/>. Accessed on march and april 2008
22. SIPp traffic generator for the SIP protocol. Available at: <http://sipp.sourceforge.net/>. Accessed on march and april 2008
23. PROTOS Test-Suite: c07-sip. Available at:
<http://www.ee.oulu.fi/research/ouspg/protos/testing/c07/sip/#h-ref18>. Accessed on april and may 2008
24. RFC 3550. RTP: A Transport Protocol for Real-Time Applications. Available at:
<http://www.ietf.org/rfc/rfc3550.txt>. Accessed on may 2008
25. RFC 2327. SDP: Session Description Protocol. Available at: <http://www.ietf.org/rfc/rfc2327.txt>. Accessed on may 2008
26. RFC 1034. Domain Names: Concepts and Facilities. <http://www.ietf.org/rfc/rfc1034.txt>. Accessed on march and april 2008
27. IDMEF: Intrusion Detection Message Exchange Requirements. Internet draft: draft-ietf-idwg-requirements-02.txt. Accessed on may 2008

28. P. Oechslin. Making a Faster Cryptanalytic Time-Memory Trade-Off Laboratoire de Sécurité et de Cryptographie (LASEC)
29. Cuppens (F.), Ortalo (R.), LAMBDA : A Language to Model a Database for Detection of Attacks. Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID'2000), Toulouse, France, October 2000
30. RFC 2865. Remote Authentication Dial In User Service (RADIUS). Available at: <http://www.ietf.org/rfc/rfc2865.txt>. Accessed on may 2008
31. RFC 3588. Diameter Base Protocol. Available at: <http://www.rfc-editor.org/rfc/rfc3588.txt>. Accessed on may 2008
32. RainbowCrack 1.2. Available at: <http://www.antsight.com/zsl/rainbowcrack/>. Accessed on march and april 2008
33. J. Garcia, S. Castillo, G. Navarro and J. Borrell. Mechanisms for Attack Protection on a Prevention Framework.
34. F. Cuppens, F. Autrel, Y. Bouzida, J. Garcia, Gombault (S.) et Sans (T.), Anticorrelation as a criterion to select appropriate countermeasures in an intrusion detection framework. Annales des Telecommunications, n. 61, pp 197-217, March 2006.
35. SIP Express Router - SER. Available at: <http://www.iptel.org/ser/>. Accessed on may 2008



The High Cybercafe: Internet in the Nepal

Himalayas

SVF-3903

Tanel Saimre

Master of Philosophy in Visual Cultural Studies

Faculty of Humanities, Social Sciences and Education

University of Tromsø

May 2012

Acknowledgements

My fieldwork, film and thesis were made possible through the help and participation of many people. My gratitude goes to: my wonderful informants and others from Khumbu Lodge for permission and help with conducting fieldwork in this establishment; teachers of the Shree Himalaya Primary School in Namche, especially the computer and mathematics teacher and 5th grade children; my hosts at Moonlight Lodge for a wonderful and cosy stay; translators Ramesh Simkhada, Dik Rai and Prakash Sapkota; Bente Sundsvold for supervision, time and devotion, Ane Lyngstad Oltedal for the Fleck 2000 tip; and all my other classmates (except Esben:)). VCS 2010 will be in my heart forever.

Table of Contents

1 Introduction.....	1
1.1 The Topic.....	2
1.2 Reflexive Aspect.....	2
1.3 Technology, Society and ANT.....	3
1.4 Modernity, Globalisation and Cyberspace.....	5
1.5 High-Context and Low-Context.....	7
1.6 Actor-Networks, Black Boxes and Punctualisation.....	8
1.7 My Methodological Toolbox.....	10
2 The Setting.....	11
2.1 Nepal, Khumbu and the Sherpas.....	11
2.2 Namche Bazaar and the Role of Tourism.....	13
2.3 Modernity and Change?.....	16
3 Fieldwork in Namche.....	19
3.1 Two Months of Yaks, Cables and Cybercafes.....	19
3.2 Khumbu Cyber, Sameer and Rajesh.....	20
3.3 Namche Primary School and Dawa.....	21
3.4 Trips Along the Network.....	21
3.5 Note-taking.....	22
3.6 Video Camera Use.....	22
4 High Context Communication at the School.....	25
4.1 Computer Lessons at the School.....	25
4.2 It's Government Service and Therefore Useless?.....	28
4.3 Summary	32
5 Punctualisation Battle at the Cybercafe.....	33
5.1 Mediators to the Cyberworld.....	33
5.2 Actors and Networks.....	35
5.3 Summary.....	38
6 The Internet as an Actor-Network.....	39
6.1 What Is It Used For.....	39
6.2 Accompanying Effects: Language, Alphabet and Calendar.....	41
6.3 Sameer's Facebook Friendship.....	43
6.4 Monks and Nuns.....	45
6.5 Chronological View.....	46
6.6 Discussion.....	47
7 Conclusion.....	49
8 References.....	51

1 Introduction

It was a long hike. I had planned to walk the distance from the airfield to my fieldwork site in a day, maybe two. But the reality of my 30 kg backpack and hard mountain trails going either up or down but never really level for any length, dictated my tempo and it took me three days instead.

I was 3000 m up in the Himalayas of Nepal, on a hike from the airfield at Lukla to my fieldwork location in Namche Bazaar. Tourists carrying light day packs were constantly whizzing past me. I was moving more or less at an equal tempo with the porters, poor Nepali men, slowly taking their careful and measured steps uphill, carrying their incredible loads of up to 90 - 100 kg. I had refused to hire a porter myself. It's cheap for a westerner, but I wanted to carry all my own gear. "To get to know what it feels like to be a porter," I had thought. "What an idiot," was I thinking now, my lungs gasping for oxygen in the thin atmosphere and all my strength squeezed out of me. This is the High Himalayas. There are no roads, only frail mountain trails. The only means of transport besides your own feet are charter helicopters or yak caravans for those who can pay.

Anthropologists are not among the wealthy and the powerful, so I walked. On the third day, totally exhausted, I reached Namche Bazaar. A beautiful place, as I remembered from my visit as a tourist about one year ago: a crescent-shaped little cluster of houses on the side of a mountain above a huge gorge, with a nice stream running through it. It takes a day or two to hike to the airfield, a week to the nearest bus stop. Mountainous terrain isolates this place from the transportation networks, roads and airports of our planet. But it is located 30 km from Mount Everest, and therefore it is visited by tens of thousands of tourists per year. It is a place interwoven into the global cultural fabric of economic ties, personal acquaintances and communication networks. The latter was the aspect I came here to study – communication networks, more specifically the internet. For more or less than eight years the people of Namche Bazaar have had access to the internet, and it is slowly working its way into the everyday life of the people here. Smartphones, wifi and cybercafes are a part of everyday life for a growing part of the population here. Physically isolated but still interwoven – this is my research setting.

1.1 The Topic

The aim of this paper is to look more closely at the processes involving the adoption of internet in a Nepali mountain village. Two and a half months of fieldwork was performed at a primary school and a cybercafe in Namche Bazaar, Solukhumbu district, Nepal. My aim is to be informative about the processes revolving around the internet in a general global sense. Therefore my thesis is more concerned with the internet than the particular cultural setting of Nepal or Namche.

Internet is a form of information and communication technology. The fact that it is information AND communication technology is of importance here. While no device is able to purely transmit a message without modifying it, communication technology coupled with information technology is designed to not just relay information, but also to process it. Chat messages are saved in log files, e-mails are scanned for viruses, uploaded digital photos are automatically shared with certain people, etc. Technology affects our communication in ever more ways.

Computers and the internet originate in the achievements of white North-American military engineers and mathematicians of the Cold War period. By now, however, 44.8% of internet users are from Asian countries, with over a billion Asian people using the internet. Europe and North-America account for just 34.1% of internet users (Miniwatts Marketing Group n. d.). Contrary to these statistics we still tend to assume that internet is “English” and “Euro-American”, and if people from other countries start using the internet, then it makes “them” more similar to “us”. So I went on fieldwork to have a look and find out what is really happening: how does the adoption of internet, as an alien technology (i.e. used but not produced locally), take place and what other processes are related to it?

1.2 Reflexive Aspect

I think the best research is done on subjects to which the researcher has a special relation. It does not have to be a wholly positive one, a relation of love and kisses only, but it better be a long, eventful and emotional one. I find my immersion into the topic to be long enough – I wrote my first computer programs about 25 years ago. It was done in the BASIC language on a personal computer with a green monochrome monitor and using a cassette tape recorder as a

storage device. From using 2400 baud modems to connect to nodes on a precursory internet-like formation called the FidoNet on the ancient analogue pulse-dialling phone network of the collapsing Soviet Union, to the 12 Mbaud ADSL connections of the digital broadband of today, I have witnessed the development of computers, and the ensuing altering of the face of the society, of the ways we talk and think. My generation has, in a way, grown up as the same time as computers, spending our childhood in the computer-free age, suffering our adolescent hardships and self-discovery period together with the computers coming out of the nerd niche and breaking into mainstream use, and living as grown-ups in a world of which the computers and the internet are a natural part.

My homeland Estonia is seen as a quick developer among other post-Soviet countries, the so-called “Baltic Tiger”, with one of its characteristics being the fast adoption of information and communication technologies (including internet banking, voting on elections and doing administrative things such as declaring your taxes and establishing limited liability companies over the internet). The concept of e-state (which means that the Estonian state is accessible to its citizens online in many convenient ways) together with the fact that the popular internet voice-call software Skype was founded and is being developed in Estonia, form a very important part of the identity of Estonians.

This aspect of our identity makes me feel interest in the internet, after having witnessed on the one hand its going through such an enormous change during the past couple of decades and on the other hand my own society being so thoroughly transformed by it. Therefore it dawned on me, when travelling in Nepal as a tourist and seeing the internet popping up everywhere – here I have an opportunity to see this kind of a phenomenon again!

1.3 Technology, Society and ANT

Broadly speaking, I will be exploring society and technology. I will not give precise definitions of these terms for now, and I will come back to this in the end of this section.

There is a spectrum of theories about the mutual interaction between society and technology, with extreme technological determinism constituting one end of it, and the social construction of technology (SCOT) on the other.

Social construction of technology started out as a protest against the mindset according to

which social factors were needed only when explaining a false belief or adaption of a non-working technology. The claim of the founders of SCOT was that social input is needed for the proper explanation of *all* beliefs, true and false. SCOT recognises that the criteria for judging whether a technology actually works are the socially conditioned. One of its core concepts is the “interpretive flexibility”, meaning that different groups of people can have very different understandings of a technology, including, but not limited to, the symbolic meaning of a technology like a car or aircraft (MacKenzie and Wajcman 1999, 21). The internet, for example, can be an information-finding-engine, a communication device, a political tool, a status symbol, etc.

Technological determinism is in many ways the opposite of SCOT. It claims that technology matters, not just to the material and biological conditions of our lives, but also to the way we live together socially. Langdon Winner, in his essay “Do artifacts have politics?” discusses some very valid points about political consequences that certain technologies may have, bringing some (by now, classic) examples of technology either enabling the settling of a certain issue: the low bridges of the parkways of Long Island, New York, that were arguably designed to keep out the buses of the public transportation system (and people from the lower classes with them) or huge plazas on US university campuses that were designed to defuse student demonstrations; or technologies that *inherently* contain a certain political model, like the centralised nuclear power with a techno-scientific-military elite versus the egalitarian, more democratic, decentralised solar power (Winner 1980, 33-34).

Technological determinism does have some valid aspects and our SCOT-inspired sensitivity towards the influence of social relations on technological artefacts should not result in neglect of the valid points of the opposite view. Further, technological determinism would be very tempting to use in the case of this research. This seems to be a clear cut case of “foreign” technology coming in “determining” certain kind of social change. But to view matters this way would mean to lose sight of half of the situation – why and how does the “foreign” technology “come”. We must preserve a holistic view on the situation and remember that changing technology will always be only one factor among many others: political, economic, cultural, and so on (MacKenzie and Wajcman 1999, 5).

The reason I refused to define “society” and “technology” above is that they are not isolated spheres, with only some kind of “influence” moving between them. There is a tight

interwovenness there – technology and society are mutually constitutive (MacKenzie and Wajcman 1999, 23). In that sense the word “society” also includes “technology”, among all the other persistent and patterned relations that make a troop of primates of the biological species *Homo sapiens* into a society. The phenomena denoted by these two words should not be ripped apart by theory.

One approach which attempts to theorise society and technology (and much else) holistically is the actor-network theory, often abbreviated as ANT. It starts from the standpoint that to reduce all social relations to either human or material/technological actors is an unacceptable reductionism. Social networks are not composed only of people, but also of machines, animals, money, texts, architecture, etc. (Law 1992, 2-3). Both society and technology are made of the same “stuff”: networks linking human beings and non-human entities (MacKenzie and Wajcman 1999, 24).

1.4 Modernity, Globalisation and Cyberspace

Technological change, especially such where new technology (like the internet) is adopted by a more traditional society (like Nepal), can be seen as an act of two things: modernisation (a pre-modern society adopting modern characteristics) and globalisation (a Nepali village becoming interwoven into the interconnectedness of the global modern world through internet and telecommunications). So both modernity (the state of being modernised) and globalisation, and the relation of internet to them, are relevant and require some explanation.

“Globalisation” is an umbrella term denoting a complex set of phenomena. One of these is disembedding, the movement towards a more abstract world. A world with writing, that allows knowledge without a given person, a “knower”. A world with the clock, which externalises time, chops it up into units that are identical for everybody, anywhere and any time, independent of the ebbs and flows of experienced time. A world with units of measurement – abstract concepts of otherwise subjective experiences like length and weight. Anthony Giddens defines disembedding as the “lifting out” of social relations from local contexts of interaction and their restructuring across indefinite spans of time-space (1990, 21). In other words, it is a transition from a concrete society, based on intimate, personal relationships, memory, local religion and orally transmitted myths, to an abstract society

based on formal legislation, archives, a book religion and written history (Eriksen 2007, 18). A lifting out of things from their context into abstractness.

The above description of globalisation as a movement towards a more abstract world also fits modernity. In fact, modernity is in many ways a similar thing, except it takes place on a smaller scale, like that of a nation or a state (Eriksen 2007, 25-27). So globalisation is modernity with a global spread (Eriksen 2007, 30). So globalisation is (among the plethora of other phenomena) global disembedding.

What role does the internet play in this? A good amount of theory has been produced on the relations between the internet and globalisation. One of the authors in this field, Manuel Castells, says the conditions for an accelerated and intensified globalisation were created primarily by three processes: the deregulation of world markets, the end of the Cold War, and the growth of information technology (1996, 3). Now, thanks to the latter, distance between cause and effect can be enormous and space is relativised through the use of telecommunications.

Now, let us clarify one final notion, that of “cyberspace”. The word was coined by the science fiction author William Gibson. In his 1984 science fiction novel “Neuromancer” he depicted cyberspace as a global monolithic placeless medium in a future world which had degenerated into a similarly placeless capital-driven dystopia ruled by transnational corporations, where words denoting locations and nationalities were used as mere adjectives or trademarks: Danish vodka, Japanese neurosurgery, German steel. The novel demonstrated eerie foresight and the word entered mainstream use together with the internet in the 1990-s. Initially it denoted the idea of what internet use was going to be like: global and placeless, almost a psychedelic experience of extreme disembedding from the offline reality. As shown by many studies (Miller and Slater 2000, Ahola 2005, Burrell 2012), this is a fruitless view. The term “cyberspace” has by now become ubiquitous and its field of meanings has exploded to the point of uselessness. Meanwhile, the view of internet as a disembedded online reality has fallen out of use, as people do not see the internet as a “place” where you go and become detached from your physical surroundings. In my work the situation appears to be similar – this new media appears to be deeply embedded in the rest of the participants' social lives, not separated from it.

1.5 High-Context and Low-Context

The concept of context played an important role above: we saw how globalisation and modernisation signify the “lifting of the social relation out of their local contexts of interaction”. The movement towards abstraction is all about losing the context, hiding it or decreasing its importance.

In current research the approach of high context versus low context communication plays an important role. This approach was developed in the 1970-s by Edward Twitchell Hall, Jr., an influential American anthropologist and cross-cultural researcher. Hall was an adherent of a view (quite trivial now, but important at its time), according to which culture is a model constructed by its bearers, that helps us make sense of the real world. It protects our senses from information overload, categorises reality and puts order into it (Hall 1976, 9-15).

According to Hall, culture defines how we experience reality through the use of context. Some cultures rely on context more than others. He distinguishes between two different styles of communication, high-context and low-context: “A high-context (HC) communication or message is one in which most of the information is either in the physical context or internalised in the person, while very little is in the coded, explicit, transmitted part of the message. A low-context (LC) communication is just the opposite; i.e., the mass of the information is vested in the explicit code. Twins who have grown up together can and do communicate more economically (HC) than two lawyers in a courtroom during a trial (LC), a mathematician programming a computer, two politicians drafting legislation, two administrators writing a regulation, or a child trying to explain to his mother why he got into a fight.” (Hall 1976, 86-91)

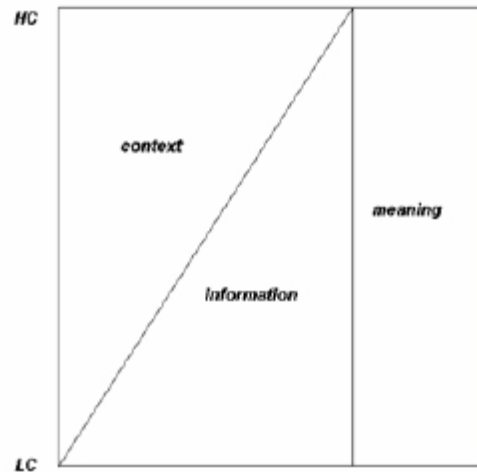


Illustration 1: The LC-HC axis (Hall 1976, 102)

Change pushes our communication towards the LC end of the scale. Since change makes a lot of context dysfunctional we have to be more and more explicit (LC) in our communication. Globalisation, i.e. change towards disembedding “lifts our social relations out of the local context”, and places it into new ones. So on the one hand, globalisation moves us toward a more abstract world of less context, but on the other, context will always be there, especially in more and more complex situations, since complexity will result in overload, if we do not turn towards more HC style of communication and let context take care of some of the meaning-making.

The notion of modern life being abstract or context-independent has come under criticism. Bruno Latour has even used the statement “we have never been modern” as a title of his book (1993). In other words he claims that we have always been dependent on context. With modernity as a move towards a more abstract world we have not reduced the amount of context, but just turned to a different kind of one. The actor-network theory is meant as a tool to help bring that context back into view and explore it.

1.6 Actor-Networks, Black Boxes and Punctualisation

In order to start seeing the context, ANT proposes that we think of the world as made up of actors, and the relations between them. In contrast to other similar approaches, like Manuel

Castells' network society (1996), the actors do not necessarily have to be human. Actors are simply “entities that do things” – be they humans, animals, artefacts, machines, groups, institutions, etc. According to Latour, the distinction between human and non-humans, embodied or disembodied skills, impersonations or “machination,” are less interesting than the complete chain along which actions are distributed (Latour 1992, 243). The emphasis here is on relations between entities, not entities themselves.

An important aspect of actors is that they consist of a number of elements – a human is made of organs and body parts, knowledge, habits, preferences; a technical system is made of its components and relations. These components can in turn be considered actors. So any actor is simultaneously a network of other actors; and any actor is simultaneously a network of its components. That is the meaning of the term actor-network: the world is made up of collections of components, which in turn are components in bigger collections, bigger networks. Naturally, an actor obtains its essence only in relation to the network it is in. Actors and networks are mutually constitutive.

