Flow Control in a Hybrid Satellite-Terrestrial Network: Analysis and Algorithm Design

by Gabriel L. Olariu
Advisor: John S. Baras

CSHCN M.S. 97-3
(ISR M.S. 97-7)

Abstract

Title of Thesis: Flow Control in a Hybrid Satellite-Terrestrial Network: Analysis and Algorithm Design

Name of degree candidate: Gabriel L. Olariu

Degree and year: Master of Science, 1997

Thesis directed by: Professor John Baras
Institute for Systems Research

The Internet is making its way into our day-to-day life. Start-up companies and industry leaders in communication networks are competing in the market place for offering new and high performance solutions to their increasing number of customers. Individuals that, due to their job characteristics have access to the Internet via their workplace desktop have quite a different experience with the cyberspace versus those that are accessing networking services from their homes.

Typically, a common Internet "surfer" that connects from home, will be frustrated by the speed at which his/her service is working. This is due to the limits imposed by the classical dial-up connection, via an Internet Service Provider.

To-date, various alternatives to this situation have been reported: access via cable television networks and access via satellite links, not to mention ISDN line solutions. The first alternative is more likely to be implemented in crowded areas where the cable infrastructure may be already in place. However there are studies that show that a large investment is necessary to cover vast areas with this kind of infrastructure. Major costs are primarily due to the effort of laying out cable.

Satellite Internet on the other hand is not restricted to work in a given area. Satellites "see" virtually everywhere. However there are trade-offs concerning the available bandwidth and its allocation. Also, satellite Internet is primarily dedicated to persons that receive more data than they generate for output.

In this thesis we pursue a study of the flow-control in a satellite-terrestrial network. An analysis study is first performed for the DirecPC Hybrid Internet service. Then different bandwidth allocation strategies are compared, with the performance criterion being the delay in interactive sessions. The best service obviously minimizes the delay.

We present theoretical and analytical background on the interactive traffic modeling problem. Fractal-type traffic is fed into the network models and different performance metrics are measured and discussed.

We end by concluding that in the event that a satellite-terrestrial network would exclusively be used for interactive users, the optimal policy is to first serve the connection that suffers the largest instantaneous delay.

Flow Control in a Hybrid Satellite-Terrestrial Network: Analysis and Algorithm Design

by

Gabriel L. Olariu

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland at College Park in partial fulfillment
of the requirements for the degree of
Master of Science
1997

Acknowledgments

I am greatly indebted to Professor John Baras, my advisor, for his belief and support shown to me in these two years of graduate study in the Institute for Systems Research at the University of Maryland at College Park. Not only he has revealed to me the importance of communication networks research – and within this the major challenges that hybrid architectures pose for flow control strategies – but he gave me the opportunity to work with one of the leader companies in satellite-terrestrial networks, namely Hughes Network Systems.

It was during my technical internship with the DirecPC Department of Hughes Network Systems when I realized the importance of having an effective and efficient implementation of flow control algorithms. In this concern, I want to thank Assistant Vice-President Douglas Dillon for all his help during the summer of 1996.

I equally wish to thank Professor Nick Roussopoulos and Professor Scott Corson for their support and valuable comments that led to generating the final version of this thesis.

I also want to mention that most of the editing and design phase work was done in the Systems Engineering Laboratory, where I enjoyed the high technology endowment and the support of a wonderful group of colleagues–Jagdeep Rao, Isatou Secka, Mingyan Liu, Narin Suphasindhu, Manish Karir.

This work was supported by the NASA Center for Satellite and Hybrid Communication Networks, under contract NAGW-27777, Hughes Network Systems and the Sate of Maryland under a cooperative Industry-University contract from the Maryland Industrial Partnerships Program (MIPS).

Most importantly, thanks to my wife who undertook the responsibility of caring for our young son, so that I would be able to commit myself to working on this thesis.

Table of Contents

List of Tables

List of Figures

Flow Control in a
Hybrid Satellite-Terrestrial Network:
Analysis and Algorithm Design

Gabriel L. Olariu

August 3, 1997

Chapter 1

Introduction

Hybrid Satellite-Terrestrial Networks present challenging case studies for flow control algorithms. Satellite channel particularities can easily become constraints on the type of applications that such a network can support.

The widespread use of the TCP/IP protocol suite within the Internet community enforces its use for hybrid networks as well. Researchers have already addressed the study of TCP/IP for this specific situation, [17], [16]. These are protocol level studies, for networks very similar to the one that is the subject of our work. The protocol modifications proposed are very interesting and most likely to be transformed into add-ons for the existing protocols. We do not address this part of the flow control. It is worth mentioning however that the simulator we have built for the analysis part, is implementing part of the TCP/IP congestion control algorithm. Another important control issue is the allocation of resources to users. In the design part of the thesis we will perform a comparative study of three allocation algorithms. The control literature in general, and in particular that part which addresses communication networks control contains many studies on this problem. For one of the allocation algorithms, we used the "fair allocation" principle introduced by Raj Jain [12], [13].

Due to the "deterministic" delay introduced by the satellite channel, it is interesting to investigate if interactive applications can be ported to these networks without altering the service requirements. In this thesis, we will consider the case where the only users of the network are the interactive ones. This allows us to treat them all as equal; our assumption makes the analysis and design easier. It has been shown that interactive, Internet-type applications generate self-similar traffic, [8], [18]. For this reason, we departed from the method of using classical Markovian traffic models. We dedicate part of this work to presenting theoretical and implementation issues of self-similar traffic. This is however a research area in its own right and it is beyond the goals of this thesis to go into deeper details on this subject. We refer to [24], [25], [26] for detailed analysis and results in this area.

In the following section we introduce some hybrid networks terminology, and we end this introductory chapter with the thesis outline.

1.1 Hybrid Satellite-Terrestrial Networks

Hybrid Satellite-Terrestrial Networks – HSTN – are relatively new in the rapidly evolving telecommunications arena. Flow control in HSTN is the subject of this thesis. This is a research area of large interest due to several facts. First, as mentioned above, HSTN are becoming increasingly popular. Second, while introducing new network architectures, classic transport protocols need to be adapted to the new environment. We can mention, as an example, the efforts done towards modifying TCP congestion control for having a proper behavior on "long, fat pipes", [14], as a satellite link is.

Although evident, we will briefly describe a typical HSTN. This is with the purpose of giving a clear picture of the typical network architecture addressed in this thesis. Let us consider a source-destination pair from the set of communicating nodes. The particular case considered here is characterized by the asymmetry in link capacities in a sense to be defined next. Even if all HSTN need to have, by definition, terrestrial and satellite links as well, by choice of design, flows may propagate in a uni-directional or bi-directional sense. The attribute uni- or bi-directional refers to the wireless links. If the satellite link is unidirectional and the terrestrial one is slow then we have an asymmetric HSTN. DirecPC^{TM}, a commercial product of Hughes Network Systems is a typical example. Most of this thesis is inspired from the study of the DirecPC^{TM} flow control algorithm. In the bi-directional case, the satellite link supports a full duplex communication channel. This requires more specialized and expensive reception equipment. One reason for investigating the asymmetric case is its immediate availability and rapid market expansion. Asymmetric Internet access offers an alternative to the classical Internet access via bi-directional dial-up lines. The structure of the network will be given in the analysis chapter.

1.2 Related Work

The topics addressed in this thesis are mainly identified as follows: modeling of self-similar traffic, performance analysis of TCP congestion control for asymmetric communication networks and bandwidth allocation algorithm design. In the next paragraphs we will conduct a brief survey of the research done in each of these areas.

Markovian methods for modeling packet arrival processes have lost popularity within the networking community due to the remarkable results on the self-similar nature of LAN [18], WAN [22], VBR video traffic [4], [10], and WWW traffic [8]. As a result, traffic self-similarity became a "hot" research area. Several papers addressed modeling and analysis aspects of the self-similar traffic [11], [20], [21], [23], [24], [25], [26], [27], [28]. In this thesis, we introduce basic notions of "fractal-type" traffic in Appendix A, and design a self-similar traffic generator in section 4.2.3.

Analysis of asymmetric communication networks, performed at control protocol level, is the subject of [16], [17]. These papers address the TCP congestion avoidance algorithm. In [19], [29], the case of ATM queues is investigated in the context of self-similar input traffic. In Appendix C we give an example of network dimensioning, and, at the same time, we verify for accuracy our traffic generator algorithm by comparison with the results in [19]. Our results prove to be identical to those in the paper mentioned above; this is the accuracy test for the traffic model.

The topic of bandwidth allocation algorithms has been the subject of extensive research. One of the criteria used in this thesis for performance evaluation is the fair bandwidth allocation, introduced in [12], [13]. In [6], the fair bandwidth allocation is treated as a particular case in a more general context of rate allocation algorithms. In this thesis, we show that the fair bandwidth allocation strategy provides smaller average queueing delay than the equal bandwidth allocation strategy. However, our experiments reveal the fact that, the policy that first allocates bandwidth to the connection with largest queueing delay gives better results. This outcome must be placed in the context of interactive users, modeled as **ON-OFF** processes, where certain restrictions on the distributions of the **ON** and respectively **OFF** periods apply, as stated in section 3.1.

1.3 Outline

Chapter 2 is the Analysis of the DirecPC Hybrid Internet prioritization and flow control algorithm. It is a simulation effort in its entirety. Here we present the asymmetric connection via a network model. Then we map this model to a simulation one. The simulation approach is described together with the experiments performed. Data pertaining to the analysis is graphically depicted in Appendix B.

Chapter 3 is the design part of the thesis. We introduce the traffic and service facility models, and the three bandwidth allocation policies that will be compared and ranked.

Because experiments are performed using our own simulators, we dedicate chapter 4 to the design of experiments.

Finally we present the thesis contributions, conclusions and future work plans in chapter 5.

Experimental results are given in the appendices.

Chapter 2

Analysis Considerations

2.1 Introduction

Hughes Network Systems operates an HSTN that offers Hybrid Internet Service. Part of the traffic carried by in this system is generated by interactive applications.

In this chapter we address the analysis of the flow control algorithm for this system. The analysis is a simulation effort in its entirety. The interactive users are assumed to use the TCP congestion avoidance mechanism. Hybrid Internet adds more to this mechanism in order to cope with the satellite link delays. In the following section, we give architectural details for the Hybrid Internet Access and the simulation that has been built.

2.2 Architecture of the Hybrid Internet Access

Fig. 2.1 shows a typical Hybrid Internet connection.

The communication objects that participate in data flow are:

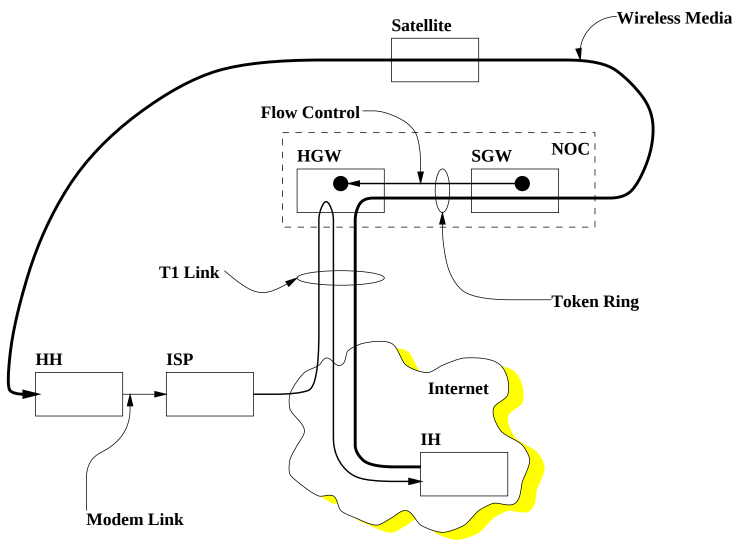

Figure 2.1: A Typical Hybrid Internet Connection

IS: Internet Server;

NOC: Network Operations Center;

HGW: Hybrid Gateway;

SGW: Satellite Gateway;

HH: Hybrid Host;

ISP: Internet Service Provider.

DirecPC Hybrid Internet uses terrestrial and satellite links to deliver information to HH's. Reliable data delivery is based on TCP flow control. It is known that TCP behaves poorly on satellite links because of the well known large bandwidth-delay product and the transmission quality of satellite links; see for example [14]. To alleviate the problem, TCP spoofing is introduced [2], [3]. Spoofing means that the HGW acknowledges data to the IS on HH's behalf. This mechanism is applied under the assumption that the satellite link will correctly deliver data to the destination. An error on the satellite link will be noticed by the HH after about .5 s and retransmissions will begin from the HGW.

