TKN Telecommunication Networks Group

Technical University Berlin

Telecommunication Networks Group

Modeling of Burst Assembly Process in Optical Burst-Switched Networks

Ahmad Rostami

rostami@tkn.tu-berlin.de

Berlin, February 2007

TKN Technical Report TKN-07-01

TKN Technical Reports Series

Editor: Prof. Dr.-Ing. Adam Wolisz

Abstract

We analyze burst assembly process, as the main building block of optical burst switching (OBS) paradigm. The analysis is performed for time-based, volume-based as well as hybrid burst assemblers. Under the assumption that packets arrival to the assembly buffer is Poisson, exact analytical expressions are derived for length and interdeparture time of bursts that are generated by these three classes of assembly algorithms. Furthermore, we consider the issue of generating burst traces, which arises during performance evaluation of OBS networks through discrete-event simulation. In such a simulation study a significant part of the simulation time, particularly in case of a network with a large number of ingress nodes, is used by the implementation of the burst assembly algorithms. This is due to the fact that each data burst is result of aggregating several short-length packets which - in a straightforward approach - have to be generated individually and afterwards "melted" into the burst. We present a novel approach to fast generation of bursts, which is based on the analytical models developed for burst length and interdeparture time distributions as well as an efficient generation technique (composition) supporting generation of these distributions. The analysis is followed by numerical results that validate the accuracy of developed models and demonstrate the speed-up gains of using proposed burst generation algorithms.

Contents

Chapter 1

Introduction

Advances in optical technology and wavelength division multiplexing (WDM) have resulted in a huge amount of capacity available for transmission over optical links. Unfortunately the possibilities of efficient usage of this capacity for data and multimedia traffic are frequently limited by the fact, that switching techniques with fine switching granularity – like packet switching – cannot be efficiently realized in the optical domain using current technology. The reason being that switching times of optical devices are still too large compared with the average packets length and that there is no equivalent to random access memory (RAM) in optical domain to realize the so-called store-and-forward switching technique. To address the issue, optical burst switching (OBS) has recently been proposed ([1], [2]) and attracted much attention from the networking research community.

OBS is a switching paradigm that allows for dynamic allocation of resources in the optical domain at sub-wavelength granularity. It does so by combining three principles, namely burst assembly at the edge, one-way out-of-band signaling and cut-through switching in the core. Burst assembly refers to the process of aggregating small-size packets into bursts at ingress edges of the optical network. By increasing the size of data units this aggregation makes it possible to relax the requirements on the speed of optical switching. Once a new data burst is ready for transmission at an ingress node, a signaling message is generated and released to the network ahead of the burst. The data burst then follows the message an offset time after it has been sent without waiting for an acknowledgement, i.e., one-way reservation. The role of the signaling message is to inform all switching nodes along the path of exact arrival time of the burst so they can configure their ports so as to switch the burst in a cut-through fashion upon its arrival. To that end, each signaling message is processed electronically at every intermediate node after passing through an opto-electrical conversion. However, in order that data bursts can bypass such conversions at intermediate nodes, signaling messages are transmitted on dedicated WDM channels, i.e., out-of-band signaling.

In this report we focus on the burst assembly process as one of the main building blocks of OBS architecture. Assembly algorithms change the statistical characteristics of the input traffic that, in turn, influences the performance of the network. In order to thoroughly analyze performance implications of the assembly process in OBS networks, the first step would be to study the burst assembly algorithm itself. Nevertheless, only few analytical studies exist in this direction, which have resulted in using simulative investigations as the main tool to understand the performance implications of different burst assembly algorithms.

Yu *et al.* [4] investigate the assembly process and its implication on the performance of the network, however, they only discuss the approximate distributions of bursts length and interdeparture time under time-based and volume-based algorithms. Laevens [6] provides an approximation of burst length distribution for the slotted-time Bernoulli packet arrival to the assembly buffers. Also, authors in [13] study a variation of hybrid burst assembly which is periodic and bursts are generated only at the end of each period. They present an approximate expression for the distribution of number of bursts that are generated at the end of each assembly period.

On the other hand, in a simulative experiment, the process of burst generation takes a significant amount of simulation time, since each data burst can be created only by generating multiple individual packets. This issue becomes especially crucial when OBS networks with a large number of ingress nodes have to be simulated over longer operational time periods. The most common approach to overcome the problem is to oversimplify it by ignoring the impact of burst assembly process on traffic characteristics, and simply assuming that burst traffic has the same statistical characteristics as the packet traffic - in fact it is usually assumed that bursts arrive at an OBS core node according to the Poisson process with burst lengths exponentially distributed. This approach, however, may undermine the credibility of simulation results and produce misleading results. In [7], we have demonstrated that the impact of the assembly process on the data loss rate at a core OBS node may be as high as several orders of magnitude!

The contribution of this report is two-fold. First, we derive analytical models for length and interdeparture time distributions of the bursts that are generated by the most popular assembly algorithms. Namely, we will consider time-based, volume-based as well as hybrid burst assembly algorithms. Then, we apply a technique, which is referred to as *composition technique*, to the models in order to develop simple algorithms that mimic real assembler's behavior by generating burst traffic of the same statistical characteristics. The algorithms can be exploited as burst traffic generator in a per flow basis in discrete event simulation models to accelerate the simulation.

In developing analytical models in this work, we shall make the common assumption that packet-level traffic arrives at the assembly buffer according to the Poisson process and packet lengths are exponentially distributed. The assumption of Poisson packet arrivals can be justified taking into account that, as we will observe in the next Chapter, burst assemblers multiplex packet-level traffic of a large number of micro flows. Moreover, there are some recent measurements of traffic in the Internet suggesting that at sub-second time scales packet arrivals in the core network follow the Poisson process [5]. We note here that the time scale associated with the burst assembly process is indeed far below a second. In addition, while exponential assumption for packets length distribution allows us to develop tractable models, through simulation we will present results for the case where arriving IP packets have real measurement-based trimodal distribution, and show that the developed models provide a good approximation also for this case.

The rest of this report is structured as follows. In Chapter 2 we discuss popular algorithms for burst assembly and their operation. In Chapter 3, we present our analytical models for probability density functions of interdeparture time and length of the bursts generated by different types of assembly algorithms. In Chapter 4 we apply the composition technique to the models of Chapter 3 to develop fast burst generators that can be used in simulation

models. In Chapter 5 we present numerical results that validate the accuracy of developed models and demonstrate that large speed-up gains can be achieved by employing proposed burst generators. Final comments included in Chapter 6 conclude the work.

Chapter 2

Burst Assembly in OBS

Burst assembly process is one of the distinguishing features of OBS architecture that could largely influence performance of the network. In order to generate bursts, each OBS ingress node contains a burst assembly unit. The unit receives packet-level traffic, which is usually collected from low speed links, at its input and aggregate them into bursts. In its simplest form, an assembly unit is composed of a packet classifier, an assembly controller and several assembly buffers, see Fig. 1. Once a packet arrives to the unit, its destination address (and possibly the QoS class that it belongs to) is checked by the packet classifier. According to the result of the classification, the packet is enqueued in one of the available virtual destination queues (VDQs), where each VDQ is a buffer dedicated to packets destined to a certain egress node and is referred to as an assembly buffer. Therefore, the unit should contain one assembly buffer per destination (and per QoS class). Note that the term destination here refers to an egress node in the OBS cloud, thus packets of the same attributes from different micro flows can be multiplexed into the same assembly buffer.

The assembly controller is responsible for making the decisions regarding when contents of each assembly buffer should be aggregated into a burst and released to the network. The controller takes care of scheduling burst generations based on several criteria, of which some are imposed by the network, e.g., maximum and minimum burst length, and others are imposed by QoS requirements of incoming packet traffic, e.g., maximum delay that a packet can tolerate in an assembly buffer. Accordingly, various proposals have been presented and investigated for this purpose, see e.g., [3], [4]. The controller may apply different algorithms to different assembly buffers.