The next important concept is a black box. The term is derived from cybernetics, where it signifies a piece of machinery or a set of commands that might be very complex but can be substituted by a box because it is regular and stable (Wiener 1948). In studying sociotechnical phenomena, a black box could be a computer, a car, a television or any other object that operates as it should. When this occurs, the complex sociotechnical relationships that constitute it are rendered invisible, or black-boxed. The more a box appears to be closed, the more are the networks it includes assumed to be reliable and stable themselves. "The more automatic and the blacker the box is, the more it has to be accompanied by people" (Latour 1987, 137). Relating to the previous section about context: a black box is an actor-network, a sociotechnical system, that is lifted out of its context, or whose context has been hidden from view. In a sense, it is a high-context unit, context that has been sealed up.

Punctualisation refers to the process by which complex actor-networks are black boxed. The process of punctualisation thus converts an entire network into a single point or node in another network (Callon 1991, 153; Law 1992, 4-5). For example, a university is a network consisting of personnel, technical devices, financial resources and facilities. When seen from the level of a state government, all these actors are punctualised into a single entity: the university. So punctualisation is the verb which leads to the noun of the black box. Of course,

as we shall see, a black box is not a state, a configuration of things, but more like a process in itself, so to a certain degree these two concepts can be considered synonyms.

1.7 My Methodological Toolbox

In describing the sociotechnical system I am using concepts taken from the actor-network theory. ANT grew out of an environment of high technological impact (science and technology studies) so I find it suitable for studying the social relations around a computer network. ANT is also appropriate for explaining social processes in situations of abrupt change, where sets of concepts have not yet been developed (Latour 2005, 149) or where previously existing concepts and categories like the classical anthropological concepts of kinship, power, magic, tradition or religion, might be misleading.

To summarise my approach: as an anthropologist interpreting my experiences during fieldwork in Namche, I will be concentrating on the sociotechnical system consisting of the internet, computers and people (both Nepali and tourists). Point one: my approach is informed by the distinction between modern and traditional, which in simplified terms is the degree of reliance on context. For characterisation I am using Edward Hall's categories of high-context and low-context. Point two: my approach is also informed by the critique of point one, of the idea of low-context communication, of context-independence, of abstract modernity. For this I rely on the actor-network theory.

2 The Setting

2.1 Nepal, Khumbu and the Sherpas

By going to a “developing country” or “emerging market”, in other words by doing my research in a cultural environment which is different from the one where ICT technologies are being spawned and developed I hope to gain a better insight into the adoption (as opposed to creation) processes. Besides, most of the growth in internet usage now is happening outside the traditional “technologically advanced” geographical areas of the world (which gives reason to question the validity of such a division). According to a report on internet trends by a private venture capital consultation company Kleiner Perkins Caufield & Byers, the fastest growers in the amount of internet users during 2007-2010 were China, India, Nigeria and Russia (Meeker 2011, slide 7). Nepal is located between the two fastest growers, China and India, and can be considered as non-contributing to the ICT sector (China and India contribute by housing factories and software R&D departments of both national and multinational companies). So the internet in Nepal can be considered something purely consumed and not locally produced. This gives me an opportunity to observe an adoption of a totally alien technology.

Nepal is an interesting field for anthropological study. Until 1949 the government of Nepal rigorously excluded foreigners from travelling outside of Kathmandu by rule of the then governing Rana dynasty that closed Nepal to all foreigners (in a successful attempt to protect it from British colonial influence). From then on just a few mountaineering expeditions were granted passage. In the 1950-s came a change of attitude and tourists were welcomed in (According to Stevens 1993, 356 this was in relation to King Mahendra's coronation in 1955; according to Sherry Ortner 1978, 28 this happened in 1952). From then on this previously isolated and technologically stagnant country has seen a growing torrent of tourists, and been heavily affected by it. In a few decades it made a quantum leap from a medieval-like setting to a 20th century life with electricity, internal combustion engine, telephone and now, the computers.

My fieldwork area was in the region of Khumbu, famous as the land of the Sherpas.

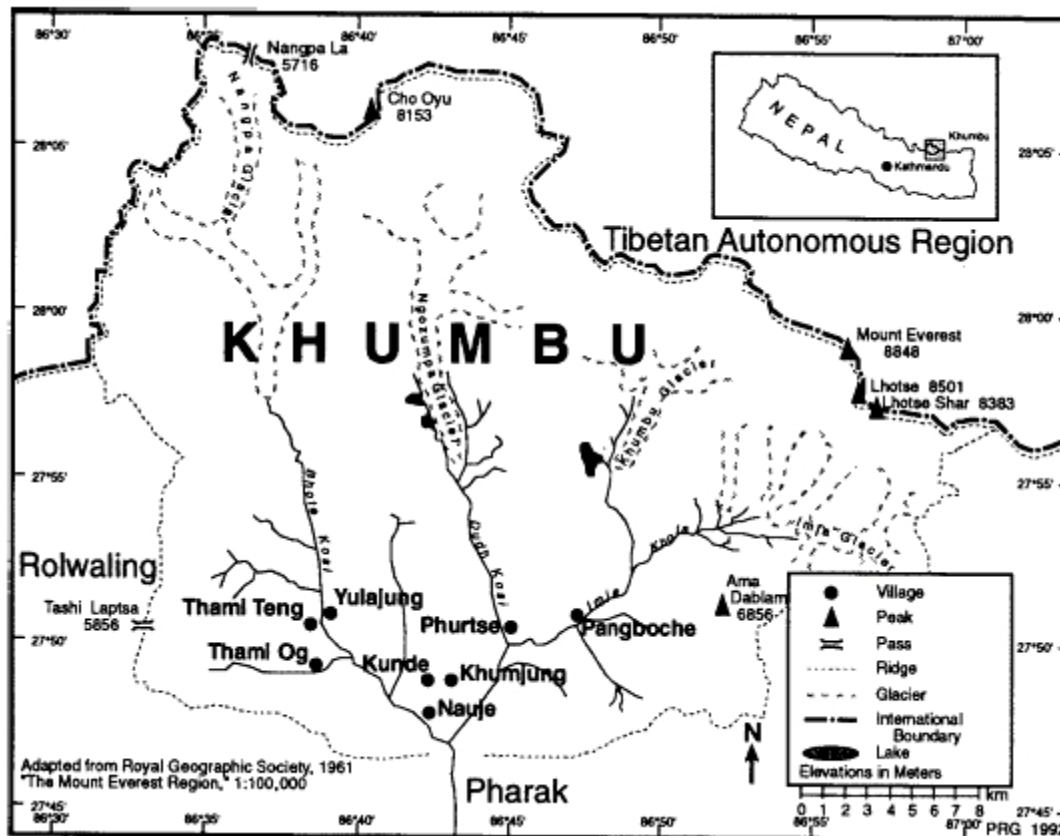


Illustration 2 - Location of Khumbu and Namche (called Nauje on this map). From Stevens 1993, 25.

Khumbu is also the location of Mount Everest that has sparked interest in Europeans ever since they had the social and technological organisation sufficient to cater for such a dangerous and irrational pastime as climbing inaccessible peaks. In the 1950-s when China occupied Tibet and Nepal ended the isolation policy, mountaineering expeditions were launched from Nepal instead of Tibet. The region of Khumbu became a staging ground for the race to “conquer” the world's highest mountain. The local people, the Sherpas, became entangled in that race.

“When men first were drawn to Everest, it was an unknown quantity. It lay between two unknown countries – Tibet and Nepal.” This is a quote from the narrator's text in the 1953 UK documentary “The conquest of Everest”. This Freudian slip describes the British attitude of the time: the mountaineers were considered the first “men” to arrive at Mount Everest, the local population living there for centuries was something more like a natural resource to be used. Over time the role of Sherpas has, however, come to be more and more recognized and now they enjoy worldwide fame. Both the Sherpas and their land of Khumbu have been researched extensively by anthropologists. The race to the top of Mount Everest and the social

and cultural effects of this and other mountaineering activities are described by anthropologist Sherry Ortner in her excellent study “Life and Death on Mt. Everest: Sherpas and Himalayan Mountaineering” (1999). She has also done extensive research on Buddhism – the religion of the Sherpas, and described her findings in “Sherpas Through Their Rituals” (1978) and “High Religion” (1989). Stanley Stevens has done cultural ecological research on the Sherpas, their sustenance (agriculture and pastoralism), and the effects of tourism described in “Claiming the High Ground” (1993). I will also be referring to Christoph von Fürer-Haimendorf, one of the first anthropological explorers of the region, a man with a very interesting life story. At the breakout of World War II he found himself, a citizen of Austria and the word “Führer” in his name, trapped in Allied India. Fortunately for anthropology, the British authorities gave him a reasonable amount of freedom and he conducted his fieldwork while being confined to India and Nepal, much like during the previous World War Malinowski was trapped on the Trobriand Islands. He did extensive research among the Sherpas and has written several books about his findings, including: “The Sherpas of Nepal” (1964) and “The Sherpas Transformed” (1984).

2.2 Namche Bazaar and the Role of Tourism

The location of my fieldwork was Namche Bazaar, Nepal. The name is alternatively Namche, Namche Bazaar or Nauje. Sherry Ortner and Stanley Stevens use Nauje, and this is considered to be the Sherpa name. I use the Nepali name Namche, because during all my time there I did not hear anybody refer it to in any other way than that (probably because I communicated in English, not Sherpa).

As mentioned in the introductory section, Namche is physically isolated by mountainous terrain. There is an airfield about one or two day's hike away, at the village of Lukla. There is also a short airstrip just above Namche, but only small Pilatus Porter aircraft and certain helicopters can use it, because of its short length and the fact that it is located so high in the thin atmosphere. The usual route, however, is to fly from Kathmandu to Lukla (120 USD for the ticket) and hike from there. Another option is to hike from Jiri, which is the nearest bus stop. This takes about a week, depending on the physical ability of the hiker.

Namche village is located on a hillslope, with the lowest point at 3450 m ASL. It is the main economic hub of the region, with a weekly market on Saturdays. As this is the heart of

Sherpa territory, the inhabitants are mostly Sherpas, with a historic Rai minority. The economic and tourism boom of the recent decades has brought people from many other ethnicities (or castes, as they were rather seen by my informants) to seek jobs there. These people are mostly Rais and Tamangs, but it is possible to find many other peoples as well. Namche is in fact a very multicultural place, even excluding the tourists. The general language of communication is Nepali. Sherpas using their own language among themselves, but are switching more and more to Nepali (since they cannot read and write in Sherpa, but only Nepali, or some of them in English, Tibetan script is taught at the primary school now, so this situation might change in the future). Rais, Tamangs, Chaudharys, etc. all have their own languages, but not all of them are able to speak their indigenous languages, so they also use Nepali. In this paper I will use the term Nepali to denote a Nepali person regardless of the caste/ethnicity.

Namche is on the trail to the Base Camp of Mount Everest. Everest Base Camp is a very popular trekking destination for tourists and is called EBC for short. Most trekkers want to see Mount Everest and be able to say that they have stood at the foot of it. Besides EBC there are other attractions: smaller peaks and high passes for enjoying the views and putting yourself to the test.

There are two tourism seasons in a year: March to May and September to December. Summer is rainy season (with snow in the higher regions) and wintertime is too cold for comfortable trekking. Catering to the flow of trekkers can be considered the main economic activity in Namche. This takes three different forms: offering porter and guide services, keeping a lodge and a restaurant, or running a tourist shop. There is now tens of lodges in Namche, providing accommodation and food for all the trekkers and hikers. According to Stevens (1993, 363-364) these were originally, until the 1970-s, just ordinary Sherpa houses with a sign up front, inviting trekkers in for a meal and a bed. Nowadays, however, lodges are purpose-built houses with separate rooms and have little in common with traditional Sherpa houses (i.e. houses used before the rise of tourism). By the way, it is important to note, that previously Sherpas depended heavily on trade with Tibet as a source of income. In 1967 the Chinese government restricted trade with occupied Tibet, which was a major economic blow for the Sherpas. Fortunately this coincided more or less with the rise in tourism, which provided, after a short period of doubt and transition, an alternative source of income (Fürer-Haimendorf 1975, 3). Tourism, just like trade, could be made to fit well with traditional

agricultural and pastoral activities to diversify the sources of income. Now, limited over-the-border trade is again being carried out with Tibet, which has diversified the Sherpa economy even further.

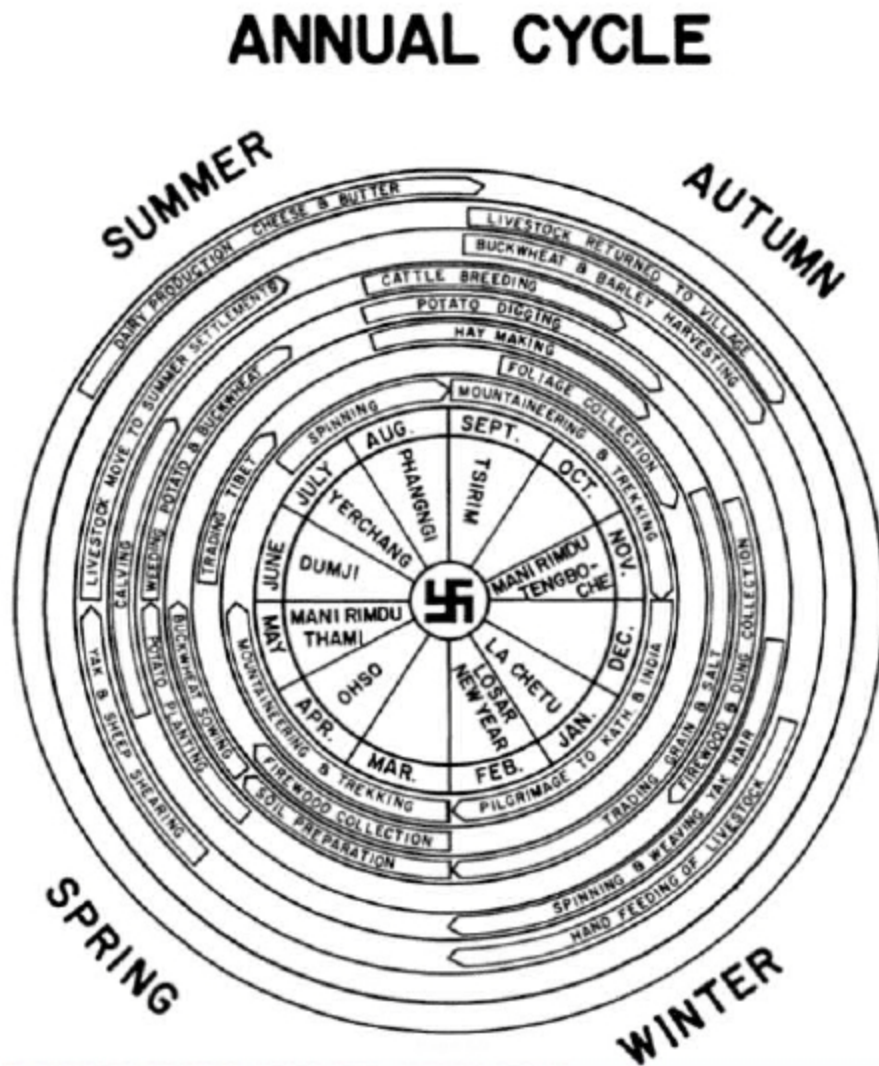


Illustration 3: The position of mountaineering and trekking (Mar-May and Sept-Nov) in the Sherpa Calendar according to Fisher (1990).

According to Stevens, even households which earned most of their income from tourism still continued to grow crops. He estimates that tourism accounts for 90% of monetary income for the Khumbu region, but nevertheless it is a supplementary activity. The Sherpas are aware of tourism's unstable character, its responsiveness to the ebbs and flows of global economy and have stated their readiness to return to agro-pastoralism if tourism loses its attractiveness

(1993, 371). While I did not conduct specific research on this topic, I noticed that my host family was cultivating potato, and all the terraces in Namche were being used for either crops or grazing cattle, so people were involved with agropastoralism at least to some extent.

Nevertheless, to think of this area as a “traditional” culture would be clearly a mistake.

As of now, Namche is visited by thousands of tourists every year (according to a wall chart in the National Park Office in Jorsale, the average tourist count of the last 13 years is about 23 000 per year, excluding the mountaineering expeditions, which probably amounts to a couple of hundred people per year). These tourists come from all over the world (I listed 20 different countries without really concentrating on it) and usually spend around two days in Namche to rest and acclimatize to the high altitude. The local people who speak English have plenty of contact with tourists during the high season. So despite its geographically isolated location it is much more in contact with other countries than most rural Nepali villages. And I haven't even mentioned the internet yet.

2.3 Modernity and Change?

Although due to the history of Nepal, some fifty years ago Namche Bazaar could surely have been considered a pre-modern society, together with concrete relations, memory-based history, personal and oral history and intimate relationships, it would be a grave mistake to still regard it as such. The “pure, untouched” state is described by the early explorers like von Fürer-Haimendorf (1964). He also described with much tragedy the social change resulting from the increasing tourist flow and contact with the money-based impersonal polluting western culture (1984). Whether to regard the social change as a fall from grace or a welcome progress in technology, medical facilities and social organisation remains up to the interpreter. But it is indisputable that the social change has been remarkable.

On the opposite side, it could be argued that the Sherpa society has always experienced change. Stevens describes a widely accepted view (1993, 213-214), according to which the Sherpa agropastoral history can be divided into three relatively static phases: early period of mixed agropastoralism brought from Tibet, the “potato revolution” starting in the mid-1800 (described by von Fürer-Haimendorf as a powerful event, resulting in a population explosion,

settling new and higher areas and a new level of prosperity, including a new number of temples and monasteries being built) and the “tourism period” where reliance on traditional subsistence has decreased and transformed by effects of mountaineering and mass tourism.

This view already would implicate a fair amount of change over the history, but Stevens argues against it and believes the truth to be even more dynamic. His research (1993, 214-215) has showed the local history not to be a static traditional existence with just two periods of rapid change (first potato, then tourists), but as a much more diverse and eventful evolution: “Sherpas do not support this view on their past. Khumbu oral traditions and oral history [...] tell a story instead of a more dynamic history of innovation and adaptation.”

While there has been a fair amount of change in the lifeways of the people there, in 1950 the characterisation as “pre-modern” still holds, since none of the change by then can be described as disembedding or lifting the social relations out of local context. There were mountaineering expeditions and job-seeking trips to Darjeeling in India, but the scale of these was small, compared to the onslaught of tourism triggered by the construction of the airstrip in Lukla in 1964 (Ortner 1999, von Fürer-Haimendorf 1984, Fisher 1990). This resulted in an explosion of change (see Illustration 4).

3 Fieldwork in Namche

It is early morning, about 6 o'clock. I have already been to the monastery to see the monks and their early morning chanting. Now I am sitting in front of Khumbu Cybercafe and waiting for my informant, Sameer, to come and open it. The mountain air is crisp and clean, the sun is just about to appear from behind the mountain-tops. A very enjoyable smell of juniper smoke is hanging in the air of the whole village. It is rising from small altars in front of the houses here and there. It is a Sherpa custom to burn juniper branches. It pleases the gods, they say. It certainly pleases me, at least.