We consider that a connection was initiated by the HH, and now the IS sends the requested data. Data flows out from the IS's LAN and arrives via a T1 line in the NOC. The first NOC object that receives data is the HGW. It plays the router role for the NOC LAN. The HGW acknowledges data on behalf of the HH and this is called TCP "spoofing". The HGW delivers flow control information from the NOC to the IS using acknowledgment messages. The flow control data is calculated for each on-going connection, using values of various state variables. It is the role of the HGW to perform these calculations. All Hybrid Internet packets received in the HGW are forwarded to the SGW. A packet prioritization scheme runs in the HGW and sorts traffic into two classes: high priority and low priority. The SGW is used as point of departure for the packets to the satellite link. The SGW jobs are a mixture of Internet and exogenous traffic. The latter one is mainly package delivery and data feed traffic. Package delivery is a mechanism by which an organization broadcasts messages to its subscribers (clients). Data feed is basically multimedia traffic. This exogenous traffic produces fluctuations in the bandwidth allocated to Hybrid Internet because it is treated as high priority and receives service as long as the queues dedicated to it are not empty. It is important to mention that even if all streams pass through the SGW, they are internally routed to different queues. Thus, the SGW maintains four queues at any moment in time: two for package delivery and data feed and two for the two priority levels of the Hybrid Internet. Packets leaving the SGW, reach the destination HH after traveling in satellite channel frames. All data is acknowledged to the HGW. The HH sends acknowledgments via the dial-up to the ISP. After that, they are sent via Internet backbone to the HGW. Upon receiving the acknowledgments, the HGW drops the corresponding packet copies.

2.3 Analysis Model

The analysis model reflects the most important characteristics of a Hybrid Internet connection. An important issue is the level of detail that one must use in modeling. This is dependent on the analysis goals. Fig.2.2 depicts the architecture used in our simulations.

The following components can be identified:

1 Data connection: IS sends data to the corresponding HH;

2 Acknowledgments: circulate from HGW to the IS;

3 Acknowledgments: circulate from HH to the HGW.

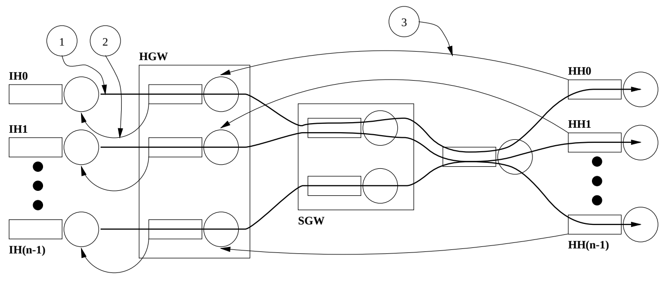

Figure 2.2: Analysis Model

The SGW is shown having two queues. One is **high priority** and the other is **low priority**. The HGW assigns to incoming packets one of these two priority levels according to the following policy:

if the number of un-acknowledged bytes for a connection is less than a configurable, but fixed, threshold value, then these packets are high priority.

This "encourages" connections to operate with small windows. This ultimately may lead to an increase in the number of users. It is well known that small TCP windows are generally avoided in the context of satellite networks, due to large delays on the wireless link. These delays can generate unnecessary retransmissions, followed by all sorts of bad consequences – large delays, congestion. However in this architecture, small windows do not pose problems because the HGW is acting as proxy for the HHs. A connection operating with large windows has an equal number of un-acknowledged bytes. The flow control strategy of the HGW will immediately react to these large windows by reducing the advertisements to the IS. It is thus probable that a connection will have both high and low priority packets during its existence.

Chapter 3

Design Considerations

3.1 Traffic Model

As discussed already, our concern is in the modeling of the interactive applications. The small Web-pages transfer and Telnet traffic are the most common interactive applications. It has been shown in Crovella et. al.[8], Leland et. al.[18], that this type of traffic is suitably modeled by self-similar processes.

We consider a set of sources (IS's) that send data to the destination HH's via the NOC. For our work it is important to find the number of sources that are allowed to send data without producing overflow in the NOC. The NOC model will be discussed in the next section, because its appropriate description depends on the type of traffic that is fed into it.

The IS's are indexed $IS^{(i)},\ i \in 1, \dots, M$, where M is the number of sources to be determined. Sources are independent and operate in an ON-OFF fashion. At a given time instant, a source is either busy or idle. A sequence **(busy,idle)** forms an **operation cycle**. We will refer to figure 3.1 when describing the traffic process.

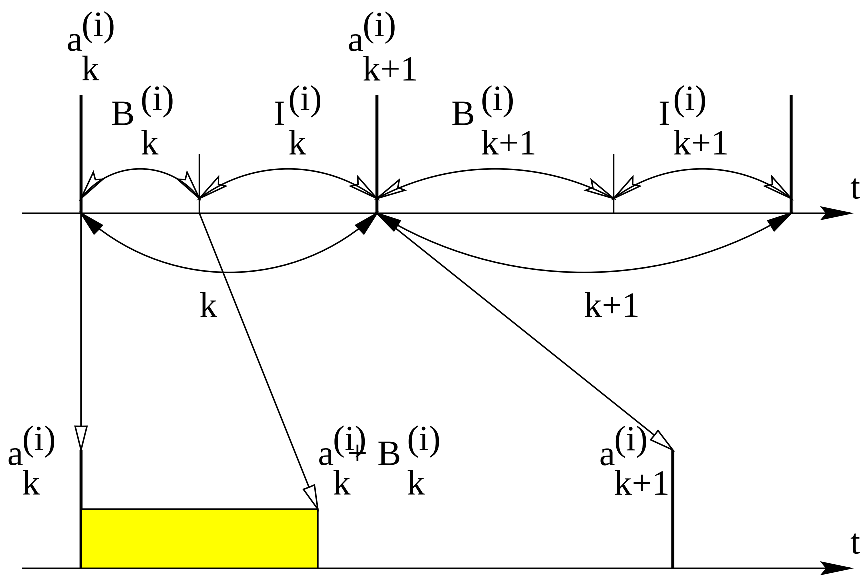

Figure 3.1: The Source (IS) Operation Cycle

3.1.1 Traffic Model for the Individual Source

The interactive source generates traffic that has "heavy tail". Appendix A gives the definition for this type of distributions and some facts about the Pareto distribution.

Let us consider the i^{th} IS. We describe the busy and idle periods for this typical source.

The Busy Period: During this period the IS sends data at a constant rate μ_{IS}.

Definition 1 *A busy period is a random variable* $B^{(i)} = \{B_k^{(i)} | k \in Z\}$, $i \in \{1, \ldots, M\}$ *where* $B_k^{(i)}$ *are iid and Pareto distributed:*

$$P[B^{(i)} \geq t] \sim t^{-\alpha}, \ as\ \alpha \rightarrow \infty, \ and\ 1 < \alpha < 2. \tag{3.1}$$

We will use the generic name B to describe the random variable busy period. B has finite mean ,

$$\mu_B = E[B] < \infty, \tag{3.2}$$

and infinite variance $Var(B)$. Details on Pareto distribution and statistics are given in appendix A.2.

The Idle Period: This is the time during which the source performs one or more of the following:

1. waits for the client to ACK data;
2. calculates control variables.

Definition 2 *An idle period is a random variable* $I^{(i)} = \{I_k^{(i)} | k \in Z\}$, $i \in \{1, \dots, M\}$ *where* $B_k^{(i)}$ *are iid distributed, with a heavy tail distribution.*

We will use the generic name I to describe the random variable **Idle** period.

$$\mu_I = E[I] < \infty. \tag{3.3}$$

It is important to anticipate that the idle period must be "longer" than the busy period.

The Operation Cycle: Fig. 3.1 shows the operation cycle.

Definition 3 *An operation cycle is the random variable,*

$$\begin{aligned} &O^{(i)} = \{O_k^{(i)} | k \in Z\}, i \in \{1, \dots, M\}, \\ &O_k^{(i)} = B_k^{(i)} + I_k^{(i)}. \end{aligned} \tag{3.4}$$

The source arrival epochs are denoted by $a_k^{(i)}$. Then the ordered sequence $\{a_k^{(i)} | i \in \{1, \ldots, M\},\ k \in Z\}$ forms a stationary point process. Stationarity is implied by the same property of the busy and idle random variables (the latter was assumed). The process $a^{(i)}$ is a stationary renewal process.

In the following, let $\lambda_k^{(i)}$ denote the packet generation rate at source i and time k. The first two moments for this random variable are:

$$\begin{aligned} E[\lambda_k^{(i)}] &= \mu_{IS} \frac{\mu_B}{\mu_B + \mu_I}, \\ E[(\lambda_k^{(i)})^2] &= \mu_{IS}^2 \frac{\mu_B}{\mu_B + \mu_I}. \end{aligned} \tag{3.5}$$

where we assume that the source generates packets with a constant rate μ_{IS} . Moreover, all sources exhibit this behavior. That is, they have the same constant rate:

$$\lambda_k^{(i)} = \begin{cases} \mu_{IS} & \text{IS is in busy state,} \\ 0 & \text{IS is in idle state.} \end{cases} \tag{3.6}$$

and the arrival processes are stationary.

Then (3.5) follows from:

$$\begin{aligned} E[\lambda_k^{(i)}] &= E[\lambda_k^{(i)} | \text{Source i is Busy}] P[\text{Source i is Busy}] \\ &+ E[\lambda_k^{(i)} | \text{Source i is Idle}] P[\text{Source i is Idle}] \\ &= \mu_{IS} \frac{\mu_B}{\mu_B + \mu_I} \\ &= \mu_{IS} p, \text{where } p = \frac{\mu_B}{\mu_B + \mu_I}. \end{aligned}$$

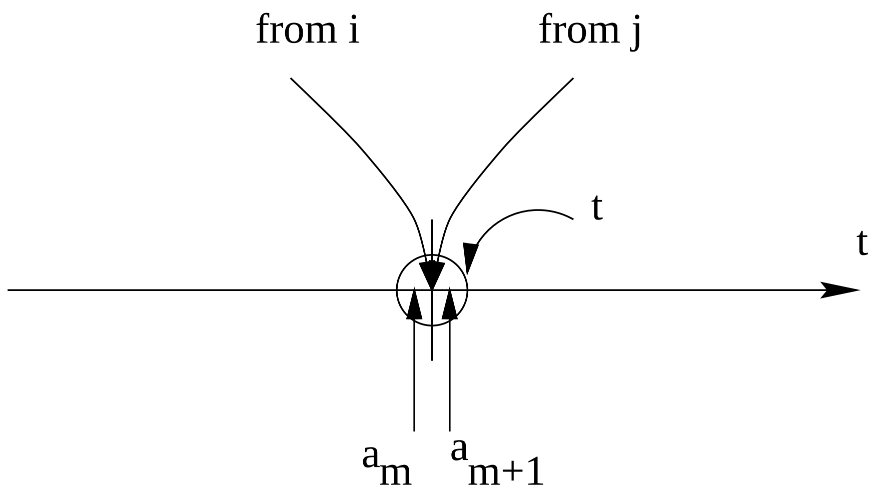

Figure 3.2: Arrival epochs from two sources begin at the same time instant t

3.1.2 The Aggregate Process

The M sources send data to the NOC. Therefore we need to model the superposition of the traffic from individual sources.

For this, consider that time is discrete, and that events take place at $k \in Z$. The cumulative arrival process considers the individual arrival epochs together. Each arrival instant is generated by a source which by convention, has an index in $\{1, \ldots, M\}$. If it happens that two or more sources arrive at the same time, then in the aggregated model arrivals are ordered according to the source index, as shown in fig. 3.2. There, two sources $i < j$ arrive at the same time instant t.

The aggregate arrival traffic is the integer valued random point process $a(M) = \{a_k(M) | k \in Z\}$. However, the renewal property is lost by superposition.

Each arrival epoch has a mark attached to it, having the meaning of the duration of the busy period (Pareto distributed). The following marked point process describes the aggregated traffic:

$$(a(M), B(M)) = \{(a_k(M), B_k(M)) | k \in Z\} \tag{3.7}$$

We are interested in the intensity of the aggregated traffic. An important theoretical result is the one obtained by Likhanov et. al. [19]. It is described here in a rather informal way.

