Simulation analysis of quality of service parameters for on-board switching on ATM network for multimedia applications

Zainab Abbas Pota
The University of Toledo

Follow this and additional works at: http://utdr.utoledo.edu/theses-dissertations

Recommended Citation
http://utdr.utoledo.edu/theses-dissertations/943

This Thesis is brought to you for free and open access by The University of Toledo Digital Repository. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of The University of Toledo Digital Repository. For more information, please see the repository's About page.
A Thesis

entitled

Simulation Analysis of Quality of Service Parameters for On-board Switching on ATM Network for Multimedia Applications

by

Zainab Abbas Pota

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Master of Science Degree in Electrical Engineering

______________________________
Dr. Junghwan Kim, Committee Chair

______________________________
Dr. Patricia R. Komuniecki, Dean
College of Graduate Studies

The University of Toledo
December 2010
An Abstract of

Simulation Analysis of Quality of Service parameters for On-board switching on ATM Network for Multimedia Applications

by

Zainab Abbas Pota

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Master of Science Degree in Electrical Engineering

The University of Toledo

December 2010

In this thesis the Quality of Service parameters for transmission of multimedia applications like voice, video and data are evaluated on ATM networks using various On-board switching algorithms and Queueing techniques. Due to the increasing interest in multimedia traffic it has become essential to consider satellite on-board switching capabilities for managing the multibeam input and output. Multimedia application has different requirements in terms of delay, jitter, and bandwidth and packet loss [1]. The network through which the data is transmitted and received has a large number of nodes, and the nodes have high variability and can cause blocking so blocking probability is another important parameter. Considering the expansion of internet there is demand for development of new protocols, architectures and network systems at the terrestrial level and also on the on-board processor keeping in mind the complexity level and cost factor. To improve these parameters different techniques like Differentiated services and On-
board switching algorithms like Dijkstra’s algorithm, Bellman-Ford algorithm and K-shortest path algorithm have been implemented on ATM networks like Banyan network, Benes network and Batcher-Banyan network using a single server M/G/1 queue. Simulation for MF-TDMA uplink and TDM downlink and evaluation of delay in algorithms, Queueing delay, Burstiness factor and Blocking probability has been performed. The results show that for small and medium size networks Bellman-Ford algorithm is effective in terms of delay whereas for large networks K-shortest path algorithm is most effective. To achieve low blocking probability and burstiness factor the Batcher-Banyan ATM network for single server M/G/1 queue is most efficient.
Acknowledgements

There are a lot of people I am grateful to who have helped me in my research and completion of my thesis. I want to sincerely thank Dr. Junghwan Kim for his constant support, guidance and understanding throughout the span of my Master’s program. I would like to thank Dr. Mohsin Jamali and Dr. Ezzatollah Salari for their time and serving as my committee members.

I am really grateful to have great friends like Anandi Gandhi, Mohammed Majid Wadood and Disney Maxwell for being there whenever I needed them.

Last but not the least I want to thank my family Abbas, Munira, Tasneem and Mohammad and my new family Husein, Sakina and Abdultayab for providing me with abundance of love, encouragement and support. Without them everything would be meaningless.
# Table of Contents

Abstract .................................................................................................................................................. iii  
Acknowledgements ............................................................................................................................... v  
Table of contents .................................................................................................................................... vi  
List of Tables ........................................................................................................................................... viii  
List of Figures ......................................................................................................................................... ix  
1 Introduction ........................................................................................................................................... 1  
   1.1 Multibeam technology....................................................................................................................... 1  
   1.2 Current Networking Scenario........................................................................................................... 4  
   1.3 On-board switching............................................................................................................................. 5  
   1.4 Outline of Thesis................................................................................................................................. 7  
2 Techniques for On-board Switching....................................................................................................... 8  
   2.1 On-board processing........................................................................................................................... 8  
   2.2 Queues and Queueing Analysis......................................................................................................... 11  
      2.2.1 Synchronous single server queues............................................................................................. 12  
   2.3 Switching networks............................................................................................................................ 14  
   2.4 ATM switching fabrics....................................................................................................................... 17  
      2.4.1 Banyan network........................................................................................................................... 17  
      2.4.2 Batcher-Banyan network............................................................................................................. 19  
      2.4.3 Benes network............................................................................................................................. 20  
   2.5 ATM switching delays....................................................................................................................... 2
1  2.6 Differentiated Services.................................................................23
    2.7 User terminal transmission, scheduling and switching.....................25

3 Shortest path algorithms.........................................................................28
   3.1 Shortest path algorithms..............................................................28
   3.2 Dijkstra’s algorithm........................................................................34
   3.3 Bellman-Ford algorithm...............................................................38
   3.4 K-shortest path algorithm.............................................................39

4 Simulation Scenario and results............................................................44

5 Conclusion and future work....................................................................55
   5.1 Conclusion....................................................................................55
   5.2 Future work..................................................................................56

References....................................................................................................57

Appendix A.................................................................................................59
List of Tables

Table 1.1: Various communication services.................................................................5
Table 2.1: Network taxonomy .........................................................................................16
Table 4.1: Medium Data rate chart for data, voice and video used for simulation.......47
List of Figures

Figure 1.1: Two-way interactive multimedia satellite communications system........3
Figure 2.1: System block diagram of an on-board processing satellite payload..........10
Figure 2.2: Simulation model for on-board queue, server and switch.....................11
Figure 2.3: Structure of an 8 X 8 Banyan switch.............................................18
Figure 2.4: Batcher-banyan network..................................................................20
Figure 2.5: 8 X 8 Benes network......................................................................21
Figure 2.6: $w_{ij}$ evaluation for each Per Hop Behavior......................................24
Figure 2.7: Scheduler block diagram.................................................................26
Figure 3.1: A transport network as a connected graph......................................30
Figure 3.2: Illustrating Dijkstra’s algorithm through a 6 node network.................36
Figure 3.4: Finally all the nodes are updated and the shortest path to the destination node 6 is found to be 12 units.................................................................36
Figure 3.5: Illustration for Bellman-Ford example...........................................40
Figure 3.6: Initializing by making the source node at distance 0 and other nodes at infinity.................................................................40
Figure 3.7: Relaxing each edge until finally we obtain the shortest path to each node using the steps mentioned in the Bellman Ford algorithm..........................41
Figure 3.8: Illustration of K-shortest path, with k= 6........................................44
Figure 4.1: MF-TDMA for the uplink………………………………………………………47
Figure 4.2: Time delay for data link versus number of nodes on the network…………48
Figure 4.3: Time delay versus number of nodes for Bellman-Ford algorithm………49
Figure 4.4: Time delay versus number of nodes for K-shortest path algorithm……50
Figure 4.5: Total delay for Dijkstra’s, Bellman-Ford and K-shortest path without queueing delay……………………………………………………………………………51
Figure 4.6: Time delay for algorithms with Queueing versus number of nodes……52
Figure 4.7: Queueing delay versus number of nodes…………………………………53
Figure 4.8: Burstiness factor versus traffic load………………………………………55
Figure 4.9: Blocking probability for Batcher-Banyan ATM network versus call holding time…………………………………………………………………………………56
Figure 4.10: Blocking probability versus number of servers for the queueing system…56
Chapter 1

Introduction

In modern satellite communication systems the Quality of Service (QoS) management has become a crucial topic due to the increasing interest in multimedia traffic. Satellite communication has become an integral part of terrestrial data networks. Hence, it has become essential to consider satellite on-board switching capabilities for managing the multibeam input and output. Also, multimedia application has different requirements in terms of delay, jitter, and bandwidth and packet loss [1]. The network through which the data is transmitted and received has a large number of nodes, and the nodes have high variability. Considering the expansion of internet there is demand for development of new protocols, architectures and network systems at the terrestrial level and also on the on-board processor keeping in mind the complexity level and cost factor.