The village street is quite empty, I saw one porter carrying some plywood, probably to a construction site somewhere. And then a *dzopkio* – a crossbreed of a yak and a cow. These wander around here on the streets all the time, on their own. I can hear morning village sounds: cocks crowing and somebody working with a saw somewhere. After ten minutes Sameer appears. He is happy to see me. He opens the padlock on the cybercafe door and we go in. Sameer has lots of switches to flip – the lights, all seven computers and monitors, and some extension cables powering the internet switches and wifi routers.

Half an hour later we are sitting behind a computer and surfing around on Facebook. Sameer has been upstairs, in the lodge, to eat breakfast. There is a monitor in the kitchen connected to a CCTV camera in the cybercafe, so he would see if a client appears. But there has been no clients – it is off-season and there is neither tourists nor locals on the move this early. So we are sitting and surfing the Facebook for now. Later some clients appear and I go sit in my usual spot in the corner, take my trusty notebook and start observing.

3.1 Two Months of Yaks, Cables and Cybercafes

To prepare for my project, I contacted an Estonian woman who is doing voluntary work in an orphanage in Kathmandu. She put me in contact with a hotel-owner in Kathmandu, who in turn referred me to a lodge-owner in Namche. The fact that just two people are enough to establish a link between an Estonian student and a Sherpa man in the Himalayas speaks something about the interconnectedness of our world.

I spent a total of ten weeks in Namche, starting in April with the tourist season in full

swing, and ending in June when the rainy season had started, stream of tourists had almost stopped and a significant part of the population of Namche had done the seasonal migration back to their home villages. Namche was half-empty at that time.

My lodgeowner, with whom I stayed, told me that the main man about everything to do with internet here is Nima Gyaltzen Sherpa from Khumbu Lodge: a local business leader, lodgeowner and internet service provider. He is operating a radio link which provides internet connection to Namche Bazaar and some surrounding villages. So I set up an interview with him, and started doing observation at his cybercafe, Khumbu Cyber.

3.2 Khumbu Cyber, Sameer and Rajesh

Khumbu Cyber is where I spent most of my time. It is located in the very center of Namche, on the ground floor of Khumbu Lodge and is a part of Nima's business operation consisting of a lodge, restaurant and a cybercafe, as is very usual here. Khumbu Cybercafe contains seven computers in one room about 10 m x 5 m. There are two small booths, one for international phone calls (VoIP) and another for standard intra-Nepal calls (NTC landline). There is also a copy-machine, a printer, a scanner and recharge cards for pre-paid mobile phones.

Most of my time in Khumbu Cyber I was sitting on a bench in the corner, passively observing and taking notes. Sometimes I had my camera out on a tripod and would point it and start recording, if something of interest was happening. People, tourists from different countries, Sherpas and other Nepali people, came and went, sitting behind computers and browsing the internet, buying recharge cards for mobile phones, making phone calls. Sameer, the clerk, helped anyone who needed assistance, and took care of the payments. He came and talked to me whenever he had a spare moment. I did not understand any of the Nepali dialogues except place names and numbers, so I would often ask him about what had happened earlier – he would explain to me who had said what. Of course, a lot of the information was still lost because of my inability to understand Nepali, but it was during these conversations I got most of the information I learned.

Sameer was one of my main informants. The other one was Rajesh, the other employee of Khumbu Cyber. Whereas Sameer sat in the cybercafe and attended to clients, Rajesh was the technician. He was usually out and about on errands, fixing the cable here and there or taking

care of some equipment. Rajesh is Chaudhary, which is a subset of the Tharu ethnic group (Gautam et al 1994, 328). He himself says that “Tharu is another word for Chaudhary”. His father is the headmaster of a school in a village halfway between Namche and Lukla. Nima knew his father and six years ago, when he needed somebody to take care of the technical side of his internet connection business, he asked around. So Rajesh has been working in Namche for six years now, intermittently also studying in a college in Kathmandu.

Sameer is a friend of Rajesh. They are both from Lahan, in Tarai region of Nepal. One year ago Nima asked Rajesh to invite a friend to work at Khumbu Cyber, they needed more hands. So Sanijv asked Sameer to come.

3.3 Namche Primary School and Dawa

The other important fieldwork arena besides Khumbu Cyber was the village school. The computer/mathematics teacher there, Dawa Sherpa, became my third informant. He was very friendly and gave me permission to film and observe his computer lessons. I attended three 5th grade computer lessons (I chose the 5th grade, because it was the oldest at this school). I also asked the lama teaching at the school for permission to film his lesson, and he agreed. So I also filmed one of his Tibetan language lessons, for comparison. It is quite usual in Nepal for schools to be English, and all subjects except Tibetan and Nepali were taught in English, which of course suited me very well.

3.4 Trips Along the Network

I took two trips with Rajesh, when he was doing his errands. First, it turned out that Khumbu Cyber was also providing internet connection to the school and he needed to go check why it is not working. We spent half a day checking the cable, fixing it here and there and configuring the network to get the connection to work.

On another day we took a hike with Rajesh to the radio link stations. Nima's internet connection works via a radio link between Namche and Udayapur in South Nepal. This is due to the fact that Nima and his associates had worked out how to achieve direct line of sight along a valley all the way from Namche to South-Nepal, where they have subscribed to internet connection from a Nepali ISP called Worldlink. Direct line-of-sight is required for the

radio link to work. The radio link is 120 km long, a fact that Nima is proud of and many people find it hard to believe according to him. This radio link extends the Worldlink connection onwards over the mountains into Namche. There were two of these radio link receiving stations (one for backup) and they were connected to Namche by optical cable. One was just on Namche border, the other about two hours away. I learned a lot about the technical aspects of the internet connection on this trip. From the receiving stations here at Namche another radiolink goes on to Chukkung, another to Imja – remote but popular trekking/mountaineering destinations – so Nima does not provide internet connection only to Namche but also to neighbouring villages.

3.5 Note-taking

For note-taking I used the method described in “Writing Ethnographic Fieldnotes” by Emerson, Fretz and Shaw (1995). First, I made jottings of keywords, names, figures and mnemonics during conversing/observing or immediately after, which I later converted into expanded fieldnotes (during a break or in the evenings). I also transcribed video interviews and dialogues into text.

In addition to fieldnotes I also wrote analytic texts, based on the method of Developmental Research Sequence, described by James P. Spradley in his “Participant Observation” (1980). This included extracting “terms” – words of cultural significance – and combining them into domains and taxonomies – categories of meaning. These domains and taxonomies can then be combined into paradigms by finding out the similarities and differences between the terms included in them. All this analytical work should continuously be checked and backed by observations.

As my fieldwork progressed I tried to follow Spradley's DRS method and keep up with the writing tasks he prescribes. I started out with a lot of enthusiasm which gradually decreased. It seems to me that DRS is more useful in prolonged or more intensive fieldwork situations with more material, in my case the organising potential of DRS did not really become viable, although some of the analytic texts I wrote were useful.

3.6 Video Camera Use

I had the camera with me almost always, but approximately half of the time I did not take

it out of the bag. Taking out the camera seemed to send a powerful message to the participants that “this now is important” which affected those participants who had not been exposed to the camera often enough to be “immune” to it.

There were two types of camera work – open observation and planned shots. I observed situations that I deemed interesting with a camera, without much thinking about what kind of shots I need, simply pointing the camera's attention where my informants seemed to be pointing theirs. In the cybercafe I often felt like I am intruding the privacy of the internet users, however. So I tried to establish some kind of a eye contact or nod-greeting instead of just starting to shoot a complete stranger (filmmaking is still more about working with people than working with a camera).

Planned shots formed a small part, maybe a tenth, of the shooting and were mostly scenic shots not involving important human activities. I kept a list of shots I needed, adding items into this list in some creative mood sometimes late at night when thinking about my project or reading something (Hampe 2007 gave me ideas often). Then I would sometimes go out with the camera specifically to “get that shot”.

The camera gave me the following abilities:

Comprehend unknown languages. Although my informants spoke some English, a lot of the intricate detail of social situations was being lost due to my lack of Nepali ability. I made a friend in Namche, a trekking guide named Ramesh, with whom I met in Kathmandu after my fieldwork period in Namche was over. We watched through my footage and he made me a rough translation of the Nepali dialogues. So the camera worked as a (delayed, but still) translation device, helping me reveal many important details about what had happened. While it would naturally be better for me to understand the language and hence the situations live, being able to understand foreign language situations is still a wonderful ability afforded to us by technology.

Intricate details of communication – when I observed the communication between participants, I was trying to concentrate on the content of what is being said. Watching the same dialogues from video tape over and over gave me insights about the unspoken communication – proxemics, body language.

As is typical in ethnographic analysis, or rather any analytic work, I used writing as an “extension of the mind”, writing out, formulating, phrasing, rephrasing and analysing my

thoughts in the text editor. I felt the need for a similar ability with video. A need for the ability to review my video material on a big screen and doing some preliminary rough form of editing – which is equivalent to “thinking” in a text editor, only its “thinking” with footage. Lightweight editing software allows this already quite easily and soon it will probably be a standard procedure on visual-anthropological fieldwork.

Metje Postma describes two different, to a certain degree conflicting, audiovisual styles in ethnography: description and narrative. The first is concentrated on “descriptive data” and its authority is determined by its representativeness and the precision of its description in relation to the “real event” (2006, 321). It is “technical” footage (not raw or unedited!). The other is an ethnographic documentary, a film with a particularity that lies in cross-cultural representation of the reality of the members of the other community, based on ethnographic understanding (2006, 325). That means a film in the “filmic” sense – with a narrative and characters. The former suits well for depicting cultural actions, the latter for people. Both are beneficial for written ethnography as well. Trying to find stories, concrete narratives presentable in the film also helps to order the knowledge about a cultural situation.

Of the text to follow, most of chapter 5 was discovered through working with the footage for filmmaking purposes, diving the material into scenes and finding narrative elements. Knowledge about dynamic elements of culture are easier to discover through a dynamic medium.

4 High Context Communication at the School

4.1 Computer Lessons at the School

I start describing my findings with the primary school because that really helped me open up the reality in Namche. I started my observation there a few weeks into my fieldwork, after meeting one of the teachers in the library and discovering through casual small-talk that they teach computers there.

The first thing to catch my attention as I entered the classroom was a poster on the wall, hand written on a huge red sheet of paper, with a black marker. The poster consisted of the following text (spelling mistakes in original):

“COMUTER

Definitions of “Computer”

- 1) The word Computer is taken from the word “Compute” means to calculate
- 2) A Computer is an electronic machine which can read, write and compute data.
- 3) A Computer is an electronic device which accepts data and instructions, processes them and gives processed output as information.

PARTS OF COMPUTER

- 1)CPU = Central Processing Unit
- 2)Monitor (VDU = Visual Display Unit)
- 3)Keyboard
- 4)Mouse
- 5)CD-ROM Drive
- 6)Floppy Disk Drive
- 7)Printer
- 8)Volt Guard

9)Speaker

More info:

Data = information about something

Instruction= Command, order

Processing= Changing data into useful information

Unit = Component

Produced by: Dawa Sherpa”

This is certainly a very formal approach to the topic of computers. The lesson itself followed a similar pattern of formality in speech. Here is a transcript of the video taken during the lesson:

Teacher: What are the capabilities of a computer. You! Tell me number one.

Pupil A: Storage, reliability... [teacher writing the words on a whiteboard as the pupil reads them]

Teacher: Accuracy... And number four?

Pupil A: Versatility...

Teacher: Number five?

Pupil A: Diligence and speed.

Teacher: And speed. [turns to others] all things are included or not?

Pupils together: Yes.

Teacher: And what do you mean by accuracy?

Pupil A: Accuracy means accurate. And mathematical...

Teacher: I have shown you a very good example in your computer yesterday. Bring your chair [sitting down behind a computer, showing, pupils gather around him] We are going to do mathematical calculation which gives the accurate results. Now we are going to do the mathematical problems operations under the addition, multiplication, subtraction and division in a computer with the help of a computer. Yes or no? [starts a calculator program in Windows] It is?

Calculator! Yes or no? When we are going to write $5 + 5$, it equals 10.
It is correct or not? Not correct yes?
Pupil B: Correct.
Teacher: Correct, good. That means accurate. Accurate answer or not?
Pupil B: Yes.
Teacher: So computer, one of the characteristics of a computer is accuracy. Yes or no?
Pupils together: Yes.

The rest of the lesson followed a similar pattern of learning abstract knowledge about the topic of computers, not so much about a real concrete computer as a tool or device to be used in practice. Here is an excerpt from the textbook¹, section “Main Points to Remember” of the chapter being studied during that lesson:

“A computer is an electronic device that accepts data and instructions, processes them and gives processed output as information.
Every computer has two types of memory: Primary memory and secondary memory. RAM and ROM represent primary memory. Hard disk, floppy disk, CD ROM are examples of secondary memory.”
(Khanal 2009, 14-15)

This was the first computer lesson at the school that I observed. On the same day I also started observations at one of the cybercafes. Here is an excerpt about the cybercafe for comparison.

Nima's [the owner's] sister is watching some Hindi music videos from youtube. The band is called Heartbeat. We are sitting with Sameer and chatting about his past. He shows me pictures of Dharan [a town he lived in for a while]. He tells me it is nice and clean, not like Kathmandu. He shows me pictures of several places in Nepal that he has a connection to. He googles “[placename]+pictures” to find photos

¹ The textbook is printed in 2009 but it is written in 1996, and while it is very thorough about the computer technology of 1996, it has seen only minor revisions. It still contains information about floppy disks (5¼" and 3½" with sizes in Kb), which have been totally obsolete for years. The textbook even contains a separate chapter on MS-DOS together with a “Practical”, where students learn to navigate around in the folders using MS-DOS command prompt commands. The teacher surely has freedom to skip the obsolete parts and Dawa seems to be aware of what is outdated.

of the places. Quite often (but not always) he double-clicks internet links unnecessarily. Never does he pay any attention to web pages with text. Only photos.

Computer use at the Khumbu Cyber is casual. Nothing about “versatility” and “diligence”. Just Youtube videos and photographs. I see people, both local and tourists, reading their e-mail, chatting on Skype and communicating to their friends on Facebook. No “accuracy” or “speed”. Just interpersonal communication. The computer class and the cybercafe seem to be two different worlds.

I tried to understand why these two worlds are so different. For one, the education setting in Namche in general struck me as quite authoritative. The teacher is addressed by the kids as “sir”, always. Before entering the class, the student asks for permission from the teacher. The schoolday starts with a lineup of all students, some marching exercises, a headcount and a singing of the Nepali anthem. All students must wear a school uniform.

Singing of the anthem every morning reminds us of the fact that the school is very sharply a government setting. It is an establishment conforming to the operational program set by the Nepali Government.

4.2 It's Government Service and Therefore Useless?

May 31st, we are sitting in Khumbu Cyber. Two young men enter and bring a device with them, that to me looks like a receiver from a radio dish antenna. After a brief chat with them, Rajesh explains to me, that it indeed is the receiver from the NTC mobile phone tower, from the antenna upholding the radio link south from here. Something is wrong with the device and they brought it here to fix it. The two guys turn out to be NTC employees. They chat with Sanjiv for about 10 minutes, then leave. The broken device stays here. Rajesh then explains to me, that the mobile phone tower is not working (I check my mobile phone and, indeed, there is no coverage). Rajesh tells me that tomorrow they will fix it. “If this was private company, they would fix it now. But it is government office, so they go tomorrow.”

This sentiment appears to be widespread: if something is government, then it is reason

enough for it to be not good. In an interview with Nima, the owner of Khumbu Cyber, he told me about the two mobile phone operators in Namche. There is Nepali Telecom or NTC, which is a “government” service and therefore “worthless”. And then there is Ncell, which is “private” and “a good service, owned by TeliaSonera, a Swedish company.”

“But it is government office, so they go tomorrow.” This statement seems to legitimise fixing a problem later rather than sooner. A “government service” and “worthless” are not two properties that often appear together. Instead, one property is the cause for the other: being “government” is a cause for being “worthless”.

This generalisation naturally brings to my mind the school. In the previous section we saw that I got a strong impression of the sharp contrast between the computer class and real-life at Khumbu Cyber. The school lessons were something that to me appeared to be quite pointless, as I did not see the kids learning any skills applicable in real-life situations. Obviously the school must be a completely worthless establishment, I thought.

Unless I am missing something.

It turns out I was missing something indeed, and understanding that took me a while. I will present my train of thought from this point on to realising how to interpret the school properly.

During fieldwork I had a talk with Natang, the owner of the lodge where I was staying. He told me that out of the eight teachers working at Shree Himalaya Primary School three teachers are paid by the government and five by the school management committee (meaning donations from the local people and from various international NGOs). The textbooks used in the computer lessons are standard textbooks used all over Nepal. In other words, the study program is fixed on state level, although the teachers make a selection out of that (like Dawa leaving out the outdated parts). According to the foreword of the textbook the computer class is compulsory only starting from class 9 (Khanal 2009, 3), but Namche school has decided to start computer education from class 3 already.

So in the field it seemed that the computer class is a mix of government policy from above and a local initiative (the teacher and the school management committee). I attributed the “uselessness” of the lessons to the “government” part of the education system. This is what I wrote in my analytic text straight after fieldwork:

The computer class at the school is not just another government service that does not work. It is a mix of a “worthless government service” and local initiative. The study program is provided by the government (in the form of a government approved but outdated textbook), and this is why the content of the lesson is in such a sharp contrast with real life internet experience at Khumbu Cyber.

But at the same time I remembered that Natang and Nima speak of the school quite highly and this kept bothering me. Why? Why did they not see it as useless as other government services? What was I missing?

I was missing a perspective to understand how education in Nepal, and to some extent the whole Nepalese culture, works. I found this perspective, during post-field analysis, from an article titled “Understanding Cheating in Nepal” written by a Peace Corps volunteer Chadwick Fleck (Fleck 2000). He, an American science teacher, was volunteering to teach English and science in rural Nepal. During exams he was surprised to see his pupils openly behaving in ways which according to his American perspective was blatant cheating – children were copying answers from each other and discussing questions among each other. This was in spite of him having explained it very clearly that he expects all his pupils to work independently and the children had all seemed to understand and explicitly agreed with that. Apparently Fleck had problems with correctly interpreting the events at a Nepalese school, just as I did. But if in my case it might have resulted just in a misinterpretation (and the writing of a bad master's thesis!), then in his case it resulted in a failure to work. And as so often in ethnography, we learn more from failure than we would learn from success. If Fleck had been able to contain his frustrations and continue working somehow, he might not have come to seeking help with interpretation. But he did seek help, turned to the theory of Edward Hall, and came to a revelation. Using Hall's notion of high-context communication, which I described in section 1.5, our American volunteer teacher was able to gain a better understanding of the Nepalese school and reconcile himself with it:

Textbook knowledge is not as highly or widely revered as it is in the U.S. Therefore, most people do not see education as valuable in and of itself; instead they believe that it is a means to an end. Education is valuable because of the social status a person gains by reaching higher levels of study. [...] More importantly, in Nepal's hierarchical culture,

education is a measure of social rank rather than knowledge. For example, a girl of the right caste, whose family has a good reputation and who has herself finished X years of schooling, may be a more attractive bride than a girl who has not been to school. Why is that so? Being educated is important not because the girl will become a good match for a boy intellectually, but because society — via the school — has recognized her and respects her. Those village girls who have completed some schooling will earn higher dowries for their families, too.