Denote by $\xi_k(M)$ the counting process of the number of busy periods that arrive at a generic queue at time k. Then, if the idle period is "longer" than the busy period, the distribution of $\xi_k(M)$ tends to be Poissonian as the number of sources goes to infinity. Also, if we take the limit as $M \to \infty$ in (3.7), and denote by (a_s, B_s) the limiting process, then the process B_s is independent from a_s and ξ_k, $\forall s \in Z$.

The mathematical formulation of this result is given below:

Lemma 1 *Let $K \subset Z$, $n \in N$, and $x_i \in N \cup \{0\}$, $i \in 1, \dots, n$.*

- *Then, if $M \to \infty$ such that:*
  1. $\lambda = \frac{M}{E[B]+E[I]} = const.$;
  2. $E[B] = const. \Leftrightarrow P[B \leq \tau] = const.\ for\ \tau < \infty$;
  3. $E[I] \to \infty \Leftrightarrow P[I \leq \tau] \to 0\ for\ \tau < \infty$.

  the following hold:

  - $P[\xi_{k_i}(M) = x_i]_{\overrightarrow{M \to \infty}} P[\xi_{k_i}(\infty) = x_i] = \frac{e^{\lambda} \lambda^{x_i}}{x_i!}$;
  - $P[\xi_{k_1}(M) = x_1, \dots, \xi_{k_n}(M) = x_n]_{\overrightarrow{M \to \infty}} \prod_{i=1}^{n} P[\xi_{k_i}(\infty) = x_i]$.

- *For $M \to \infty$, the following are true for the process (a_s, B_s):*

– *The random variables B_s are independent of a_s;*

– *The random variables B_s are independent of ξ_s,*

for $\forall s \in Z$.

This result is useful in modeling the service facility as we will see in the next section.

3.2 The Service Facility

3.2.1 Source Level Analysis

The service facility is a generic name for the NOC. The direction of study for the service facility is not intended to address the protocol level. We only consider the case where the traffic is self-similar. This assumption is strongly supported by results in Leland et. al, [18], Crovella et. al. [8]. In other words we consider that due to various reasons which are not investigated here, the traffic has this behavior. A queueing approach is now introduced. Most of this information is from Likhanov et. al. [19].

The general picture is given in fig.3.3.

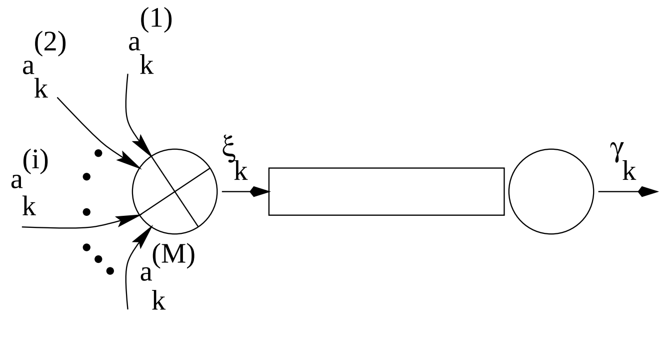

Figure 3.3: General Picture of the Service Facility

The individual sources are superposed into the aggregate traffic Λ_k. Each arrival brings a service requirement described by the random variable γ_k. This is an i.i.d random variable and independent of the arrival process.

This represents the service obtained by the packets in the SGW before being sent to the satellite link. It is important to distinguish between packet service and source service. The work to be done brought by a source is definitely composed of packets, and the real situation is that packets are sent through the satellite link.

Since it may be important to find the number of interactive connections (sources, Internet Servers) that can be accommodated simultaneously in the NOC, Lemma 1 of section 3.1.2 provides an approach for this problem.

If we consider the individual source process in part then the queue would be $G/D/1$. This is due to the general distribution of the individual arrival stream and the deterministic service brought about by the constant size of the packets. However, the aggregated traffic was shown to be Poisson. Then the model for the system is mapped into a $M/G/1$, and the state dynamics are described by a Markov Chain.

This model can be easily solved for the stationary state-occupancy probabilities, if the steady state exists, using the Polyachek-Kinchin formula (3.9).

Let $X = \{X_{k_i} | k_i \in Z\}$ be the state space, represented by the number of sources in the queue at time t_i. The arrival process is the aggregate process ξ_{k_i} from section 3.1.2. The arrival rate is the λ given by Lemma 1. The service process is heavy-tail distributed. In this case a Pareto distribution is considered, with appropriate parameters (in practice obtained by fitting to data) –see appendix A.2.1. The resulting Markov Chain is stationary if:

$$\rho = \lambda \mu_B < 1. \tag{3.8}$$

Figure 3.4 shows the state transition diagram and the events associated with state transitions.

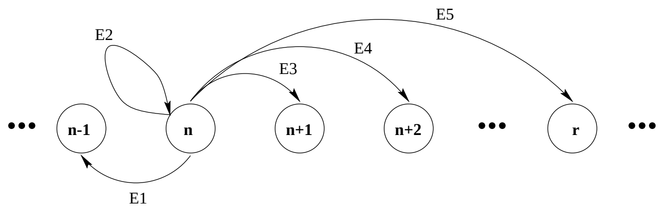

Figure 3.4: The state transition diagram and associated events

The following events drive the state transitions:

E1 Zero source arrivals in one service time;

E2 One source arrivals in one service time;

E3 Two source arrivals in one service time;

...

E5 $r \Leftrightarrow n + 1$ source arrivals in one service time;

If (3.8) holds, then the steady state probabilities exist. Let $q_i = P[X_k = i]$ denote these probabilities, where i is a positive integer.

Then in the Polyachek-Kinchin formula we have:

$$Q(\omega) = B(\lambda \Leftrightarrow \lambda\omega) \frac{(1 \Leftrightarrow \rho)(1 \Leftrightarrow \omega)}{B(\lambda \Leftrightarrow \lambda\omega) \Leftrightarrow \omega} \tag{3.9}$$

where,

$$Q(\omega) = \mathcal{Z}\{q_i\} = \sum_{i=0}^{\infty} q_i \omega^i$$
$$B(\omega) = \mathcal{L}\{p_B(j)\} = \sum_{j=1}^{\infty} P[B = j] e^{-\omega j}.$$

Also the balance equations for the Markov chain in fig.3.4 yield:

$$\begin{aligned} q_j &= \sum_{i=0}^{j} p_i q_{j-i+1} + p_j q_0 \\ \sum_{j=0}^{\infty} q_j &= 1, \end{aligned} \tag{3.10}$$

where,

$$\begin{aligned} &j \in N \cup \{0\}, \\ &p_i = \sum_{j=1}^{\infty} P[B = j] \frac{(j\lambda)^i}{i!} e^{-j\lambda}. \end{aligned} \tag{3.11}$$

In (3.11), p_i is the probability that i new sources will enter the queue during one busy period. This relation, (3.11) can provide network dimensioning information. It gives an estimate for the number of connections that can be busy during a typical ON period. Appendix C shows an example of network dimensioning, based on this result.

From (3.9), the probability that a departure leaves the queue empty is:

$$q_0 = Q(0) = 1 \Leftrightarrow \frac{\lambda}{\mu_B}. \tag{3.12}$$

Then from (3.10) we can solve for q_{j+1} recursively:

$$q_{j+1} = \frac{1}{p_0}[q_j \Leftrightarrow \sum_{i=1}^{j} p_i q_{j-i+1} \Leftrightarrow p_j q_0] \tag{3.13}$$

Equation (3.13) and the second one in (3.11), will be used in the next chapter for simulations.

3.2.2 Packet Level Analysis

In this section we are interested in developing bounds for the packet loss probability in the generic queue of section 3.2.1. While the results in the previous section can be used to generate source loss probability in a queue with a specified capacity measured in number of sources, the results in this section apply at the packet level.

We are using the approach in Likhanov et. al. [19]. The following relation gives the loss probability in a finite capacity queue:

$$P_{loss} \simeq \frac{c}{\alpha(\alpha+1)} i\lambda^{\alpha}(R\mu_B)^{1+\alpha}L^{1-\alpha}. \tag{3.14}$$

where L is the buffer length in packets and the other quantities are as defined earlier.

3.3 Bandwidth Allocation Strategies

In this chapter we present three control algorithms for bandwidth allocation. Each of them assumes that the controller is fully aware of the (per connection) queue status. The queue length is used to determine buffer space availability for newly arrived packets. All packets that are not dropped are considered part of the demand. If there are any other packets in the queue that did not receive service up to the current simulation clock, they are part of the demand also. The controller has the current demand information available. Past demand information is needed to compute the current one. Until this point, all control policies behave identically. From now on, based on the current demand, we investigate three bandwidth allocation strategies:

1. Equal Allocation;
2. Fair Allocation;
3. Most Delayed Queue Served First Allocation.

In the following subsections we describe each of these strategies.

3.4 Equal Bandwidth Allocation

According to this algorithm, whenever the demand from a connection is nonzero, it counts towards the sum of sources that participate in the bandwidth allocation.

This algorithm is given below:

Step 1 Find the number of connections with non-zero demand;

Step 2 Allocate the whole bandwidth equally to connections in the set generated at Step 1.

Steps 1 and 2 are performed on-line. The statistical nature of the connections necessitates large computing resources for such simulations, and for the real-world implementations of this strategy. The idle periods are statistically longer than the busy ones, which in turn implies that demands may be zero for a large set of simulation clock instants. Allocating the whole bandwidth to a restricted set of connections leads to cutting the portion of demand brought about by packets that do not receive service. This has a positive impact on delay. However, there is a significant waste of bandwidth occurring while operating under the equal allocation strategy.

Example: If there is only one connection with nonzero demand, which is almost sure less than the total bandwidth, then the difference between the total bandwidth and the demand is waisted. Physically, the connection in this case will use only the amount needed, but the service facility is not aware of this fact and spends resources unwisely.

3.5 Fair Bandwidth Allocation

This algorithm was reported by Raj Jain in a series of papers [13], [12].

This algorithm is given below:

Step 1 Find the number of connections with non-zero demand;

Step 2.1 If the sum of the individual demands is less then or equal to the total bandwidth, allocate as requested; **End**

Step 2.2 If the total of the individual demands exceeds the resource capacity, then go to Step 3;

Step 3 Divide the total bandwidth by the number of connections in the set generated at Step 1; This generates the **Fair Share**;

Step 4.1 For all connections with individual demand less than or equal to the **Fair Share**, allocate bandwidth to cover the entire individual demand;

Step 4.2 If Step 4.1 cannot be performed, then allocate the **Fair Share** to all connections in the set;

Step 5 Find the remaining bandwidth after allocating according to Step 4.1 and go to Step 6;

Step 6 Re-start from Step 3 with the set of non-zero demand connections for which bandwidth has not been allocated yet, and the total bandwidth as calculated at Step 5; repeat until each connection in the original set is served.

A fair allocation example is given below.

Example:

Assume that five connections have the following demand vector: $[1, 2, 5, 8, 3]$ and the total bandwidth to be shared is 15.

Step 1: The number of connections with non-zero demand is 5;

Step 2.1: Skipped – Total Demand = 19 > Total Bandwidth = 15;

Step 2.2: Tested as TRUE;

Step 3: Fair Share = 15/5 = 3;

Step 4.1 Allocate 1, 2 and 3 for connections 1, 2 and 5 respectively ;

Step 5 Remaining Bandwidth is $15 \Leftrightarrow 1 \Leftrightarrow 2 \Leftrightarrow 3 = 9$

Step 3 Fair Share = 9/2=4.5;

Step 4.2 Allocate 4.5 for connections 3 and 4.

Connections 1, 2 and 5 are served as they requested. Connection 3 gets 0.5; less than requested. Connection 4 gets 3.5; less than requested.

If we have operated under equal bandwidth allocation, then each connection would have received 3 units of bandwidth. Thus, connection 1 gets 2 units more, connection 2 gets 1 more, connection 3 gets 2 less, connection 4 gets 5 less and connection 5 is entirely covered. This example illustrates the superiority of the fair allocation strategy both in satisfying connection requests and minimizing the waste of bandwidth.

3.6 Most Delayed Queue Served First Bandwidth Allocation

The name of this algorithm indicates the operation of this strategy. The controller inspects the delays in the queues and allocates bandwidth starting with the one that has packets with longest delay.

