3. Policy Based SLA for DiffServ Networks

A Service Level Agreement (SLA) [73] is a service contract between a customer and a service provider that specifies the forwarding service a customer should receive. A customer may be a user organization (source domain) or another DiffServ domain (upstream domain). SLA is a static procedure, usually performed manually which has a major drawback i.e., DiffServ is used to handle more connectivity dynamically through its service classes. Thus there arises a compelling need to automate the SLA trading. Bandwidth Brokers are used in DiffServ Architecture to automate SLA Trading between different DiffServ Domains. But the drawback in automating SLAs using Bandwidth Brokers is that only Bandwidth constraints are negotiated. Here, a mathematical model is presented to support decision activities on SLAs with many performance metrics. This framework uses a Policy Server to take decisions on the service trading issues. The decision on what and when to choose are written as policies. Policies are stored in the directory server, and are retrieved by the policy server using LDAP, the service agreement adhering to the policies taken are then negotiated between the SLA Agents (SLAA).

3.1 SLA Framework for Wired Networks

Internet Service Provider (ISP)'s face many challenges when they are interconnected with each other. For resource sharing, they have to come up with a contract, or a Service Level Agreement (SLA) that specifies what service is performed during which period and at what cost. Although this
principle is simple, the setup of SLAs is not straightforward and in today’s Internet it is limited by a range of issues:

- SLAs are set up manually
- The traffic itself is not characterized by other parameters, than by terms of bandwidth.
- Setting up multiple peers to enhance reliability or to exploit load balancing, which is difficult using Border Gateway Protocol (BGP) [74] and Routing Policy Specification Language (RPSL) [75]. This is reflected in the many detours found in the Internet [76].
- And finally, price refers to this static environment.

SLA Trader (SLAT) architecture and a mathematical model uses additional performance metrics along with bandwidth for effective SLA trading. The proposal to automate the task of SLA negotiation has gained popularity in the framework of DiffServ. Assuming that both basic contract and physical connectivity exist between two ISPs, their SLATs set up new bilateral agreements as demand and load of the networks dynamically changes.

SLA traders may choose to exploit alternate paths at the inter-Autonomous System (AS) level, balance load between several peers and improve operational reliability. QoS routing should not only happen inside but also between Autonomous Systems [77]. The objective is to “encourage simple, consistent and stable interactions between Autonomous Systems” [78]. Attaching price to offered SLAs during the negotiation process gives ISPs a powerful tool to evaluate the potential benefits of new SLAs. So, the cost of the SLA is also attached along with the agreement.
Based on these observations a new, integrated method for ISP interaction by using SLAs as a central element, carrying service provisioning, routing information, performance metrics and pricing information is modeled.

An illustration of SLA trading is specified:

Let us consider that ISP-1 has a quantity of aggregated traffic to send to ISP-5 (see Fig. 3.1) and that the direct link between ISP-1 and ISP-5 could not accommodate the traffic. A mesh topology is considered because more number of trading will take place and effectiveness of the trading algorithm and the proposed mathematical model can be tested. However any topology can be used. Fig. 3.1 has multiple paths to reach the destination network (ISP-5). As observed, some ISPs find themselves in competition to others. Thus an effective SLA needs to be traded between one of the ISPs which offer service. Here, bilateral agreements in the form of SLAs build up in a nested manner finally providing an end-to-end service. Cost and delay increase at each ISP (additive metric) along the path while the bandwidth metric is concave and stays at its minimum. From [79], "... the observation is that multilateral agreements rarely work...", furthermore, SLA trading happens at a medium time scale (several minutes to hours). Restricting the routing, provisioning, measurement and pricing activities to the AS-level solves many scaling problems. The Internet has about 6,000 ASes and there are about ten times as many routed IP networks and 10,000 times as many hosts. Dividing further by applications and traffic source leads to an explosion of micro-flows at the backbone [80].
SLA trading protocols and the traders themselves may change from location to location. Some of the previous work [73] consider only few parameters like bandwidth; link etc, for defining an SLA in DiffServ architecture. SLAs at each ISP are defined by the following service performance metrics which are crucial for an effective SLA:

- Bandwidth
- Buffer Capacity
- Time Delay (Delay)
- Queuing Delay
- Packet Loss
- Congestion
- Throughput
- Jitter
- No. of Physical Paths
- Demand

Here, the focus is on the interaction between DiffServ domains. Resource allocation within each domain is not considered. To achieve an end-to-end architecture that allows extending this approach among ISPs to end-users,
individual profiles and contracts need to be aggregated to serve as input for SLA traders. To keep this step as simple as possible, important results from user behavior surveys have to be considered [81].