(Fleck 2000, 3)

So according to Fleck the school in Nepal is not so much about acquiring explicit coded knowledge as it is about attending formalities, obtaining social rank, building group ties and social relations. But of course, education in Nepal should not be regarded as merely “showing up” at the school. Gregory Bateson (1972) has coined the term “deuterolearning” to denote a certain type of a byproduct of the learning process – a certain kind of “learning to learn”. This means acquiring certain kinds of appreciative habits and abstract patterns of thought. And certainly a lot of deuterolearning goes on at Namche primary school and other Nepal schools: kids are learning about social hierarchy, authority, group values, etc.

One example from Fleck's account strikes me as particularly relevant:

During a school day, the average Nepalese teacher spends all of his or her class time lecturing to students — even to first graders — and expects them to sit quietly. Students respond in unison to the teacher's rhetorical questions, usually in the affirmative.

Male teacher: Nepal is a mountainous country in South Asia. Yes or no?

Students: Yes, Sir.

Male teacher: The world's highest peak, Mt. Everest, is in Nepal, isn't it?

Students: Yes, Sir.

[...] Communication between teachers and students is very limited, and kids learn to say, "Yes, Sir," regardless of their understanding of a statement or agreement with it, or disagreement with it. In a high-

context culture, the students do not challenge or dispute a teacher's point (Hall, p. 111). Questioning a point that a teacher has made is not seen as inquisitive; rather it is seen as confrontational. Questioning a teacher's ideas publicly may be an even greater offense.

[...] Students do ask questions in school, of course, but there are many unwritten rules for when, where, and how it is appropriate to do so. It depends on the context.

(Fleck 2000, 4; his reference is to Hall 1976)

Compare this to my account of what happened in the classroom. The similarity is striking!

4.3 Summary

So, I went to the computer class expecting to find something about how the people there see, use or teach computers and internet. I did not learn so much about the specific topic of computers, because the lessons were not implicitly about it. Thus they seemed useless to me. After interpreting my field data through Hall's concept of high-context communication I learned the meaning of these lessons plus a valuable lesson about how the Nepali society works in general: the school lessons seemed useless to me, because I was viewing them from my low-context point of view. In the context of Nepali society they were completely relevant and, through a deuterolearning-style mechanism, served their purpose (which was also the view conveyed to me by my informants): the kids were learning about attending formalities, obtaining social rank, building group ties and social relations.

5 Punctualisation Battle at the Cybercafe

So, during my fieldwork I considered the computer class at the school a somewhat strange phenomenon, a curiosity, something that did not really function as it was meant to. In contrast, I considered the goings on at Khumbu Cyber to be “real life”, internet in actual use. After gaining the insight about LC and HC communication styles, I now have an insight about how to interpret the situation at Khumbu Cyber as well.

In this chapter I will analyse the situation in Khumbu Cyber, my second observation arena. Khumbu Cyber is a place of many intersecting interests and influences. Traditional Namche village life meets the internet, tourists meet locals, Namche people meet people in other places over the telephone and internet.

5.1 Mediators to the Cyberworld

When some people, especially of the younger generation, generally speak good enough English and possess the computer skills necessary to function on the internet, then others, need help in navigating this new and alien terrain.

May 10th, 12:08. A Sherpa girl comes in, talking to Sameer in Nepali. Sameer opens yahoo.com, types the username as the girl dictates it. The girl then types in the password, but it's wrong. She then spells it to Sameer, letter by letter, using English alphabet spelling. Sameer types it in, still unsuccessful. The girl then tries again herself, this time it works, Yahoo mail opens. They proceed to read the e-mails: Sameer reads them to the girl loudly. They proceed through about three e-mails, all of them automated mail messages from Yahoo about administrative matters. The girl then pays to Sameer and leaves. Sameer tells me that she was waiting for a specific letter which had not arrived.

The above describes a more or less typical account of one of Sameer's main activities at the cybercafe – helping the Nepalis who come to the cybercafe to use the internet but do not have the skills for it. Using the internet demands a set of skills – a degree of command of the

English language on the one hand, and experience and knowledge on how the websites and various services work on the other. Sameer acts as an assistant, a mediator for them. There are many references to this function in my fieldnotes: “Sameer and a young man compiling a Nepali document in Word” or “Sameer helps a Sherpa girl fill out a PDF form – the Nepali passport application”. Here is another account:

June 8th. An old Sherpa man stands next to Sameer who is sitting at a computer. They have the old man's Gmail account open, Sameer has compiled an e-mail with several photos from the old man's digital camera attached. As Sameer later explained, they tried to send the photos to somebody in Kathmandu, but the e-mail address was wrong and the e-mail was not delivered. The old man makes a call from the STD phone to ask about the address. They try once more and then give up, the old man leaves. Sameer tells me that probably the first part of the e-mail address was right (the part that comes before the @ sign), but the latter was incorrect (confused gmail.com up with yahoo.com or some other).

Not all assistance efforts work out, but nevertheless the effort is there. Here is how Sameer himself commented on his assisting-mediating function in an interview:

“Nepali people did first not have idea about internet, they are coming to my cybercafe and they ask how do I send the e-mail. I am here, this computer in my cybercafe, and I am opening, creating accounts: Yahoo, Hotmail, Gmail, and so on. And every week they come here and I give them idea, teach. You can click here, first you can write e-mail id, and subject, and you can write mail and send. So little is one by one is using the internet, the Nepalis.”

So Sameer has, according to his own words and my observations as well, a role of a mediator or an interpreter for the people less skilled in internet use. Although the verb “teach” was used by Sameer here, when I try to elicit the role of a “teacher” in an interview, I meet with opposition:

Me: So as part of you work at the cybercafe you also work as a teacher? You teach the Nepali people?

Sameer: No no no, not like that. Only if they have a problem, we...
Me: Yeah that's what I mean.
Sameer: Like we show how to send an e-mail. But we are not teacher like...
Me: Yes, I do not mean you are officially like a teacher but I mean teaching is part of your job?
Sameer: No no no we don't.
Rajesh interferences here: There are basic courses sometimes off [the tourism] season by Prashu. And he is teaching.

My awareness of the difference between HC and LC communication styles already pays off here: whereas me, in my LC way meant that anybody who “teaches” other people is a “teacher”, for Sameer the word “teacher” has a totally different connotation, one entailing a certain status that has to be earned or attained through an elaborate social process. Therefore he refused to use this term for himself, although he agreed that he does “teach” users at the cybercafe.

Prashu Tamang, mentioned by Rajesh above, is a former employee of Khumbu Cyber who started his own cybercafe – Buddha Communication, a few hundred metres away from Khumbu Cyber – seven years ago. He comes to Khumbu Cyber sometimes and often participates in running Khumbu Cyber either by taking care of the technical matters or working as an attendant when either Sameer or Rajesh are away from Namche. He also runs a basic computer use course for Nepalis (unfortunately, during my time there none took place so I was unable to witness it). Sameer and Rajesh expressed it very strongly, that all the teaching of internet use belongs to Prashu. Apparently his status at the establishment was sufficient (he has been active there for nine years) to bear the title “teacher”.

5.2 Actors and Networks

Now let us come to the approach mentioned above: the actor-network theory. According to ANT we should consider the social to be made of actors influencing one another. An actor is something whose influence on other actors leaves a trace and can be seen. Khumbu Cyber can be considered to be an actor, since its influence on other actors can be seen. It draws people to itself, makes people talk about it and use its services. It is continually securing a

supply of needed resources – people, computers, furniture, electrical energy – for itself.

Another feature of the ANT approach is that a network of actors can be punctualised, perceived as a singular actor on another level. Khumbu Cybercafe is seen as a punctualised actor by most tourists – they come in, use the internet and other services provided there, pay and leave. For many local people (actors more closely integrated into the social networks of Namche) however, the components of the cybercafe continue to play independent roles, outside of the punctualised role of the cybercafe. Here is an example from a video transcript of a scene where a Sherpa man has entered Khumbu Cyber and is awaiting an e-mail from somebody. Apparently he has pre-arranged to use Khumbu Cybercafe's common e-mail address and they are checking that e-mail account now.

The Sherpa man and Sameer are sitting behind the computer, with Sameer operating the computer (meaning he has the keyboard and mouse). Sameer asks: “What was the name?”

Sherpa man: “Lulen.”

Sameer is scanning the list of names in the Inbox. They go through the list of e-mails that have been received, but the awaited letter is not among them. The Sherpa wants to clarify: “If it comes, will it be stored here?”

Sameer: “Yes.”

“Thank you,” and the Sherpa gets up and starts to walk towards the door.

Sameer, while still looking at the computer, says: “You have to pay for checking e-mails.” Then looks apologetically-smilingly at the Sherpa:

“Just 10 rupees.” [For comparison: a bowl of rice costs 200 rupees].

The Sherpa explains, with a smile and extended arms: “Look, sometimes I get from you, sometimes you get from me, so just let it be.”

Sameer explains with a quiet voice mixed with a bit of uncomfortable laughter: “Earlier we had good income from the tourist and it was OK not to pay for using the internet for just 5-10 minutes. But right now there are no tourists and we have to depend on just the

Nepali people...”

The Sherpa says now with a friendly-authoritative tone: “In my case it is different, because it wasn't me checking my e-mail. It was you.” Smiling, he backs away toward the door, about to leave. Sameer is unsure what to say, just says “Yes...”

Here we have two competing views: on the one hand, the old Sherpa tried to interpret the situation as simply one person (Sameer) doing a favour to another (him) – one in a long list of back and forth reciprocations (“sometimes I get from you, sometimes you get from me”). Sameer, on the other hand, refused this interpretation and instead provided his own version, where he presented Khumbu Cyber as an establishment that usually lives off of tourists², but was temporarily forced to extend the tourists' role onto the Nepali people (“usually the tourists provide income, but now it is off-season so we have to rely on Nepali people”). The clever Sherpa's answer was to show that he did not conform to the role forced onto him (“it was not me reading my e-mail, it was you”). He used the fact that he did not sit behind the computer and surf the web and do all the other things that tourists there do, and thus the role does not apply to him. After all, he was just sitting and watching. He pointed to the context to support his point. This was an example of HC communication, or in other words, the Sherpa's behaviour made sense in a HC way of thinking.

I think it is noteworthy that it is a failure that allows us to explore this situation: a black box is a system that operates as it should. If it does not operate as it should, it will also fail to be a black box. I witnessed many tourists using the internet, paying for it and leaving. In their perception (and also mine, at that moment), Khumbu Cyber was punctualised into a black box – an establishment where one uses the internet and then pays for it. It never occurred to me that it is in fact an effort, a process. It was only due to the fact that Sameer (representing the actor-network, Khumbu Cyber) failed to negotiate the terms in its favour this time, that I noticed this as a significant event, and thus perceived the continuous process, the ongoing effort of punctualisation on behalf of Khumbu Cyber.

In an interview, Nima the owner, mentions “social responsibility”: they are trying to earn

² For more about tourists as a source of sustenance, see Fisher 1990. On page 123 he mentions an analogy used by Sherpas: “...tourists are like so many cattle, representing highly mobile, productive, and prestigious, but perishable, forms of wealth. Like cattle. tourists give good milk. but only if they are well fed”. The term “cattle”, used by people with such long pastoral traditions, is devoid of any derogatory meaning here.

money from tourists and use it to improve life in Namche. The example above also illustrates that, when Sameer says to the Sherpa that usually they depend on tourists for income but since its off season, he has to charge him for reading e-mail. In other words, explanation is required for charging money for a service offered – this reveals how Khumbu Cyber is still entangled in pre-money economy contexts.

5.3 Summary

We saw two aspects of the cybercafe's operation in Namche: that many Nepali persons need mediators to use the internet and that a “punctualisation battle” is continually waged by Khumbu Cyber in order to continue its existence.

About the mediators – we saw social hierarchy in action, when Sameer and Rajesh refused to let themselves be called “teachers”. This is a title, dependent of the social context of the person carrying it, not a job description to be awarded abstractly to anyone who conforms to the verb “to teach”. As such the title is reserved only for persons higher up the ladder – like Prashu, the older employee who also had his own cybercafe by now and ran computer courses for the Namche people. This is an instance of the high-context aspect of the society.

About the “punctualisation battle” – the actor-network of Khumbu Cyber has to withstand forces that are trying to demolish it – for example, other actors who try to use its components for their own advantage. We saw that the stability of social structures is not a given, a granted feature. The actor-networks have to continually negotiate their terms of existence, continually align other actors into favourable positions. Here we also witnessed something about “lifting the social relation out of the local context” – Khumbu Cyber, in its attempt to extend the tourist-cybercafe relationship onto a Sherpa man experienced a setback, a failure. In this case it failed, sometimes it succeeds, and as such is illustrative of the battle of turning concrete personal relationships between people into contract relationships between a client and a service establishment, an instance of the “lifting out of social relations from the local context”. For the tourists this battle has already been fought, more or less, and in their minds Khumbu Cyber has been black boxed into a service establishment with payment as input and internet service as output.

6 The Internet as an Actor-Network

Khumbu Cyber is not, of course, a lone actor trying to align other actors in Namche Bazaar into a network that suits it. No, it functions only as a part of another, bigger network, a global community of internet users. As a cybercafe, it is attractive and interesting for people only if there are all these other people also using internet all over the world. To see how it functions in relation to this larger actor-network, we will take a look at how internet is used in Khumbu Cyber.

6.1 What Is It Used For

Based on my time spent in Khumbu Cyber, I compiled a cultural domain according to Spradley's DRS method:

Functionalities of the Internet in Namche

Nepalis	Trekks, tourists
1. Interpersonal communication: e-mail, Facebook, Skype. 2. Nepali pop music on Youtube 3. News sites: E-Kantipur, MySansar, Nepalnews 4. Photo services on Facebook, Flickr, Picasa 5. Calendar conversion websites for converting Bikram Sambat to the Gregorian calendar.	1. Interpersonal communication: e-mail, Facebook, Skype.

From an interview with Prashu:

Me: So who are the people who are using internet here?

Prashu: Internet... everyone using it. Especially tourist people... They want to keep in touch with each other and with family, some people doing business, with internet. And every people. Really necessary to

use internet now, yes.

Me: Would you say local people use internet?

Prashu: They use too, yea. They... since nine years ago quite a few people were using, but right now everybody is using. We give them really good knowledge, we teach them, you know, about internet, about the computers. Right now they can use very well, yea. [...] They want internet in any way. Like Facebook and things you know. And keep in touch with family, sending e-mail, getting e-mail, like. Yea. Its really necessary.

So the interest seems to be strong enough in the people, that they come to learn to use computers and internet use, it is not something that Prashu and other cybercafe owners have to advertise. People are willing to pay for the courses which can be just one session or a longer course of several sessions (Prashu: “Depends on the payment, if they want single course or more”). Tourists are travelling, so they used internet mostly just for short communication sessions (e-mail, Facebook). But some Namche people spent longer sessions online and their activities also seemed to be more varied (especially members of Nima's household and other related people who did not have to pay for it or had some special deal). As to with whom they are communicating depended on the person. People who work in the trekking and guiding business naturally develop friendships to their clients from other countries in the line of work, and these tourists and trekkers continue to communicate with their newly acquired Nepali friends after they have returned home. Others have friends and relatives who have gone to other countries to work. This seemed to be a significant amount, according to a discussion with Rajesh. Since long distance phone calls are expensive, the internet provides the only really affordable channel of communication.

Nepali government institutions have also adopted internet for communicating with its population. Rajesh used a government SMS service to find out the SLC results (School Leaving Certificate exam or end of elementary school exam results). The results are announced on a certain day for the whole country, so this is a huge undertaking. Earlier these results were published in a newspaper, but now there are online options. There is a webpage, where you type in the SLC number of the pupil and are supposed to get the result, but instead, somewhat typically for a “worthless government service” an SQL database error is displayed. So Rajesh uses an SMS service instead – he sends the SLC number and, on the second try,

gets the result.

Another example of this is the passport application procedure. In order to apply for a Nepali passport, a person needs to fill out an application form (a PDF on the Ministry of Foreign Affairs website), take it to a “government office” who checks it and sends it to Kathmandu. Then, after a certain wait period, if everything works out, the person should go to Kathmandu to collect the passport.

I went to Khumbu Cyber at around 14 o'clock. A lot of people were there. Apparently a government official had come to Namche and for about one week's time he would be here accepting the passport applications [I later learned that this was an all-Nepal effort to provide people with passports]. Many people are coming to Khumbu Cyber to fill out the form and print it out. Sameer is busy helping them. At the moment he is working with a 20 year old Sherpa girl Dawa. They are filling the application both for her and her sister (Dawa provides the information to fill in for her sister as well). After filling in the blanks they print it out and then she takes it to the government official.

Notice that Sameer is busy working as a mediator here again.

6.2 Accompanying Effects: Language, Alphabet and Calendar

Nepalis in Namche are using English alphabet to write Nepali words. Typing Nepali characters on the computer is technically possible but too complicated for most users for comfortable text input. For short phrases of personal communication (e-mail, text chats and Facebook) English alphabet works fine. Sometimes Sameer types Nepali documents in for someone – this is a service for a fee. The client later comes to collect the printed out text or has it sent to him as a file.

Dawa the computer teacher said there are Sherpa graphic designers in Kathmandu who even use Tibetan fonts to make Sherpa texts and graphics (Sherpa language is very different from Nepali and related to Tibetan), so he says it is possible but he has tried and not found a way to use Tibetan. This is accessible only for professionals, who cannot work without being able to use Sherpa writing.

The onslaught of the English language (and alphabet) is of course a part of a larger process³. It seems to be prestige language anyway, with or without the internet. An excerpt from the fieldnotes illustrates this:

I leave from Khumbu Cyber in the evening and go home to eat supper. My host family is sitting in the common room and watching TV in silence. I see some screen graphics with the contour of Nepal in it, so it must be a Nepalese channel. The TV show consists of two beautiful women, a filmstar and TV show host, walking on a beach somewhere and discussing movies, filmstars, what is it like to live like a filmstar, etc. I understand the conversation because it is in English. I eat my *dal bhat* and start to realise the function of English here as a prestige language.

Laura Kunreuther, who has worked extensively in Kathmandu, says that speaking in a mix of English and Nepali is typical of young educated Nepalis. Her informants said that “English is the international language” (2006, 331). I heard the same categorisation from everybody I talked to about this: Nepali is the national language and English is the international language. Internet is not the only factor enforcing the English language, just one of many.

Another thing enforced by the internet use is the Gregorian calendar. One morning Sameer logged into Facebook and got a birthday greeting from a friend. It was not actually his real birthday, he had just entered a random date when creating his account, because it is required by Facebook. When this date now arrived, his friends had gotten a notification about Sameer's birthday and had sent him greetings. This is an example of the black boxed technology breaking down and revealing something valuable again: the black boxed birthday notification system on Facebook did not work as intended (and thus failed to be a black box) and illustrated the calendar issues that I otherwise might not have thought of.

³ I compiled a domain of English words used in Nepali speech, which shows mostly words related to computers and mobile phones, but that might be due to my working context. The list of words in random order is: photocopy, recharge (as in a prepaid mobile), charge (as in charge batteries), (tele)phone, network busy (an error message often received on one's mobile), mp3 (emm-pee-three), mobile, card reader, webcam, Facebook, hello, sir, solar battery, inverter, tower (used as a synonym for “mobile network signal”). Numbers are said in English in a phone number context or as a western calendar year number.

So now I noticed that all the websites and software presume the usage of the Gregorian calendar, which is incomprehensible to someone used to Bikram Sambat, the Hindu calendar in use in Nepal. Of course they will learn it gradually through computer use. Interestingly, spelling the month names and date numbers in English is the norm when talking about Gregorian dates in Nepali, so that “June eleventh” is said as an English expression in the midst of Nepali speech.