This algorithm is given below:

Step 1 Sort the connections in decreasing order of the delay encountered by the packet in the head of the queue;

Step 2 Allocate bandwidth starting with the first queue in the ranking generated at Step 1;

Step 3 Repeat Step 2 until either the entire bandwidth is allocated or, all connections have received service.

To continue on the example from section 3.5, assume that the given vector is sorted in the sense of Step 1. Then connections 1, 2, 3, 4 get 1, 2, 5 and 7 units of bandwidth respectively. Connection 5 is not served, and connection 4 gets 1 unit less than it requested.

All bandwidth allocation strategies are simulated and numerical results discussed in chapter 4. Numerical results are presented in Appendices D.1, D.2 and D.3.

Chapter 4

Design of Experiments

The results in this thesis are in their great majority via simulation. It is thus important to dedicate a chapter to the design of experiments. This issue is common to analysis and design. However the implementation differs in the two cases. This difference is justified by the level of detail we want to attain in each of them. While the analysis is intended to closely reflect all communication processes, the design is especially addressing the testing of control algorithms. Moreover,analysis and design differ in what concerns the language chosen for simulation implementation. We do not give much detail on the programming aspects but it is worth mentioning that analysis is done using a C++ simulator while design is in Matlab. From the programming point of view, both are object oriented implementations.

4.1 Analysis Phase

The experiments on the flow control mechanism of the Hybrid Internet use the model given in fig.2.2.

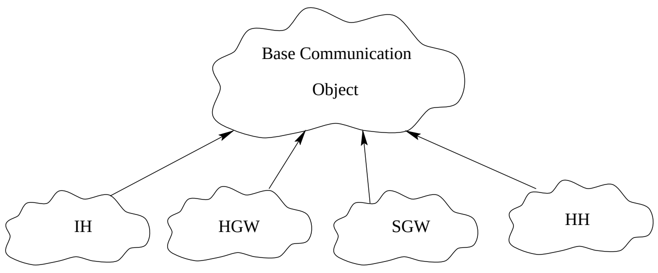

Figure 4.1: Communication Objects are Derived from a Base Object

4.1.1 Obtaining Models for Communication Objects

The design of the communication objects models relies on the fact that they all have common components being either protocols or hardware. Based on this remark, we decided to use object oriented design concepts. This can be followed in fig.4.1.

Each communication object obeys the increase-decrease congestion control algorithm of TCP. Also, each of them has data and acknowledgment queues. The SGW is the only object with two data queues. This reflects its actual configuration. The queue itself is an object in the programming sense and it is endowed with specific functions. Information flows between objects according to the architecture specified earlier. To allow for this, communication objects are designed to access the queue structures of their neighbors for sending packets. However all other operations at the queue level are private to the object that owns them.

4.1.2 Queue Model

Data circulates between communication objects, and packet queueing is needed to store the extra amount that cannot be sent at a time. For this reason, the queue model accuracy is one of the most important issues that has to be addressed. Both analysis and design use queues. For analysis, the queue object is much more complex. It has the capability of managing linked lists of messages, that in turn are distinct objects. Various queue operations are implemented with the following being the most important:

1. Addition of packets to the queue;
2. Keeping copies of unacknowledged messages;
3. De-queueing of packets;
4. Packet delay monitoring;
5. Queue length monitoring.

4.1.3 Simulation Setup

Most of the discrete event simulation is running inside a loop. Aside of the loop there are only initialization and object linking operations done prior to the beginning of the simulation in order to establish the network architecture and object properties. For this, a user input file is provided.

The simulation loop is kept very simple, all implementation details being transfered to the communication objects. They are equipped with an interface used to transfer the simulation clock, which is the only exogenous parameter needed.

4.2 Design Phase

4.2.1 Simulation Setup

The accuracy of the results is strongly related to the simulation setup. First, this is a discrete event simulation. Events are drown out from an event set, $\mathcal{E} = \{a, d\}$ where a, d stand for arrival and departure respectively. The state space is the queue length at the service facility, $\mathcal{S} = \{(x_1, \dots, x_M)\}$. Each component of the state vector refers as usual to the per-connection queue length. The NOC operating policy dictates that a connection level investigation should be performed. Fig. 4.2 graphically depicts the simulation process.

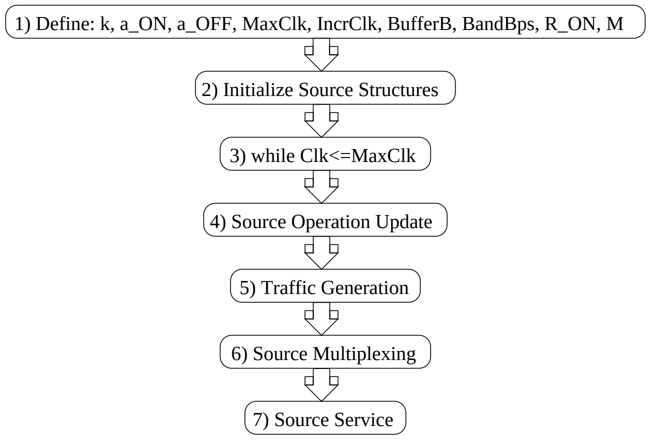
1) Define: k, a_ON, a_OFF, MaxClk, IncrClk, BufferB, BandBps, R_ON, M
2) Initialize Source Structures
3) while Clk<=MaxClk
4) Source Operation Update
5) Traffic Generation
6) Source Multiplexing
7) Source Service

Figure 4.2: The Simulation Loop

The constants k, a_ON, a_OFF, $MaxClk$, $IncrClk$, $BufferB$, $BandBps$, R_ON, M need to be selected at initialization. Their meaning is given below:

k**:** Pareto distribution parameter;

a_ON**:** Pareto distribution parameter - refers to the **Busy** period;

a_OFF**:** Pareto distribution parameter - refers to the **Idle** period;

$MaxClk$**:** Simulation length;

$IncrClk$**:** Simulation clock increment value;

$BufferB$**:** Total buffer space available in the service facility [**Bytes**];

$BandBps$**:** Total bandwidth available at the server **Bytes/s**];

R_ON**:** Source rate while in **Busy** period;

M**:** Number of sources to be multiplexed;

From a programming point of view we found convenient to model the source as a structure, even if the object-oriented approach would have been even more appropriate. Because here we will not discuss the programming aspects of the simulation, no more details on this problem are given. The second box in fig. 4.2 refers to the initialization of these structures. The most important aspect of the initialization is the choice for the source to begin in a **Busy** or **Idle** state at Clk = 0. This is because of the **ON-OFF** source behavior. After the initial decision is made, the session process is deterministic: a **Busy** period is followed by an **Idle** one all the way throughout the simulation process.

The third step in the discrete event simulation is to start the loop. Three main things have to be done within the simulation loop. First, and shown with number 4 in fig.4.2 is to update the source operation mode. By this, it is ensured that the operation is indeed **ON-OFF** at the session or connection level. The operation mode update is applied to any source that has an expired clock. We will show that the timing of transitions is done by maintaining clock structures "inside" each source. Second, traffic is generated for each source that is **Busy** at the current clock value. Traffic generation is the subject of the next subsection.

Third, the individual data streams are multiplexed into the aggregated traffic. Aggregated traffic is not used in deriving conclusions on performance metrics. It is mainly needed for comparison with the analytical results in section 3.1.2. Finally, the individual traffic streams are serviced in the NOC.

4.2.2 The Source Object

It is convenient from the programming point of view to encapsulate the properties of the individual sources into objects. The data type chosen to represent a source is close to the common **C** structure. It is not identical because we are using Matlab which even if **C** syntax-based and implemented does not strictly follow it. We will discuss the main properties –fields– of this structure with the intent of clarifying the simulation details. The source object will be referred to simply as source. Each source maintains an internal clock for synchronization purposes. Initially, that is when a transition from an operation mode to the complementary one appears, this clock is set to a value equal to the value of the Pareto distributed random variable. Then, the clock is decreased with $IncrClk$ while the simulation clock Clk advances. When the internal clock expires, an operation mode transition occurs. Each source maintains a queue for the packets that have been generated. This conveniently models the service facility in the sense that we do not distinctly need to create and manage a service object. This saves processing time and memory. Each source keeps track of the total number of packets generated. The status of each packet is also stored. Once generated, a packet joins the processing queue if there is space or it is dropped if not. Each packet has two time stamps. One reflects the arrival at the NOC and the other one the departure. A dropped packet is marked with infinite delay,

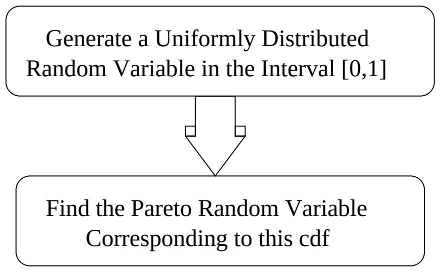
Generate a Uniformly Distributed Random Variable in the Interval [0,1]
Find the Pareto Random Variable Corresponding to this cdf

Figure 4.3: Generation of Pareto Distributed Random Variables

for consistency purposes. If delay is finite, it is calculated and stored. This concludes the description of the source object.

4.2.3 Generation of Fractal Traffic

Each source generates traffic according to the discussion in section 3.1.1. The **Busy** period has a Pareto distributed length. The **Idle** interval is also Pareto distributed but more heavy-tailed. The "lengths" of these periods are generated as shown in fig. 4.3.

In the first step a uniformly distributed random variable is generated:

```
F=rand(1) ;
```

In the second step, F from above is used to generate through the transformation method the random variable that corresponds to it. One reference for the transformation method is Leon-Garcia [9].

The pseudo-code for this second step is given below:

```
while y < k
  y = k/((1-F)^(1/a));
end
```

Here, k is the constant in the Pareto cdf. The random variable must be larger than k for the analytical results to hold in the simulation environment. The values for the Pareto random variables are double data types. The simulation time is discrete with increments $IncrClk$. It is important to determine as closely as possible the times when operating modes change and this happens at individual source clock expiration. All these statements lead to the need of adjusting the values returned by the random variable generator to be "multiples" of the $IncrClk$. This is shown in the following piece of pseudo-code which assumes that $IncrClk \in (0, 1)$ is of the form $\frac{1}{10^n}$, $n \geq 1$ and t is a double value.

```
t=t/IncrClk;
t=floor(t);
t=t*IncrClk;
```

First, the decimal point in the clock value is shifted towards right with n places; n is the same as in the last sentence. Then this value is rounded to the nearest integer that is less than it. Finally, this result is brought to the initial scale.

4.2.4 Running Experiments

The main goal in our design is to compare the three control algorithms. Given that sources have random behavior, we need to first find statistical quantities (metrics) that can be used to compare the three bandwidth allocation strategies. As usual, these statistical measures are average quantities, which in a simulation environment are time averages. For these time averages to converge to statistical averages, simulations have to run for a long time. As seen before, these simulations necessitate a powerful computational platform. For a number of sources in the order of hundreds experiments become themselves a problem.

The alternative, used in this work, is to define a fundamental environment for each control strategy to run under. Fig. 4.4 shows this approach.

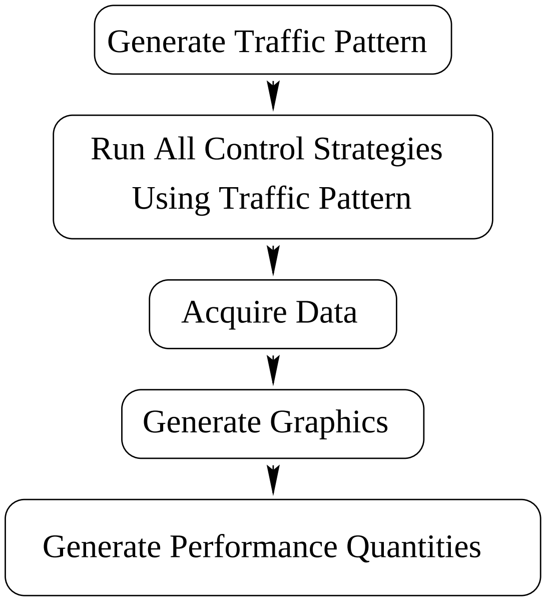
Generate Traffic Pattern
Run All Control Strategies Using Traffic Pattern
Acquire Data
Generate Graphics
Generate Performance Quantities

Figure 4.4: The Benchmarking Process

In the following we concretely describe the process.