1.1 Multibeam Technology:

With the idea of developing a new protocol comes the next factor; choosing the appropriate network. There are various networks implemented but in the current scenario
the ATM (Asynchronous transfer mode) network is chosen. Asynchronous Transfer Mode (ATM) is an International Telecommunication Union–Telecommunications Standards Section (ITU-T) standard for cell relay wherein information for multiple service types, such as voice, video, or data, is conveyed in small, fixed-size cells. ATM networks are connection-oriented [2], that is use a protocol to establish an end-to-end logical or physical connection before any data may be sent.

As mentioned above, satellite communication systems are playing a significant role in the exchange and delivery of many types of information, from voice communications to high definition television and Internet services. A multimedia service consists in transporting different types of information between the end-users by keeping time a time relation in the transport of different types, for example voice and data or video coupled with sounds and texts. High reliability and flexibility for setting up links make satellites unique for specific applications [3]. High-gain multibeam antennas at the satellite have made possible the use of small antennas (0.5–1.2 m) and low-power transmitters at the user terminals (UTs), reducing their costs, suitable for satellite-based two-way interactive multimedia communications. Multibeam technology also helps to increase the system capacity resulting from frequency reuse between beams, which allows a larger supportable population of users. While this approach improves the transmission aspects, it introduces the requirement for full connectivity between User Terminals in different beams. The satellite acts as a provider of dynamic links connecting any pair of UTs in the network whenever the need arises. From the network management standpoint, the utility of satellite resources has to be maximized in supporting a large population of UTs. On the
other hand, the user quality-of-service (QoS) requirements should be maintained. Real-time applications like voice and videoconferencing, for example, are delay and jitter sensitive.

Data applications, on the other hand, are more tolerant to delay and jitter. Service differentiation and QoS guarantees are, therefore, the currently important issue [1].

Figure 1 shows an example of a multibeam satellite communications system to support various services for both residential and business applications. The supported service groups can be classified into two main categories: broadband satellite access (BSA) for two-way interactive multimedia services and private business network (PBN) [1].

![Figure 1.1: Two-way interactive multimedia satellite communications system [1]](image-url)
Asynchronous Transfer Mode (ATM) networks will support the wide range of applications, including real-time (voice, video) services and different types of data-service. The traffic types are heterogeneous in the sense that they have different bit-rate profiles and QoS requirements. Bit-rate profiles are generally described in terms of their peak bit-rate, burstiness and average burst duration [1].

1.2 Current Networking Scenario:

There are various parameters to the telecommunication services. New services have been thought and network-engineered in a span of years, and also the tremendous progress in VLSI technology makes it difficult to foresee the new network capabilities that the end-users will be able to exploit even in the very near future.

Communication services could be classified based on their information capacity, which corresponds to typical rate (bits/s) at which the information is required to be carried by the network from the source to the destination.

Real-time services are sound and image services involving interactions between two or more people. On the other hand, data services as well as unidirectional sound and image services are not real-time services, since even a high delay incurred by the information units in the transport network does not impair the service itself, rather it somewhat degrades the quality. There are important factors here for checking Quality of Service one of them is the Burstiness factor. Burstiness factor by definition is the ratio between the average information rate of the service and the channel peak rate.
### Table 1.1: Various communication services [8]

<table>
<thead>
<tr>
<th>Class</th>
<th>Mbit/s</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low speed</td>
<td>0.0001-0.001</td>
<td>Telemetry/POS</td>
</tr>
<tr>
<td></td>
<td>0.005-0.1</td>
<td>Voice</td>
</tr>
<tr>
<td></td>
<td>0.001-0.1</td>
<td>Data/Images</td>
</tr>
<tr>
<td>Medium Speed</td>
<td>0.1-1</td>
<td>Hi-Fi sound</td>
</tr>
<tr>
<td></td>
<td>0.1-1</td>
<td>Videoconference</td>
</tr>
<tr>
<td></td>
<td>0.1-10</td>
<td>Data/images</td>
</tr>
<tr>
<td>High speed</td>
<td>10-50</td>
<td>Compressed TV</td>
</tr>
<tr>
<td></td>
<td>100-500</td>
<td>Uncompressed TV</td>
</tr>
<tr>
<td></td>
<td>10-1000</td>
<td>Data/images</td>
</tr>
</tbody>
</table>

1.3 On-board Switching:

On-board switching is used to provide connectivity between two or more users of a communications system. One of the most basic requirements on the capabilities of on-board processing satellites is the ability to do switching. On-board switching is a technology enabling to support a single satellite hop network topology in multi-beam satellite systems. In the payload of the satellite, the on-board switch is responsible for the establishment of physical or logical paths between its inputs and outputs usually associated with the up-link and down-link beams respectively.

On-board switching considerably enhances the resource utilization and performance of the multi-beam satellite systems and displays the following advantages:
• Support of a single satellite hop full meshed network topology for ground terminal to ground terminal(s) data connection,

• Halving of transmission delay in end-to-end ground terminals data connections,

• Halving of the radio resources involved in end-to-end ground terminals data connections, and therefore increase of the overall satellite system transmission capacity,

• Active on-board support of traffic differentiation and Quality of Service (QoS),

• On-board duplication of traffic bursts for efficient multicast support, and therefore optimization of the radio resources usage for multicast traffic transports [3].

In a Multibeam configuration, the satellite must switch the received traffic to the appropriate downlink beam. The type of switching performed depends on the satellite payload technology.

For a Non-regenerative (or repeater or bent-pipe) satellite, the received signals are not demodulated, and uplink-to-downlink connectivity is performed by frequency switching. In this case, the assigned frequency-time slot positions in the MF-TDMA frame (or multiframe) can be further used to map to a specific downlink destination beam. Connectivity is performed by frequency switching. Due to the burstiness of multimedia traffic, this map must be dynamically changed to cope with the traffic variation. The dynamic change of this connectivity pattern is limited by the achievable switching speed of the physical on-board matrix, e.g., microwave switch matrix. This dynamic switching based on frequency-time slot positions can be considered as a fast circuit switching scheme. Furthermore, the User Terminal needs to sort the traffic packets according to
their downlink destinations and transmit them in the corresponding assigned time-frequency slots.

In short the objectives are as follows [4]:

1) To route (switch) the traffic from various source regions to destination regions without conflict;

2) To schedule the given traffic demand so that the maximum throughput (or utilization) of satellite transponders is achieved; and

3) To keep the number of switching matrices small.

1.4 Outline of the thesis:

After the introduction in Chapter 1 about the networking and switching scenario today and the demand for Quality of Service parameters from the User end and the need for testing new switching networks and algorithms, this thesis will give a detailed explanation of the working of the On-board processor switching scenario in Chapter 2 with different switching networks, switching fabrics, queuing techniques and service techniques for QoS. Chapter 3 gives an in-depth explanation of the algorithms considered in this thesis for the efficient switching of data packets with worst-case complexity analysis. Chapter 4 concentrates on the simulation and results. Chapter 5 gives the discussion of the results and future work.
Chapter 2

Technique for On-board Switching

On-board processing (OBP) and switching (OBS) is still fairly young in satellite communications and the end-application of one payload to another can vary significantly, there is no uniformly accepted definition of an “on-board processing” payload. Thus, a moderately detailed definition of what an on-board processing payload encompasses with respect to this research work is mentioned.

2.1 On-Board Processing (OBP)

The following system blocks will comprise the on-board processing payload [5]:

- Receive Antenna Sub-unit: responsible for the reception of the uplink carrier signal(s).
- Receive Sub-unit: performing the initial uplink carrier signal processing, which typically includes,
  - Low Noise Amplifier(s) (LNA): responsible for amplifying the incoming signal, while introducing as little noise into the signal as possible.
- Dehop module: responsible for reassembling or “dehopping” the original carrier signal as it was prior to hopping.
- RF Downconverter: responsible for translating the uplink RF frequency down to some IF for further processing.