3.1.1 Mathematical Model

A mathematical model considered for this framework is based on Linear Programming and uses Simplex Method [82] to solve it. In this model, 'traffic splitting' is not considered as a trading parameter as it will be taken care by the ISP which provides the service. This is because the number of physical paths is also taken as one of the performance metric for trading. Similarly this model assumes that the best optimal routing is available. Moreover, (if) any problem in routing during data transfer will obviously affect the performance metric listed above which is considered for trading and hence the traded SLA can be withdrawn due to poor performance.

In the case of a small network, all the performance metrics can be considered in the trading process with higher priority given for Bandwidth, Delay and Packet Loss. Whereas, in the case of a very large network, it is not practically possible to meet out all the constraints on all the performance metrics, it would be effective if only Bandwidth, Delay and Packet Loss are considered for trading.

Let,

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{ij}$</td>
<td>Total (maximum) Bandwidth available from ISP $i$ to ISP $j$.</td>
</tr>
<tr>
<td>$U_{ij}$</td>
<td>Bandwidth being used for traffic between ISP $i$ to ISP $j$ at instant $t$.</td>
</tr>
<tr>
<td>$R_{ij}$</td>
<td>Reserved bandwidth from ISP $i$ to ISP $j$ for applications with very high priority.</td>
</tr>
</tbody>
</table>
Hence the bandwidth that can be leased to other ISPs \( G_j \) is given by

\[
G_j = T_j - U_j - R_j
\]

Let,

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( D_{E_j} )</td>
<td>Demand for service from ISP ( i ) to ISP ( j )</td>
</tr>
<tr>
<td>( D_j )</td>
<td>Delay from ISP ( i ) to ISP ( j ) at instant ‘( t )’.</td>
</tr>
<tr>
<td>( C_j )</td>
<td>Cost of reaching ISP ( j ) through ISP ( i ) in $ as bid by ISP ( i ) per bandwidth unit.</td>
</tr>
<tr>
<td>( F_j )</td>
<td>Fraction of bandwidth bought from ISP ( i ) to reach ISP ( j ).</td>
</tr>
</tbody>
</table>

The objective is to minimize the cost of borrowing service from other ISPs in order to meet with the service demand.

\[
\text{Minimise } \sum_{i,j} F_j G_j C_j \tag{3.1}
\]

As stated earlier, \( C_j \) in the above equation represents the cost per bandwidth unit ISP \( i \) charges to reach ISP \( j \) through \( i \).

### 3.1.2 Constraints

The primary constraint is that the demand for service to reach ‘ISP \( j \)’ through ‘ISP \( i \)’, \( D_{E_j} \) should be less than or equal to the amount of bandwidth ‘ISP \( i \)’ is ready to offer for cost to reach ‘ISP \( j \)’, \( G_j \).

\[
D_{E_j} \leq \sum_{i,j} G_j \tag{3.2}
\]

The other constraints check if the performance metrics in the service offered by the ‘ISP \( i \)’ to reach ‘ISP \( j \)’ fall within the pre-calculated boundaries as expected by the ‘ISP \( k \)’ which needs the service. These boundary constants can also be set dynamically and SLA negotiated accordingly by means of a Service Agent (SA) and a policy server.
Buffer Capacity $B_y$ should not be less than the constant $N$ (Number of packets that can be buffered),

$$B_y \geq N \quad (3.3)$$

The time delay $D$ is set to a limit expressed by ‘$p1’ [83] as expected by the ‘ISP $k$’ which needs the service.

Where,

$$p1 = \text{Propagation Time} + \text{Transmission Time} + \text{Queuing Delay}$$

(+ Setup Time)

Where,

- Propagation Time: Time for signal to travel length of network = Distance/Speed of light
- Transmission Time = Size/Bandwidth

Therefore,

$$D_y \leq p1 \quad (3.4)$$

Queuing Delay $Q_y$ should not exceed ‘$p2’$, (where, $p2 = \frac{D}{2} \times (N - 1)) [84].$

$$Q_y \leq p2 \quad (3.5)$$

The Packet Loss $P_y$ for the service provided should not exceed ‘$p3’ and Congestion in the channel offered for service $C_o_y$ should be within ‘$p4’ both of which are arrived as shown below [85].