4.2.5 Defining Common Input Data

Common input data is fed to all control strategies. Each control algorithm is tested for:

1 The same buffer space;

2 The same total bandwidth;

3 The same number of sources having:

3.1 The same succession of **ON-OFF** periods,

3.2 And the same constant arrival rate.

Table 4.2.5 summarizes these elements. Fig. 4.5 shows connection states.

Buffer per Connection	500 packets
Total Bandwidth	15 packets/unit time
Number of Connections	5 connections
Constant Arrival Rate	10 packets/unit time
Mean of the Uniform Arrival Rate	5 packets/unit time
Delay Imposed to Queued Packets	0.1 unit time

Table 4.1: Common Input Data

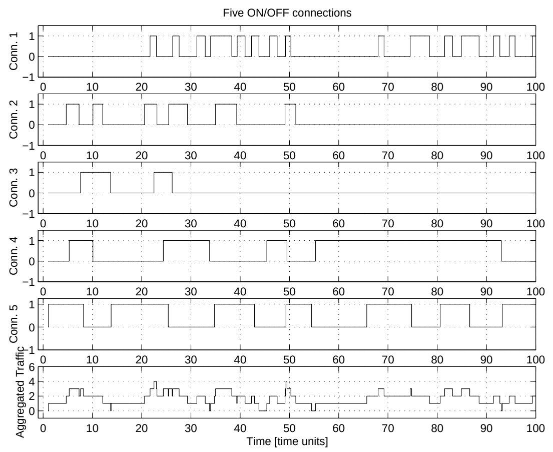

Figure 4.5: The source level behavior; This **ON-OFF** pattern has been applied for all control strategies.

4.2.6 Queue Dynamics

The service facility is a distinct entity, but it is not enough to model it as a single queue. In fact, it maintains a queue for each on-going connection. Therefore, in our simulations, we have 5 queues that share buffer space and bandwidth. We have considered equal allocation of buffer space for all connections. That is,

$$Buffer(i) = \frac{Total\ Buffer\ Space}{Number\ of\ Connections} \forall i \in \{1, ..., Number\ of\ Connections\}$$

This buffer allocation policy is not considered to be optimal. However we chose this strategy for the following reason: the statistical behavior of all connections is identical; the rate of packet generation while busy is the same for all connections. In other words, the interactive users have similar statistical behavior. Therefore there is no need to prioritize one connection in the detriment of the others.

In section 3.3 we mentioned the three bandwidth allocation strategies that have been simulated and compared. All of those rely on the same queuing dynamics. This is described in the following paragraph.

All packets received from sources are initially stored into the service facility buffer, where as seen above, space is equally divided among connections. Upon arrival, packets are time stamped. They may receive or not service depending on the bandwidth allocation policy. A packet that has received service is considered to be sent over the satellite channel, but a copy is maintained in the queue waiting for acknowledgment. Even if there is known data about uplink-downlink satellite delay, this would not help us very much as we chose to simulate a reduced number of connections. However the simulation can be easily modified to reflect real data.

The following are important quantities used throughout the simulation:

State The connection state at time t;

Queue Queue length at time t;

Demand The demand at time t;

Band The bandwidth allocated by the bandwidth allocation strategy at time t.

The interaction between these quantities is shown in the following two figures: 4.6 and 4.7.

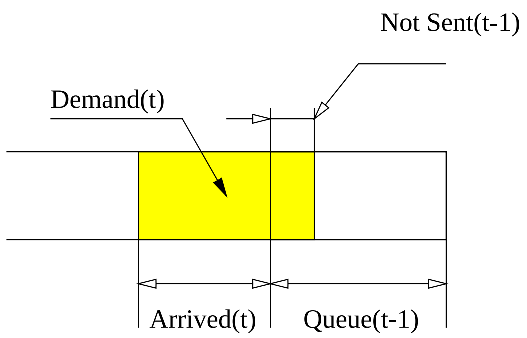

Figure 4.6: Demand at time t has two components: number of packets that were already in the queue but have not been sent and unacknowledged, and number of packets that have just arrived

In reality, it is most probable that copies are stored in separate queues. This is not important as long as the effect of their presence is kept unchanged. The most important effect is that these copies occupy buffer space that otherwise would have been distributed to the new coming packets.

The service is FIFO at the connection queue level. That is, packets are sent in the order of their arrival within each connection. The fact that we maintain

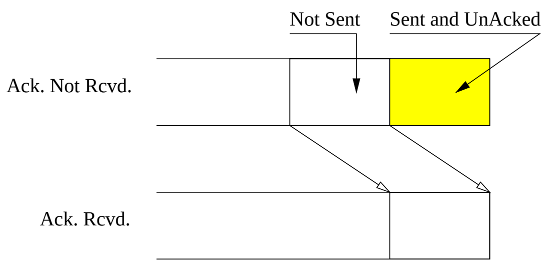

Figure 4.7: Copies for packets that have been acknowledged, are dropped; Then the whole queue must shift to the right

the copies as well as the new packets in the same logical entity is not disturbing the service policy. This is because the queue mechanism keeps track of the next packet to be sent out at any given time. Once the packet is sent over the satellite link, it will incur a deterministic delay. Therefore, for simplicity, all simulation data regarding delay is referring to the queueing component and not to the propagation one.

Chapter 5

Contributions of the Thesis, Conclusions and Future Work

5.1 Contributions of the Thesis

The entire analysis study has been performed using a simulation software built for this purpose. The first version has been developed while the author was with Hughes Network Systems. Since then, and mainly for the purpose of this thesis, the traffic model has been changed to reflect the self-similarity of interactive applications.

The testbed for bandwidth allocation policies was developed by the author of this thesis in the Center for Satellite and Hybrid Communication Networks of the Institute for Systems Research at the University of Maryland, College Park.

In order to compare the three bandwidth allocation strategies, we have used the result of section 3.2.1 to reduce the dimension of the simulations. It was shown that a negligible probability is associated with the event that a large number of sources start requesting service during a **Busy** period.

We proved that the policy that starts allocating bandwidth with the queue that has the largest instantaneous delay, is optimal in the case of interactive users.

5.2 Conclusions

First, the simulator used in analysis has been verified for accuracy with real data from Hughes Network Systems. In this thesis we did not include real traffic data for proprietary reasons. Even if the experiments performed and presented here are for a reduced set of users, the simulator can cope with a large number of connections. Moreover, there is a built-in capability for assigning different behavioral characteristics to users. In other words, they are not constrained on being interactive, but as seen, we can easily assign them this quality. Using a limited number of users allowed us to easily present simulation results. However, the main reason for this is the theoretical result on the number of connections that will enter the service facility in one **Busy** period, which in a sense limits the number of simultaneous sessions at a given time instant.

Second, design, containing both theoretical and experimental parts has proven what one's intuition would have dictated. Serving the user with the largest delay first allows for minimal queueing delay and minimal bandwidth wasting. This is a very important result, because it shows that there exist policies that can serve the interests of both the users and the service provider. Three different bandwidth allocation strategies have been investigated. The most efficient, in the sense if minimizing queueing delay is the **Most Delayed Queue Served First** bandwidth allocation. The fair allocation strategy gives better results than the equal allocation. Both the fair allocation and the most delayed queue first provide zero bandwidth wasting. This is very important because usually the interactive users are not the only users of a satellite-terrestrial network. The fair and most delayed queue first allocations are more computationally intensive than the equal one. This is the price paid for having better performance on the user side.

An important and interesting point is the comparison between analysis and design results. The actual implementation of the Hughes Network Systems Hybrid Internet seems to behave close to the equal bandwidth allocation policy. Further experiments can be performed to thoroughly sustain this, using more accurate simulators that closely reproduce the TCP congestion avoidance strategy. However, our conclusion is that the quality of the interactive applications using the Hybrid Internet architecture can be improved. It is important to mention that we did not model any extraneous processes that usually interfere with the Hybrid Internet operation, such as video or other services that compete for bandwidth consumption. These are present in the real system and what is worst, they are often given priority over the Internet traffic, usually based on revenue predictions.

5.3 Future Work

In the real system, interactive applications – like TELNET and small Web pages transfer – coexist with various other kind of processes. It is likely that we will encounter FTP transfers and multimedia applications trying to take their share of satellite bandwidth. In this case, a dynamic bandwidth allocation based on levels of priority may apply. This is an area needing further study.

Another interesting problem is the validation of the simulation experiments with real data acquired from the system. We decided to model traffic with Pareto distributed **ON-OFF** periods, based on the results in other papers. To prove that self-similarity exists in the HSTN, an extensive traffic monitoring should be done and the traces investigated with the appropriate mathematical and signal processing tools.

Appendix A

Self-Similar Processes

A.1 Definitions

In this appendix we give definitions for self-similarity and long-range dependency. For this, we follow the paper of Crovella, et.al [8].

Definition 4 *Let $X^{(m)} = \{X_k^{(m)} | k \in N\}$ be the aggregate process given by:*

$$X_k^{(m)} = \frac{\sum_{j=km-m+1}^{0} X_j}{m}, \ k \in Z. \tag{A.1}$$

with autocorrelation function $r^{(m)}(k)$. Then, X is called asymptotically self-similar with Hurst parameter $H = 1 \Leftrightarrow \frac{\beta}{2}$ $0 < \beta < 1$, if:

$$\begin{aligned} \lim_{m\to\infty} r^{(m)}(1) &= 2^{1-\beta} \Leftrightarrow 1, \\ \lim_{m\to\infty} r^{(m)}(k) &= \frac{[(k+1)^{2-\beta} \Leftrightarrow 2k^{2-\beta} + (k \Leftrightarrow 1)^{2-\beta}]}{2}, \ k \geq 2. \end{aligned} \tag{A.2}$$

For a self-similar process, the autocorrelation function does not change with aggregation. That is,

$$r^{(m)}(k) = r(k). \tag{A.3}$$

Definition 5 *Let $X = \{X_k | k \in N\}$ be a discrete time WSS process, with mean $\mu = E[X_k]$, variance $\sigma = E[(X_k \Leftrightarrow \mu)^2]$ and autocorrelation function $r(k) = E[(X_n \Leftrightarrow \mu)(X_{n+k} \Leftrightarrow \mu)]/\sigma^2, k \in N \cup \{0\}$. Then X is a long-range dependent process if*

$$\lim_{k \to \infty} \frac{r(k)}{k^{-\beta}} = const., \ 0 < \beta < 1. \tag{A.4}$$

Let $\beta = 2 \Leftrightarrow 2H$ in A.4. Then, $H = 1 \Leftrightarrow \frac{\beta}{2}$, with $0.5 < H < 1$. The parameter H is called Hurst parameter and completely defines A.4.

The long-range dependence shows that the autocorrelation decays hyperbolically which is slower than the exponential one, when $0 < \beta < 1$.

Due to relation (A.3), self-similarity is specifically referring to self-similarity in distribution. That is, for different aggregation scales, the distribution remains unchanged. This is the reason why traffic that exhibits self-similarity is called fractal like traffic. Long-range dependency is a result of self-similarity.

A.2 Heavy-Tailed Distributions

Definition 6 *Let X be a random variable with distribution $F_X(x)$. Then X has a heavy-tailed distribution if*

$$\begin{aligned} &1 \Leftrightarrow F_X(x) = P[X \geq x] \sim x^{-\alpha}, for \\ &x \to \infty \ and \ 0 < \alpha < 2. \end{aligned} \tag{A.5}$$

A.2.1 Pareto Distributions

The Pareto distribution is extensively used in modeling ATM and interactive traffic like Web transfers and Ethernet. Tsybakov et. al. [19], [29], Crovella et. al. [20], [8] use it in analysis and/or simulations. Here we follow the work in Johnson [15] for some definitions.

The following defines the **Pareto distribution of the first kind**.

Definition 7 *Let X be a random variable with distribution $F_X(x)$. Then X is Pareto distributed if the cdf is given by:*

$$\begin{aligned} &F_X(x) = 1 \Leftrightarrow k^{\alpha} x^{-\alpha}, where \\ &k > 0,\ \alpha > 0,\ x \geq k. \end{aligned} \tag{A.6}$$

Then the pdf is

$$p_X(x) = \alpha k^{\alpha} x^{-(\alpha+1)}. \tag{A.7}$$

In general the r^{th} moment of the Pareto random distribution is finite if $r < \alpha$. Here we use random variables with finite mean and infinite variance, which translates into $1 < \alpha < 2$. In this case, the expected value is given by the following:

$$E[X] = \frac{\alpha k}{\alpha \Leftrightarrow 1}. \tag{A.8}$$

Appendix B

Analysis Results

There are various quantities that the analysis simulator can provide. Here we present traces from the un-acknowledged output of the SGW. For comparison purposes we used the same setup as in the design study. There are five logical queues, one per connection. As opposed to the design implementation, these are not distinct entities. Packets coming from the IS may end in either the high or low priority queues of the SGW. This is a fundamental difference between operation policies. Allocation of bandwidth to Hybrid Internet traffic is done on a fixed algorithm basis. It resembles the actual algorithm running in the NOC.