So the larger actor-network of the global internet is aligning the local actors in Namche into using English alphabet and even some English expressions, plus the Gregorian calendar. It is mutually reinforcing with other global influences brought by tourists, enforcing English language and calendar. The internet also provides its mediator actors – date converter websites for the calendar conversions and cybercafe clerks like Sameer for typing the text.

6.3 Sameer's Facebook Friendship

Now let's see an example of a human relationship in this new medium. About one month before I started my observations at Khumbu Cyber, Sameer had acquired a friend on Facebook, a girl named Rose from Indonesia. This previously unknown girl had sent a friend request to Sameer, indicating her wish to communicate with him. Sameer accepted the request and they started chatting. I tried to find out about the circumstances of their becoming friends, but Sameer could not offer me any more explanation than that. She had simply requested friendship and then they started chatting. This ease with which they started an online relationship reminds me of Laura Kunreuther's account of spontaneous friendships started over the phone network in Kathmandu (2006, 336):

During the mid-1990s, youths began using the phone to connect with otherwise inaccessible acquaintances through what is colloquially known as a *blaf kal* (bluff call). The friendships that develop through these bluff calls are often between two people who have been introduced by a mutual friend, or they are the result of a misdialled or randomly dialed number. A 23-year-old daughter of a family I frequently visited often invited a young man to the house and to family events. [...] One evening I asked him how he and Sarjana had met. He replied matter of factly, “On the phone.” He had apparently

misdialed his friend's number and reached Sarjana instead. They began to talk and quickly became regular "phone friends." Only after a year or so of this phone relationship did they begin to meet each other in person.

Kunreuther then refers to Joshua Barker's work from Indonesia. It turns out the bluff call or similar phenomenon is also known there (Barker 2002, 166). These bluff calls have one common characteristic both in Indonesia and Nepal: they are used by young people to get away from the traditional social control of the community. Barker on Indonesia:

The disciplines normally brought to bear on public and domestic spheres (by way of overhearing and seeing, for example) became far less effective in controlling communication (Barker 2002, 166).

And Kunreuther on Nepal:

Relationships that develop over the phone in these bluff calls are powerful because they appear to be unentangled and circumvent the usual pressures of family and social control. [...] The ability to connect over the phone becomes a context that many young people in Kathmandu describe in subjective terms as a "freer" and "more real" emotional attachment (Kunreuther 2006, 336).

Although Kunreuther draws parallels between the bluff call phenomenon and the internet chats (2006, 349) in that both are characterised by intimate conversations among strangers, we cannot simply draw a parallel to Sameer's Facebook friendship. Telephones were anonymous (at least before the caller ID function became ubiquitous on mobile phones), and so were the internet chat-rooms popular during Kunreuther's research, but Facebook is different because everything a person says carries his/her name and can be seen by that person's friends (since I am on Sameer's friend list I gained access to their conversation). So while a Facebook relationship might be freer and unentangled from usual family and social control, it just as well might not be. Depends who else is watching. By the time their relationship dwindled to a stop in July (because Sameer left Namche for a seasonal trip to Kathmandu and did not spend that much time behind a computer any more), Sameer had acquired six Indonesians from Rose's friends list and Rose in turn had acquired four Nepalis from Sameer's list. This indicates that their relationship was in no way private, and was of interest to their friends. Yes, Facebook users in Nepal are mostly young people, as probably in many places in the world,

so even if there are other people watching, the nature of social control can be very different, but since it remains a very implicit phenomena, it is hard to explore.

We can explore the relationship of communication to context, however. An excerpt from my fieldnotes:

We are sitting in Khumbu Cyber again with Sameer, passing time on Facebook. Sameer sent a joke message to Rose about having sent her a box of candy, he ran this message through Google Translate from English into Indonesian. Now Rose replied in Indonesian. Google Translate translates it into something to the effect of “Thanks, next time send me real ones hahaha”.

This actually is an exception in the sense that usually their conversations proceeded in English. But here the box of candy, although HC communication, certainly works.

The excerpt continues:

Sameer starts telling me: “I don't like Muslims [referring to the fact that Indonesia is a Muslim country]. They are naughty, they want to make their own community. In Gaighat [Sameer's hometown] there is many of them. They are always on the mic [the call to prayer from the loudspeakers]. Its a headache for me. They are dirty, always bargaining.”

So the fact that Indonesians are Muslims was ran through Sameer's local context, attributing certain characteristics to them. But still he entertains the following thought, and speaks it in front of my camera:

“She is already married [according to her Facebook profile]. Too bad for me. I don't want to meet her. Maybe she has a baby.”

While it is possible to communicate with any person from any country, relations on the internet are still bound by the offline ties – Sameer mostly converses with people he knows also outside of the internet.

6.4 Monks and Nuns

Another important fact is the relation to an older hierarchical system: that of the Sherpa

Buddhist religion. There was a monastery in Namche which was inhabited by monks and nuns seasonally (during certain events and ceremonies). These monks and nuns would come to Khumbu Cyber but only to buy recharge cards for their mobile phones. Only once during my whole fieldwork did I see someone from the religious establishment using the internet.

When monks and nuns did come into the cybercafe, they were being treated a little like children – spoken to with simple sentences, asked several times the same question, etc. They seemed to be living in their own world, a parallel system, and the internet apparently had nothing to offer them, internet has failed to align them, at least for now.

6.5 Chronological View

To understand the situation thoroughly we will now look at the historical development of internet at Namche as well. Rajesh says in an interview:

At first it was the tourists, coming here and asking where is the internet. Nepali people then see the tourist, how they use internet, how to use mail, how to send pictures. So now is good, little by little the Nepalis is similarly using the internet now.

I interviewed Nima, Sameer and Rajesh about the history of internet in Namche. According to them it were the tourists who first brought the demand for the internet. Internet connection was then provided by Sherpa entrepreneurs and lodgeowners to cater to tourists but also to the Nepalis who had contacts with tourists and became friends with them (this usually means porters, trekking guides and lodgeowners). Tourists transfer the knowledge of how to use the internet to these Nepalis. So in a way, tourists bring with them both a need (by becoming friends with Nepalis and wanting to communicate with them) and the means to satisfy that need (teaching the computer skills needed to use the internet).

Since the opportunity was then already established, the Nepalis also communicate with each-other, including Nepalis working in other countries. Nepalis often had another rate than tourists. (At the time of my research in Khumbu Cyber the “local price” was half of the “tourist price”). Therefore Nepalis used the internet mainly for interpersonal communication (Facebook, e-mail), just like the tourists on their travels. During early times internet connection worked via a satellite link and thus was quite expensive, so tourists did not spend a long time online. Typical usage was a quick session, quickly going through your e-mail and

sending a few “I am okay, its wonderful here!” e-mails to friends. Now the prices have gone down and tourists take longer sessions, but since they are on the move it is still the locals who are using internet more extensively and have also developed more elaborate uses for internet than tourists.

Now as revealed by fieldwork, together with the internet (and tourism) came the adoption of other things: English alphabet also for communication between the Nepalis on the internet (as the Nepali alphabet is more difficult to learn and use on a computer), Gregorian calendar (as all the websites and most tourists use this calendar as opposed to the Bikram Sambat) and English terms and concepts (memory card, printer, camera, etc) that are working its way into everyday Nepali language.

Also internet has started to replace other communication channels not originally meant to be affected: people now avoid making long distance phone calls to friends and relatives abroad, and instead use internet chat or voice programs. Also newspapers are no longer carried to Namche since this takes many days due to the physical isolation. People prefer to read the online versions instead of two days old news on paper.

Recently NTC started offering its cheaper but also less reliable ADSL connection in Namche, so there is now even competition between the internet connection providers. There is a large number of cybercafes in Namche. Some of them still rent a part of the radio link connection from Khumbu Cyber, but some of use the NTC connection now. The bandwidth that Nima is using for his radio link used to be 3 MB/s, but is now reduced to 2 MB/s (according to Rajesh). This reflects the decreasing number of clients (dropped from 26 to 15). Rajesh says this is a result of NTC offering its cheaper, although less reliable (NTC is, after all, a government owned company) ADSL connection in Namche.

6.6 Discussion

We learned that the demand for internet was brought by tourists, who wanted to use the internet during their visit, and wanted to communicate to their newly made friends in Namche after they went back. Then, since the internet was already there, it quickly found other uses: Nepalis started to use it for communication among each other, and with friends and relatives abroad. It also came to replace newspapers and to some extent the telephone. The Nepali government is also using the internet to communicate with its population now. Government is

endorsing computer education at the schools. Real life development and use of internet is happening at cybercafes, though. Some people need help with using internet, and mediating the cyberworld to these users is also part of the job of cybercafe clerks like Sameer.

This history makes sense. Since actors and networks are mutually constitutive, no network can form by itself out of thin air. There were actors that were already aligned into using the internet – tourists, their digital cameras, friends and family back home. Their existence resulted in a local actor-network – the Khumbu Cyber – being formed out of local and imported actors (hardware, people, etc.). Khumbu Cyber in turn started affecting other local actors and aligning them into its network (except some, like the religious establishment, which is offering resistance). Now there are other actors that are making use of these already aligned actors – the people of Namche are already used to the internet, thanks to Khumbu Cyber. NTC has started offering its ADSL connection, Nepali government is making some of its administrative services available online, etc.

Other effects are being felt as well – the bigger, global actor-network of internet is aligning the actors in Namche into using English language and Latin alphabet – succeeding only partially since Nepalis use Latin alphabet but mostly still Nepali language. Gregorian calendar is enforced on most websites. Mediator actors are used for this – convertor websites and cybercafe employees.

New kind of human relations develop online, in a different social control environment disembedded from the village setting – it is different people who are watching, therefore participants try to conform to different rules. It is possible to talk to people from all over the world but most conversations naturally happen between people who know each other outside of the internet as well.

7 Conclusion

My aim with this paper was to explore how internet is adapted into an environment culturally different from the one where it was spawned. For this purpose I borrowed the actor-network theory taken from science and technology studies, and followed how actor-networks are created and maintained. I also needed the concept of high-context and low-context communication in order to make sense of the cultural processes.

Internet used to be seen as a monolithic placeless cyberspace which would make us all similar to each other. My main finding is that this is not always the case. It is a collection of different people doing different things while embedded in their social contexts. Here is how I came to understand that.

First I observed the computer lessons at the Namche primary school, which seemed very useless to me, children did not seem to actually learn much about computers or their use. At the cybercafe, an arena of “real life and hands on” computer and internet use, completely different things were going on. Cybercafe clerks were acting, for clients less skilled in the internet use, as mediators to the cyber world, as guides to people who did not know how to behave in this new cyberenvironment, both technically (setting up e-mail accounts, teaching where to click, etc.) and culturally (with the English language and Gregorian calendar), creating a new role for themselves in the Namche society. In spite of that they denied their role as a “teacher”.

The explanation for both these two findings is that the society in Namche works in a more high-context way than I was used to. A teacher is not simply someone who teaches but someone who has gone through the necessary social processes and has earned the title of “teacher”. Similarly, the purpose of attending school is not only to obtain knowledge, but to learn a lot more through deuterolearning: about formalities, social rank, group ties. The meaning of the pupils' going to the school is not derived only from the explicit knowledge they learn at the school, but more from the fact of going to the school itself, and from learning to behave in certain ways and interacting with certain people. Context is also why Sameer and Rajesh do not agree to let themselves be referred to as teachers – although they do transfer knowledge and skills to clients in the cybercafe they lack the necessary context that a person

with the title of “a teacher” needs to have. They are simply cybercafe clerks who help their clients. Prashu, on the other hand, has the necessary “level” to be honoured with this title.

With my awareness honed to this high-context style, I turned to the cybercafe. I found that it is not simply a business establishment in a low context sense – its meaning is not simply to provide a service and charge a fee for that. It had to continually fight “a punctualisation battle” in order to exist as an actor-network, and not be demolished into component actors.

The cybercafe is, of course, not a lone actor, but is in turn a part of a larger actor-network, the global internet, which on the one hand gives meaning to the cybercafe and on the other hand enforces certain effects – English language, Latin alphabet and Gregorian calendar.

Networks do not form out of thin air – they form there where actors are already present and trying to do something. Nima established Khumbu Cybercafe because there were tourists already present. Nima and his radio link connection have accustomed the people of Namche to the internet, and now other actors are coming in to take advantage of this. Another service provider is entering Namche to take its share of the client base. And the Nepali government is also using the internet to communicate to its population.

These shifting alignments and alliances between actors bring about new social hierarchies and roles. Sameer, Rajeesh and Prashu have the role of mediators or gatekeepers to the cyberworld. It is them – relatively young men, teaching old men, a somewhat of a reversal of usual social roles. This new cultural practise brings along a cultural power shift, arise of a new powerful skillset: proficiency in English plus skills of computer usage.

8 References

Films

Conquest of Everest, UK 1953, Dir George Lowe (available on www.youtube.com/movie?v=Kj9OFJsyXio)

Literature

- Ahola, Alphonse Ndem. 2005. Cyber Dreams. Online and Offline Dealings in Cyber Cafes in Ngaoundere (Cameroon). Master Thesis, University of Tromsø.
- Bateson, Gregory. 1972. Social Planning and the Concept of Deuterolearning. In *Steps to an Ecology of Mind. Collected Essays in Anthropology, Psychiatry, Evolution, and Epistemology*. Gregory Bateson, 127-138. San Francisco: Chandler Publishing.
- Barker, Joshua. 2002. Telephony at the Limits of State Control: "Discourse Networks" in Indonesia. In *Local Cultures and the "New Asia"*. ed. C. J. Wan-Ling Wee, 158-183. Singapore: Institute of Southeast Asian Studies.
- Callon, Michel. 1986. Some elements of a sociology of translation: domestication of the scallops and the fishermen of St Brieuç Bay. In *Power, action and belief: a new sociology of knowledge?* ed. John Law, 196-223. London: Routledge.
- Callon, Michel. 1991. Techno-Economic Networks and Irreversibility. In *A Sociology of Monsters: Essays on Power, Technology and Domination*. ed. John Law. 132-165. New York: Routledge.
- Castells, Manuel. 1996. *The Rise of the Network Society*. Oxford: Blackwell.
- Emerson, Robert M., Rachel I. Fretz, Linda L. Shaw. 1995. *Writing Ethnographic Fieldnotes*. University of Chicago Press.
- Eriksen, Thomas Hylland. 2007. *Globalization. The Key Concepts*. Oxford: Berg Publishers.
- Fischer, James F. 1990. *Sherpas: Reflections on Change in Himalayan Nepal*. Berkeley: University of California Press.
- Fleck, Chadwick. 2000. Understanding Cheating in Nepal. *Electronic Magazine of Multicultural Education*. Vol. 2, No. 1. <http://www.eastern.edu/publications/emme> (accessed March 8, 2012)

- von Fürer-Haimendorf, Christoph. 1964. *The Sherpas of Nepal. Buddhist Highlanders.* University of California Press.
- von Fürer-Haimendorf, Christoph. 1975. *Himalayan traders: Life in Highland Nepal.* London: John Murray.
- von Fürer-Haimendorf, Christoph. 1984. *The Sherpas Transformed. Social change in a Buddhist society of Nepal.* New York: Sterling Publishers.
- Gautam, Rajesh and Asoke K. Thapa-Magar 1994. *Tribal Ethnography of Nepal. Volume I.* Delhi: Book Faith India.
- Gibson, William. 1984. *Neuromancer.* New York: Ace Science Fiction Books.
- Giddens, Anthony. 1990. *The Consequences of Modernity.* Cambridge: Polity Press.
- Hall, Edward Twitchell. 1976. *Beyond Culture.* New York: Anchor Books.
- Hampe, Barry 2007. *Making Documentary Films and Videos. 2 edition.* New York: Holt Paperbacks.
- Khanal, R. C. 2009. *Computer Concept for Class V.* Ekta Books, Kathmandu, Nepal.
- Kunreuther, Laura. 2006. Technologies of the Voice: FM Radio, Telephone, and the Nepali Diaspora in Kathmandu. *Cultural Anthropology* 21 (3): 323-353.
- Latour, Bruno. 1987. *Science in Action: How to Follow Scientists and Engineers Through Society.* Cambridge, MA: Harvard University Press.
- Latour, Bruno. 1992. Where are the Missing Masses? The Sociology of a Few Mundane Artifacts. In *Shaping Technology / Building Society. Studies in Sociotechnical Change.* ed. Wiebe. E. Bijker, John Law. 225-259. Cambridge, MA: MIT Press.
- Latour, Bruno. 1993. *We Have Never Been Modern.* Cambridge, MA: Harvard University Press.
- Latour, Bruno. 2005. *Reassembling the Social.* New York: Oxford University Press.
- Law, John. 1992. *Notes on the Theory of the Actor Network: Ordering, Strategy and Heterogeneity.* Lancaster: Centre for Science Studies, Lancaster University.
<http://www.comp.lancs.ac.uk/sociology/papers/Law-Notes-on-ANT.pdf> (accessed April 26, 2012)
- Law, John and John Hassard. 1999. *Actor Network Theory and After.* Sociological Review Monographs). Oxford: Blackwell Publishers.
- MacKenzie, Donald and Judy Wajcman. 1999. Introductory Essay. In *The Social Shaping of Technology, Second Edition.* ed. MacKenzie, Donald and Judy Wajcman. 3-28.

- Buckingham: Open University Press.
- Meeker, Mary. 2011. Internet Trends: presentation at Web 2.0 Summit, San Francisco, October 18, 2011. <http://kpcb.com/insights/internet-trends-2011> accessed at 9th Nov, 2011.
- Miller, Daniel and Don Slater. 2000. Internet: An Ethnographic Approach. Oxford: Berg Publishers.
- Miniwatts Marketing Group. N. d. Internet World Stats. www.internetworldstats.com (accessed May 14, 2012).
- Ortner, Sherry B. 1978. Sherpas Through Their Rituals. Cambridge University Press.
- Ortner, Sherry B. 1989. High Religion: A Cultural and Political History of Sherpa Buddhism. Princeton: Princeton University Press.
- Ortner, Sherry B. 1999. Life and Death on Mt. Everest: Sherpas and Himalayan Mountaineering. Princeton University Press.
- Postma, Metje. 2006. From description to narrative: what's left of ethnography? In *Reflecting Visual Ethnography: Using the Camera in Anthropological Research*. ed. Metje Postma, Peter I. Crawford, 319-357. Leiden: CNWS Publications.
- Spradley, James P. 1980. Participant Observation. San Diego: Holt, Rinehart and Winston.
- Stevens, Stanley F. 1993. Claiming the High Ground: Sherpas, Subsistence, and Environmental Change in the Highest Himalaya. Berkeley: University of California Press. Available online at: <http://ark.cdlib.org/ark:/13030/ft8b69p1t6/>
- Wiener, Norbert. 1948. Cybernetics: or Control and Communication in the Animal and the Machine. Cambridge, MA: MIT Press.
- Winner, Langdon. 1980. Do artifacts have politics? *Daedalus*, 109: 121-136.



Voice Over IP Overview: Services, Architectures, Ordering, and Billing

Ming Lai

Order and Service Management Systems

732-699-2626

mlai@telcordia.com

May 19, 2003

An SAIC Company

VoIP Sea and Islands

VoIP Net

Island 1

VoIP Net

Island 2



IP
Continent

PSTN
Continent

Introduction

VoIP is here now and is growing rapidly.