$T_{\text{min}}$ : Minimum Inter-Arrival Time observed by the receiver

$P_o$ : Out of order packet

$P_i$ : Last in-sequence packet received before $P_o$

$T_g$ : Time between arrival of packets $P_o$ and $P_i$

$n$ : Packets missing between $P_i$ and $P_o$

If $(n + 1)T_{\text{min}} \leq T_g < (n + 2)T_{\text{min}}$ then $n$ missing packets are lost due to transmission errors and so ‘$p3’='n’, and
\[ P_{ij} \leq p^3 \]  \hspace{1cm} (3.6)

Else, \( n \) missing packets are assumed to be lost due to congestion and so

\[ 'p4' = 'n', \text{ and} \]

\[ C_{ij} \leq p^4 \]  \hspace{1cm} (3.7)

Throughput \( TH_{ij} \) should be greater than or equal to \( p^5 \) [86] given by

\[ p^5 = \left\{ \frac{MSS}{RTT} \right\} \times C / (\sqrt{p}) \]

Where,

\begin{center}
\begin{tabular}{|l|p{0.8\textwidth}|}
\hline
\textbf{MSS} & Maximum Segment size in bytes. Typically 1460 bytes \\
\hline
\textbf{RTT} & Round Trip Time in seconds, measured by TCP. \\
\hline
\textbf{P} & Packet loss \\
\hline
\textbf{C} & Constant assumed to be 1. \\
\hline
\end{tabular}
\end{center}

\[ TH_{ij} \geq p^5 \]  \hspace{1cm} (3.8)

The jitter \( J_{ij} \) should be within \( p^6 \) [87] given by

\[ p^6 = p^6 + (|D(i-1,i) - p^6|)/16 \]

Given

\[ D(i,j) = (R_j - S_j) - (R_i - S_i) \]

Where,

\( S_i, S_j \) are sender timestamps for packets \( i, j \), and

\( R_i, R_j \) are receiver timestamps for packets \( i, j \). Therefore,

\[ J_{ij} \leq p^6 \]  \hspace{1cm} (3.9)

The number of physical paths \( PP_{ij} \) is within the boundary constant \( p^7 \) which represents the minimum number of physical paths needed,

\[ PP_{ij} \geq p^7 \]  \hspace{1cm} (3.10)
3.1.3 Non-Negativity Constraints

The following are the non-negativity constraints applied in the model.

Cost $C_{ij}$ should always be positive.

$$C_{ij} \geq 0 \quad (3.11)$$

Fraction of bandwidth obtained from ‘ISP $i$’ to reach ‘ISP $j$’, $F_{ij}$ is positive.

$$F_{ij} \geq 0 \quad (3.12)$$

The bandwidth that can be offered to other ISPs $G_{ij}$ by ‘ISP $i$’ is also positive.

$$G_{ij} \geq 0 \quad (3.13)$$

3.2 Architectural Framework

The architectural framework of the Policy-based SLA for DiffServ is shown in Fig.3.2. For representation convenience, the architectural elements between two ISPs alone are considered which may further be extended between any numbers of ISPs. Each ISP has a SLAT which trades the SLA with other ISPs. Every SLAT can buy and sell SLAs. This trading should be automated to suit with the dynamicity of DiffServ architecture. There is a component, SLAA which is a part of SLAT. It communicates with other ISPs by means of agreements. The SLAA after getting the agreement proposal from other ISPs requests the policy server to take a decision on which agreement to choose.

The mathematical model implemented in the Policy Server facilitates taking decisions on which agreements suit the best, satisfying all the constraints. The policy decisions to choose a suitable agreement among the feasible ones are installed as Policies in the Directory Server. These policies are queried by the policy server through LDAP and then suitable SLA is accepted. The SLAA then finalizes the agreement with the other ISP upon a service. Based on the
agreement the SLAT will set up DiffServ classifiers/markers at ingress nodes and remarkers at the egress nodes. The Policy Management Console (PMC) provides the administrator in the ISP with an interface to define the various policies or policy groups. It is typically the function of the PMC to validate the syntactic and semantic correctness of the policy input, to ensure consistency among the high-level policies and to check for compatibility of the various policies. Signaling demand and supply (trading) between ISPs needs an appropriate protocol to transport SLA messages. Any existing protocol for SLA trading such as new BGP attributes, the Internet Open Trading Protocol (IOTP) and others can be used in the framework. The framework does not suggest any specific protocol or does not define a new protocol altogether to suit the architecture.

Fig. 3.2 Architecture of the Policy Based SLA

3.3 SLA Trading Algorithm

An SLA Trading algorithm for provisioning and profitability analysis is given in Table 3.1, 3.2. A passive provisioning algorithm does wait for requests from its customers to select as to which resources to buy. Whereas an active provisioning algorithm tries to forecast future needs, and will buy resources in
advance, before becoming scarce. Buying in advance may be based on statistical information or on trend analysis.