Fig.B.1 presents the throughput sensed by the HH, under the error-free satellite channel assumption.

Fig.B.2 presents the delay per connection.

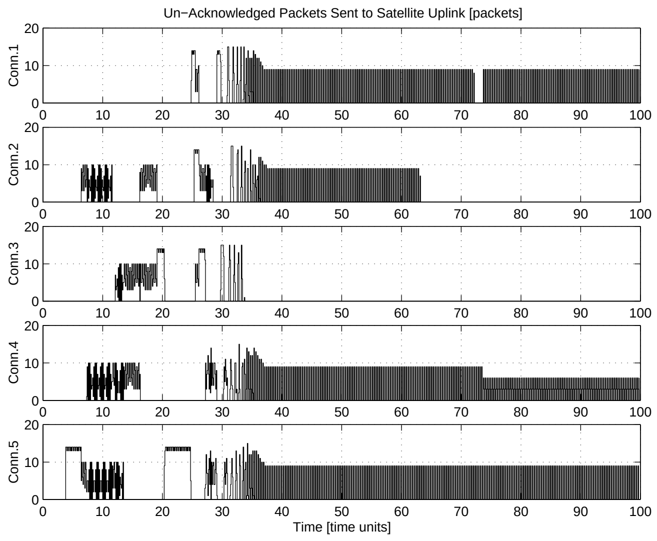

Figure B.1: Un-Acknowledged Output (Throughput) of the SGW.

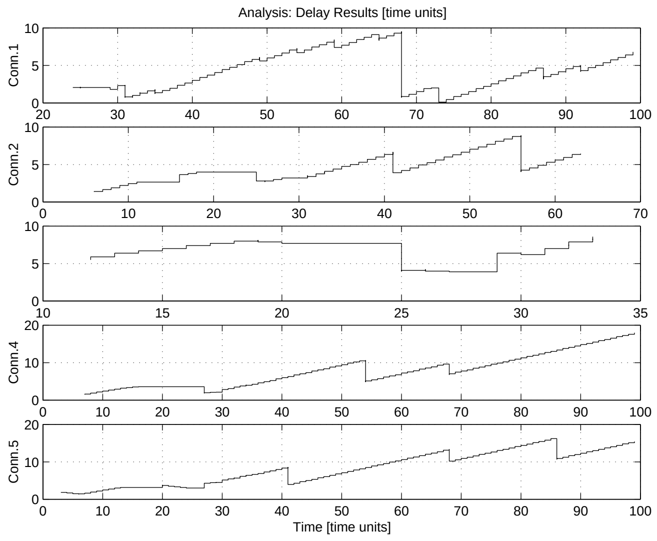

Figure B.2: Per connection queueing delay.

Appendix C

An Example for Network Dimensioning

Given that one has information about the users' behavior the results in section 3.2.1 provide approximations for the number of sources that can be in the system at the same time.

The information about the user translates into a known Pareto distribution. For illustration purposes, we choose $\alpha = 1.5$, $k = 1.447$, and 10 connections. α and k are parameters that define the Pareto distribution.

The following two figures show the probability that i new sources enter the service facility in a busy period with mean α and the logarithmic plot of the stationary queue distribution.

Fig. C.1 shows that there are few chances that a big number of sources will try to join the queue during a busy period of mean α. This can also help in limiting the number of sources that will be used for simulating various bandwidth allocation algorithms.

Fig. C.2 shows that the probability of the number of sources in the queue decreases algebraically fast and not exponentially as in the classical Markovian models.

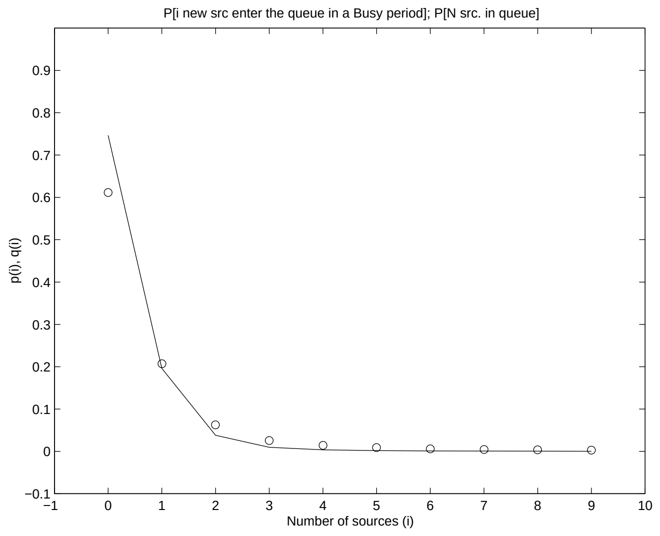

Figure C.1: Probability that i new sources enter the service facility during a Busy period – solid line; Stationary queue distribution – circles

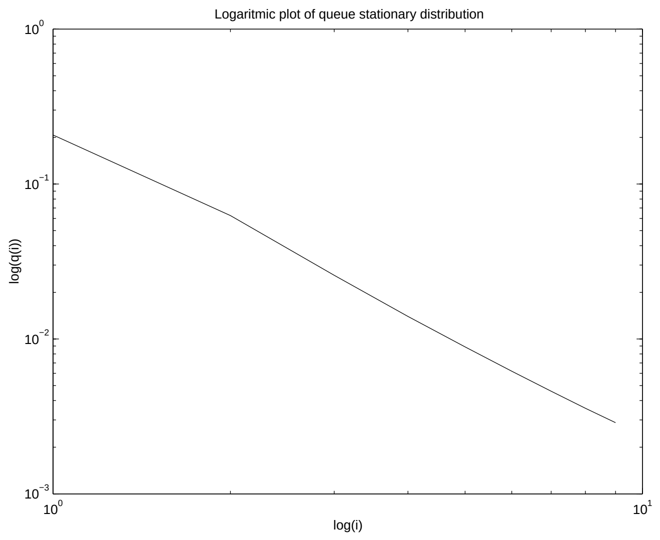

Figure C.2: Stationary queue distribution in logarithmic coordinates

The remark illustrated in figure C.1, that just few number of sources will actually enter service, is sustained also by the following experiment. We have considered 100 sources and plotted the aggregated traffic. Each source sends packets at the rate of 1 packet/simulation clock; therefore the figure is basically representing the number of sources in Busy state at any moment.

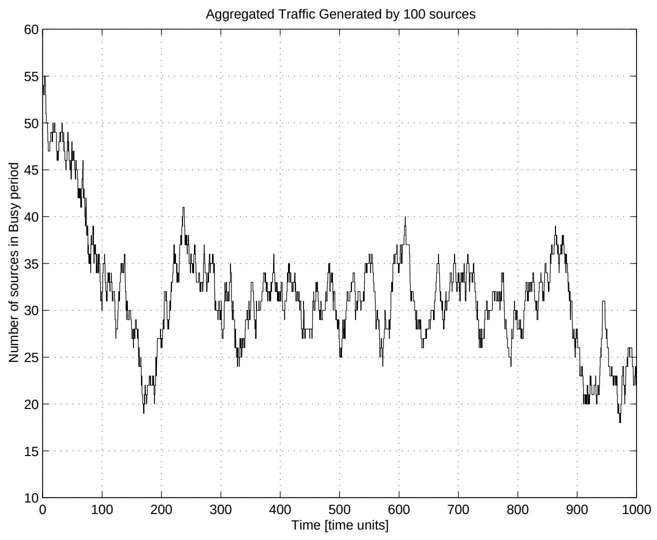

Figure C.3: Aggregated Traffic Generated by 100 ON/OFF sources

In the long-run the number of sources that will enter the service facility decreases to zero. This is theoretically sustained by the assumption that the Idle period has infinite mean.

Appendix D

Design Results

This appendix and the following two present experimental results obtained with different bandwidth allocation policies. The following quantities are shown in figures D.1 ... D.5:

Connection State: At any time, a connection can be either Busy, shown with 1 or Idle, shown with 0. While Busy, the source generates packets at constant rate. All connections use the same rate.

Queue Length: The queue length is an important quantity which may ultimately determine if packets will be dropped or not.

Demand: As presented earlier (see chapter 4), demand represents the number of packets that are admitted in the queue and they are either new packets or ones that have not received yet service.

Bandwidth: This is the number of packets that a queue is allowed to output at a time; It depends on the bandwidth allocation policy; The packets that are sent to the satellite link are not actually deleted from the queue until the acknowledgment is coming from the destination node.

Delay: This is the instantaneous delay suffered by a packet that is sent out and not yet acknowledged; In other words, this is the queueing delay.

Acked: Is the number of packets sent and acknowledged.

Unacked: Is the number of packets sent and un-acknowledged.

The total delay suffered by a packet in its way from the source to the destination is the sum of the queueing delay and propagation delay. While the latter is constant, the former is dependent on the bandwidth allocation as sections D.1, D.2 and D.3 illustrate.

D.1 Equal Bandwidth Allocation

Table D.1 presents the average delays for each of the five connections with the service facility running in equal allocation mode.

Connection Index	Average Delay
1	1.4469
2	2.0720
3	1.6941
4	2.0541
5	1.7182

Table D.1: Average queueing delays with the service facility running the equal bandwidth allocation policy

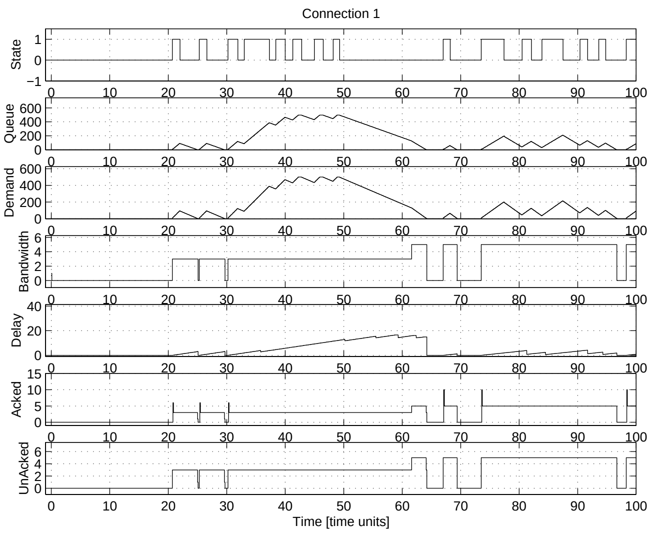

Figure D.1: EBA, Connection 1: State, Queue, Demand, Bandwidth, Delay, Acknowledged packets, Unacknowledged packets; All quantities are given in number of packets

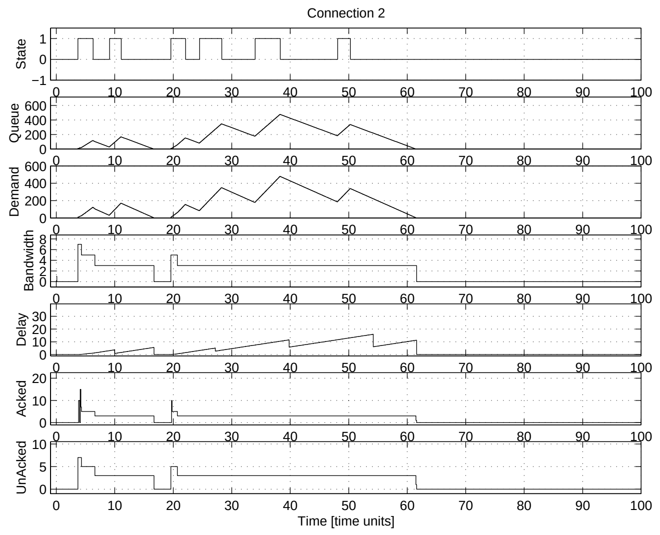

Figure D.2: EBA, Connection 2: State, Queue, Demand, Bandwidth, Delay, Acknowledged packets, Unacknowledged packets; All quantities are given in number of packets