The regulatory issues are being addressed. The benefits from VoIP technology of efficient use of networks and new enabled services are much greater than the cost cutting via circumventing regulatory charges

Retail VoIP service ordering and billing solutions for different VoIP architectures are being developed with different maturity.

Interconnection and wholesale/resell business processes and data exchange among VoIP related service providers lack industry wide coordination.

OBF can play an important role to connect existing and emerging VoIP islands cost-effectively in ordering and billing.



Outline

What and Why VoIP

VoIP Services

VoIP Market and Evolution

VoIP Technology Overview

State of VoIP Technology Adoption

VoIP End Point Connection Types

Key Differences of VoIP Ordering and Provisioning from
Circuit Switch Voice Services

Key Differences of VoIP Billing from Circuit Switched Voice
Services

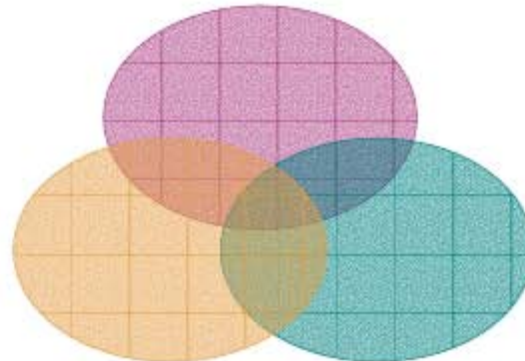
Issues: Regulatory, Business, Standards

What is Voice over IP (VoIP) ?

Voice over IP is the technology to transmit voice over IP networks and the associated services enabled by the technology

VoIP Services

Telephony Services



Other IP Voice Applications

Unified Communication Services

VoIP Services

Telephony Services

- **POTS/Class/Voice Mail Services (Internet/IP Telephony)**

 - PC to PC

 - PC to Phone

 - Phone to Phone

 - Phone Card (Pre-Paid or Post-Paid)

- **Centrex/PBX Services (IP Centrex/PBX)**

- **IN Services**

 - Toll Free, Time-of-Day Routing, Voice VPN, Area Code Selection, Voice Dialing, SCP-enabled services,

Unified Communication Services

- **Multimedia/Mixed Media Communication**

 - Instant Messaging, On-demand Conferencing, Presence Management, Collaboration, Media Streaming, Unified Messaging, Caller Image/Info Delivery,

- **Web/Data/Voice Integration**

 - Click to Talk from Web or E-Mail, Directory Dialing, ENUM (E.164+DN), Real-Time Feature Parameter Changes, Call Control and Logging, Automated Attendant,

IP Voice Applications

- **Enterprise Applications**

 - Distance Training/Learning, IP Contact Center, Voice Portal, Voice Enabled Transaction and Content Services, Voice Web Advertisement, Tele-medicine

- **Personal Applications**

 - Multi-Modal Navigation/Map, Voice Enabled Information Services, Gaming with Voice and Data

Why VoIP

1. Cost Savings

- Efficient Use of Network Bandwidth for Voice and Other Traffic
- Enabling Customer Self Service to Cut Down CSR Costs
- Leveraging the Maturity of IP Technology, Competition of IP Equipment, and Broadband Internet Access
- Enabling Business Customers to Reduce Telecom Management Costs of Voice and Data
- Lowering the Toll Costs for Customers Thru IP Network

2. New Services

- New and High-Margin Telecom Service Revenues for Carriers
- Satisfying Customer's Needs for More Convenience and Unified Communications
- Enabling Disaster Recovery and Remote Operations for Business Customers

VoIP Service Market

1. IP telephony service providers handle about 1.12 B calls/month (1/2002); 6.8 B calls in 2001; \$1.7 B revenue from intl. calls - about 5% of total International minutes
2. Other VoIP services reach about \$25B in 2008, growing from < \$2B in 2003

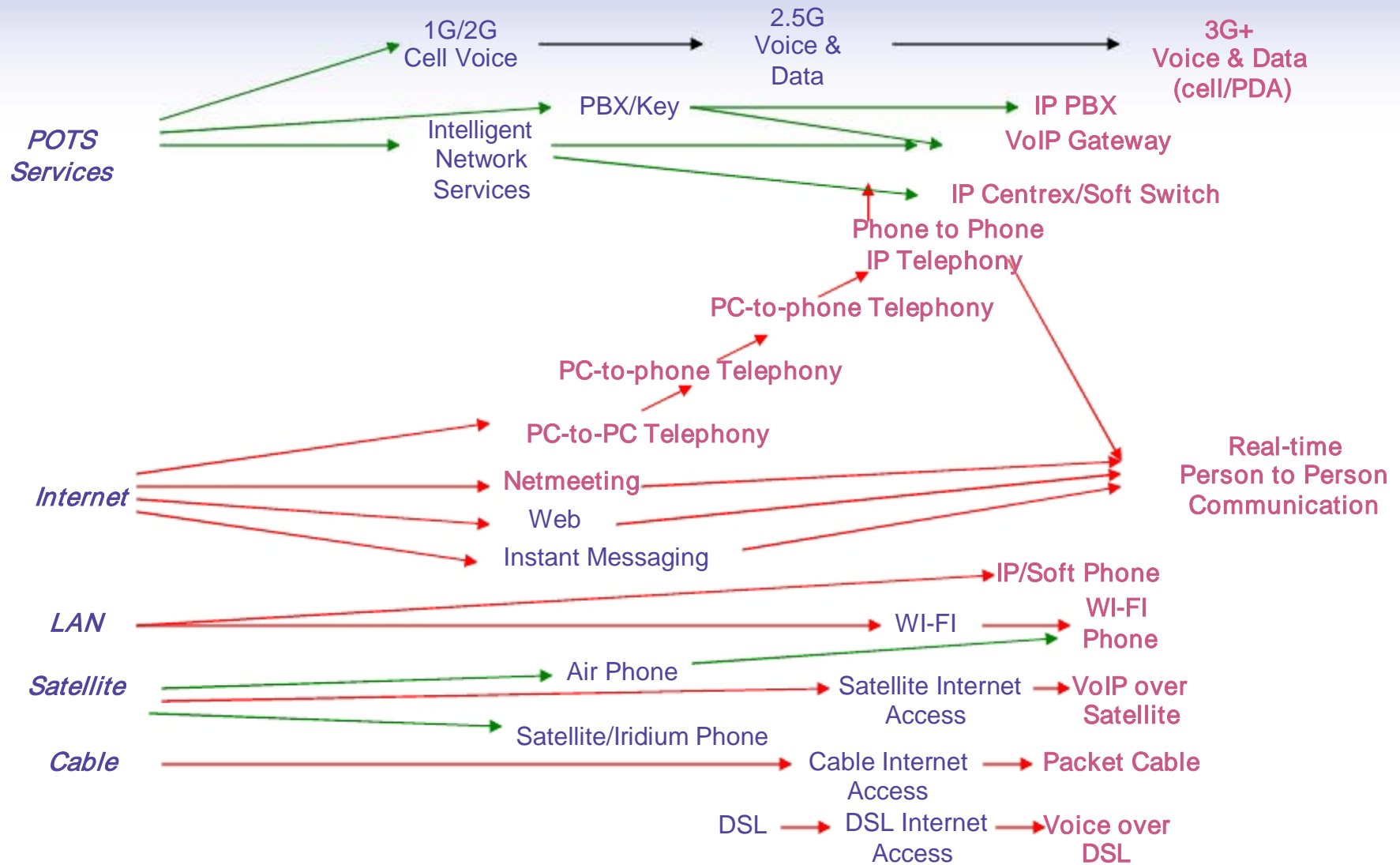
Sources:

(1) iLOCUS, 2002

(2) Telcordia Technologies analysis from industry sources: IDC, Frost and Sullivan, Yankee Group, EletroniCast, CyberEdge Information Systems

(3) IDC, U.S. Contact Center Consulting and Implementation Service Forecast and Analysis, 2002-2006, April 2002.

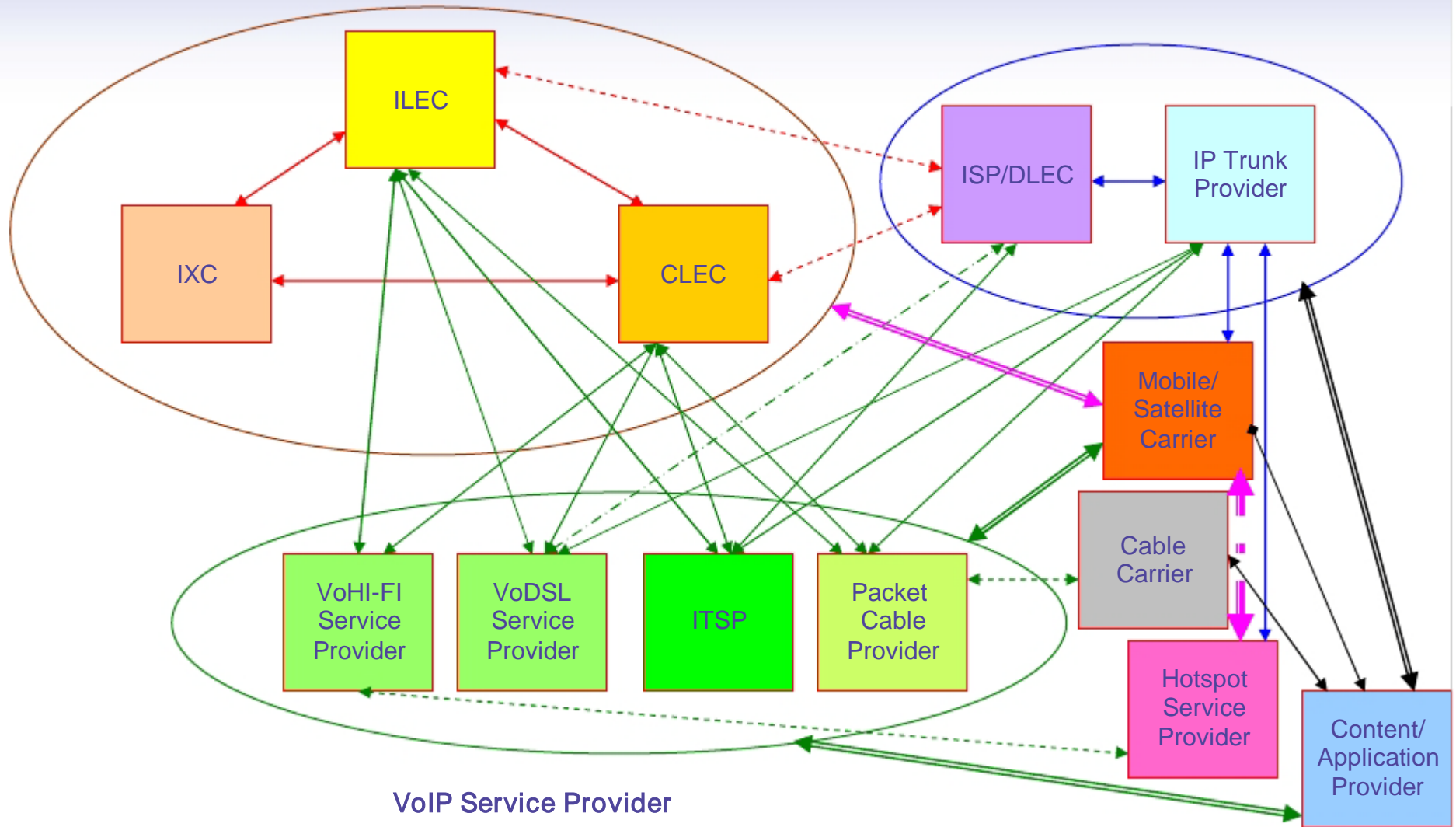
Voice over IP Evolution - End User View



Business Relationships Among Carrier Types – Ordering and Billing

Wireline Voice Service Provider

IP Service Provider



VoIP Technology Overview

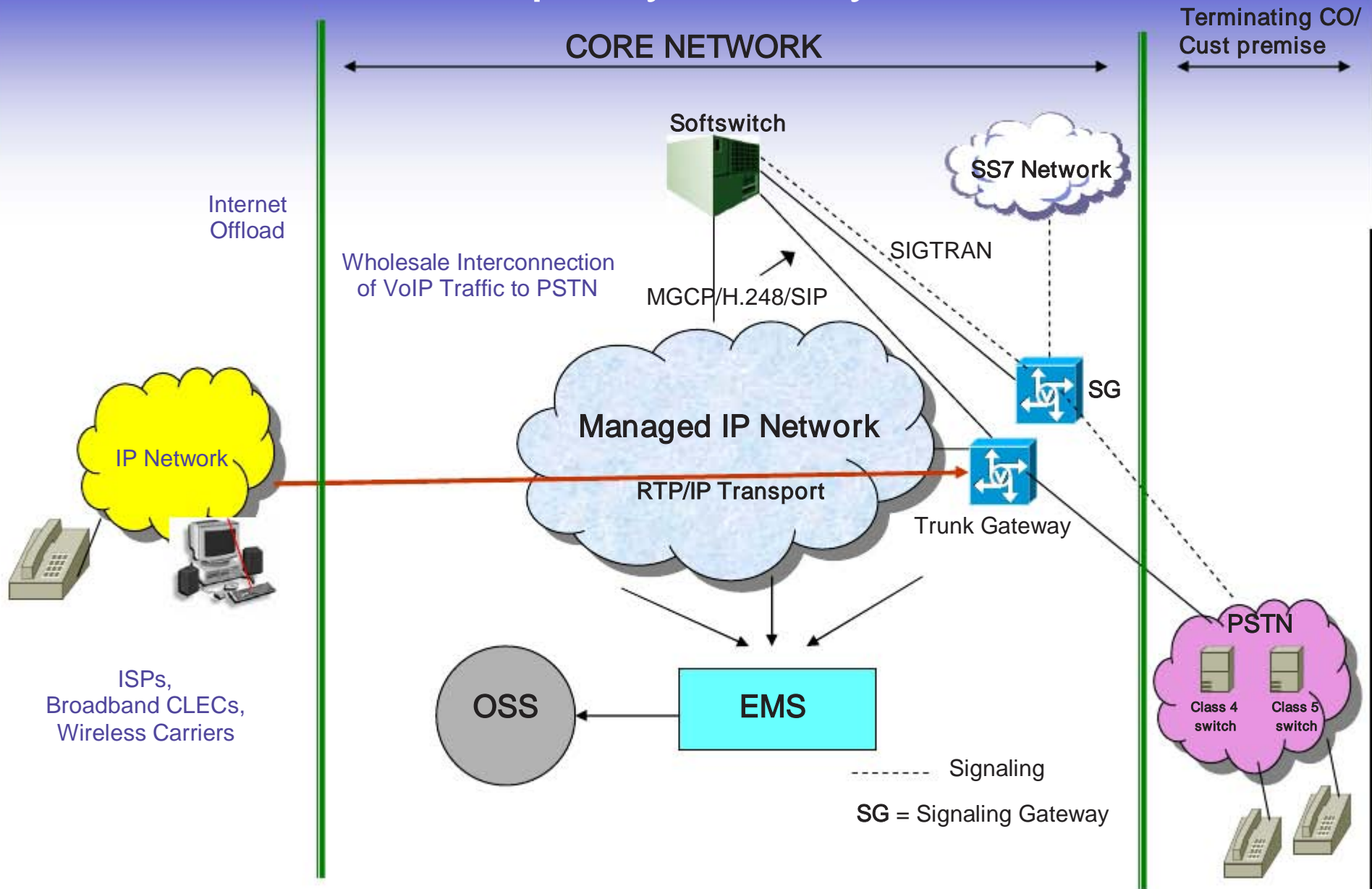
VoIP Architectures

- Class 4 Internet Telephony Gateway
- Class 4 Packet Tandem
- H.323 Gateway/Gatekeeper
- SIP Server
- Class 5 Soft Switch*
- IP-Centrex with Circuit Switch
- Packet Cable

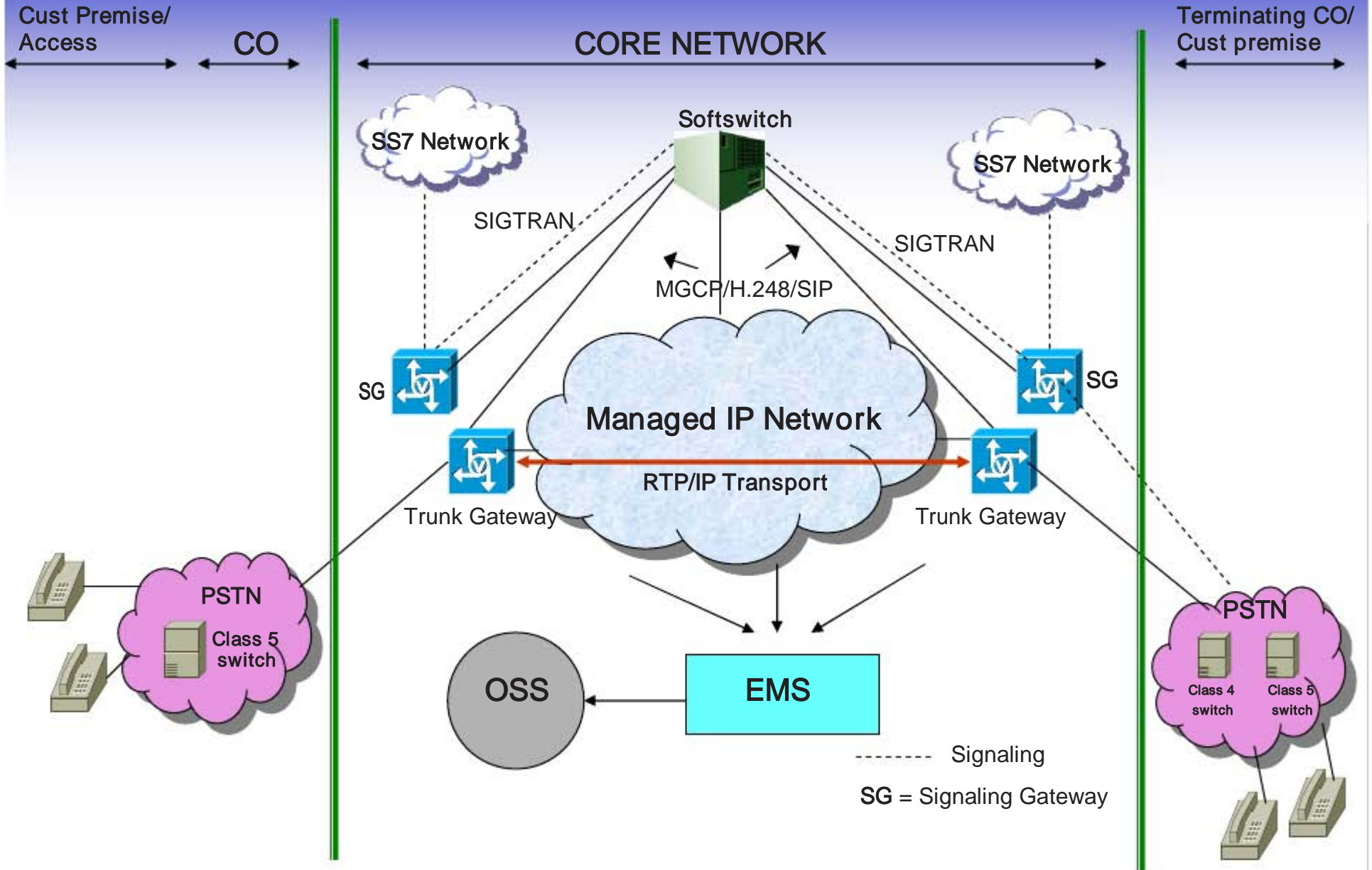
VoIP Architecture Components

- Access Transport Medium (DSL, Cable, LAN, WI-FI, Satellite)
- Voice Terminal (POTS phone, cell/smart phone, PDA, IP phone, PC+USB phone, PC, WI-FI phone)
- Network Protocols (ITU H.323, H.248, BICC, IETF SIP, MGCP, SIGTRAN, IEEE 802.11e, 3GPP 3G-324M, Cable Lab NCS)
- Network Systems (Soft Switch; Gatekeeper; Gateway -Trunk, Signaling, CAS, PRI, Analog, GSM; Residential Gateway-IAD, MTA, Loop Start; SIP Server; Service Server - Feature, Conference, Packet Voice Mail, Media, Announcement, Wiretap, IVR Server)