<table>
<thead>
<tr>
<th>Table 3.1: Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPDATE_PERIOD          : Time for updation</td>
</tr>
<tr>
<td>volume()               : volume function for an SLA object (time×bandwidth)</td>
</tr>
<tr>
<td>send bid()             : sends an offered SLA to peer</td>
</tr>
<tr>
<td>accept bid()           : sends an accept message</td>
</tr>
<tr>
<td>known dests            : reachability list</td>
</tr>
<tr>
<td>MIN_REQUIRED_DBW       : Minimum required bandwidth</td>
</tr>
<tr>
<td>MAX_ALLOWED DELAY      : Maximum allowed delay</td>
</tr>
<tr>
<td>MAX_ALLOWED_PL         : Maximum allowed packet loss</td>
</tr>
<tr>
<td>MIN_REQUIRED_BC        : Minimum required buffer capacity</td>
</tr>
<tr>
<td>MAX_ALLOWED_QD         : Maximum allowed queuing delay</td>
</tr>
<tr>
<td>MIN_REQUIRED_TP        : Minimum required throughput</td>
</tr>
<tr>
<td>MAX_ALLOWED_JI         : Maximum allowed jitter</td>
</tr>
<tr>
<td>MAX_ALLOWED_C          : Maximum allowed congestion</td>
</tr>
<tr>
<td>MIN_REQUIRED_PP        : Minimum required physical paths</td>
</tr>
<tr>
<td>MAX_ALLOWED_COST       : Maximum allowed cost</td>
</tr>
</tbody>
</table>

Once an SLA trader identifies its needs to buy some resource from any one of its peers, it will have to select one of the bids and buy it. The selection of the bid is made based on the bid’s value for the SLA trader and its price. For bids of equal value, if no special policy is applied, the bid with the first lower price will be selected.

The SLA trader will also have to evaluate whether the selected bid is worth buying using a profitability analysis algorithm. This algorithm helps justifying whether buying that bid is profitable; to make money through selling of derived services. This algorithm also ensures that SLA traders won’t build service loops.
Table 3.2: SLA Trading Algorithm

```c
struct bid {
    ISP_dest, // ISP destination
    bw, // bandwidth
    delay,
    packet_loss,
    buffer_capacity,
    queuing_delay,
    throughput,
    jitter,
    congestion,
    physical_paths,
    cost
}

process trading () {
    while (true) {
        for each d in known_ISP_dests {
            /* buy bids */
            if((bw_to_ISP(d)>MIN_REQUIREDBW) and
               (delay_to_ISP(d)<MAX_ALLOWED_DELAY) and
               (packet_loss_to_ISP(d)<MAX_ALLOWED_PL) and
               (buffer_capacity_at_ISP(d)>MIN_REQUIRED_BC) and
               (queuing_delay_to_ISP(d)<MAX_ALLOWED_QD) and
               (throughput_to_ISP(d)>MIN_REQUIRED_TP) and
               (jitter_to_ISP(d)<MAX_ALLOWED_JI) and
               (congestion_to_ISP(d)<MAX_ALLOWED_C) and
               (physical_paths_to_ISP(d)>MIN_REQUIRED_PP) and
               (cost_to_ISP(d) <MAX_ALLOWED_COST)) {
                bid = [find bid with highest volume/cost ratio]
                if bid and is_profitable(bid) accept_bid(bid)
            }
            /* make bids */
            for each n in neighbours {
                if !bid_already_sent(n, d, bw, delay, packet_loss, buffer_capacity, queuing_delay, throughput, jitter, congestion, physical_paths, cost)
                make_bid(n, d, bw, delay, packet_loss, buffer_capacity, queuing_delay, throughput, jitter, congestion, physical_paths, cost)
            }
        } sleep(UPDATE_PERIOD)
    }
}

process bid_recv() {
    while (true) {
        wait(bid_received(bid)) {
            if !is_ISP_dest_known(bid.ISP_dest)
                known_ISP_dests.add(bid.ISP_dest)
        }
    }
}
```
boolean is_profitable(bid) {
    expected_volume = volume(bid)*EXPECTED_USAGE;
    expected_income = price(bid.dest,bid.bw, delay,
                packet_loss, buffer_capacity,
                queuing_delay, throughput, jitter,
                congestion, physical_paths, cost ) *
                expected_volume
    if bid.price < expected_income return true
    else return false
}

void make_bid(neighbour, ISP_dest, bw, delay, packet_loss,
        buffer_capacity, queuing_delay, throughput,
        jitter, congestion, physical_paths, cost ) {
    bid = new Bid(ISP_dest, bw, delay, packet_loss,
            buffer_capacity, queuing_delay,
            throughput, jitter, congestion,
            physical_paths, cost )
    bid.price = volume(bid)*price(ISP_dest, bw,
            delay, packet_loss,buffer_capacity,
            queuing_delay, throughput, jitter,
            congestion, physical_paths, cost)
    send_bid(neighbour, bid)
}

Trading is performed by the method 'process trading()' which identifies the bid with the highest volume/cost ratio and finds out if that bid is profitable using the 'is_profitable(bid)' method and if found profitable accepts the bid using the 'accept_bid(bid)' method. Then for each neighbor (ISPs) if bid is not already sent then bid is sent for every UPDATE_PERIOD. The profitability is tested by comparing the bid price with the expected income. Bidding with the neighboring ISPs is done using the 'make_bid()' method.