In general, burst assembly algorithms may be classified into three major categories, namely time-based, volume-based and hybrid algorithms. In a time-based algorithm, the controller is equipped with a timer. Once a packet arrives to an empty assembly buffer, the timer associated to the buffer is set to a time threshold T_{Th}. This threshold is determined considering maximum delay that a packet can tolerate in the ingress node. Then, as soon as the timer expires, all packets in the buffer are aggregated into a burst and sent out. The timer is deactivated when the buffer is emptied. If length of the burst generated in this way is below a given level L_{min}, padding has to be used to fulfill the minimum burst length requirements. The value of L_{min} is dictated by the network architecture [3], and depends on the ratio between number of data and control channels of WDM links in the network. Specifically, L_{min} has to be selected large enough so as to avoid possible conflicts between

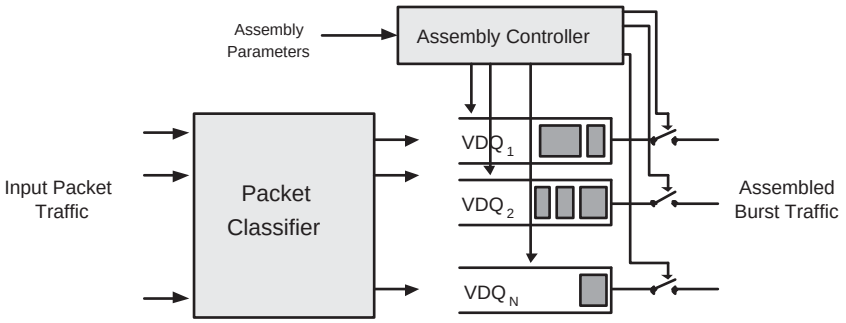

Figure 2.1: Burst assembly unit in an ingress OBS node.

reservation messages of different data bursts over the control channel.

In a volume-based algorithm, the controller checks the aggregate length of packets in the assembly buffer each time a new packet arrives. As soon an the aggregate length exceeds a predefined threshold L_{Th} all packets in the buffer are assembled into a new burst. This threshold should clearly be larger than the minimum burst length requirement of the network and is usually selected with respect to maximization of utilization of resources over the network [3]. Algorithms of this category are usually not recommended for delay-sensitive traffic, because there is no guarantee on the upper-bound delay that a packet may experience in the buffer.

Alternatively, in a hybrid assembly algorithm, the control unit keeps track of both aggregate volume of packets in the buffer and the time elapsed since the first packet has arrived. That is, the timer is set to T_{Th} once a packet arrives and finds the buffer empty, and length of the queue is compared against a length threshold L_{Th} upon each new arrival. Then, a new burst will be generated when either the timer expires or the volume threshold is exceeded. In either case, the timer will be deactivated after the buffer is emptied. In an algorithm of this category load intensity determines which criterion, between time and volume, will be used to generate a new burst at a given time. That is, if the load intensity is below a specific level, a new burst will be generated T_{Th} units of time after the first packet has arrived; however, if the load intensity is heavy enough, bursts of length L_{Th} will be generated back to back so that no packet will encounter maximum assembly delay of T_{Th}. In the former case, it is likely that the timer expires while the aggregate length is less than the minimum burst length requirement. If such does happen, padding has to be used.

Chapter 3

Burst Traffic Modeling

In this chapter, we consider modeling of burst traffic generated by the time-based, volume-based and hybrid assemblers. Consider a single assembly buffer in an OBS ingress node. Let packets arrive to the buffer according to the Poisson process with rate λ, and also packet lengths be exponentially distributed with mean μ^{-1}, i.e., the offered load to the assembler is $\rho = \lambda\mu^{-1}$. In the following analysis we let X and Z be random variables denoting the length and interdeparture time of the bursts leaving the assembler's output, respectively.

3.1 Time-Based Algorithm

Consider a time-based assembly algorithm with the assembly parameter and the minimum burst length requirement set to T_{Th} and L_{min}, respectively . According to Fig. 2, from the instant of time that first packet arrives to the buffer (i.e., t), it takes T_{Th} units of time until a new burst will be generated. Thus, we can assume that each burst contains $(N+1)$ packets where N is the total number of packets that arrive during the interval after the first packet has arrived and before the timer expires. Hence, N would be a random integer that follows the Poisson distribution with mean λT_{Th}. That is,

$$P_T[N = n] = \frac{(\lambda T_{Th})^n}{n!} e^{-\lambda T_{Th}} \quad (n = 0, 1, \ldots). \tag{3.1}$$

The average number of packets per burst is equal to $\lambda T_{Th} + 1$. If total number of packets in a burst is given by k, then length of the burst is sum of k *i.i.d.* exponentially distributed random variables that is known to follow the Erlang distribution [9] with the density function given by

$$f_T(x|k) = \frac{\mu(\mu x)^{k-1}}{(k-1)!} e^{-\mu x}. \tag{3.2}$$

Accordingly, the total volume of the packets in the buffer when the timer expires would have the following density function.

$$f_T(x) = \sum_{n=0}^{\infty} \frac{\mu(\mu x)^n}{n!} e^{-\mu x} . P_T[N = n] \quad (x > 0). \tag{3.3}$$

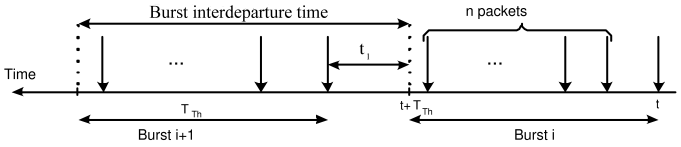

Figure 3.1: Burst formation process under the time-based assembly algorithm.

The probability that the burst length is smaller than L_{min}, i.e., padding is required, would be equal to:

$$\begin{aligned} P_{T,Pd} &= P(X < L_{min}) \\ &= \int_0^{L_{min}} f_T(x)dx \\ &= \sum_{n=0}^{\infty} \frac{\gamma(n+1, \mu L_{min})}{n!} P_T[N = n] \end{aligned} \tag{3.4}$$

where $\gamma(a, x) = \int_0^x t^{a-1}e^{-t}dt$ is the incomplete gamma function. Therefore, the density function of bursts after padding would be given by

$$f_{T,Pd}(x) = P_{T,Pd}\delta(x - L_{min}) + \sum_{n=0}^{\infty} \frac{\mu(\mu x)^n}{n!} e^{-\mu x}.P_T[N = n] \quad (x \geq L_{min}) \tag{3.5}$$

where $\delta(.)$ is the Dirac delta function.

Now, we turn to the burst interdeparture time. As depicted in Fig. 2, by making use of the memoryless property of the packet arrival process, we notice that the interdeparture time between two consecutive bursts is composed of a random value t_1 that is exponentially distributed with mean λ^{-1} plus a deterministic value T_{Th}. The exponential part accounts for the time required until a new packet arrives to the buffer after the previous burst has been sent. In fact, the burst interdeparture time distribution is the same as the distribution of packet interarrival time, shifted by the assembly parameter. Therefore, the density function of burst interdeparture time can be written as the shifted-exponential density given in (3.6).

$$f_T(z) = \lambda e^{-\lambda(z - T_{Th})} \quad (z \geq T_{Th}). \tag{3.6}$$

Note that $f_T(z)$ is independent of μ^{-1}. In addition, while the random part would have a negligible effect on interdeparture times in a heavily loaded assembler, it plays an important role under the light load situations. That is, under the small arrival rates the random part may be comparable to the fixed part T_{Th}.

3.2 Volume-Based Algorithm

Let us now consider traffic generated by a volume-based assembly algorithm with assembly parameter L_{Th}. It is assumed that packet-level traffic with the same characteristics

as described in the previous section arrives to the assembly buffer. Let us first derive the distribution function of burst interdeparture times.