- On-board Processing sub-unit
  - Demultiplexer/Channelizer: separation of each incoming carrier signals for individual processing.
  - IF Downconverter: responsible for transforming the pass band signal back into the baseband equivalent for further processing.
  - Demodulator: recovers the information down to the bit level from the baseband carrier wave.
  - Forward Error Correction Decoder: responsible for performing error detection and correction in order to reconstruct the original digital data sequence that was transmitted.
  - Message Router: responsible for extracting address routing information from the data packet headers in the data sequence. The information is then used to determine the destination of the data, which could be one of the satellite’s crosslinks or one of the multiple downlink transmitting beams.

- Retransmit Sub-unit: responsible for preparing data for downlink transmission. This section is comprised of modules which perform the inverse of functions described already in the receive sub-unit. Thus, for brevity their descriptions will be omitted.
  - Encoder
  - Modulator
- Multiplexer
- Upconverter
- Rehop module
- Amplifier(s)
- Beam selector / switching matrix

- Transmit Antenna Sub-unit: responsible for the transmission of the downlink and intersatellite link (ISL) carrier signal(s).

The overall topology of the system is shown in Figure 2.1.

Figure 2.1: System block diagram of an on-board processing satellite payload
2.2 Queues and Queueing analysis

For the IP-based network in our satellite based system diverse applications like voice, video and data require different levels of service. To achieve efficient performance and maintain Quality of Service requirements of the user On-ground and On-board queues are considered in this thesis. The Queueing performance in supporting IP-based multimedia is evaluated.

At the uplink as well as downlink servers, voice and video are regarded as first priority, and this real-time traffic is forwarded to the ATM switching network for switching as soon as it arrives at the server. Non-real time traffic that is data are treated second priority by the server and send to the queue before it can be possible for the server to forward it to the switch.

![Simulation model for on-board queue, server and switch](image)

Figure 2.2: Simulation model for on-board queue, server and switch [6]
In this thesis, we consider a discrete-time buffer system with first-in-first-out queuing discipline. Time is divided in fixed-length slots separated by slot boundaries [7]. Message arrivals and departures occur on slot boundaries. The arriving customer on slot boundaries is called arrival instants and the time instant between two arrival instants is called inter-arrival time. The transmission time or service time of a message is equal to one slot, so messages can leave the buffer no earlier than one slot of their arrival instant.

2.2.1 Synchronous Single-server Queues [8]:

This thesis considers the M/G/1 queue for queuing its data packets because it has variable service time which is an important factor on comparison with the M/D/1 queue which has a unit service time.

For synchronous queues the customers join and leave a queue at discrete epochs nt (nt= 1, 2, 3...), that is they are integrals of the slots with start and end occurring at slot boundaries. The main random variables characterizing a queue are as follows:

- $A$: arrival size, that is number of service requests offered to the queue in a given time period;
- $W$: waiting line size, that is number of customers in the queue waiting for the server availability;
- $Q$: queue size, that is total number of customers in the queue;
- $\theta$: service time, that is amount of service requested to the queue by the customer;
- $\eta$: waiting time, that is time spent in the queue by a customer before its service starts;
• $\delta$: queueing time, that is the total time spent by a customer in the queue (waiting time + service time)

• $\lambda$: average arrival rate to the queue (for a synchronous queue it is the probability that a service request is received in a slot by the queue)

• $\rho$: service utilization factor, the time fraction in which each server in the queue is busy

• $\Pi$: loss probability, the probability that the new customer is not accepted by the queue

In M/G/1 queue the customers join the queue according to a Poisson process with average arrival rate and the service times are independent and identically distributed (IID) with an arbitrary probability distribution [8]. The number of customers left behind by the $n$-th departing customer who leaves the system at epoch $n$ is described by [8]

$$Q_n = \max \{0, Q_{n-1} - 1\} + A_n \tag{2.1}$$

$A_n$ is the number of arrivals during the service time of the n-th customer.

The average number of customers in the system is then given by $E[Q]$, which results in the well-known Pollaczek–Khinchin (P–K) mean-value formula [8]:

$$E[Q] = \rho + \frac{\lambda^2 E[\theta^2]}{2(1 - \rho)} \tag{2.2}$$

The mean value of the queueing delay is obtained via Little’s formula [8]:

$$E[\delta] = E[Q] = E[\theta] + \frac{\lambda E[\theta^2]}{2(1 - \rho)} \tag{2.3}$$
\( \lambda \) is the arrival rate, \( \theta \) is the service time and \( \rho \) is the service utilization factor.

The average number of customers waiting in the queue for the server availability is obtained from considering that the average number of customers in the server is \( E[\theta] \) and the corresponding waiting time is given by Little’s formula, that is [8]

\[
E[\eta] = E[\delta] - E[\theta] = \frac{E[W]}{\lambda} = \frac{\lambda E[\theta^2]}{2(1 - \rho)} \quad (2.4)
\]

### 2.3 Switching Networks

It has been recognized that onboard Asynchronous Transfer Mode (ATM) switching is going to play an important role in the near future of satellite communication system.

Until now lots of ATM switching fabrics have been applied in Optical Fiber channel, for instance, Batcher-banyan networks and Benes networks that are used on the On-board processor [9]. The network considered is a satellite ATM network.

Communication scientists have been studying structures referred to as connecting networks for use in switching systems characterized by a very large size, with thousands of inlets and outlets. Hence there is a need for the development of large interconnection networks for switching systems in which a distributed processing capability is available to set up required permutations [8].

The driving forces for the above scenario are as mentioned below:

1. Switching fabrics capable of carrying aggregate traffic on the order of hundreds of Gigabits,
2. Tremendous progress achieved in CMOS VLSI technology that makes the distributed processing of interconnection networks feasible for very large networks too.

The connection capability of a network is usually expressed in two indices referring to absence and presence of traffic carried by the network:

1. Accessibility: full accessibility is when inlet can be connected to the outlet when no other I/O connection is established in the network and limited is when such a property does not hold

2. Blocking: if at least one I/O connection between an arbitrary idle outlet cannot be established by the network owing to internal congestion due to the already established I/O connections.

Table 2.1: Network taxonomy [8]

<table>
<thead>
<tr>
<th>Network Class</th>
<th>Network type</th>
<th>Network states</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-blocking</td>
<td>Strict-sense non-blocking</td>
<td>Without blocking states</td>
</tr>
<tr>
<td></td>
<td>Wide-sense non-blocking</td>
<td>With blocking states</td>
</tr>
<tr>
<td></td>
<td>Rearrange able non-blocking</td>
<td>&quot;</td>
</tr>
<tr>
<td>Blocking</td>
<td>Others</td>
<td>&quot;</td>
</tr>
</tbody>
</table>

- Strict-sense non-blocking (SNB), if the network can always connect each idle inlet to an arbitrary idle outlet independent of the current network permutation that is independent of the already established set of I/O connections and of the policy of connection allocation.
• Wide-sense non-blocking (WNB), if the network can always connect each idle inlet to an arbitrary idle outlet by preventing blocking network states through proper policy of allocating the connections.

• Rearrangeable non-blocking (RNB), if the network can always connect each idle inlet to an arbitrary idle outlet by applying, if necessary, a suitable internal arrangement of the I/O connections already established.

From the table above [8], we see that SNB network satisfies at the same time the definition of WNB and RNB networks, but not vice versa. The above three non-blocking network types are characterized by a decreasing cost index, starting from strict sense non-blocking and ending with rearrangeable non-blocking.

The cost index of the network has assumed to be the number of cross points in the network, as reasonable in the space-division switching systems of the sixties and seventies. Nowadays such a performance index alone does not characterize the cost of an interconnection network for broadband applications, owing to the extreme degree of integration of the electronic components in a single chip.