3.3.1 Policy Specification for the SLA Trading

// On a trading event the action trade sends a pop-up displaying that the trading is being analyzed to check if all the constraints are satisfied with minimal cost based on the mathematical model. The policy specification is given in Table 3.3 and the corresponding java code which enforces this policy is given in Table 3.4.
The rule /Policies/TradingPolicy will invoke the action trade within the /PMAs/TradePMA’s engine, when the event trading is dispatched to the PMA from the trading event service.

This Policy framework has the flexibility to choose among alternative solutions based on the user’s preference of one performance metric to be given more importance than others.
3.4 Observations on Results

A simulation of the above is performed using the QualNet Network Simulator [88] for the policy specifications given in Table 3.3. The test environment has five ISPs (ISP1 to ISP5), as in Fig.3.1.

Table 3.5: Performance Metrics and other parameters of the hosts

<table>
<thead>
<tr>
<th>Performance Metrics</th>
<th>ISP1</th>
<th>ISP2</th>
<th>ISP3</th>
<th>ISP4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Bandwidth (MBps)</td>
<td>3</td>
<td>18</td>
<td>12</td>
<td>20</td>
</tr>
<tr>
<td>Used Bandwidth (MBps)</td>
<td>3</td>
<td>10</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>Reserve Bandwidth (MBps)</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Remaining Bandwidth (MBps)</td>
<td>0</td>
<td>6</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Demanded Bandwidth (MBps)</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Throughput (x 10^3 Bits/Sec)</td>
<td>4.2</td>
<td>4.3</td>
<td>4.2</td>
<td>4.2</td>
</tr>
<tr>
<td>Delay (x 10^-3 /Sec)</td>
<td>2.9</td>
<td>1.8</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Jitter (x 10^-4 /Sec)</td>
<td>4</td>
<td>3.8</td>
<td>3.9</td>
<td>3.9</td>
</tr>
<tr>
<td>No. of Physical Paths</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Packet Loss Factor</td>
<td>0.3</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>Congestion Factor</td>
<td>0.3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>Queueing Delay (x 10^-4 /Sec)</td>
<td>0.2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Buffer Capacity (No. of Packets)</td>
<td>100</td>
<td>110</td>
<td>95</td>
<td>90</td>
</tr>
<tr>
<td>Cost ($)</td>
<td>-</td>
<td>4</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Fraction of Bandwidth that can be given (MBps)</td>
<td>- 2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

The algorithm was tested for the setup where the maximum number of trading will occur and hence efficiency of the model and the algorithm are identified.

Since the objective is to trade for meeting with excess demand for sending data to ISP5 the performance metrics are shown for only ISP1, ISP2, ISP3 and ISP4. In the simulation environment ISP1 faces a demand more than its available bandwidth to ISP5. All the simulations were performed with respect to traffic flow from ISP1 to ISP5. As in Fig.3.1, all the other ISPs trade with ISP1 to provide service to reach ISP5. Performance metrics, constraint parameters, the adjustable variables are given in Figs. 3.3 to 3.10.
The trade for the Service is decided by using the Simplex method to solve the Linear Programming model and a feasible solution is obtained. The performance constraints of the four ISPs (ISP1 to ISP4) are shown graphically from Fig. 3.3 to Fig. 3.6. The constraints imposed by the ISP1 for the required service are shown in Fig. 3.7 and Fig. 3.8. According to the constraints given by ISP1 for the required service the simplex method is used for solving the LP model and the best bid among the bids offered by the three ISPs (ISP2, ISP3 and ISP4) is selected. Since only the bid for the service offered by ISP2 satisfy the constraints of ISP1, SLA between ISP1 and ISP2 takes place. The process of solving the LP model by simplex method by adding slack variables is shown graphically in Fig. 3.9. As only the trade provided by the ISP2 satisfies all the constraints with the objective of minimum cost the Service offered by ISP2 is agreed upon for trade.
From the performance metrics (Fig. 3.3 to Fig. 3.6), and the constraints (Fig. 3.7 & 3.8) on performance metrics, the objective of minimizing cost is arrived (Fig. 3.10). Thus an effective SLA is traded between ISP1 and ISP2 satisfying the constraints on the performance metrics which affects the service.