Since both packet lengths and interarrival times are assumed to be exponentially distributed, the problem of finding distribution of burst interdeparture times can be regarded as equivalent to that of finding the burst length distribution for the time-based assembler with the assembly parameter L_{Th}, in which packet arrival process is Poisson with mean μL_{Th} and packet lengths are exponentially distributed with mean λ^{-1}. Therefore, the density function of burst interdeparture times is given by

$$f_L(z) = \sum_{n=0}^{\infty} \frac{\lambda(\lambda z)^n}{n!} e^{-\lambda z}.P_L[N = n] \quad (z > 0). \tag{3.7}$$

where $P_L[N = n]$ is given by

$$P_L[N = n] = \frac{(\mu L_{Th})^n}{n!} e^{-\mu L_{Th}} \quad (n = 0, 1, \ldots). \tag{3.8}$$

Similar to the burst interdeparture times under the time-based assembler, burst length distribution in this case is the same as the distribution of packet length, shifted by the assembly parameter L_{Th}. Thus, the density function of burst length can be written as

$$f_L(x) = \mu e^{-\mu(x - L_{Th})} \ (x \geq L_{Th}). \tag{3.9}$$

In this case, $f(x)$ is independent of the packet arrival rate. Note that in viewing the process from this perspective, N, which has the Poisson distribution, is the total number of arrivals during an interval of length L_{Th}. However, as described in Chapter 2 a new burst will be generated upon arrival of the packet that makes the aggregate length exceed L_{Th}. Thus, the burst interdeparture time should be calculated using an Erlang distribution with $(n + 1)$ phases. This also accounts for the exponential term in calculating the burst length. As a result, the average number of packets per burst in this case is equal to $\mu L_{Th} + 1$.

3.3 Hybrid Algorithm

Consider a hybrid algorithm that its minimum burst length requirement, volume threshold and time threshold are set to L_{min}, L_{Th} and T_{Th}, respectively. In our analysis we assume that $L_{min} < L_{Th}$. As depicted in Fig. 3, once a packet enters an empty assembly buffer, the timer is set to T_{Th}. Therefore, the probability that the timer expires before the volume criterion, i.e., L_{Th}, is met can be expressed as

$$P_T = 1 - P(\tau < T_{Th}) \tag{3.10}$$

where τ is the random variable characterizing the time needed until enough packets arrive to the assembly buffer so that a burst can be generated using the volume criterion. We recall from the volume-based assembler that the conditional density of τ given the number of arrivals follows the Erlang distribution. That is,

$$f(\tau \,|\, n) = \begin{cases} \delta(\tau), & n = 0 \\ \frac{\lambda(\lambda\tau)^{n-1}}{(n-1)!} e^{-\lambda\tau}, & n \geq 1 \end{cases} \tag{3.11}$$

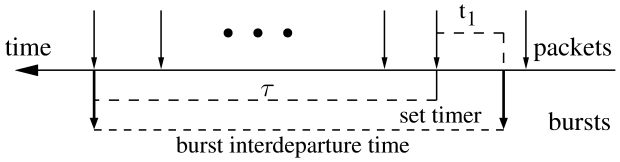

Figure 3.2: Burst formation process under the hybrid assembly algorithm.

where n is the number of arrivals during a period of L_{Th} with the probability function given in (3.8). Note that $n = 0$ is associated with the situation that size of the first arrival to the buffer is larger than the volume threshold. In that case, the timer will be deactivated immediately after it is set, thus τ is equal to zero. For the case of $n \geq 1$ we would have

$$\begin{aligned} P(\tau < T_{Th} \,|\, n) &= \int_0^{T_{Th}} \frac{\lambda(\lambda\tau)^{(n-1)}}{(n-1)!} e^{-\lambda\tau} d\tau \\ &= \frac{\gamma(n, \lambda T_{Th})}{(n-1)!} \quad (n \geq 1). \end{aligned} \tag{3.12}$$

Therefore, P_T can be computed as

$$P_T = 1 - e^{-\mu L_{Th}} - \sum_{n=1}^{\infty} \frac{\gamma(n, \lambda T_{Th})}{(n-1)!} P_L[N = n]. \tag{3.13}$$

Now, let us derive the density of length of the bursts that are generated by the assembly buffer under study. Those bursts that leave the assembly buffer before the timer expires would be larger than L_{Th}, thus their density would be equivalent to that of the bursts generated by the pure volume-based algorithm, as given in (3.9). However, if the timer does expire, the burst length would be smaller than L_{Th}. Therefore, the density function of bursts length before possible padding would be equal to

$$f_H(x) = \begin{cases} P_T f_a(x), & x < L_{Th} \\ (1 - P_T)\mu e^{-\mu(x - L_{Th})}, & x \geq L_{Th} \end{cases} \tag{3.14}$$

where $f_a(x)$ is the density of the bursts length when the timer expires, which is equal to the conditional density of $f_T(x)$, as given by (3.3) , given $(x < L_{Th})$. That is,

$$\begin{aligned} f_a(x) &= f_T(x \,|\, x < L_{Th}) \\ &= \frac{f_T(x)}{K_1} \quad (x < L_{Th}) \end{aligned} \tag{3.15}$$

where K_1 is the normalization factor and can be computed as, (see (3.4))

$$\begin{aligned} K_1 &= P(X < L_{Th}) \\ &= \sum_{n=0}^{\infty} \frac{\gamma(n+1, \mu L_{Th})}{n!} P_T[N = n]. \end{aligned} \tag{3.16}$$

Accordingly,

$$f_a(x) = \frac{\sum_{n=0}^{\infty} \frac{\mu(\mu x)^n}{n!} e^{-\mu x} P_T[N=n]}{\sum_{n=0}^{\infty} \frac{\gamma(n+1,\mu L_{Th})}{n!} P_T[N=n]} \quad (x < L_{Th}). \tag{3.17}$$

To calculate the probability that padding is applied we can write,

$$P_{H,Pd} = P(X < L_{min} \,|\, X < L_{Th}) P(X < L_{Th}) \tag{3.18}$$

where $P(X < L_{Th}) = P_T$ and

$$\begin{aligned} P(X < L_{min} \,|\, X < L_{Th}) &= \int_0^{L_{min}} f_a(x) dx \\ &= \frac{\sum_{n=0}^{\infty} \frac{\gamma(n+1,\mu L_{min})}{n!} P_T[N=n]}{\sum_{n=0}^{\infty} \frac{\gamma(n+1,\mu L_{Th})}{n!} P_T[N=n]}. \end{aligned} \tag{3.19}$$

Finally, the density function of bursts length after padding padding can be expressed as

$$f_{H,Pd}(x) = \begin{cases} P_{H,Pd}\delta(x - L_{min}) + P_T f_a(x), & L_{min} \le x < L_{Th} \\ (1-P_T)\mu e^{-\mu(x-L_{Th})}, & x \ge L_{Th}. \end{cases} \tag{3.20}$$

Now we turn to the bursts interdeparture time. As depicted in Fig. 3 bursts interdeparture time is composed of two parts, namely t_1, which is the time required until a packet arrives after the last burst has been sent out, and τ, which is the time between the first packet arrives and the new burst is generated. That is,

$$Z = t_1 + \tau. \tag{3.21}$$

For those bursts that are released due to the timer expiration, τ would be deterministic and equal to T_{Th}, thus the density of bursts interdeparture time would be equal to that of the pure time-based assembler, as given in (3.6). For other bursts, however, interdeparture time would be equal to sum of t_1 and τ given $\tau < T_{Th}$. Therefore,

$$f_H(z) = P_T \lambda e^{-\lambda(z-T_{Th})} U(z - T_{Th}) + (1-P_T) f_b(z) \quad (z > 0) \tag{3.22}$$

where $U(.)$ is the unit step function. Also, $f_b(z)$ can be calculated by convolving the density functions of t_1 and τ as follows:

$$f_b(z) = f(t_1) * f(\tau \,|\, \tau < T_{Th}). \tag{3.23}$$

Since packet arrivals is Poisson, $f(t_1) = \lambda e^{-\lambda t_1}$ $(t_1 \ge 0)$. Also, from (3.11) and (3.12) we would have

$$f(\tau) = e^{-\mu L_{Th}} \delta(\tau) + \sum_{n=1}^{\infty} \frac{\lambda(\lambda\tau)^{n-1}}{(n-1)!} e^{-\lambda\tau} P_L[N=n]. \tag{3.24}$$