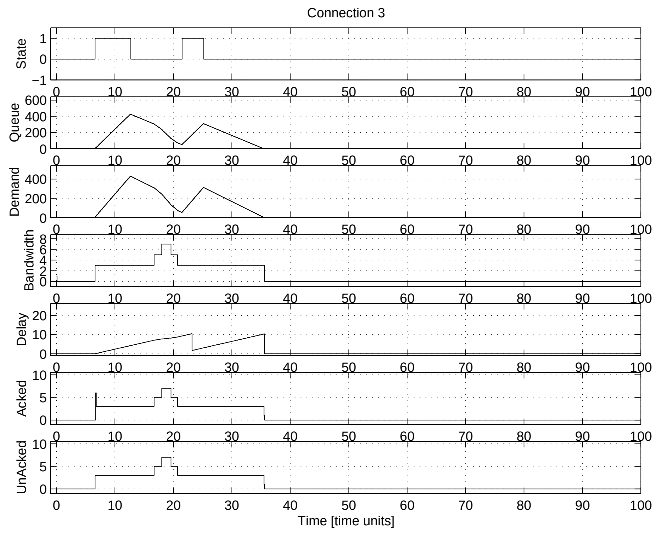

Figure D.3: EBA, Connection 3: State, Queue, Demand, Bandwidth, Delay, Acknowledged packets, Unacknowledged packets; All quantities are given in number of packets

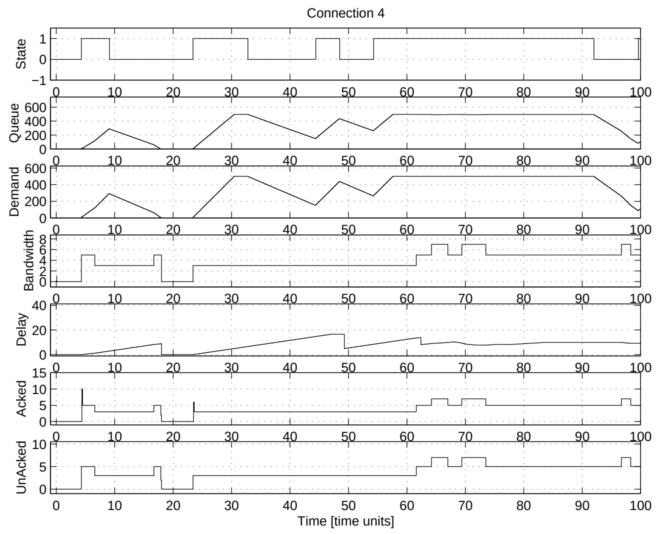

Figure D.4: EBA, Connection 4: State, Queue, Demand, Bandwidth, Delay, Acknowledged packets, Unacknowledged packets; All quantities are given in number of packets

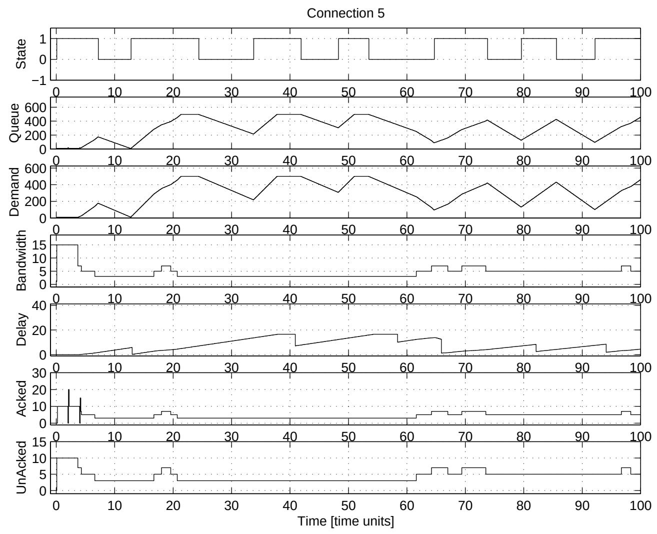

Figure D.5: EBA, Connection 5: State, Queue, Demand, Bandwidth, Delay, Acknowledged packets, Unacknowledged packets; All quantities are given in number of packets

D.2 Fair Bandwidth Allocation

In this case, delays are smaller than in the previous case. Bandwidth waist is reduced to zero. Table D.2 presents the average delays for the fair allocation algorithm.

Connection Index	Average Delay
1	1.4468
2	2.0720
3	1.6689
4	2.0524
5	1.7088

Table D.2: Average queueing delays with the service facility running the fair bandwidth allocation policy

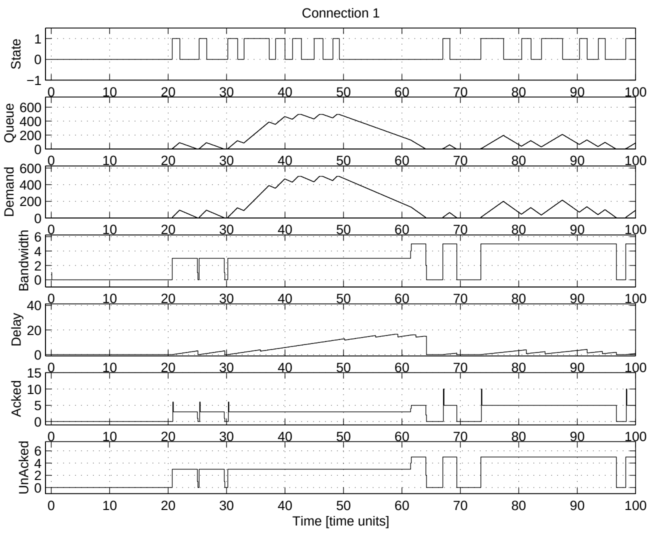

Figure D.6: FBA, Connection 1: State, Queue, Demand, Bandwidth, Delay, Acknowledged packets, Unacknowledged packets; All quantities are given in number of packets

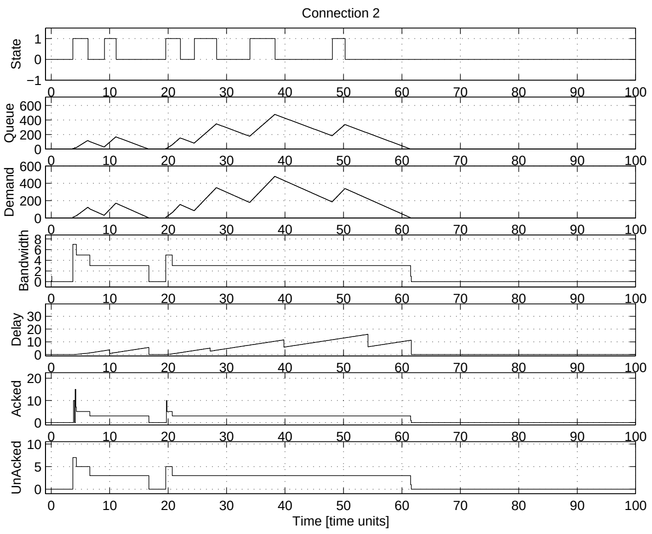

Figure D.7: FBA, Connection 2: State, Queue, Demand, Bandwidth, Delay, Acknowledged packets, Unacknowledged packets; All quantities are given in number of packets

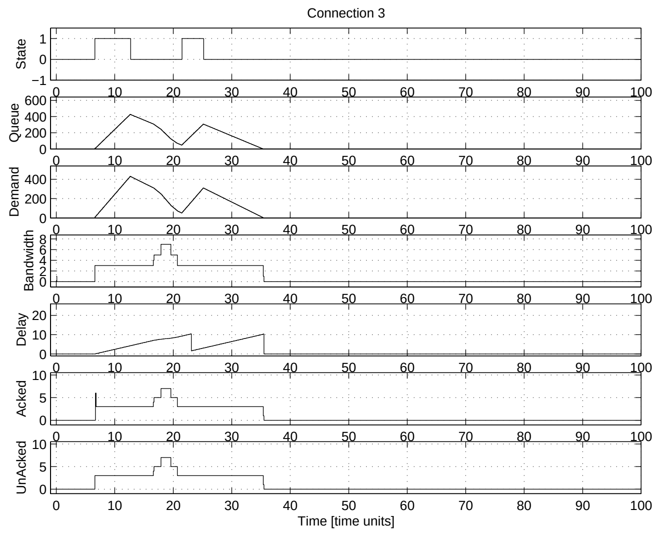

Figure D.8: FBA, Connection 3: State, Queue, Demand, Bandwidth, Delay, Acknowledged packets, Unacknowledged packets; All quantities are given in number of packets

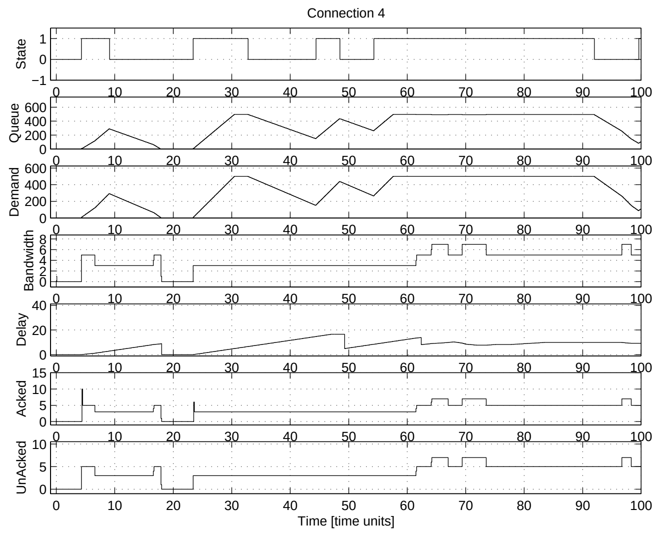

Figure D.9: FBA, Connection 4: State, Queue, Demand, Bandwidth, Delay, Acknowledged packets, Unacknowledged packets; All quantities are given in number of packets

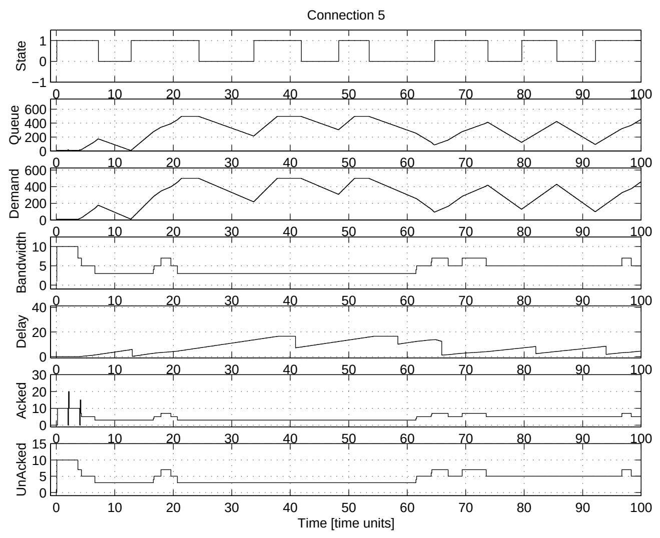

Figure D.10: FBA, Connection 5: State, Queue, Demand, Bandwidth, Delay, Acknowledged packets, Unacknowledged packets; All quantities are given in number of packets

D.3 MDQSF Bandwidth Allocation

Connection Index	Average Delay
1	0.0000
2	0.5298
3	0.0204
4	0.0741
5	0.8847

Table D.3: Average queueing delays with the service facility running the most delayed queue served first bandwidth allocation policy

This is clearly the most efficient control algorithm with respect to minimizing delay. The bandwidth waist in this case, is zero as in the fair allocation policy.