3.5 SLA Framework extensions for Wireless Networks

Wireless Ad Hoc networks [89], [90],[91] are autonomous networks operating either in isolation or as "stub networks" (end networks) connecting to a fixed infrastructure (fixed network). A typical example of Ad Hoc wireless network would be mobile users carrying their Mobile Intelligent devices (mobile hosts) such as Palmtop, Notepads etc. But the transmission range (signal) of the Ad Hoc wireless hosts is very limited. A mobile Ad Hoc host can communicate with another host only within its transmission range (communication range). Depending on the nodes' (mobile hosts') geographical positions, their transceiver coverage patterns, transmission power levels, and co-channel interference levels, a network can be formed and unformed on the fly.

The main characteristics of Ad Hoc networks are:

*Dynamic topological changes*: Nodes are free to move about arbitrarily. Thus, the network topology may change randomly and rapidly over unpredictable times [92].

*Bandwidth constraints*: Wireless links have significantly lower capacity than wired links [93]. Due to the effects such as multiple accesses, multipath fading, noise, and signal interference, the capacity of a wireless link can be degraded over time and the effective throughput may be less than the radio's maximum transmission capacity.
**Multi-hop communications**: Due to signal propagation characteristics of wireless transceivers, Ad Hoc networks require the support of multi-hop communications; that is, mobile nodes that cannot reach the destination node directly will need to relay their messages through other nodes.

**Limited security**: Mobile wireless networks are generally more vulnerable to security threats than wired networks [94].

**Energy constrained nodes**: Mobile Ad Hoc nodes rely on batteries for proper operation. As an Ad Hoc network consists of several nodes, depletion of batteries in these nodes will have a great influence on overall network performance.

To support mobile computing in Ad Hoc wireless networks, a mobile host must be able to communicate with other mobile hosts, which may not lie within its radio transmission range. Therefore in order that one mobile host in the Ad Hoc network communicate with the other not lying in its transmission range some other hosts in its transmission range should route the packets from the source to the destination host. The conventional routing protocols (set of rules for effective data delivery) used in wired networks cannot be effectively used in Ad Hoc networks [95]. Hence new routing mechanisms are required to be used for routing in Ad Hoc networks, routing issues are not considered here and it is beyond the scope of this research.

Since many mobile hosts may be within transmission range of each other, there may be multiple routes for a packet to reach a destination. Therefore the source host should decide as to which optimal route to send the packets to reach its destination. Thus, there should be a SLA between the source mobile host and the host which routes the packets to the destination host.
The decision on selecting the best possible SLA, based on constraints can be performed by extending the mathematical model given in section 3.1.1.

Two main performance metrics required in addition to the wired networks for wireless services are

- Battery Consumption
- Mobility

3.5.1 Constraints

The Battery Consumption $BC_i$ for the offered service should be within the boundary $p7$

$$BC_i \leq p7$$

(3.14)

The Mobility Factor $M_i$ which gives the idea of how long the host $i$ will be in the transmission range of host $j$ for which packets need to be routed should not smaller than a particular constant represented by $p8$

$$M_i \geq p8$$

(3.15)

This mobility factor $M_i$ plays a crucial role in Ad Hoc networks because the hosts are all mobile. We generally assume that a mobile which has joined the Ad Hoc has more probability of staying in the network than the ones which came earlier than that. But the exact nature of the mobility of a host can be predicted only based on past performances of the mobile.

3.6 Architectural Framework for Wireless Network

In Ad Hoc networks, the SLA for the services between the hosts should be made dynamic as the network is very mobile in nature for which SLAs cannot be set up manually. The mathematical model extended gives the hosts in the
Ad Hoc networks the dynamicity to set up SLAs. In order to achieve this
dynamicity in setting up SLAs each host has to deploy the mathematical
model proposed in section 3.1.1 so that it will be able to choose the
agreements which satisfy the constraints given. The architecture for the
Policy Based SLAs in Ad Hoc networks are proposed in Fig.3.11 and Fig.3.12.

Fig 3.11. Policy Based Approach through Wired Network

In Fig.3.11 policy server is placed in the wired network. Polices are stored in
the Directory server. The Ad Hoc User1 is within the vicinity of both User2 and
User3. The User 4 is not within the transmission region of User1. So when
User1 wants to send a packet to User4, the connection establishment is
through intermediary users, User2 and User3.

Assuming that both the host services satisfy the constraints of User1, it is for
the User1 to choose a SLA among the two. In this case since User1 is
connected to a base station which in turn is connected to the wired network
having the policy server, User1 can query the policy server through the base
station and then the leaf access router and edge router. For simplicity we
have shown the policy server being connected to the base station through
only few hops. But in practice it may be many hops away from it. Once the request reaches the policy server, it fetches the appropriate policies from the directory server through LDAP.