Therefore, we can write,

$$\begin{aligned} f(\tau \,|\, \tau < T_{Th}) &= \frac{f(\tau)}{P(\tau < T_{Th})} \\ &= \frac{e^{-\mu L_{Th}} \delta(\tau) + \sum_{n=1}^{\infty} \frac{\lambda(\lambda\tau)^{n-1}}{(n-1)!} e^{-\lambda\tau} P_L[N=n]}{e^{-\mu L_{Th}} + \sum_{n=1}^{\infty} \frac{\gamma(n,\lambda T_{Th})}{(n-1)!} P_L[N=n]} \quad (\tau < T_{Th}). \end{aligned} \tag{3.25}$$

Substituting (3.25) in (3.23) and solving the convolution yields

$$\begin{aligned} f_b(z) &= f(t_1) * f(\tau \,|\, \tau < T_{Th}) \\ &= \int_0^\infty f_1(z-x).f(x \,|\, x < T_{Th})dx \\ &= \int_0^\infty \lambda e^{-\lambda(z-x)} \frac{e^{-\mu L_{Th}}\delta(x) + \sum_{n=1}^\infty \frac{\lambda(\lambda x)^{n-1}}{(n-1)!} e^{-\lambda x} P_L[N=n]}{e^{-\mu L_{Th}} + \sum_{n=1}^\infty \frac{\gamma(n,\lambda T_{Th})}{(n-1)!} P_L[N=n]} dx \end{aligned} \tag{3.26}$$

In case $z \leq T_{Th}$, then we can write

$$\begin{aligned} f_b(z) &= \frac{\int_0^z \lambda e^{-\lambda(z-x)} e^{-\mu L_{Th}}\delta(x)dx + \sum_{n=1}^\infty (\frac{\lambda^{n+1}e^{-\lambda z}}{(n-1)!} \int_0^z x^{n-1}dx) P_L[N=n]}{e^{-\mu L_{Th}} + \sum_{n=1}^\infty \frac{\gamma(n,\lambda T_{Th})}{(n-1)!} P_L[N=n]} \\ &= \frac{\lambda e^{-\mu L_{Th}} e^{-\lambda z} + \sum_{n=1}^\infty \frac{\lambda(\lambda z)^n}{n!} e^{-\lambda z} P_L[N=n]}{e^{-\mu L_{Th}} + \sum_{n=1}^\infty \frac{\gamma(n,\lambda T_{Th})}{(n-1)!} P_L[N=n]} \quad (z \leq T_{Th}). \end{aligned} \tag{3.27}$$

On the other hand if $z > T_{Th}$, we would have

$$\begin{aligned} f_b(z) &= \frac{\int_0^{T_{Th}} \lambda e^{-\lambda(z-x)} e^{-\mu L_{Th}}\delta(x)dx + \sum_{n=1}^\infty (\frac{\lambda^{n+1}e^{-\lambda z}}{(n-1)!} \int_0^{T_{Th}} x^{n-1}dx) P_L[N=n]}{e^{-\mu L_{Th}} + \sum_{n=1}^\infty \frac{\gamma(n,\lambda T_{Th})}{(n-1)!} P_L[N=n]} \\ &= \frac{\lambda e^{-\mu L_{Th}} e^{-\lambda z} + \sum_{n=1}^\infty \frac{\lambda(\lambda T_{Th})^n}{n!} e^{-\lambda z} P_L[N=n]}{e^{-\mu L_{Th}} + \sum_{n=1}^\infty \frac{\gamma(n,\lambda T_{Th})}{(n-1)!} P_L[N=n]} \quad (z > T_{Th}). \end{aligned} \tag{3.28}$$

Therefore,

$$f_b(z) = \begin{cases} \frac{\lambda e^{-\mu L_{Th}} e^{-\lambda z} + \sum_{n=1}^\infty \frac{\lambda(\lambda z)^n}{n!} e^{-\lambda z} P_L[N=n]}{e^{-\mu L_{Th}} + \sum_{n=1}^\infty \frac{\gamma(n,\lambda T_{Th})}{(n-1)!} P_L[N=n]} & 0 < z \leq T_{Th} \\ \frac{\lambda e^{-\mu L_{Th}} e^{-\lambda z} + \sum_{n=1}^\infty \frac{\lambda(\lambda T_{Th})^n}{n!} e^{-\lambda z} P_L[N=n]}{e^{-\mu L_{Th}} + \sum_{n=1}^\infty \frac{\gamma(n,\lambda T_{Th})}{(n-1)!} P_L[N=n]} & z > T_{Th}. \end{cases} \tag{3.29}$$

Accordingly, bursts interdeparture time can be expressed as

$$f_H(z) = \begin{cases} P_T \lambda e^{-\lambda(z-T_{Th})} + (1-P_T) \frac{\lambda e^{-\mu L_{Th}} e^{-\lambda z} + \sum_{n=1}^\infty \frac{\lambda(\lambda T_{Th})^n}{n!} P_L[N=n] e^{-\lambda z}}{e^{-\mu L_{Th}} + \sum_{n=1}^\infty \frac{\gamma(n,\lambda T_{Th})}{(n-1)!} P_L[N=n]} & z > T_{Th} \\ (1-P_T) \frac{\lambda e^{-\mu L_{Th}} e^{-\lambda z} + \sum_{n=1}^\infty \frac{\lambda(\lambda z)^n}{n!} P_L[N=n] e^{-\lambda z}}{e^{-\mu L_{Th}} + \sum_{n=1}^\infty \frac{\gamma(n,\lambda T_{Th})}{(n-1)!} P_L[N=n]} & 0 < z \leq T_{Th}. \end{cases} \tag{3.30}$$

Eqns. (3.1), (3.8), (3.13), (3.17), (3.18), (3.20) and (3.30) collectively yield exact probability density functions for length and interdeparture time of bursts generated by the hybrid assembly algorithm under study. Now we take some practical considerations into account and approximate P_T and $f_H(z)$ with simpler expressions. In practice, the length threshold L_{Th} is usually much larger than μ^{-1}, so that $e^{-\mu L_{Th}} \simeq 0$. Therefore, P_T can be approximated by

$$P_T \simeq 1 - \sum_{n=1}^\infty \frac{\gamma(n, \lambda T_{Th})}{(n-1)!} P_L[N=n]. \tag{3.31}$$

Simplyfying (3.30) and subsituting P_T with the expression in (3.31), $f_H(z)$ can be approximated by

$$f_H(z) \simeq \begin{cases} (1 - \sum_{n=1}^{\infty} \frac{\gamma(n,\lambda T_{Th})}{(n-1)!} P_L[N=n])\lambda e^{-\lambda(z-T_{Th})} + \sum_{n=1}^{\infty} \frac{\lambda(\lambda T_{Th})^n}{n!} e^{-\lambda z} P_L[N=n] & z > T_{Th} \\ \sum_{n=1}^{\infty} \frac{\lambda(\lambda z)^n}{n!} e^{-\lambda z} P_L[N=n] & 0 < z \leq T_{Th}. \end{cases} \tag{3.32}$$

Chapter 4

Genaration of Burst Traffic in a Discrete Event Simulator

In this chapter we analyze the density functions that were derived in the last chapter in order to develop efficient algorithms that take care of the burst traffic generation process in a discrete event simulation (DES) model. Let us start with the time-based burst assembly. The density function in (3.5) is composed of two parts, where the second part includes the product of two functions namely, the Erlang density function with $(n+1)$ phases and mean $(n+1)\mu^{-1}$ and the Poisson distribution with mean λT_{Th}. Moreover, we notice that the second part in (3.5) constitutes a *convex* combination of an infinite number of Erlang density functions, each with different mean and number of phases. By definition, a density function $f(y)$ is said to be a convex combination of other density functions f_1, f_2... if it can be written as

$$f(y) = \sum_{i=0}^{\infty} f_i(y) p_i \tag{4.1}$$

where $p_i \geq 0$, $\sum_{i=0}^{\infty} p_i = 1$, and each f_i is a density function [8].