2.4 ATM Switching Fabrics

ATM satellite network concept is very similar to current ground offerings. Both systems include user terminals, access to the physical transmission medium, and interface to the switching elements [9].
The ATM switching fabric is regarded as a network, and the switching elements are regarded as nodes in the network. The two main functions of switching fabric are as follows:

1) To explore the optimal connections quickly
2) To detect blocking states on switching elements.

2.4.1 Banyan Network [10]:

The Banyan switch is a multistage self-routing architecture which uses fewer elements than the minimum number required for a rearrangeably nonblocking design. More specifically, an N×N Banyan switch uses (N/2) log N elements. Consequently, the switch cannot be nonblocking; input-to-output permutations can be constructed that cannot be concurrently routed with the switch. Therefore, smoothing buffers must lie inside the switch to achieve a reasonably low packet loss rate.

The structure of an 8×8 Banyan switch is depicted in the diagram.

![Figure 2.3: Structure of an 8 X 8 Banyan switch [10]](image-url)
Elements are arranged in three columns of four elements each; in a pattern that resembles a grid of butterflies. The inputs to the switch are the inputs to the elements in the first column, and the outputs of the last column are the outputs from the switch. In each element, one output is connected to the input of the element just horizontally on its right, and the other goes to an element whose line number, represented in binary, differs in precisely the j's bit, where j is the column number of the element (counting from 0). The path shown in bold in the diagram illustrates how to connect input 7 to output 0. Since all the bits in the binary representations of the input and the output differ, all elements along the path are set to "cross". Note that every such path is unique. Obviously, several paths cannot be routed concurrently unless they happen to require the same states of the elements.

Several remedies can be employed to attempt resolving this type of routing conflict. They are as follows:

1) Provide buffers within the elements, so that cells that cannot be immediately delivered are stored and their routing deferred according to some contention resolution policy;

2) Run the internal links at a rate that is a multiple of the cell arrival rate, sequentially establishing several paths within the duration of one cell. To provide an insight to how good these techniques can be in reducing packet loss rate, it suffices to quote the results of a computer simulation for a large (1024×1024) Banyan switch run at full input load. With the internal links running at twice the cell rate (hence capable of establishing two subsequent paths within one time slot) and a buffer size of 5 cells in each element, as many as 92 percent; of the input cells were delivered, compared to about 25 percent; for a simple unbuffered switch, and about 75 percent; for a double-rate unbuffered switch.
Still, to achieve reasonable packet loss rates (such as one packet per million), the input load would have to be reduced considerably.

2.4.2 Batcher Banyan Network [10]:

The Batcher sorting procedure involves 3 levels of sorting to produce Non Blocking input for a 3 stage Banyan network. The price for removing the wait states in the Banyan network is more nodes (more cost) and a longer travel time through the greater number of nodes. These switches are however much faster than simple Banyan switches and is much more expensive.

![Figure 2.4: Batcher-banyan network [10]](image)

2.4.3 Benes Network: \( (m = n = 2) \)

A rearrangeably nonblocking network of this type with \( m = n = 2 \) is generally called a Benes network. The number of inputs and outputs is \( N = r \times n = 2r \). Such networks have \( 2\log_2 N - 1 \) stage, each containing \( N/2 \) \( 2 \times 2 \) crossbar switches, and use a total of \( N\log_2 N - \)
$N/2$ 2×2 crossbar switches. For example, an 8×8 Benes network (i.e. with $N = 8$) is shown below; it has $2\log_2 8 - 1 = 5$ stages, each containing $N/2 = 4$ 2×2 crossbar switches, and it uses a total of $N\log_2 N - N/2 = 20$ 2×2 crossbar switches. The central three stages consist of two smaller 4×4 Benes networks, while in the center stage, each 2×2 crossbar switch may itself be regarded as a 2×2 Benes network. This example therefore highlights the recursive construction of this type of network.

![8 X 8 Benes network](image)

**Figure 2.5: 8 X 8 Benes network [10]**

### 2.5 ATM Switching Delays

For delay considerations in an ATM network there are many factors. The main elements [11] that contribute to the delay through an ATM network are the fixed switching delay of the nodes, the propagation delay experienced between nodes and the queuing delay encountered for the inter-node links.

Results are presented for generated networks disregarding delays at the nodes and for simulated ATM networks. The delay experienced by information travelling across an ATM network consists of [11]:

...
1] Packetisation and depacketisation delays,

2] Propagation delays ($P_{ij}$),

3] Transmission delay (TD),

4] Fixed switching delay and queueing delay ($K_j$).

Packetisation delay is the same for all destinations and occurs before each cell is transmitted; similarly depacketisation delays occur after a cell has been received. These two delay elements do not contribute to the routing delays and have therefore been ignored in our calculations.

Taking into account all other delays, we can estimate the total delay (in s) experienced on a multicast route through the network using the following equation which is based on analysis carried out by M. De Prycker:

$$\sum \frac{U_{ij} \times TD}{2(1-U_{ij})} + \sum TD + \sum P_{ij} + \sum TD \times K_j$$

(2.5)

where, $U_{ij}$ is the traffic load on the network.

Some network technologies (such as ATM and Frame Relay) are already based on switching, and it is not necessary to introduce another mechanism [12]. At the same time, it is useful to maintain some consistency of control through the entire network. In order to achieve this mechanism is defined to express the MPLS label within the network layer protocol fields (the VPI/VCI or DLCI).

In an ATM network, the whole MPLS packet is presented and is then segmented into ATM cells. Each cell is labeled with a VPI/VCI that is equivalent to the MPLS label. The
cells are forwarded through the ATM network and re-assembled at the end to re-form the MPLS packet.

2.6 Differentiated Services

The Internet Engineering Task Force (IETF) has proposed many service models and mechanisms in order to meet the demand for QoS.

Some key technologies have been developed to enhance the traditionally best-effort IP service model in order to provide the required QoS. The Differentiated Services (DiffServ) architecture allows IP packets to receive different forwarding treatments based on the DiffServ Code Point (DSCP) contained in the packet header. Multiple Protocol Label Switching (MPLS) is an emerging technology that can be used to further enhance the IP service model; MPLS ability to set explicit paths through a network facilitates traffic engineering while its tight integration with IP allows IP routing to be combined with fast label switching [15].

In IP networks, the Differentiated Service (DiffServ) approach seems to be the best to satisfy the QoS constraints of several traffic classes, according to the DiffServ approach, have been considered and the switch takes into account their priority, queue length and time spent inside queues [1]. We consider the Differentiated Services (DiffServ) approach that allow having scalability and keeping low complexity core nodes, like satellites, moving complexity to the edge nodes. This approach uses the DS Code Point field of the IPvX packet header in order to identify the payload type. Each node manages the packets according to this field; in DiffServ systems this operation is known as PHB (Per Hop Behavior).
Each queue is logically divided in several sub-queue related to different traffic classes; we have assumed 4 traffic classes (or PHB using a DiffServ terminology) and precisely one Expedited Forward class [17], two type of Assured Forward classes [18] with different priority and one Best Effort class.

The EF class [16] can be used to build a low loss, low latency, low jitter, assured bandwidth, end-to-end service through DiffServ domains, in order to overcome loss, latency and jitter introduced by queues in each node. The priority function has been implemented as shown in Figure 2.6 where the priority reaches the maximum value whenever one packet is inside the queue.

The AF class [16] has been introduced in order to provide inside a DiffServ domain different levels of forwarding assurances; in particular four levels of emission priorities are defined, and for each class, in each DiffServ node, a certain amount of forwarding resources (buffer space and bandwidth) is allocated.

The BE class [16] has been introduced for traffic without particular QoS requirements in terms of delay, jitter and packet loss but, despite this, a linear function growing with the number of packets in the queue has been used.