3.6.1 Architecture with no wired networks

The architecture proposed in the previous section works effectively when the User who needs service from other Users is connected to a base station through which it can request the policy server in the wired network and take decisions. But in case if we do not have any hosts connected to the base station, still an SLA can be performed within the hosts as shown in Fig. 3.12.

![Fig.3.12 Policy Based Approach in the absence of Wired Network](image)

As shown in the architecture, each host runs in it a policy service and a directory service. Supposing User1 wants to communicate with User4 which is not in the transmission range. Hence both User2 and User3 as stated previously will extend service to User1. Now User1 deploys the mathematical model proposed in section 3.1.1 and comes up with one or more solutions. From the arrived solutions, it takes decision on the choosing the most
appropriate one by querying the policy service running in it. A lightweight version of the same can be installed in each user’s mobile device.

3.6.2. Simulation Results

A simulation is performed using the QualNet Network Simulator; the test environment has four Ad Hoc hosts (mobile device users) from User1 to User4 as shown in Fig.3.11. The total bandwidth, used bandwidth, reserve bandwidth, battery consumption, mobility factor and other performance metrics of the Ad Hoc hosts are given in Table 3.9 & 3.10.

<table>
<thead>
<tr>
<th>Performance Metrics</th>
<th>User1</th>
<th>User2</th>
<th>User3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Bandwidth Allocated (MBps)</td>
<td>3</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Bandwidth used at instant (MBps)</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Reserve Bandwidth (MBps)</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Remaining Bandwidth $G_y$ (MBps)</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Demand for Bandwidth to reach User4 (MBps)</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Delay ($x 10^{-3}$/sec)</td>
<td>7</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Packet Loss Factor</td>
<td>7</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Congestion Factor</td>
<td>30</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td>Queuing Delay ($x 10^{-7}$/sec)</td>
<td>8</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>Throughput ($x 10^3$ Bits/sec)</td>
<td>100</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td>Buffer Capacity (No. of Packets)</td>
<td>9</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Battery Consumption (mWh)</td>
<td>-</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Mobility Factor-minutes</td>
<td>-</td>
<td>25</td>
<td>18</td>
</tr>
<tr>
<td>Fraction of Bandwidth that can be given $F_y$ (MBps)</td>
<td>-</td>
<td>20</td>
<td>15</td>
</tr>
<tr>
<td>Cost $C_y$ ($)</td>
<td>20</td>
<td>30</td>
<td>10</td>
</tr>
</tbody>
</table>

The performance metrics of the Users are shown in Fig.3.13 to Fig. 3.16.
From the simulation, User1 decides upon the suitable service among the offers. Since only the service offered by User2 adheres to the performance metric constraints, it is chosen.
Fig. 3.17 Maximum Value Constraints

Fig. 3.18 Constraints with Max. Value

Fig. 3.19 Minimum Value Constraints

Fig. 3.20 Slack Values of Performance Metrics - I

Fig. 3.21 Slack Values of Performance Metrics - II

Fig. 3.22 Slack Values of Performance Metrics - III
The simulations are done with respect to the packet flow from User1 to User4.

Table 3.11: Original and Final value of the Objective

<table>
<thead>
<tr>
<th>Objective</th>
<th>Original Value</th>
<th>Final Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sum_{i,j} F_y G_y C_y \text{ of 'host2' ($)}$</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>Cost of 'User2' ($)$</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Fig. 3.23 Objective

From the above figures, it can be seen that as per the constraints given on performance metrics (Fig.3.17, Fig.3.18 and Fig.3.19), the objective of minimizing the cost is arrived. (Fig.3.23). Thus, an effective SLA is traded between User1 and User2. The values of the slack variables which are added in the process of solving the LP using simplex method are shown in Fig.3.20, Fig.3.21 and Fig.3.22.

3.7 Policy Based Directory Management Provisioning

The Wireless DiffServ Architecture allows the wireless service provider to configure new services dynamically using a policy provisioning protocol. Wireless Service Providers are increasingly looking to distributed enterprise directories [96] as a way to manage information about service users and systems. Although directories offer extremely powerful capabilities, they’re essentially a generic storage system. It’s the services and applications that surround directories [97] and put their information to work and make them valuable. The supporting meta-directory products address only a portion of
the problem; they synchronize data between some systems and do not activate and deactivate accounts or deliver management features that service providers require to administer access to their existing and planned environments [98].

Ultimately what is needed is a policy based distributed control solution that can keep identity and privilege data current and take action for all managed accounts when appropriate [99]. A distributed policy based architecture for effective directory management and provisioning is presented.