An interesting feature of a complex density function that can be expressed in the form of a convex combination of other simple density functions, like that in (3.5), is that its samples can be generated using the *composition* technique [8]. In this technique, each sample of a random variable Y with the density function $f(y)$ is generated by: *i)* generating a positive random integer I such that $P(I = i) = p_i$ where $i = 0, 1, ...,$ and *ii)* returning Y with the density function $f_I(y)$. More specifically, a sample of random variable with the density function given in (3.5) can be easily generated in two steps. First, a positive random integer n is generated using the Poisson distribution with mean λT_{Th}, then a random value is generated using the Erlang distribution with $(n+1)$ phases and mean $(n+1)\mu^{-1}$.

Algorithm. 1 presents the pseudo code of a simple algorithm that exploits concept of the composition technique to generate burst traffic with the bursts length and interdeparture time density functions that are given in (3.5) and (3.6), respectively. Refereing to the algorithm $exp(\alpha)$, $Poisson(\beta)$ and $Erlang(n, n\alpha)$ are exponential random generator with mean α, Poisson random generator with mean β and Erlang random generator with n phases and mean $n\alpha$, respectively. The algorithm works as follows. First, the number of packets in the current burst is determined. Then, length of the current burst is computed using *Erlang*

random generator and taking into account that the length should not be smaller than L_{min}. Finally, interdeparture time is computed by generating an exponentially distributed random variable and adding it to T_{Th}. Following the same approach for the volume-based case, we will

Algorithm 1 Burst traffic generation according to the time-based algorithm with packet arrival rate λ, average packet size μ^{-1} and assembly parameters T_{Th} and L_{min}.

$\lambda, \mu, T_{Th}, L_{min} \leftarrow$ initialize
$k \leftarrow$ number of bursts to be generated
for $i := 1, i \leq k, i++$ **do**
  $n \leftarrow Poisson(\lambda T_{Th})$
  $x \leftarrow Erlang(n+1, (n+1)\mu^{-1})$
  if $x < L_{min}$ **then**
    $x \leftarrow L_{min}$
  end if
  $z \leftarrow T_{Th} + exponential(\lambda^{-1})$
  wait (z)
  generate and send a burst of length x
end for

get the algorithm depicted in Algorithm 2, that works in a similar way that the time-based algorithm does.

To develop an algorithm for generating bursts with the hybrid policy, which is more complicated than the time and volume based cases, we use a combination of the first two algorithms as depicted in Algorithm 3. The algorithm works as follows. In the first step, the value of P_T has to be computed using (3.13)or (3.31). Note that the value of P_T for a given set of input traffic and assembly parameters is fixed, thus it is computed only once at the beginning of the simulation. After having computed t_1 (see Fig. 3), a sample is drawn from a uniform random generator in the range $(0, 1)$. The value of U will then be used to decide which procedure, between time-based and volume-based, has to be followed to generate the current burst. In the former case, care has to be taken so as not to generate a burst whose length is larger than L_{Th}. This is achieved through a conditional probability. Similarly, in the latter case the burst interdeparture time minus t_1 must be smaller than T_{Th}. In either case, the conditional probability is achieved through implementing a while loop.

Algorithm 2 Burst traffic generation according to the volume-based algorithm with packet arrival rate λ, average packet size μ^{-1} and assembly parameter L_{Th}.

$\lambda, \mu, L_{Th} \leftarrow$ initialize
$k \leftarrow$ number of bursts to be generated
for $i := 1, i \leq k, i++$ **do**
  $n \leftarrow Poisson(\mu L_{Th})$
  $z \leftarrow Erlang(n+1, (n+1)\lambda^{-1})$
  $x \leftarrow L_{Th} + exponential(\mu^{-1})$
  wait (z)
  generate and send a burst of length x
end for

Algorithm 3 Burst traffic generation according to the hybrid algorithm with packet arrival rate λ, average packet size μ^{-1} and assembly parameters T_{Th}, L_{Th} and L_{min}.

```
λ, μ, T_Th, L_Th, L_min ← initialize
k ← number of bursts to be generated
P_T ← compute P_T from the model (0 < P_T < 1)
for i := 1, i ≤ k, i++ do
  t_1 ← exponential(λ^-1)
  U ← uniform(0,1)
  if U ≤ P_T {use time-based procedure} then
    x ← L_Th + 1
    while x ≥ L_Th do
      n ← Poisson(λ T_Th)
      x ← Erlang(n + 1, (n + 1)μ^-1)
    end while
    if x < L_min then
      x ← L_min
    end if
    z ← T_Th + t_1
  else {use volume-based procedure}
    z ← T_Th + 1
    while z ≥ T_Th do
      n ← Poisson(μ L_Th)
      if n == 0 then
        z ← 0
      else
        z ← Erlang(n, nλ^-1)
      end if
    end while
    z ← z + t_1
    x ← L_Th + exponential(μ^-1)
  end if
  wait (z)
  generate and send a burst of length x
end for
```

The main advantage of using the presented algorithms over direct implementation of the burst assembly algorithms is simulation speed-up. Specifically, if the assembly algorithm is directly implemented in a simulation model, the model first has to generate a large number of packets, and then assemble them into bursts and this process has to be repeated for every single burst. In other words, it would take $O(MN)$ time to generate N bursts, where M is the average number of packets per burst. This however implies that the more the average number of packets per burst is, the slower the corresponding simulation model works. This is fortunately not the case for our algorithms, in which time needed to generate N bursts is $O(N)$. In the next chapter, we will numerically evaluate the speed-up gain that is achieved using these algorithms.

Chapter 5

Evaluation and Discussion

In this chapter, we numerically study characteristics of traffic generated by the burst assemblers through simulation and analysis. Although the analytical models derived in this work are exact, we have performed simulation experiments for two purposes. First, we use simulation to validate the usage of our analytical models in the realistic case of trimodal packet size distribution for the arrivals as measured in the real Internet. Table I. shows the trimodal packet length distribution that has been used for this purpose [11]. Second, we evaluate the speed-up gain that can be achieved in a simulation experiment, when the algorithms of Chapter 4 are employed. All the simulation that follow results are obtained through explicitly implementing the assembly algorithms in the discrete event simulator OMNeT++ [10]. Input packets to the assemblers are generated according to the Poisson process and their length follows the trimodal distribution as given in Table I. The results of the simulation are then compared with those obtained from the analytical models with the average packet size set to 485.6 Bytes, which is equal to that in the simulation model. The capacity of the traffic flow that feeds the assembly buffer under consideration is assumed to be equal to 100 Mbit/s. We have further decided to present the cumulative distribution functions (CDFs), instead of probability density function, since they are easier to deal with.

Let us start with the time-based burst assembler. The assembly parameter T_{Th} is set to 4 msec. First we evaluate the probability that generated bursts are smaller than a given minimum burst length requirement L_{min}, thus require padding. For a given assembly threshold, the load offered to the assembler as well as L_{min} are the factors that influence this probability. Fig. 4 shows $P_{T,Pd}$ as a function of offered load for different values of L_{min}. As expected, the probability in all cases decreases with load. It is also seen that the results of the simulation match those of analytical models very closely. To analyze the impact of $P_{T,Pd}$ on the traffic characteristics we consider CDF of bursts length and interdeparture time at both light and heavy loads, while keeping L_{min} fixed at 15 KBytes, see Fig. 5. Plots of Fig. 5 clearly illustrates the impact of the padding process on the distribution of bursts size. In fact, depending on the offered load and L_{min}, fixed-size bursts could even dominate the bursts generated by the assembler, see the case with $\rho = 0.3$ in Fig. 5. From the figure it is also evident that the error introduced by the exponential packet length assumption is very narrow. CDFs of bursts interdeparture time for the same system are plotted in Fig. 6. As seen in the figure, the distributions provided by the analytical models exactly match those obtained from simulations. This is not surprising, as from (3.6) it is clear that the

distribution of bursts interdeparture time is independent of the packets length distribution.