The total waiting time has been introduced in order to solve the starvation problem for the low priority traffic class (aging method); otherwise, in case of high priority heavy traffic, low class packets could remain in starvation. This additional term is present in both AFs and BE input values computation.
Through input values all QoS requirements for each class have been introduced: higher are \( w_{ij} \) values and higher is the priority of the corresponding queue.

2.7 User terminal transmission, scheduling and switching [15]

The UT, not the scheduler, is responsible for implementing QoS mechanisms. Giving responsibility for QoS to the UT is because the UT has the most up-to-date information about current traffic conditions and is thus best equipped to make QoS related decisions. This distributed functionality approach simplifies the scheduler’s architecture, which is desirable, especially if the scheduler is located on board the satellite where real estate and computing resources are limited. In addition, this makes the scheduler design independent of the type of service offered (e.g. IP, ATM)

- Scheduler [6]: Scheduler manages the link capacity to ensure efficient link utilization. It uses the Combined Free/Demand Assignment Multiple Access (CFDAMA) scheme to assign capacity to the UTs based on their requests; in this scheme, any free (Unassigned) slots remaining in the MF-TDMA frame after all UT requests have been satisfied are distributed among the UTs.
• Traffic Selector: This module performs DiffServ traffic classification and conditioning functions. The traffic selector accepts user IP packets and classifies them into DiffServ Classes of Service (EF, AF1, AF2, AF3, AF4 or DE). Following classification, each packet is marked with the DSCP corresponding to the chosen CoS. Alternatively, the classification and marking may be performed inside the user’s network, in which case the traffic selector does not need to perform these functions. Next, the traffic in each class is metered to determine whether it conforms to the TCA. If so, the traffic is inserted into the appropriate IP queue.

• IP Queues: The IP queues hold the packets arriving from the traffic selector. There is one queue for each DiffServ CoS Regenerative payload. Regenerative payloads demodulate the uplink carriers and recover the baseband data, enabling the satellite to
access the layer-2 cell headers and use them for switching. MPLS may be used to control the switching, as described in [4]. Since the UTs segment user IP packets into fixed-length cells in the format specified for the uplink, it is unlikely that the satellite would use the layer-3 IP headers for switching: this offers no advantage over using the layer-2 headers and would require the payload to perform packet reassembly. Layer-2 on board switching offers significant advantages over frequency switching. First and foremost, it decouples the switching process from the DCA process, significantly simplifying the latter. Second, it allows individual cells to be forwarded independently, as opposed to frequency switching, which forwards entire carriers.
Chapter 3

Shortest Path Algorithms

Our shortest path algorithms are a step-by-step procedure for solving the shortest path or minimum cost flow. We are interested in finding the most efficient algorithm for this problem. Efficiency involves various computing resources needed for executing an algorithm. We consider time taken by an algorithm as our metric for measurement [12]. This chapter gives a detailed description of the algorithms implemented for this research on the previously mentioned switching elements.

3.1 Shortest Path Algorithms [12] [13]:

Transport service is a way of delivering user traffic of a certain type from the service source point to the service destination point with a certain quality. The user traffic characteristics, as well as the service quality, are agreed upon by the user and the Service Provider [13].

Path computation is the process of selecting or determining the path, and can be performed either at the time of, or ahead of service provisioning. The former is called on-line path computation and the latter, off-line path computation. The hybrid case is also
possible where some (usually initial) path computation is performed offline and some is performed on-line.

If all paths for the service are computed on one node, such path computation is called centralized path computation. A distributed path computation is performed by several cooperating computation entities either to provide a single complete path in response to a single request, or when a series of requests is issued by controllers along the path as the service is established. Routing in ATM networks is challenging since multiple metrics are specified in the connection setup request.

The routing algorithm must follow the metrics below [14]:

1. Select a path that supports the requested QoS parameters
2. Minimize the overall call blocking probability
3. Generate paths quickly so that call-setup time is minimized.

These objectives conflict, so some tradeoff must be made between global optimization of blocking, and call setup duration. The use of path-pre-computed paths enables call setup time to be reduced, while on-demand path calculation can be used to perform more complex optimization when pre-computed paths are not usable.

Path pre-computation is needed to accelerate call setup, while on-demand calculation provides a backup method should no stored path be suitable.

Different pre-computation schemes can be used, depending on the network type and size. For example, ATM networks can be treated as a virtual network by creating Virtual Path (VPs) between nodes. Assuming at least one VP exists between each source and destination node, a pre-computation scheme generating all single-link paths may suit. To support alternative routing on VPs one could include paths consisting of two VPs.
A transport network is usually presented as a connected weighted graph \( G (V, A) \), where \( V = \{ v_0, v_1, \ldots, v_n \} \) are vertices representing transport nodes (cross connects, add-drop multiplexers, and so forth) that are capable of originating, terminating, and/or switching user data traffic; and \( A = \{ a_0, a_1, \ldots, a_m \} \) are arcs needed to deliver user traffic between adjacent nodes in one direction. A path is a sequence of contiguous arcs that interconnects a pair of vertices.

For example, in Figure 3.1, XCY and XBDFY are different paths between vertices X and Y. A path contains loops if there is at least one vertex that it crosses more than once. For example, the path XBEDBCY contains a loop — node B is crossed twice.

A path is called simple or elementary if it does not contain loops. For every arc, an integer number is assigned indicating the preference of using this arc versus using other arcs. This number is called the arc weight (it is also called the arc metric) and may have positive, zero, or negative value. The less the arc weight, the more preferable this arc is, and the more likely it will appear in the selected path.
The path cost is the sum of weights of all arcs that constitute the path. For example, the cost of path XCY is 30. Suppose two paths — P1 and P2 — exist between the same pair of vertices, and path P1 has a lesser cost than path P2. We say that path P1 is shorter than path P2.

There are several problems that can be solved by path computation.

- Single-source shortest path problem: finding the shortest paths from a given vertex to all other vertices.
- Single-destination shortest path problem: finding the shortest paths to a given vertex from all other vertices.
- Single-pair shortest path problem: finding the shortest path between two given vertices.
- All-pairs shortest path problem: finding the shortest paths between every pair of vertices.

If arc CF had weight -45 the shortest path between nodes X and Y simply would not exist because the more times a path crosses nodes B, C, F, D, and B, the less it costs.

Such loops as B-C-F-D-B are called negative loops. Negative weights are an important tool and are intensively used in arc- and vertex-diverse path computation algorithms.

There are two variables (d[v] and π[v]) that a single source algorithm associates with every vertex v on the graph. d[v] is called path estimate, and upon termination of a single source algorithm it contains the cost of the shortest path between source vertex s and vertex v. π[v] is called predecessor, and it contains the identity of a vertex that is the penultimate vertex on the path from s to v. Thus, at the end of a single source algorithm it
is possible to build shortest paths between source vertex s and any vertex v that is reachable from vertex s by defining the predecessor of vertex v (π[v]), the predecessor of the predecessor of vertex v (π[π[v]]), and so forth. In fact, the set of predecessors of all vertices that belong to the graph G (V,E) identify an important sub graph G0(V0,E0), where V0 is the subset of vertices reachable from the source vertex s, and E0 is the subset of edges that interconnect vertex s with every reachable vertex v via the shortest path between vertex s and vertex v. The graph G0 (V0, E0) is called the Shortest Path Tree (SPT).

**Common Procedures**

There are two procedures that are performed by all single source algorithms:

Initialization and arc/edge relaxation. During initialization d[v] and π[v] are set to the following values:

\[
D[v] = \infty \text{ for every vertex } v \text{ except } S \quad (2.6)
\]

\[
D[S] = 0 \quad (2.7)
\]

\[
\Pi[v] = \text{NIL for every vertex } v \quad (2.8)
\]

This value must be larger than the cost of the longest path between vertex s and any reachable vertex v. In practice, it is usually set to the sum of weights of all edges/arcs of the graph.