3.7.1 Architecture for Directory Management

The architecture for Policy Based Directory Management and Provisioning for Wireless DiffServ Networks is shown in Fig. 3.24

![Architecture diagram](image)

**Fig. 3.24 Architecture for Policy Based Directory Management and Provisioning**

This architecture has a component called as Identity System that identifies and catalogues the valid mobile users in the directory structure. Mobile users are identified and catalogued into different hierarchies based on priorities like...
High, Normal, Low and Very Low Priority users. The policy based provisioning system is the core of the architecture. Once the cataloging process has taken place, the user (mobile user) information is given; privileges based on higher level Policies are designed for different users. The privileges of the users are then reflected across various systems using the connectors.

Fig. 3.25 Centralized Control over Mobile user Privileges

This architecture supports a centralized control that can keep identity and privilege of the mobile users and take immediate action for all user managed accounts when appropriate. Fig. 3.25 illustrates the end-to-end flow of control to keep access privileges up-to-date with changes in identity (user identity).

Fig. 3.26 Reconciling changes made in the field

Fig. 3.26 shows how the architecture supports in capturing changes made in the field and reconcile the information against identity (user identity) data.
3.7.2 Policy Based Provisioning Engine

Policy based provisioning System (Fig. 3.27) is an important component of the architecture. Directory Server is the server where all the higher level policies are stored. Policy Management Console (PMC) is the entity responsible for creating, modifying or deleting policy rules or entries in Directory Server. It is the responsibility of the PMC to indicate the changes in the Directory Server as and when required using an internal event messaging service. The policy server is mainly responsible for

- Retrieval of relevant policies created by the administrator through the policy console after resolving any conflicts with existing policies;
- Taking appropriate actions such as deletion of existing decision states or modification of installed parameters in the PEP. For any modifications to currently installed policies.
- Arriving at policies decisions from relevant policies for policy decision requests and maintaining those decision states.

Therefore whenever a request is made to the policy server, the policy server queries the Directory Server, fetches appropriate policies and takes decisions accordingly.
The Policy Based Provisioning System has the following specific components that contribute to system integrity and performance.

**Audit Support:** Without an audit trail or log there's no way to know who is authorized to access a system or how that authority was obtained. Within this centralized identity and privilege management model, an auditor can examine specific privileges that the mobile service provider has granted to specific mobile users on any resource in the environment. The auditor can also look at why privileges were granted and how they were established.

**Data Maintenance:** Initially, the mobile service provider expends significant resources to find and remove dormant and redundant user IDs and accounts. The service provider does this by verifying that user IDs on different systems match real people. Once the mobile service provider does this the data of the users must be maintained in a centralized database system for adding or removing a user should have a reciprocal change in privileges also. In reverse, a user account that is added or changed in the field needs to be reconciled with actual users and business policies.

**Self-Service & Change Request Management:** The mobile users are given privileges to manage user IDs and passwords without service provider's intervention. The policy Based Provisioning system achieves this achieve this while eliminating the need for a help-desk to constantly reissue forgotten passwords. In a related fashion, decision makers i.e. the users need to be able to request access changes for themselves without the intervention of the service provider.

**Password Management and Security:** Password Management and security are a key issue relating to confidentiality and integrity. The Policy Based
Provisioning system uses Distributed Substring Authentication Protocol discussed in Chapter 5 for security and password management. The decision on whether to use DSAP or not and for which users are written as policies and are taken up by the Policy Based Provisioning System.

**Reporting:** Directories contain snapshots of current information but cannot store historical data. With user identity and privilege changes over time user trends can be spotted and reports generated based on that. This reporting capability of the architecture is imperative to support the inevitable security audits for mobile service provider.

**Role-Based Access Control & Policy Management:** Role Based Access Control (RBAC) simplifies and streamlines the administrative process. Changes in policy and procedure can take effect immediately — affecting the content or privileges available to a set of users grouped under different roles. This reduces and eliminates the need for service provider to track and make changes for every individual mobile user.

**Identity Management:** Identity of the users should be maintained. This architecture supports IDSynch for identity management of the mobile users so that the mobile service providers keep track of the identity of the users.

The vision for enhancing wireless DiffServ network through integration with the directory management and provisioning service is to provide network-enabled applications appropriate information from the directory.
3.8 Summary

SLA trading framework for both wired and wireless environments were presented. The directory management and provisioning has summarily the following:

- Support for applications that have the capability to leverage the wireless network infrastructure transparently on behalf of the end user
- Provides a robust, extensible foundation for building network-centric applications
- Provides dynamic SLA trading between different DiffServ domains for effective Quality of Service
- Enables end-to-end differentiated network services on a per-user basis
- Enables wireless network-wide service creation and provisioning
- Enables the network-wide management

In Chapter 4 policy provisioning in routers, with efficient data handling capabilities are presented.