Fig. 7 depicts CDF of interdeparture time of the bursts that are generated by the volume-based assembler with the assembly threshold set to 25 KBytes. The results are shown at both light and heavy loads. We observe that as the load increases, the slope of the CDF of bursts interdeparture time decreases. Also, there is a good match between results of analysis and those of simulation, which again justifies the exponential assumption for the packets size in developing the models. Corresponding bursts length distributions are shown in Fig. 8. From (3.9) it is obvious that burst length is independent from packet arrival rate, and hence from the load in this case. CDF in the analytical case is a shifted exponential distribution, whereas it is a shifted-trimodal distribution in case of trimodal packets arrival. The amount of shift in either case is the same and is equal to the assembly threshold. Taking into account that the assembly threshold is always far larger than a single packet size, the difference in the distributions of Fig. 8 can be regarded as negligible.

Let us now consider the hybrid assembly algorithm. The hybird assembler applies a combination of time and volume criteria to decide for generating bursts, thus it is important to see the interplay between these two criteria under different settings. In Fig. 9 values of P_T are plotted as a function of T_{Th} at different loads, with the length threshold L_{Th} set to 25 KBytes. It is seen that as the offered load decreases, P_T becomes a smoother function of L_{Th}. In other words, if the assembler is heavily loaded, then a small change in T_{Th} can greatly affect the fraction of bursts that are generated due to the timer expiration. This can be explained by recalling that in the pure volume-based burst assembler, the range of variations of burst interdeparture time decreases with arrival rate, as depicted in Fig. 7.

Another important metric that has to be considered is the fraction of bursts that undergo padding before transmission, i.e., $P_{H,Pd}$. The value of $P_{H,Pd}$ is quite important, because it is pertinent to the amount of overhead that is introduced by setting of the assembler's parameters. Refereing to (3.18), for a given length and time threshold, $P_{H,Pd}$ depends on the value of L_{min}. Fig. 10 depicts the values of $P_{H,Pd}$ as a function of L_{min} for two different values of T_{Th} at load=0.5 and L_{Th}=25 KBytes. Note that at L_{min}=25 KBytes, $P_{H,Pd}$ is equal to the whole fraction of bursts that are generated due to the timer expiration, i.e., P_T.

In order to study the impact of P_T and $P_{H,Pd}$ on the distributions, two different scenarios are considered, as depicted in Table II. In either case, the offered load and L_{Th} are fixed and are equal to 0.5 and 25 Kbytes, respectively. Table II also shows the values of P_T that are achieved when each of the scenarios are applied. In fact, we are more interested in the settings that result in a moderate value of P_T because in the extreme cases the hybrid assembler collapses either to the time-based or to the volume-based assembler. Distributions of length of the bursts at the output of the hybrid assembler are plotted in Fig. 11. The burst length distributions in Fig. 11 can be divided into two regions, which is also in accordance with the expression derived in (3.20). The first region, i.e., burst size between L_{min} and L_{Th}, is associated with those bursts that are generated when timer is expired; however, the second region, i.e., burst size greater than L_{Th}, is associated with the bursts that are generated because the length threshold is exceeded, thus their size has a very limited variation. The burst size variation in the first region however depends on L_{min}. In fact, the padding process results in generation of bursts of length equal to L_{min}. As seen in Fig. 11, the fraction of such bursts is quite significant in the second scenario where L_{min} is increased to 18 KBytes.

Distributions of bursts interdeparture time corresponding to the scenarios of Table II are

Table 5.1: Trimodal Packet Length Distribution

Length (bytes)	Probability
40	0.5
552	0.3
1500	0.2

shown in Fig. 12. Since the offered load and L_{Th} are fixed in both scenarios, corresponding interdepartures only differ for the times greater than 3.53 msec, which is the smallest time threshold between the two scenarios. Similar to CDF of bursts length, that of interdeparture time can also be divided into two regions, with the border of the regions being at T_{Th}. Moreover, it is seen that as this border is shifted to the left, i.e., the time threshold is decreased, the range of variations of bursts interdeparture time reduces.

Observations of Figs. 9-12 again confirm that the analytical models provide a very accurate estimation for the more realistic case of trimodal packets length distribution. As a matter of fact, in all cases – including time-based and volume-based assemblers – we deal with random variables that are resulting from aggregating a large number of independent trimodal random variables. It can intuitively be explained that in that case the major player is the first moment of the individual packets length and the role of higher moments decreases with the number of aggregated packets.

After having discussed the characteristics of the burst traffic, we now study the speed-up gain that is achieved when using the algorithms of Chapter 4 in a discrete event simulation model. The proposed algorithms are based on exact analytical models that have been developed in Chapter 3 and therefore the traffic that is generated using such algorithms exactly match those of analytical models. Therefore, for the sake of brevity here we only present and discuss the speed-up gain of such algorithms over straightforward implementation of the assembly algorithms in a simulation model.

For this purpose, we generate traces of burst traffic through the direct approach as well as the proposed algorithms and compare the corresponding simulation times with each other. Both algorithms are implemented in OMNeT++, in which all basic random generators required for implementing the algorithms are available as built-in functions. We have measured and compared time needed for each of the approaches in order to generate 10^7 bursts on a PentiumIV 3.2GHZ processor with 1024 MB RAM. Fig. 13 presents the speed-up gain that has been measured versus average number of packets per burst. It is seen that the speed-up gain for the pure time-based and volume-based algorithms could be well beyond an order of magnitude even for short bursts, and it increases with the average number of packets per burst. The reason is that in the direct implementation, as discussed before, simulation time increases with the number of packets per burst. The speed-up gain for the hybrid algorithm is also quite significant; however, it is smaller than that of the first two algorithms. In fact, the difference between speed-up gains of Algorithm 1(2) and Algorithm 3 is that in the latter case, generating the conditional probabilities requires going through a while loop that, in turn, increases the simulation time.

Table 5.2: Parameters used to evaluate the hybrid assembler

	Scenario I	Scenario II
L_{min} (KBytes)	10	18
T_{Th} (msec)	3.95	3.53
P_T	0.5	0.7

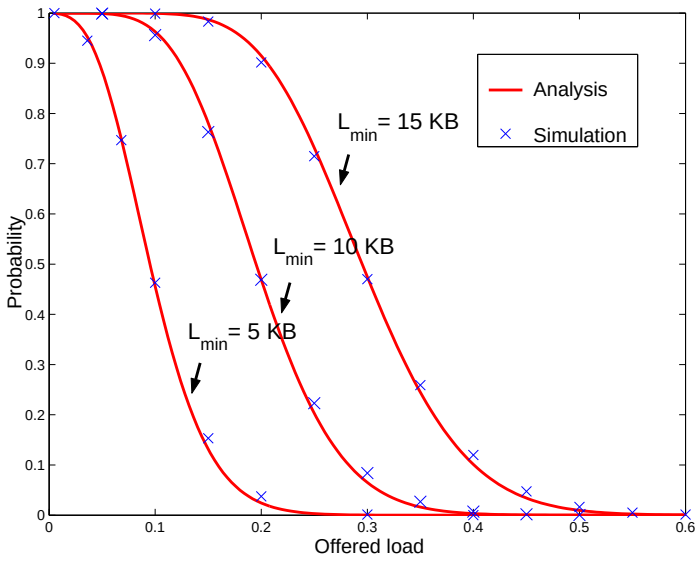

Figure 5.1: Probability that burst length is less than L_{min} as a function of load in the time-based assembler with T_{Th}=4 msec.