To describe the arc relaxation procedure let us consider an arc a with weight w (a) that interconnects a pair of adjacent vertices u and v. The process of relaxing arc a involves testing whether the selected shortest path from vertex s to vertex v can be improved by going through vertex u, that is, by traversing arc a. If this condition holds, it is said that
arc $a$ is relaxed. This is the only procedure that may change the values stored in $d[v]$ and $\pi[v]$ and can be summarized as follows.

$$\text{If } d[u] + w(a) < d[v] \quad (2.9)$$

then $d[v] = d[u] + w(a); \pi[v] = u \quad (2.10)$

Notations and assumptions used:

We consider a directed network $G = (N, A)$ with an arc length or arc cost $c_{ij}$ associated with every arc $(i, j) \in A$ and a source node $S$. The shortest path problem is to determine the shortest length of sending one unit of flow from source node to node $i$. The length of the directed path is the sum of lengths of arcs in the path.

The linear programming formulation of the shortest path problem would be:

Minimize $\sum_{(i,j) \in A} c_{ij} x_{ij} \quad (2.11)$

Subject to $\sum_{(j:(j,i) \in A)} x_{ij} - \sum_{(j:(i,j) \in A)} x_{ji} = n-1$ for $i = s$

$\{ -1 \text{ for } i \in N - \{s\} \}$

and $x_{ij} \geq 0$ for all $(i, j) \in A$.  

With the following assumptions:

1. All arcs are integers
2. The network contains a directed path from node $s$ to every other node in the network.
3. The network does not contain a negative cycle (a directed cycle of negative length)
4. The network is directed
The algorithms considered in this research are as follows:

3.2 Dijkstra’s Algorithm:

- Dijkstra’s algorithm finds the shortest paths from source node S to all other nodes in the network with nonnegative arc lengths.
- Dijkstra’s algorithm maintains a distance label $d(i)$ with each node $i$, which is an upper bound on the shortest path length to node $i$.
- At any intermediate step, the algorithm divides the nodes into two groups: those that are permanently labeled and those that are temporarily labeled. The distance label to any permanent node represents the shortest distance from the source node to that node. The distance label to a temporary node is an upper bound on the shortest path distance to that node.
- The algorithm fans out from the source node S by initially setting the source node as a permanent label of zero and each other temporary node $j$ equal to $\infty$.
- At each iteration, the label of node $I$ is its shortest distance from the source node along a path whose internal nodes are permanently labeled. The algorithm selects a node $I$ with the minimum temporary label marks it as permanent and reaches out from that node.
- The algorithm terminates when it has designated all nodes as permanent. The correctness being labeling the minimum temporary label as permanent.
- Dijkstra’s algorithm maintains the invariant property that every tree arc satisfies the condition

$$d(j) = d(i) + c_{ij} \quad (2.13)$$
with respect to current distance labels.

The formal algorithmic description of Dijkstra's algorithm is:

Algorithm Dijkstra

Begin

\[ S: = 0; S': = N \]
\[ d(i) : = \infty \text{ for each node } i \in N \]
\[ d(s): = 0 \text{ and pred}(s) := 0 \]
while \(|S| < n\) do

begin

let \( i \in S' \) be the node for which \( d(i) = \min \{d(j) : j \in S'\} \)

\[ S: = S \cup \{i\} \]
\[ S': = S' - \{i\} \]

for each \((i, j) \in A(i)\) do

If \( d(j) > d(i) + c_{ij} \) then \( d(j): = d(i) + c_{ij} \text{ and \ pred}(j): = i; \)

end;

end;

Advantages:

- Once it has been carried out you can find the least weight path to all permanently labeled nodes.
- You don't need a new diagram for each pass.
- Dijkstra's algorithm has an order of \( n^2 \) so it is efficient enough to use for relatively large problems.
Disadvantages:

- Does not work with negative weight arcs
- Acyclic graphs
- Most often cannot obtain the right shortest path

Example:

![Figure 3.2: Illustrating Dijkstra’s algorithm through a 6 node network. The figure above shows the initial state where all nodes are at infinity distance from the source node 1.](image-url)
Figure 3.3: The nodes in the first layer are updated to weights 4 and 6.

Figure 3.4: Finally all the nodes are updated and the shortest path to the destination node 6 is found to be 12 units.
3.3 Bellman-Ford Algorithm

- The Bellman-Ford algorithm solves the shortest path problem for a given graph G (V, A) by producing shortest paths from any given vertex S to all other vertices reachable from vertex S.
- It allows negative metrics for some arcs provided that they do not produce negative loops reachable from vertex S. This is a valuable quality to detect the negative loops rather than assuming that there are no such loops on the graph.
- First it initializes all nodes, where the source node is stored as zero and the other nodes as $\infty$. Then walking through all nodes performing arc relaxation for every arc. This process is repeated N-1 times where N is the number of nodes.
- It tests for negative loops by walking through all arcs and verifying if there is still an arc, for which the arc relaxation would re-label or modify the distance estimate of the arc terminating vertex. The presence of such an arc indicates the presence of a negative loop.
- The algorithm notifies the presence of negative loops. The cost of the shortest path are stored in $d(V)$, and an actual path from vertex S to destination vertex V is built.

The formal algorithm description is as follows:

Bellman-Ford (G,s)

begin

do for every vertex $v \in V$

\[ D[v] = \infty; \pi[v] = NIL \]

\[ D[s] = 0 \]

do for every $i \in (0, 1, ..., V-1)$


do for every arc $a (u, v) \in A$

\[ d[v] > d[u] + w(a) \text{ then } d[v] = d[u] + w(a); \pi[v] = u \]

do for every arc $a (u, v) \in A$

\[ d[v] > d[u] + w(A) \text{ then return FALSE /* negative loop present*/} \]

return TRUE /*no negative loops*/

end

Advantages:

1. Detects the presence of negative loops
2. Simple to implement, no complex data structures like the min-priority queue

Disadvantages:

1. Runs slower than Dijkstra’s algorithm.
2. It does not scale well.
3. Changes in network topology are not reflected quickly since updates are spread node-by-node.
Example:

Figure 3.5: Illustration for Bellman-Ford example. It calculates the shortest path to all the nodes from the source node 1

Figure 3.6: Initializing by making the source node at distance 0 and other nodes at infinity
Figure 3.7: Relaxing each edge until finally we obtain the shortest path to each node using the steps mentioned in the Bellman Ford algorithm. The shortest path from node 1 to 5 is of weight 2.

3.4 K-shortest Path Algorithm

- During the process of path selection for a particular service there may be various constraints and preferences that a user may demand for a particular service: available bandwidth, link protection quality, etc.

- One shortest path from source to destination may not be satisfying these demands. One way out is to compute several shortest paths between the source and destination and choose the one that meets all the constraints. Thus, the K-shortest path problem determines k shortest paths in the increasing cost order.
• The initial step is the same as other shortest path algorithm to select the path between source and destination.

• If $k > 1$, compute the next shortest path between source and destination nodes by removing from the graph one arc $a \in A$ and running the shortest path again on the modified graph. Repeat this step $k$ times.