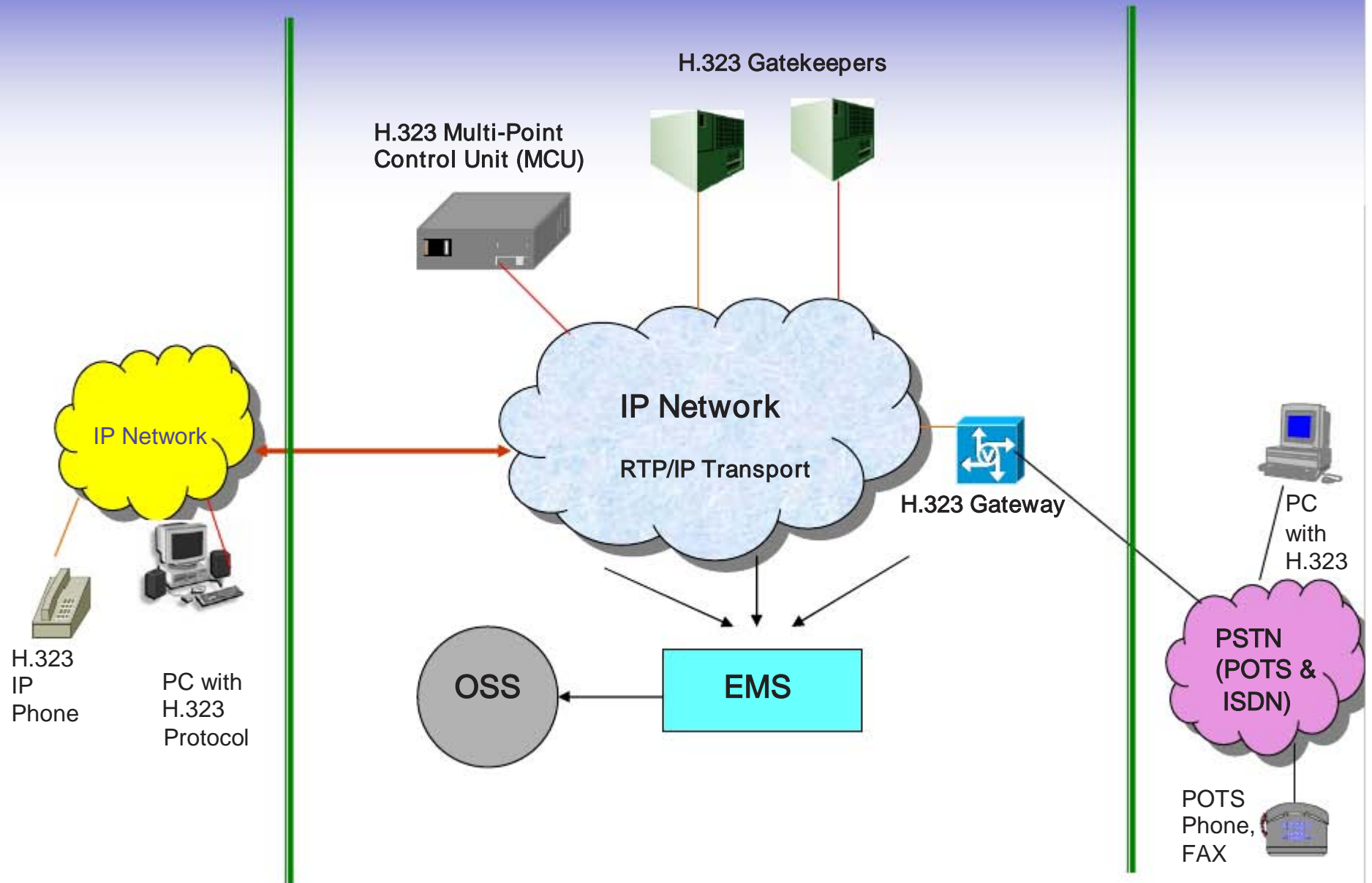
Class 4 Internet Telephony Gateway VoIP Architecture



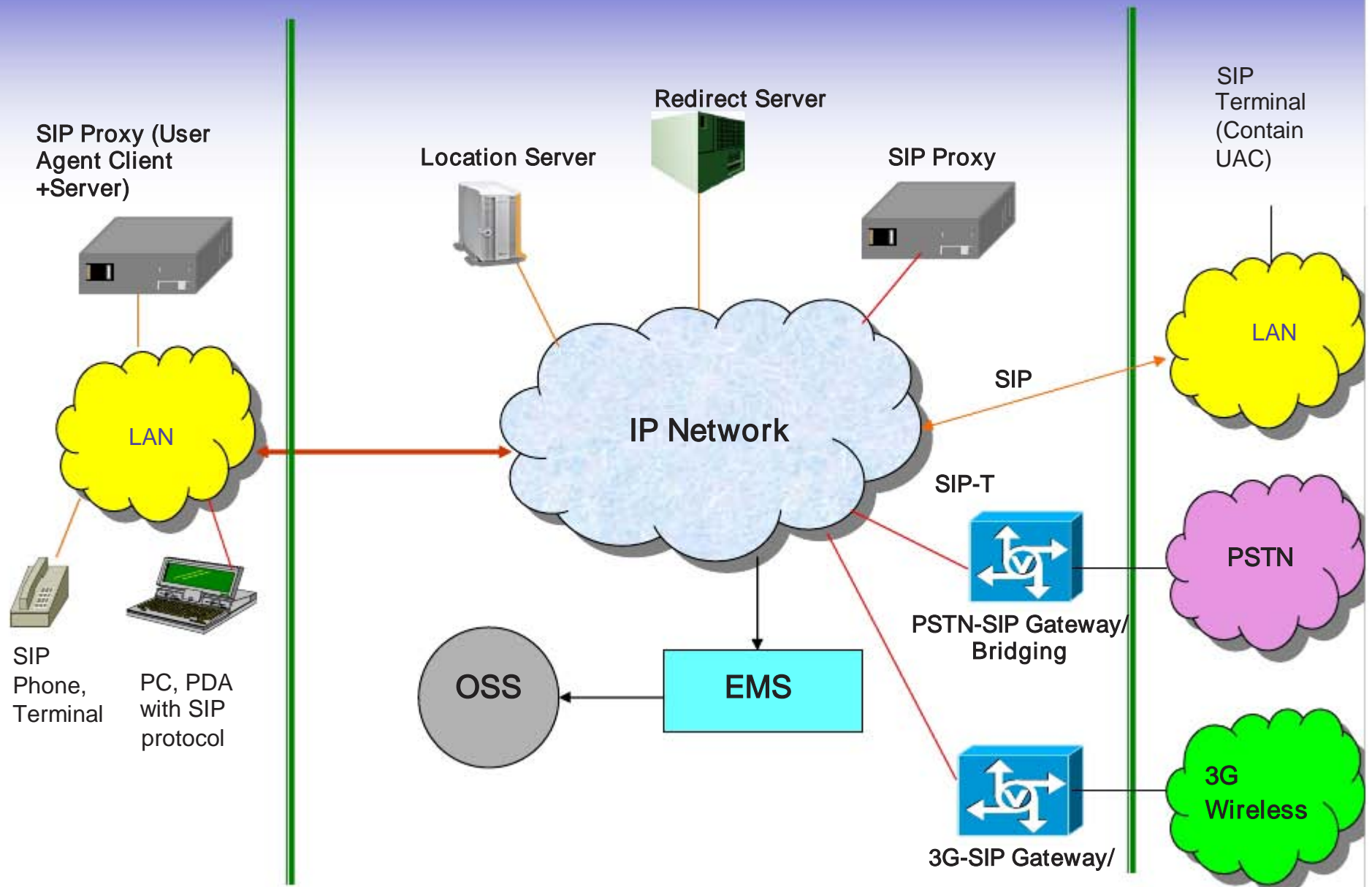
Class 4 Packet Tandem VoIP Architecture



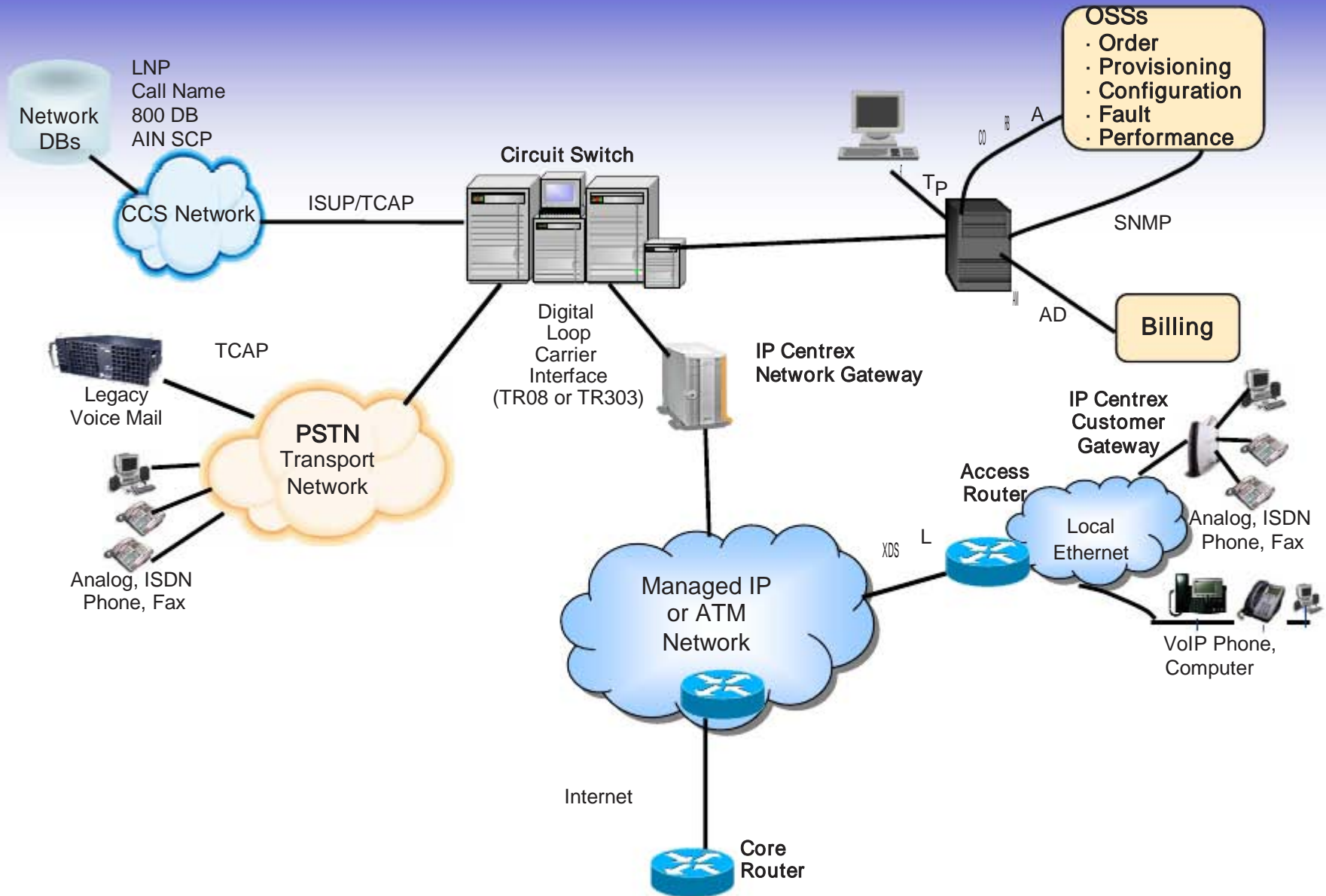
H.323 Gateway/Gatekeeper VoIP Architecture



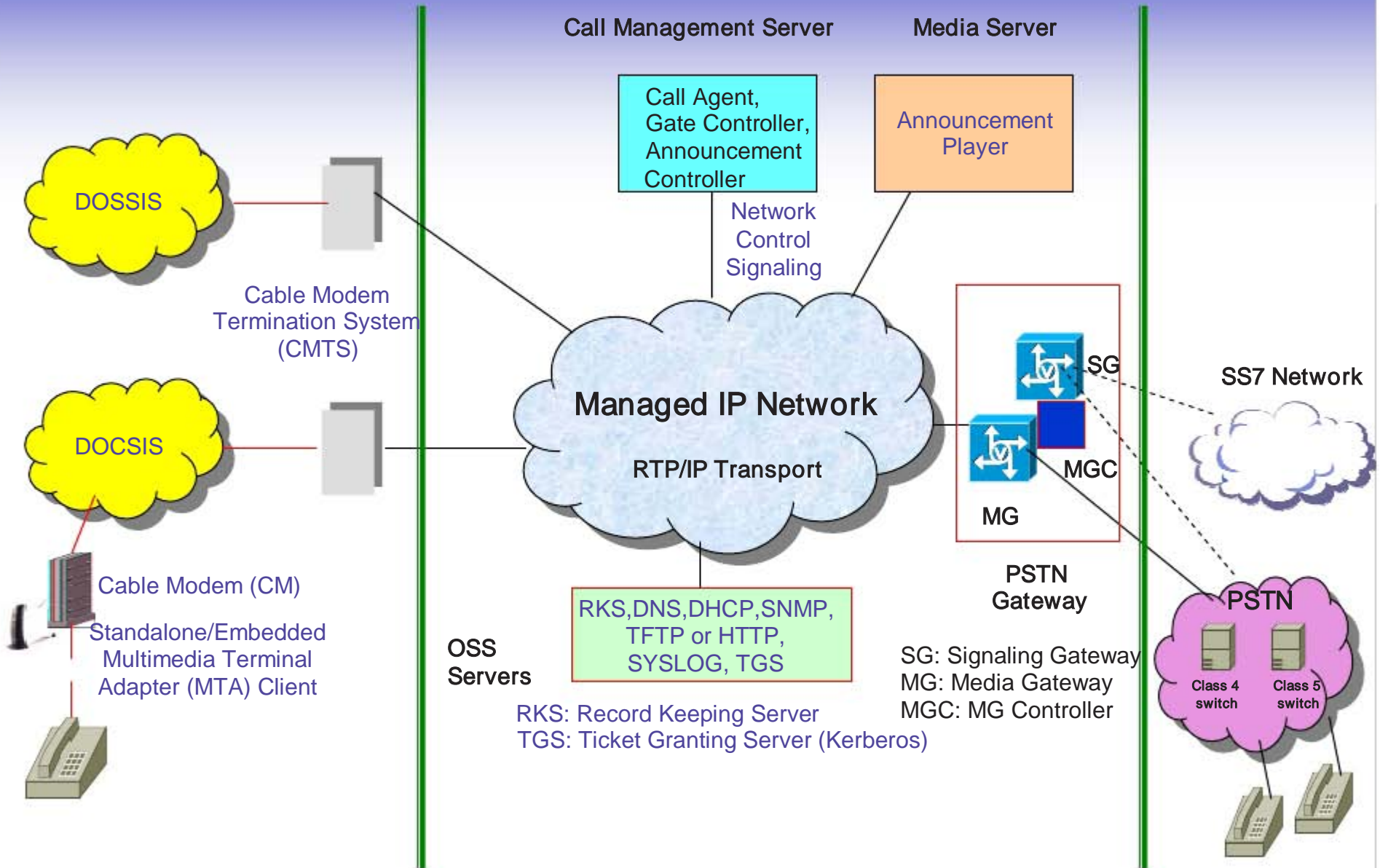
SIP VoIP Architecture



IP Centrex with Circuit Switch VoIP Architecture



Packet Cable VoIP Architecture



State of VoIP Technology Adoption

Internet Telephony Service Providers (since 1996, H.323 mainly)

LEC/IXC (Class 4 and 5 Soft Switches, Gateway, IP Centrex – H.323, MGCP, Megaco, SIP)

Cable Carriers (Packet Cable)

Satellite Carriers (H.323, SIP)

Wireless Carriers (mainly in IP trunking, 3G-324M, SIP later)

Hot Spot Providers (early stage, SIP)

Internet Service Providers (early stage, SIP based)

Application Service Providers (IP Contact Center, H.323, SIP)

Broadband CLECs (early stage)

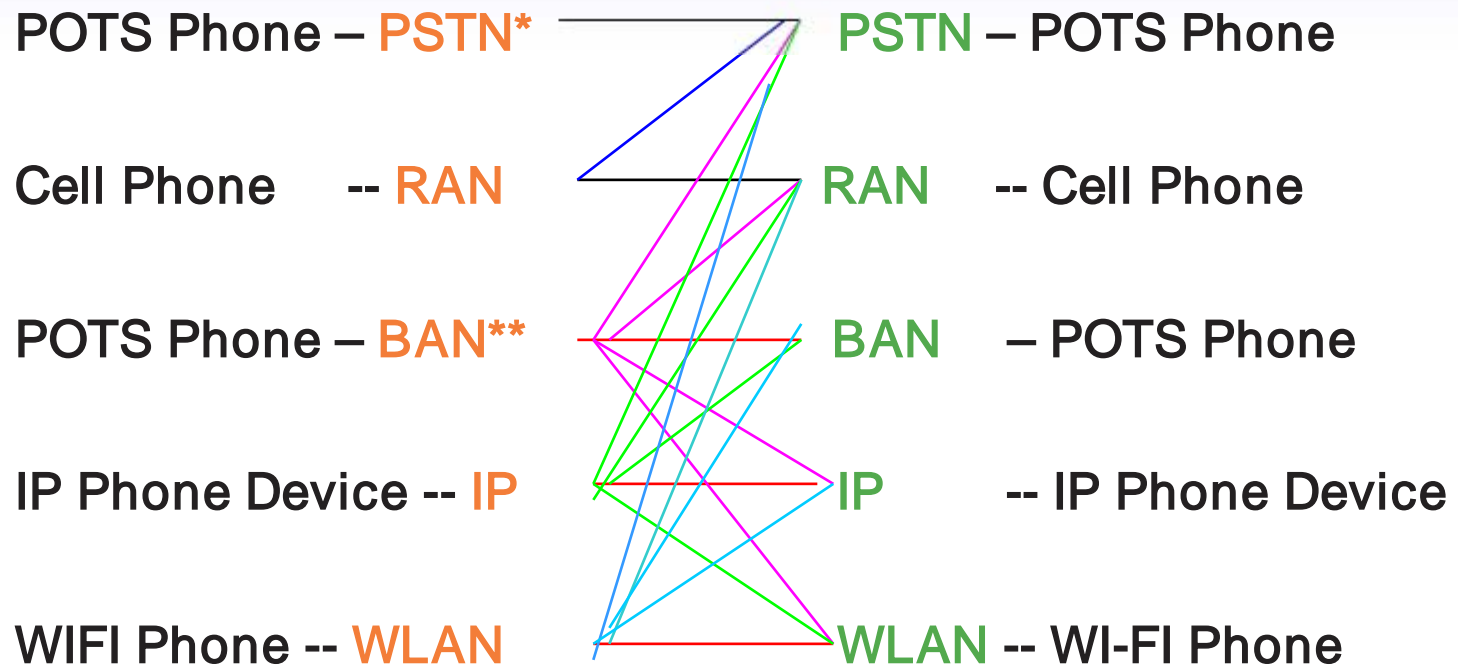
Enterprise Companies (H.323 mainly)