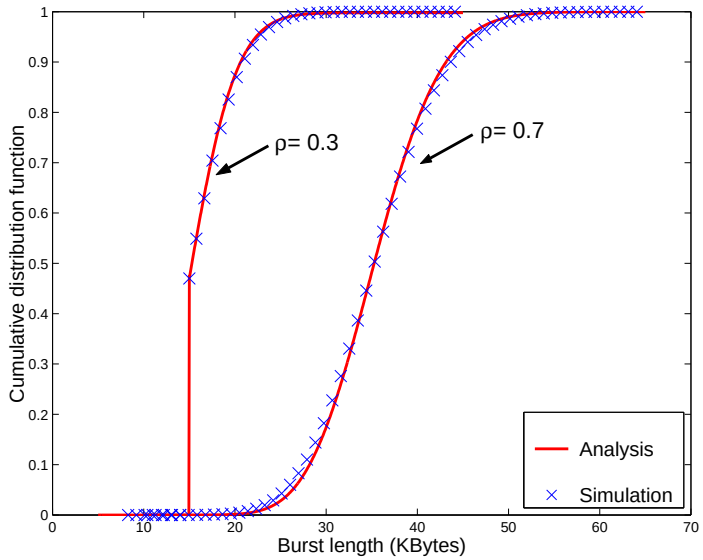

Figure 5.2: CDF of length of the bursts generated by the time-based assembler with T_{Th}=4 msec and L_{min}= 15 KBytes.

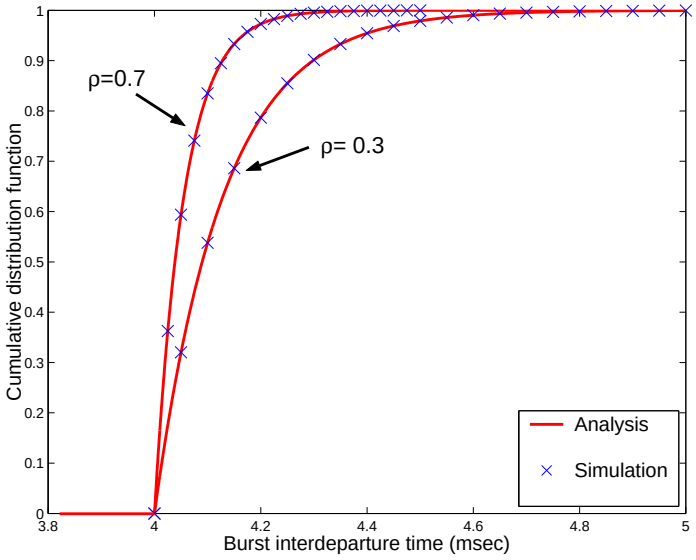

Figure 5.3: CDF of interdeparture time of the bursts generated by the time-based assembler with T_{Th}=4 msec and L_{min}= 15 KBytes.

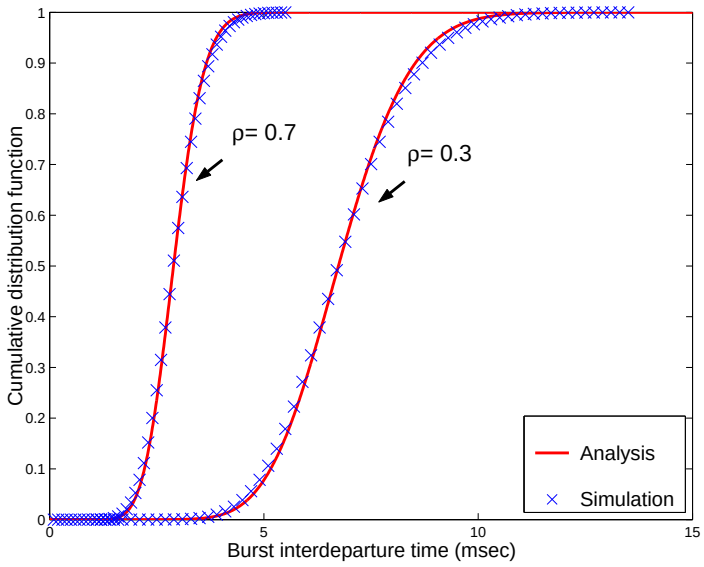

Figure 5.4: CDF of interdeparture time of the bursts generated by the volume-based assembler with L_{Th}= 25 KBytes.

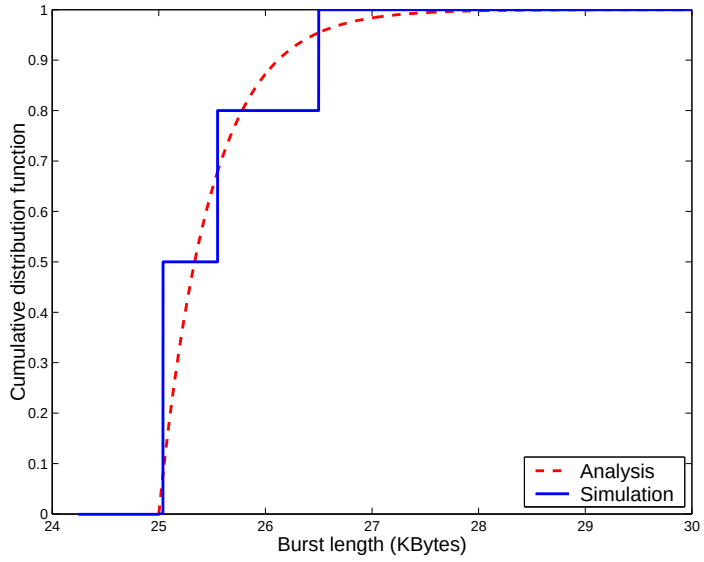

Figure 5.5: CDF of length of the bursts generated by the volume-based assembler with L_{Th}= 25 KBytes.

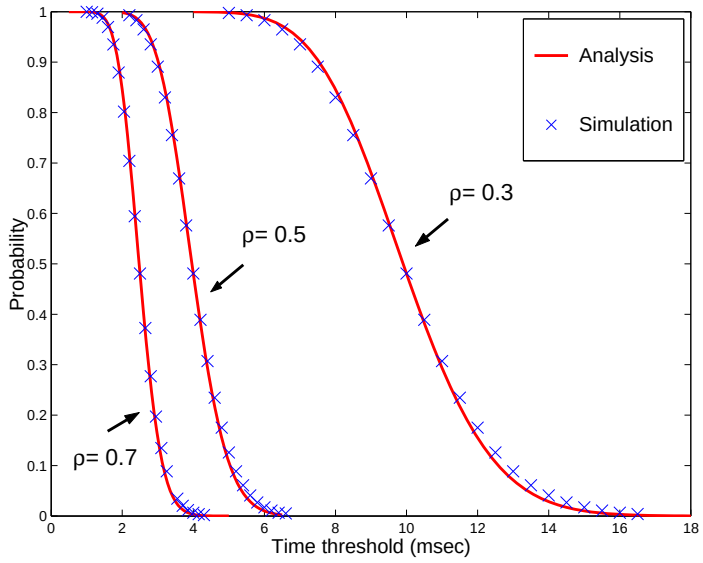

Figure 5.6: Fraction of the bursts that are generated due to the timer expiration in a hybrid assembler with L_{Th}= 25 KBytes.

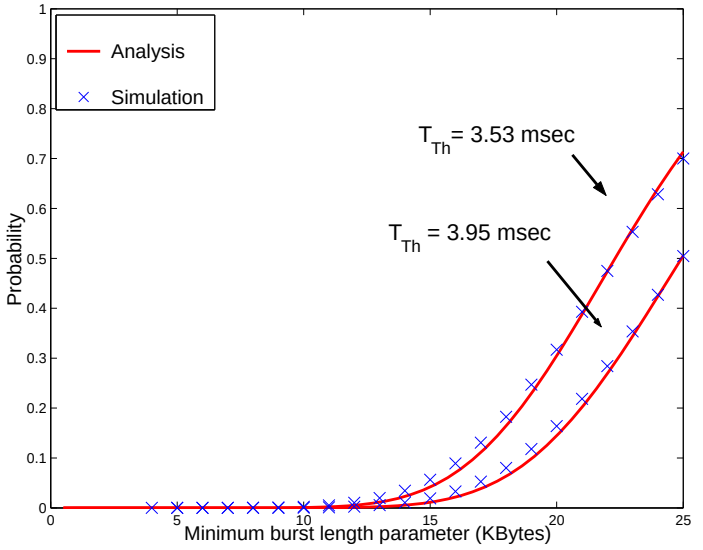

Figure 5.7: Probability that burst length is less than L_{min} in a hybrid assembler with $L_{Th} = 25KBytes$ at load 0.5.