The formal algorithm is as follows:

**Variable initialization**

$P = 0$ /*list of previously returned paths*/

$Q = 0$ /*$Q$ is min-priority queue of path candidates*/

$p_{\text{prev}} = 0$ /* $p_{\text{prev}}$ is a path returned in the previous iteration */

$\text{branch} = 0$ /* branch is current branching point */

$\text{stem} = 0$ /* stem is a path segment that starts at source vertex and common to all previously returned paths */

$\textbf{KSP\_NEXT\_SHORTEST\_PATH}(G, s, z)$ /* $G$ is graph $G(V, A)$ without negative loops, $s$ — source vertex, $z$ — destination vertex */

if $p_{\text{prev}} = 0$

then $p_{\text{prev}} = \text{BFS}(G, s, z)$

return $p_{\text{prev}}$

do for every $p \in P$
do for $I = 0; i<number\_of\_arcs(p); I = I + 1$  
/*number\_of\_arcs(p) is a total number of arcs that constitute path $p$ */

if $v\_orig(a(p, i)) = = \text{branch}$

$A = A - a(p, i)$  
/* $v\_orig(a)$ is a vertex that originates arc $a$, $a(p, i)$ is the $i$-th arc of path $p$ starting from the path head */

do for $i = 0; i<number\_of\_arcs(p\_prev); i = i + 1$

$A = A - a(p\_prev, i)$

$P = \text{stem} + \text{BFS}(G, b, z)$

if $p! \in P \&\& p! \in Q$
then INSERT$(Q, p); A = A + a(p\_prev, i)$

if $Q = = 0$ then return 0

$p\_new = \text{EXTRAC\_MIN}(Q)$  
/* $p\_new$ is a path that will be returned in current iteration */

do for every $p \in P$

    do for $I = 0; i<number\_of\_arcs(p); I = I + 1$

        if $v\_orig(a(p, i)) = = \text{branch}$ then $A = A + a(p, i)$

    if $|P| = = 0$ then stem = $p\_prev$

    do for $I = 0; i<number\_of\_arcs(p\_new); i = i + 1$

        if $a(p\_new, i)! = a(\text{stem}, i)$ then break

    $\text{branch} = v\_orig(a(\text{stem}, i))$  
/* $\text{branch}$ is set to be the originating vertex of the first diverting arc */
do while i<number_of_arcs(stem)

stem = stem _ a(stem, i)

P = P + p_prev

p_prev = p_new return p_new

Advantages:

1. Can have more than one shortest paths
2. Can meet the constraints of user more efficiently than other algorithms

Disadvantages:

1. Can produce redundant routes if shortest path not needed
2. Increases complexity

Example:

![Diagram](image_url)

Figure 3.8: Illustration of K-shortest path, with k= 6
Chapter 4

Simulation Scenario and Results

To evaluate the performance of the Switching algorithms along with DCA scheduling, M/G/1 queuing and capacity request schemes we have used MATLAB simulation model for a user terminal transmitting multimedia information. The model consists of a number of User terminals, Master Control station, scheduler, queues, server, and switching network.

There are some assumptions made for optimal evaluation of our QoS parameters. The assumptions are as follows:

- We make an assumption that the video and voice transmit efficiently without being significantly affected by delay by following the DiffServ method to maintain the QoS parameters for delay and jitter.
- We assume that the satellite does not delay the traffic received from the UTs. We also assume that the destination gateway immediately forwards traffic received from the
satellite to the terrestrial network. With these assumptions, it is possible to calculate the upstream end-to-end delay.

For QoS parameters like BER, packet loss and jitter:

• In our discussion of the scheduler we made no mention of DiffServ and QoS.

• This is because the UT, not the scheduler, is responsible for implementing QoS mechanisms.

• Giving responsibility for QoS to the UT is a logical approach because the UT has the most up-to-date information about current traffic conditions and is thus best equipped to make QoS related decisions. This distributed functionality approach simplifies the scheduler’s architecture, which is desirable, especially if the scheduler is located on board the satellite where real estate and computing resources are limited. In addition, this makes the scheduler design independent of the type of service offered (e.g. IP, ATM)

For data, delay is an important factor because it is given least priority in multimedia services. Hence, to evaluate the delay for data and improve it using efficient algorithm various computing resources are used for executing an algorithm. Since time is a dominant computing resource we use the time taken by an algorithm as our metric for measuring the most efficient algorithm.

There are two kinds of simulations performed for OBS algorithms:

I) First, only the transmission of data for ATM network from User terminal to receiver sub-unit to switching network to transmission sub unit is evaluated for delay.
II) Next the effect of other delay causing parameters like Queueing delay along with switching delay is evaluated.

**System parameters:**

- The capacity per uplink (or downlink) beam is 512 traffic slots in 24 ms-frame. Each slot can accommodate one traffic packet of 48 bytes of information.
- The uplink uses MF-TDMA with eight carriers and each carrier can accommodate a peak information rate of 1.024 Mb/s, i.e., 8 X 64 frequency-time slots per MF-TDMA frame.
- The downlink beam uses one carrier equivalent to an information rate of 8.192 Mb/s with 512 slots per TDM frame. Modulation is QPSK. Since we consider only data for delay we use 0.1-10 Mbps bit rate for transmission.

**Table 4.1: Medium Data rate chart for data, voice and video [8] used for simulation**

<table>
<thead>
<tr>
<th>Medium Speed</th>
<th>0.1-1 Mbps</th>
<th>Hi-Fi sound</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1-1</td>
<td>Videoconference</td>
<td></td>
</tr>
<tr>
<td>0.1-10</td>
<td>Data/images</td>
<td></td>
</tr>
</tbody>
</table>

The MF-TDMA for the uplink with 8 carriers in different frequencies and 512 time slots as depicted below:
Simulation results:

Five parameters have been considered for evaluating QoS performance for the three algorithms with M/G/1 queue on Batcher-Banyan network. They are as follows:

I] End-to end Delay from algorithms on ATM network without queueing

II] Delay from queueing in single-server M/G/1 queue for different queue lengths

III] End-to-end Delay from algorithms on ATM network with queueing

IV] Burstiness factor for varying traffic load on the network

V] a) Blocking probability for varying holding time in the single-server queue

   b) Blocking probability for varying number of servers

Figure 4.2 gives End-to end delay for Dijkstra’s algorithm for various numbers of nodes ranging from 32 nodes to 256 node network
The delay observed is minimal considering that Dijkstra’s algorithm starting from 0.112 sec for 32 nodes to 0.16 sec for 256 nodes.

Figure 4.3 gives the graph for Bellman-Ford algorithm for end-to-end delay versus number of nodes without any queueing. For 32 nodes the delay is approximately observed to be 0.12 sec whereas for 256 node network the delay is close to 0.6 sec. The change in delay for increasing number of nodes is steep.
Figure 4.4: Time delay versus number of nodes for K-shortest path algorithm

Figure 4.4 shows the delay for K-shortest path algorithm versus number of nodes. For 32 node network the delay observed is very close to 0.12 sec but for 256 nodes the delay is comparatively smaller of approximately 0.2 sec.

Figure 4.5 shows the results for total delay for the three algorithms with number of nodes versus time without queuing delay on Batcher-banyan network:
Figure 4.5: Total delay for Dijkstra’s, Bellman-Ford and K-shortest path without queueing delay

As shown the Dijkstra’s and Bellman-Ford give lowest delay for approximately of 0.020 sec and 0.025 sec respectively for up-to 80 nodes. From nodes to 100 onwards K-shortest path and Dijkstra’s give the least delay.

Figure 4.6 gives the results for time delay versus number of nodes for the three algorithms with queueing delay on Batcher-Banyan network. For delay with entire link along with queueing the delay increases by 0.025 sec approximately, with Dijkstra’s and Bellman-Ford giving 0.070 sec at 32 nodes. After 90 nodes the delay for Bellman Ford increases exponentially up to 256 nodes while Dijkstra’s and K-shortest path are comparatively minimal.
Figure 4.6: Time delay for algorithms with Queueing versus number of nodes

Figure 4.7: Queueing delay versus number of nodes
Burstiness factor is the ratio of total average bit rate to the beam capacity.

Here, average bit is for real time and non real time data are considered from [6] and from our 8 channel beam, the beam capacity is of 8192 Mbps. Figure 4.8 gives the graph of burstiness factor for data, voice and video for low, medium and high traffic load.