System Integrator/Outsourcing Firms (IP Centrex/PBX)

Voice over WI-FI over IP Pipes



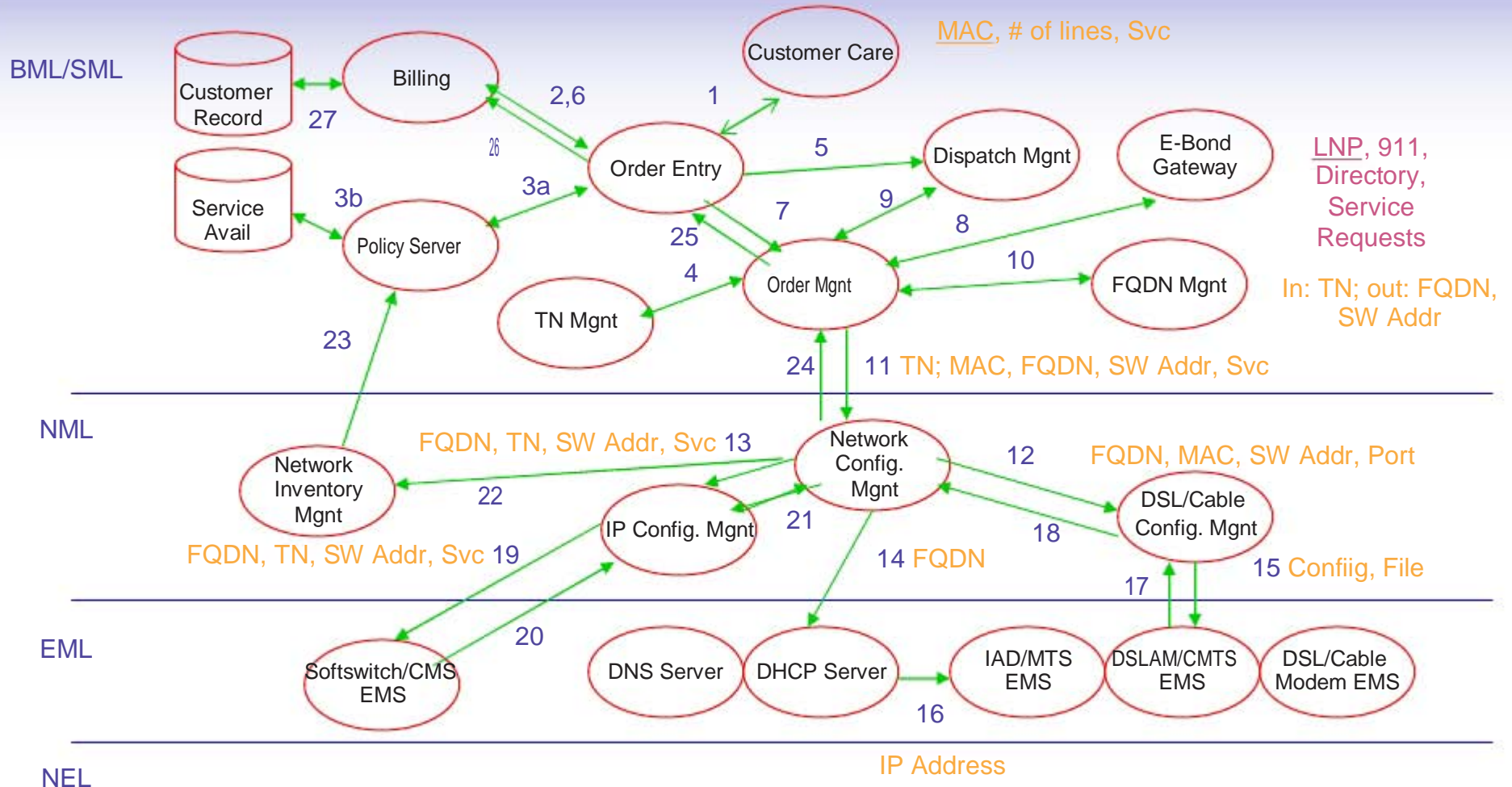
VoIP End Point Connection Types



* Include ISDN

** Include DSL and Cable

VoIP Initial Provisioning Process Example – No Service Server for VoDSL and Packet Cable in Class 5 Soft Switch Architecture



*derived from a Cable Lab OSS figure

Key Differences VoIP Ordering and Provisioning From Circuit Switched Voice Services

Provisioning and Ordering for Various VoIP Architecture and Components

Varieties of IP Transport Network Infrastructure and QoS (e.g. MPLS, IP with DiffServ, ATM, DWDM,)

Provisioning Mechanisms for Diversity of CPE (e.g. IAD, MTA, Residential Gateway, IP Phone -- subscribe/notify scheme*)

Interaction of Existing LAN Devices with VoIP CPE

Multiple VoIP Protocols

Assignment of FQDN Besides Phone Numbers

Use of DNS/DHCP Servers for Static and Dynamic IP Address

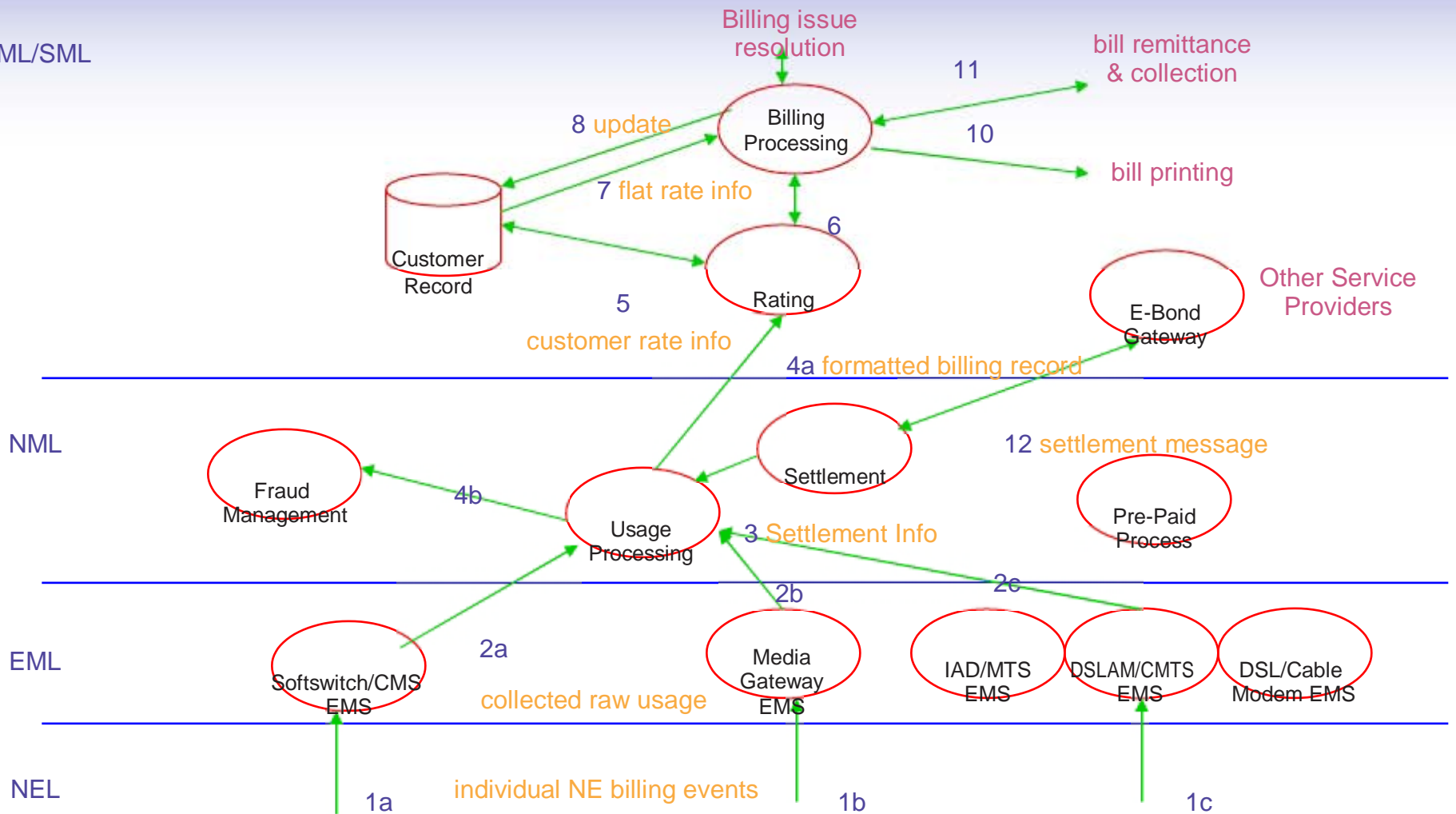
New/Change/Cancel Orders (Port Numbers, Features)

Customer Self Provisioning and Personal Call Control for Ports and Features from PC/PDA/Phones

* Notify using SNMP, SIP/Notify while Subscribe using TFTP, HTTP

VoIP Billing and Accounting Process Example – Post-Paid for VoDSL and Packet Cable in Class 5 Soft Switch Architecture

BML/SML



*derived from a Cable Lab OSS figure

Key Differences VoIP Billing From Circuit Switched Voice Services

Besides AMA and proprietary CDR, Radius Billing is used in IP telephony services, which mainly measure call duration based on location if not flat fee.

IPDR can be used for VoIP services beyond IP telephony, which may need to count packets, messages of different types, QoS, variable parameters for content and applications

Access charges apply only when a call is originated or terminated from PSTN. Today internet access (for voice or data) is taxed, but not metered.

No settlement charges for international calls when transmission is carried in IP
Billing VoIP services in legacy billing systems needs mediation with a Charging Gateway that collects packet counts from non-store-and-forward service nodes and performs a policing function for flat rate and excess usage charge tariffs

VoIP usage recording is done in a more distributed fashion than circuit switch voice – uses Universal Coordinated Time (UTC) in soft switch and time zone info for usage record – BAF Table 866 and Module 611

VoIP calls do not involve the dedicated use of the network circuits – elapsed time is computed from connect/disconnect timestamp

- soft switch needs to do notification of clock adjustments
- handles elapsed timing adjustment for internal system delay (circuit switch = 0.5 sec in GR508)
- timing irregularities in a VoIP usage record from multiple usage measurement
- long duration call records are produced based on Connect and Disconnect usage measurements besides connection status reports (for in-progress call) from soft switch.

NGN Accounting Management Generic Requirements (GR-3058-CORE, Issue 2, 12/2001)

Table 4-1. Service Type to NGN Call Type Code and Structure Code Mapping

Type of Call	Service Type	Call Type Code	Structure Code
Basic Local Service	006	240	0001
Originating Exchange Access	110	241	0625
Terminating Exchange Access	119	242	0625
Originating Connecting Network Access	589	243	0625
Terminating Connecting Network Access	720	244	0625
Basic Long Distance/Ingress	060	245	0625
Basic Long Distance/Egress	246	246	0625
Toll Free - IC Transport	141	141	0360
Toll Free - LEC Transport	142	142	0364

* Also use BAF Sensor Type and Recording Office Type to identify a usage record is generated from a VoIP system

Issues: Regulatory -- Worldwide

VoIP is not treated uniformly around the world

As long as voice services are not resold, no regulations are applied in many jurisdictions

No FCC or PSC regulations on the toll aggregation over VoIP networks

ITU IP Telephony Workshop: service centric, technology neutral regulatory treatment – similar services are treated in the same way. Independent of underlying network or technology

EU: Voice quality of many VoIP services is poor (e.g. > 250 ms avg. round trip delay or < 99% call success rate) to be classified as “licensed” real-time voice communications service like PSTN or Mobile

US House of Representatives amended HR 1291 bill in 5/02 to stipulate that “Internet telephony service, irrespective of the type of CPE used” is no longer exempt from FCC-imposed per minute charge for the Universal Services Fund. U.S. Internet community fiercely opposes HR 1291 bill.

FCC pending VoIP related proceedings: AT&T (interstate VoIP exempt from access charges), pulver.com (VoIP, SIP based, is an info service)

National. Assoc. of Regulatory Utility Commissioners (“NARUC”) issued a resolution calling for FCC to declare phone-to-phone calls over IP networks are telecom services

Gartner Group: Tax based on carrier’s total revenue, not by minutes, for retail and wholesale

Issues: Regulatory -- U.S. States

Ohio PUC initiated an inquiry in 4/2003 on how “telecom service with an Internet protocol and/or voice over the Internet component” are being provided; invited comments on if a VoIP provider is “transmitting telephonic messages”

Virginia State Corp. Commission (SCC) considers opening a formal inquiry to determine if VoIP providers are subject to SCC jurisdiction.

Florida PSC has a pending legislation to declare VoIP to be unregulated, but would leave open if VoIP providers are liable for access charges to local phone companies when they offer end users services

NY PSC determined in 2002 that a carrier utilizing IP telephony to carry voice traffic over part of its network is subject to intrastate access charges

Colorado PUC ruled in 2000 that a carrier utilizing VoIP to deliver long distance traffic should not be subject to intrastate charges

South Carolina, Nebraska state commissions opened similar proceedings, but terminated without rulings

Issues: Regulatory -- General

FCC notice on “bill and keep” (carriers recoup all of the costs of VoIP originating and terminating traffic from their own customers, rather than from other carriers) eliminates interconnection charge

Parity rules for VoIP carriers that also operate PSTN networks

Demarcation points for VoIP services between carriers for service fulfillment and assurance

Life line support may not be supported by some VoIP service providers

Issues: Business

Diversified VoIP Resell Types

- **Retail Services from VoIP Carrier(s)**

Retail Long Distance Call Center; Outbound Contact Center; Internet Café; HotSpot Facility Providers

- **Prepaid Calling Card Business**

Minutes reselling (low margin except for routes to countries with emerging economy promise or de-regulation changes); Niche local markets by offering native language calling card

- **Virtual ITSP Business**

Private labeling; Direct account resell

- **ISP Reselling VoIP Services**

May bundle services, e.g. long distance with Internet access, in near term; Opportunity exists in integrating services for future differentiated services

- **Multiple ITSP Wholesale Business**

Use least cost routing utilizing available ITSP networks; Fail over to backup ITSP networks

- **Value Added Resell Business**

Provide value-added services; May sell its own specially designed CPE and service servers

Multiple Wholesale Support Types

- **VoIP Network Facilities and Equipment (including soft switch, broadband access)**
- **U.S./Canada/Intl. PSTN VoIP Termination (e.g. charging 1.5 cents/minute with volume discount)**
- **U.S. Toll Free PSTN Orig. VoIP Termination (VoIP gateway anywhere in the world)**
- **VoIP Service Servers (e.g. Conference, Auto Attendant,**
- **Web Sales, Account, and AAA Support**
- **Network Configuration and Monitoring**
- **Billing, Trouble, Reporting, and OSS Support via EDI, XML, E-mail, Web**

Issues: Business (Continued)

Bilateral agreements between carriers to get compensation for VoIP traffic across carrier network boundary increase the complexity of billing and ordering

ISPs may sell integrated services with VoIP free of charge

IP v6 migration when VoIP devices need a static address

- 128 Address Bit vs. 32 Bit in IP v4

Currently most IP trunking service providers charge monthly fee with QoS for data packets, not for voice call quality.

Issues: Standards

Interconnection Among Different Types of VoIP Networks and PSTN

- ITU: Bearer Independent Call Control (extend ISUP used in PSTN and ISDN for signaling in ATM AAL 1 and 2, IP voice transport), H.248, H.323, IP SCP
- IETF: SIP, MGCP, SIGTRAN
- IEEE: 802.11p (Guaranteed QoS LAN), 802.11e (Real-Time QoS WLAN)
- Cable Lab: NCS
- 3GPP: 3G-324M
- H.323 and SIP just adds supplement services
- SIP and PSTN interworking standard is being finalized

End-to-End Service Quality Assurance

- QoS Policy Server, DiffServ Code Point, MPLS, RSVP, over-provisioning, dedicated facility, G.113 ICPIF, trouble

NAT/Firewall interworking with H.323 and SIP

End-to-End Order and Provisioning

USOC and FID Extension

Issues: Standards (Continued)

End-to-End Billing

- **Signaling messages in interconnecting VoIP networks for off-net calls** (PTSN
 - using SS7 Exit Message to signal egress trunk group ID)
- **Signaling needed when usage recording for a call occurs in multiple soft switches or network elements** (use Network Call Correlation Identifier in BISUP or Global Call Reference in BICC CS2; ITU-T/ATIS T1S1.3 Inter-Nodal Traffic Group Identifier in BICC message to support special routing services like 911)
- **Procedure to support billing for custom calling, IN, and value added services beyond basic calls, LNP, and toll free in GR-3058-CORE** (may need to involve gateways; need to fill Service Feature Code in BAF Table 12)
- **Billing with interconnecting carriers**
- **Billing with reselling service providers**

Acronyms

AAA: Authentication, Authorization, and Accounting
AAL: ATM Adaptive Layer
AMA: Automatic Message Accounting
BAF: Billing AMA Format
BAN: Broadband Access Network
BISUP: Broadband ISDN User Part
CDR: Call Detail Record
CMS: Call Management System
CMTS: Cable Modem Termination System
CS2: Capability Set 2
DHCP: Dynamic Host Configuration Protocol
DNS: Directory Name Server
DSL: Digital Subscriber Loop
DSLAM: Digital Subscriber Line Access Multiplexer
FID: Field Identifier
FQDN: Fully Qualified Domain Name
HTTP: Hypertext Transfer Protocol
IAD: Integrated Access Device
ICPIF: Calculated Planning Impairment Factor
IN: Intelligent Network
ISP: Internet Service Provider
ITSP: Internet (or IP) Telephony Service Provider
LAN: Local Area Network
MEGACO: MEdia GATeway COntrol

MGCP: Media Gateway Control Protocol
MPLS: Multi-Protocol Label Switching
MTA: Multimedia Terminal Adapter
NAT: Network Address Translation
NCS: Network Control Signal
PSTN: Public Switched Telephone Network
QoS: Quality of Service
RADIUS: Remote Access Dial Up Service
RAN: Radio Access Network
RTCP: Real Time Control Protocol
RTP: Real Time Protocol
SALT: Speech Application Language Tag
SCP: Signal Control Point
SCTP: Stream Control Transmission Protocol
SIGTRAN: Signal Transport
SIP: Session Initiation Protocol
SIP-T: Session Initiation Protocol for Telephony
SNMP: Simple Network Management Protocol
TFTP: Trivial File Transfer Protocol
TN: Telephone Number
USOC: Universal Service Ordering Code
VoIP: Voice over IP
VPN: Virtual Private Network
WI-FI: Wireless Fidelity
WLAN: Wireless Local Area Network