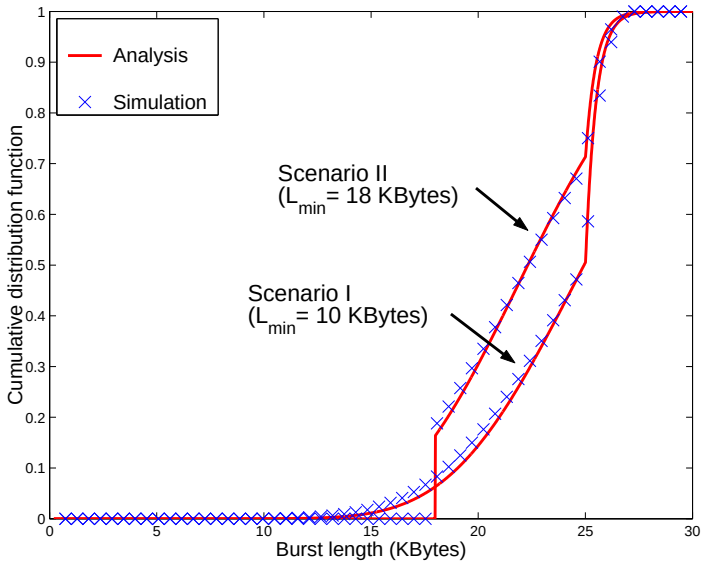

Figure 5.8: CDF of length of the bursts generated by the hybrid assembler with L_{Th}= 25 KBytes at load 0.5.

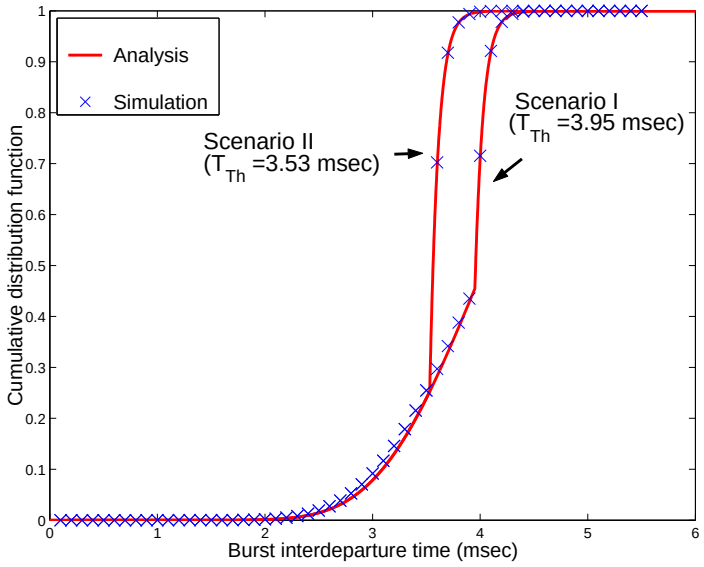

Figure 5.9: CDF of interdeparture time of the bursts generated by the hybrid assembler with L_{Th}= 25 KBytes at load 0.5.

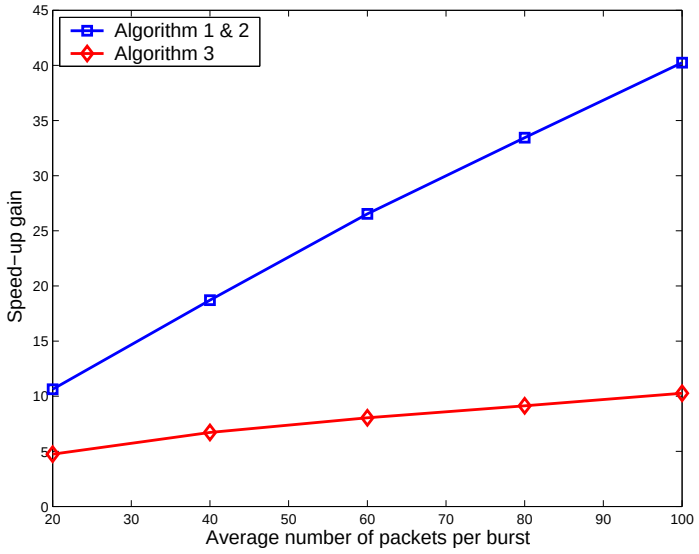

Figure 5.10: Speed-up gain of using the burst generator algorithms over direct simulating of the assembly algorithms for generating 10^7 bursts as a function of average number of packets per burst.

Chapter 6

Conclusions

We have developed an analytical model for the traffic process that leaves the burst assembly buffer in an ingress OBS node. The model includes expressions for distribution of both length and interdeparture time of bursts that are generated under different assembly algorithms. The models are exact for the case that packets length is exponentially distributed; however, through simulation we show that they also provide a very good approximation for the case of trimodal packet length distribution following real measurements in the Internet. As an important application of the models, we present novel traffic generators to be used in discrete event simulation models. The main benefit of the proposed generators is the large speed-up gain which by the way increases with the average number of packets per burst. In the specific example considered in this report the speed-up gain as high as 40 has been observed.

Bibliography

[1] C. Qiao and M. Yoo, "Optical burst switching (OBS)-A new paradigm for an optical internet," *J. High Speed Networks*, vol. 8, no. 1, pp. 69-84, 1999.

[2] J. Turner, "Terabit burst switching," *J. High Speed Networks*, vol. 8, no. 1, pp. 3-16, 1999.

[3] Y. Xiong, M. Vandenhoute, and H. Cankaya, "Control architecture in optical burst-switched wdm networks," *IEEE J. Select. Areas Commun.*, vol. 18, no. 10, pp. 2062-2071, 2000.

[4] X. Yu, J. Li, X. Cao, Y. Chen and C. Qiao, "Traffic statistics and performance evaluation in optical burst switched networks," *IEEE J. Lightwave Technol.*, vol. 22, no. 12, pp. 2722-2738, 2004.

[5] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido, "A nonstationary poisson view of internet traffic," *in Proceedings of IEEE INFOCOM*, March 2004.

[6] K. Laevens, "Traffic characteristics inside optical burst switched networks," *in Proceedings of the Optical Networking and Communications Conference (Opti- Comm)*, 2002.

[7] A. Rostami and A. Wolisz, "Impact of edge traffic aggregation on the performance of FDL-assisted optical core switching nodes," *Proc. IEEE ICC*, 2007, *to appear*.

[8] A. M. Law and W. D. Kelton, *Simulation modeling and analysis.* Mc Graw Hill, Third Edition, 2000.

[9] H. Akimaru and K. Kawashima, *Teletraffic theory and applications.* Springer, Second Edition, 1999.

[10] "Discrete event simulation system OMNeT++." [Online] Available: http://www.omnetpp.org.

[11] C. Fraleigh, *et al.*, "Packet-level traffic measurements from the Sprint IP backbone," *IEEE Network*, vol. 17, no. 6, pp. 6-16, 2003.

[12] A. Rostami and A. Wolisz, "Modeling and fast emulation of burst traffic in optical burst-switched networks," *Proc. IEEE ICC*, vol. 6, pp. 2776-2781, 2006.

[13] X. Mountrouidou and H. Perros, "Characterization of the burst aggregation process in optical burst switching," *Proc. Networking*, Portugal, 2006.