![Burstiness factor versus traffic load](image)

**Figure 4.8: Burstiness factor versus traffic load**

Figure 4.9 gives the Blocking probability versus holding time in milliseconds-single server system and Figure 4.10 gives Blocking probability versus number of servers-multiple server system. Results show that for lower holding time the probability that a call is lost is very high and for higher holding time the probability drastically
decreases. Also, for more than one server the probability increases for up-to approximately 15 servers and then improves performance for higher number of servers.

![Blocking probability versus holding time](image)

**Figure 4.9: Blocking probability for Batcher-Banyan ATM network versus call holding time**
Figure 4.10: Blocking probability versus number of servers for the queueing system
Chapter 5

Conclusion and Future work

5.1 Conclusion

In this thesis, the performance of the best algorithm of the Dijkstra, Bellman-Ford and K-shortest path on the most efficient ATM network along with queueing system are simulated and evaluated. The Quality of Service parameters used for comparison evaluation are as follows:

- End-to-end Delay from algorithms on ATM network without queueing
- Delay from queueing in single-server M/G/1 queue for different queue lengths
- End-to-end Delay from algorithms on ATM network with queueing
- Burstiness factor for varying traffic load on the network
- Blocking probability for varying holding time in the single-server queue
- Blocking probability for varying number of servers

From the results obtained in Chapter 4 for various Quality of Service parameters for the different combination of algorithms, queues and ATM networks and the Worst-case analysis of the algorithms from Chapter 3 we can conclude that Bellman-Ford
algorithm performs well in terms of delay for a network of 100 nodes with M/G/1 queue and Batcher-banyan network. As the size of network increases the delay increases drastically. For higher node network the K-shortest path algorithm performs comparatively better. We have implemented single server systems which give good performance in terms of blocking probability for small networks, whereas to achieve better blocking probability performance and burstiness factor for large networks, more number of servers should be implemented. Hence, for On-board switching algorithms along with Differentiated Services the above mentioned techniques give good QoS performance for multimedia application.

5.2 Future work

As Future work, more algorithms can be tested for the Quality of service parameters like delay, jitter, packet loss and burstiness on various ATM networks. The User terminal was assumed to not give any error for transmission for uplink scheduler and good performance for Bit Error rate was considered. Practical performance from the user terminal side can give a wider scope to the QoS parameter evaluation.
References


[7] Queueing analysis of discrete-time buffer systems with compound arrival process and variable service capacity, Bruneel, Bart Vinck and Herwig


Appendix A

MATLAB Codes:

The MATLAB Codes for the algorithms and the queueing is given below. The process of receiver section of satellite is performed and then the processed information is sent through the following steps:

%Simulation to evaluate the delay caused by on-ground and on-board queues
%The main random variables are: A, arrival size; W, waiting line size; Q, queue size; theta, service time requested by customers; ets, waiting time spent in the queue; delta, total time spent in queue = waiting time + service time
%lambda: average arrival rate to the queue

function y=thesis2()

clc; clear all;
Nb=10e6;
frame=24e-3;
lambda=(Nb/frame);
l=poissrnd(lambda);
arrival = poisstat(l);
qsize=0:500:4000;
%mean value of queueing delay

qdelay=qsize/arrival

stem(qsize,qdelay);

xlabel('mean queue size E(theta) ');
ylabel('time delay, t_1');
title('M/G/1: mean value of queueing delay');
grid on;
y=qdelay;
hold on

%average number of customers

meanservice = (53 * 8 )/(1.024*10^-6);
rho= lambda * meanservice;
meancustomers = rho+ ((lambda*lambda*meanservice*meanservice)/(2*(1-rho)))

%Dijkstra’s algorithm

% shortestPath: the list of nodes in the shortestPath from source to destination;
% totalCost: the total cost of the shortestPath;
% farthestNode: the farthest node to reach for each node after performing the routing;
% n: the number of nodes in the network;
% s: source node index;
% d: destination node index;

for t=16:32:256 %
costmatrix=randint(t,t,[1,9]);
for i=1:length(costmatrix)
for j=1:length(costmatrix)
    if i==j
        costmatrix(i,j)=0;
    end
end

n=size(costmatrix,1);
u=[];
v=[];

%considering the source node as node 1

disp('SOURCE NODE')
i=1
linkcost=0;

disp('NEXT NODE AND TOTAL COST TO NEXT NODE')
for i=1
    for j=2:n
        if costmatrix(i,j-1)~=0 && costmatrix(i,j-1)~=inf && costmatrix(i,j)~=0 &&
        costmatrix(i,j)~=inf
            if costmatrix(i,j-1)<costmatrix(i,j)
                link=costmatrix(i,j-1);
                i=j-1
            end
        else link=costmatrix(i,j);
    end
end

else link=costmatrix(i,j);

i=j
end
%the linkcost is given as follows giving the route of the
%network
linkcost=linkcost+link
end
end
disp('TOTAL LINK COST TO DESTINATION')
linkcost=linkcost+costmatrix(i,j)
end
% Bellman- Ford algorithm
for t=16:32:256
cost=randint(t,t,[-9,9]);
for i=1:length(cost)
    for j=1:length(cost)
        if i==j
            cost(i,j)=0;
        end
    end
end
end
n=size(cost,1)
% new=cost;
bell=[];
bellman=[];
disp('The initialization process');
for i=1:n
    for j=1:n
        bell(i,j)=cost(i,j);
        bellman(1,j)=bell(1,j);
    end
end
bell=[bell]
disp('______________________________________________')
disp('RESULTS FOR LEAST LINK COST TO EACH NODE IN NETWORK: ')
bellman=[bellman] ;
bellmannew=[];
%Relaxation process
for j=1:length(bellman)
    if bellman(1,j)~=0 && bellman(1,j)~=inf
        link=bellman(1,j);
        i=j;
        for k=1:length(bell)
            if (bell(i,k)+link)<bellman(1,k)
                bellmannew(1,k)=bell(i,k)+link;
            else bellmannew(1,k)=bellman(1,k);
            end
        end
    end
end
bellman=bellmannew;
end
end
for j=length(bellman):-1:2
    if bellman(1,j)==0 && bellman(1,j)==inf
        link=bellman(1,j);
        i=j;
    for k=1:length(bell)
        if (bell(i,k)+link)<bellman(1,k)
            bellmannew(1,k)=bell(i,k)+link;
        else bellmannew(1,k)=bellman(1,k);
        end
    end
    bellman=bellmannew;
    end
end
disp('');
disp(bellman)
% Kshortest path algorithm
%Find w in (G, A)
w = S(current);
for i = 1: length(P)
if w == P(i)
    indexinpath = i;
end
end
for vertex = indexinpath: length(P) - 1
    tempnetCostMatrix = netCostMatrix;
    % Remove vertices in P before vertex and there incident edges
    for i = 1: vertex - 1
        v = P(i);
        tempnetCostMatrix(v,:) = inf;
        tempnetCostMatrix(:,v) = inf;
    end
for t = 16:32:256
    cost = randint(t,t,[1,9]);
    for i = 1:length(cost)
        for j = 1:length(cost)
            if i == j
                cost(i,j) = 0;
            end
        end
    end
end
n = size(cost,1);
disp('SOURCE NODE');
i=1
k=8;
h=n-1;
l=n;

for t=1:k

linkcost=[];
linkcost=0;

for i=1

for j=2:n

if cost(i,j-1)==0 && cost(i,j-1)==inf && cost(i,j)==0 && cost(i,j)==inf

if cost(i,j-1)<cost(i,j)

link=cost(i,j-1);
i=j-1;
else link=cost(i,j);
i=j;

end

%the linkcost is given as follows giving the route of the

%network

linkcost= linkcost+link;

u= [u i];

v= [v linkcost];

end

end
w=v;

linkcost= linkcost+cost(i,j);

w= [w linkcost];
end

disp('NEXT NODE');
disp(u);
disp('TOTAL LINK COST TO DESTINATION');
disp(w);

cost (h,l)=inf;

h=h-1;
l=l-1;
end