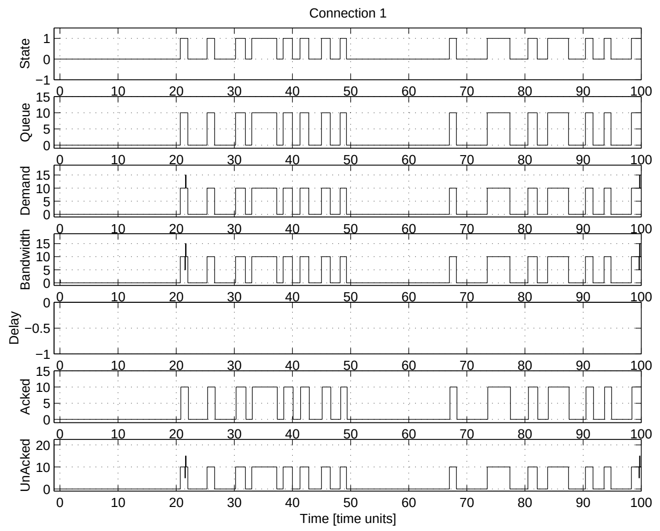

Figure D.11: MDQSFBA, Connection 1: State, Queue, Demand, Bandwidth, Delay, Acknowledged packets, Unacknowledged packets; All quantities are given in number of packets

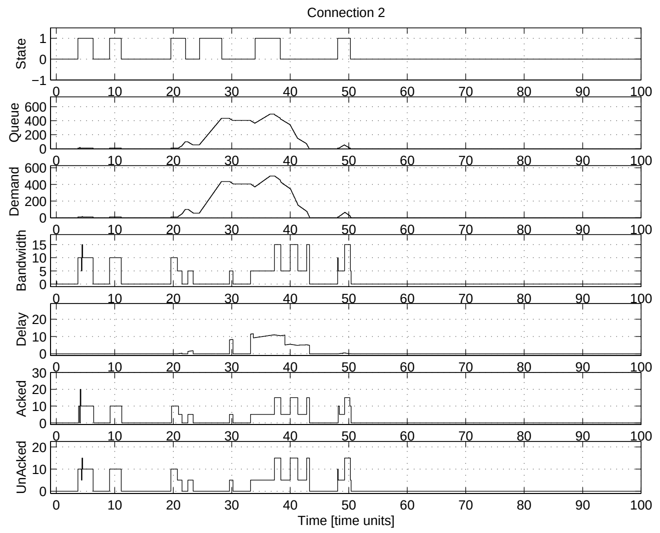

Figure D.12: MDQSFBA, Connection 2: State, Queue, Demand, Bandwidth, Delay, Acknowledged packets, Unacknowledged packets; All quantities are given in number of packets

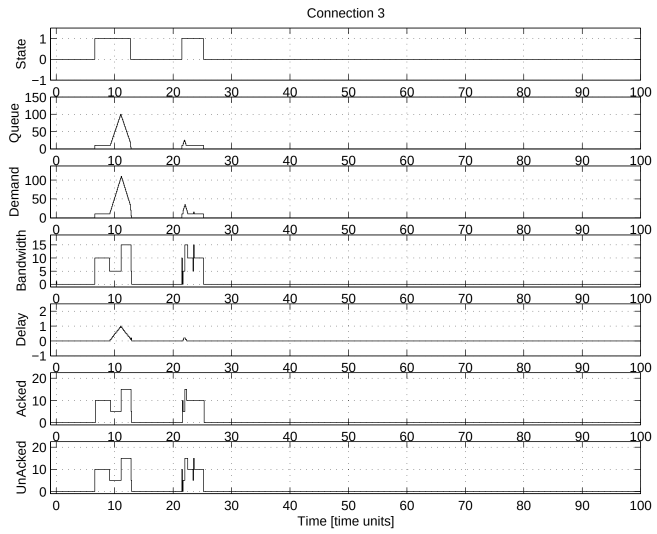

Figure D.13: MDQSFBA, Connection 3: State, Queue, Demand, Bandwidth, Delay, Acknowledged packets, Unacknowledged packets; All quantities are given in number of packets

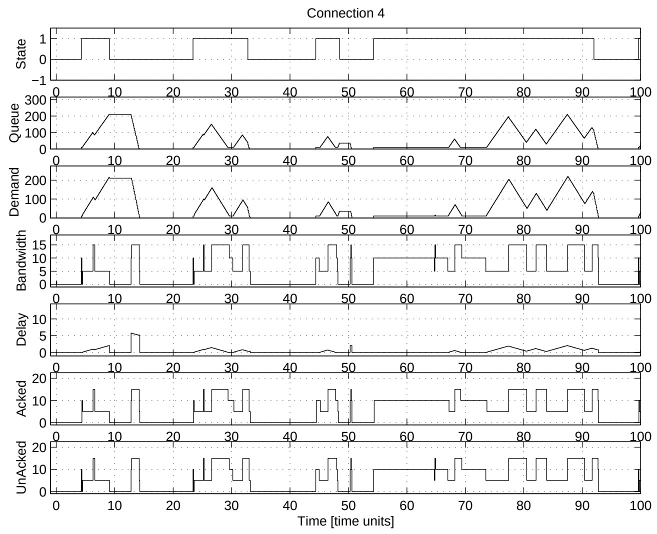

Figure D.14: MDQSFBA, Connection 4: State, Queue, Demand, Bandwidth, Delay, Acknowledged packets, Unacknowledged packets; All quantities are given in number of packets

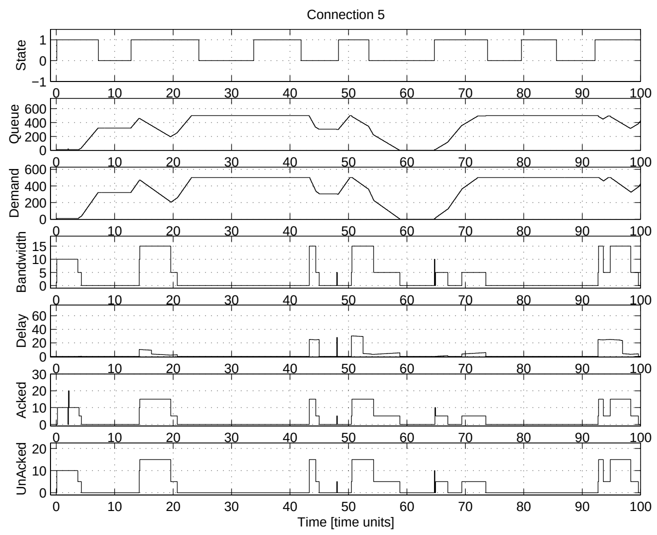

Figure D.15: MDQSFBA, Connection 5: State, Queue, Demand, Bandwidth, Delay, Acknowledged packets, Unacknowledged packets; All quantities are given in number of packets

Bibliography

[1] Anantharam, V., *On the Sojourn Time of Sessions at an ATM Buffer with Long-Range Dependent Input Traffic*, (1995), Proceedings of the 34th IEEE Conference on Decision and Control, Dec. 1995.

[2] Arora, V., Baras, J.S., Dillon, D., Falk, A., Suphasindhu, N., *Hybrid Internet Access*, (1995), Conference on NASA Centers for Commercial Development of Space, Albuquerque, NM, pp.69-74, 1995.

[3] Arora, V., Suphasindhu, N., Baras, J.S., Dillon, D., *Asymmetric Internet Access Over Satellite-Terrestrial Networks*, (1995), American Institute of Aeronautics and Astronautics, AIAA-96-1044-CP, pp.476-482, 1996.

[4] Beran, J., Sherman, R., Taqqu, M.S., Willinger, W., *Long-Range Dependence in Variable-Bit-Rate Video Traffic*, (1995), IEEE Transactions on Communications, Vol.,43, pp., 1566-1579, 1995.

[5] Bertsekas, D.P., Gallager, R., *Data Networks*, (1992), Second Edition, Prentice Hall, 1992.

[6] Charny, A., *An Algorithm for Rate-Allocation in a Packet-Switching Network with Feedback*, (1994), MIT Advanced Network Architecture Laboratory, Technical Report TR 601, 1994.

[7] Crovella, M.E., Lipsky, L., *Long-Lasting Transient Conditions in Simulations with Heavy-Tailed Workloads*, (1997), Proceedings of the 1997 Winter Simulation Conference, Dec., 1997.

[8] Crovella, M.E., Bestavros, A., *Explaining World Wide Web Traffic Self-Similarity*, (1995), Computer Science Department, Boston University, Oct. 12, 1995.

[9] Leon-Garcia, A., *Probability and Random Processes for Electrical Engineering*, (1994), Second Edition, Addison-Wesley, 1994.

[10] Garett, M.W., Willinger, W., *Analysis, Modeling, and Generation of Self-Similar VBR Video Traffic*, (1994), Proceedings of the ACM SIGCOMM, 1996.

[11] Georganas, N.D., *Self-Similar ("Fractal") Traffic in ATM Network*, (1994), Proceedings of the second International Workshop on Advanced Teleservices and High-Speed Communication Architectures (IWACA'94), Heidelberg, Germany, Sep. 1994.

[12] Jain, R., Ramakrishnan, K., Chiu, D., *Congestion Avoidance in Computer Networks with a Connectionless Network Layer*, (1988), DEC-TR-506.

[13] Jain, R., Hawe, W., *Performance Analysis and Modeling of Digital's Networking Architecture*, (1986), Digital Technical Journal, No. 3, September 1986, pp.,25-34.

[14] Jacobson, V., Braden, R., Borman, D., *TCP Extensions for High Performance*, RFC 1323.

[15] Johnson, N.L., Kotz, S., Balakrishnan, N., *Continuous Univariate Distributions*, (1994), Vol. 1, Second Edition, Wiley Series in Probability and Mathematical Statistics, pp.,573-621

[16] Lakshman, T.V., Madhow, U., Suter, B., *Window-based error recovery and flow control with a slow acknowledgment channel: a study of TCP/IP performance*,(1997), to appear, Proceedings of Infocom '97.

[17] Lakshman, T.V., Madhow, U., *The performance of TCP/IP for networks with high bandwidth-delay products and random loss*, to appear in IEEE/ACM Transactions on Networking.

[18] Leland, W., Taqqu, M., Willinger, W., Wilson, D., *On the Self-Similar Nature of Ethernet Traffic*, (1994), IEEE/ACM Transactions on Networking, No.2, pp.,1-15, 1994.

[19] Likhanov, N., Tsybakov, B., Georganas, N.D., *Analysis of an ATM Buffer with Self-Similar ("Fractal") Input Traffic*, (1995), Proceedings of IEEE INFOCOM'95, Boston, Apr. 1995.

[20] Park, K., Kim, G., Crovella, M.E., *The Effects of Traffic Similarity on TCP Performance*, (1996), Computer Science Department, Boston University, Apr. 22, 1996.

[21] Park, K., Kim, G., Crovella, M.E., *On the Relationship Between File Sizes, Transport Protocols, and Self-Similar Network Traffic*, (1996), Computer Science Department, Boston University Technical Report TR-96-016, 1996.

[22] Paxson, V., Floyd, S., *Wide Area Traffic, the Failure of Poisson Modeling*, (1995), IEEE Transactions on Networking, Vol.,3, pp.,226-244, 1995.

[23] Pruthi, P., Erramilli, A., *Heavy-Tailed On/Off Source Behavior and Self-Similarity*, (1995), Proceedings of ICC, Seattle, 1995.

[24] Ryu, B.K., Lowen, S.B., *Point Process Approaches to the Modeling and Analysis of Self-Similar Traffic: Part I – Model Construction* (1996), Proc. IEEE INFOCOM '96, San Francisco, March, 1996.

[25] Ryu, B.K., Lowen, S.B., *Point Process Approaches to the Modeling and Analysis of Self-Similar Traffic: Part II – Applications* (1997), Proceedings of the 5th International Conference for Telecommunications Systems, Modeling, and Analysis, Nashville, TN, March 1997.

[26] Ryu, B.K., Lowen, S.B., *Real-Time Generation of Fractal ATM Traffic: Model, Algorithm, and Implementation*, (1995), Center for Telecommunications Research, Columbia University, Mar., 1996.

[27] Ryu, B.K., Lowen, S.B., *Implications of Self-Similarity for Providing End-to-End QOS Guarantees in High-Speed Networks: A Framework of Application Level Traffic Modeling*, (1996), Proceedings International Zurich Seminar, Zurich, Feb.,1996.

[28] Shroff, N.B., *Traffic Modeling and Analysis in High Speed ATM Networks*, (1995), Ph.D. Thesis, Columbia University, 1995.

[29] Tsybakov, B., Georganas, N.D., *On Self-Similar Traffic in ATM Queues: Definitions, Overflow Probability Bound and Cell Delay Distribution*, (1997), IEEE/ACM Transactions on Networking, Jun. 1997.

[30] Waldby, J., Madhow, U., *Dynamic window-based flow control with end-to-end selective acknowledgments*, Proceedings of the 3rd International Work-

shop on Mobile Multimedia Communications, Princeton, NJ, (1996), September 25-27, 1996.