INFERRING CONGESTION FROM DELAY AND LOSS CHARACTERISTICS USING PARAMETERS OF THE THREE-PARAMETER WEIBULL DISTRIBUTION

By

Motlalepula Ramaisa

Submitted in partial fulfillment of the requirements for the degree

Master of Science (Applied Sciences) EEC

In the

Faculty of Engineering, Built Environment and Information Technology

UNIVERSITY OF PRETORIA

May 2005

Abstract

The increase in the number of services and traffic on the Internet amplifies the need for transport protocols to efficiently utilize network resources. One of the main problems that arise due to increased traffic is congestion. Due to the Internet's vast size, its rapid growth and heterogeneous nature, the dynamics of the Internet are constantly changing, which makes classical congestion control mechanisms inefficient. This research focused on the end-to-end delay and packet loss characteristics, with regard to their correlation with network congestion.

The first part of the work studied the end-to-end internet dynamics. Delay and loss data was gathered from 22 local and overseas sites. This data was analyzed to determine how congestion affects the behavior of these characteristics. Results showed that none of the metrics, including the round-trip time (RTT), loss distance and loss length, could be used in isolation to accurately detect congestion. Other factors which can affect these behaviors, such as distance, were investigated as well.

The delay and packet loss characteristics were modeled on the three-parameter Weibull distribution. It was found that for delay, the shape parameter, β, is less than 2.6 when there is no congestion. A value of β greater than 3.7 was typically found for a highly congested network path. For moderately congested network path, the value was normally between 2.6 and 3.7. For loss, β is always less than 2.6, but the scaling parameter, η, decreases with increasing congestion for loss distance. For loss length, η increases with increasing congestion. This suggests that η captures the fact that loss events get bursty with increasing congestion. It was observed that the burst sizes are usually small. A simulation framework for modeling delay and loss as measures for congestion was implemented using a Markov chain and validated.

This dissertation suggests that analysis of the Weibull parameter values computed for both packet delay and loss, and the bursty nature of loss, can offer some useful indication regarding the congestion state of the network path.

TABLE OF CONTENTS

List of figures

1. INTRODUCTION

1.1 Background and Motivation

In order for a transport protocol to efficiently utilize network resources it should have some form of mechanism to monitor the network status and have a means of dynamically adapting to cater for those conditions [1]. A thorough and up-to-date knowledge of the current network status puts the transport protocol in a better position to efficiently utilize network resources [2]. The aim is to transmit as much data as possible without overloading the network, and not waiting for long to get feedback concerning status changes on the network. However, since the Internet does not offer any explicit information regarding network paths, the protocol has to take its own measurements and utilize them. The most widely used measures of network status are delay and loss parameters. In Transmission Control Protocol (TCP) and other similar transport protocols, delay is used to compute a suitable Retransmission Timeout, whereas loss events are taken to be a signal of congestion [3] [4]. In a number of recent and more advanced variants of TCP, both delay and loss parameters are used to augment each other in congestion control.

Due to the rapid growth of the Internet and the increase in applications and services, the dynamics of the Internet are changing, making classical congestion and flow control mechanisms inefficient. This is particularly true for high bandwidth-delay and wireless networks. In order to develop more advanced congestion and flow control mechanisms, there is need to execute advanced studies on characterization of delay and loss parameters and to develop more representative simulation models.

1.2 Background information

An increase in round-trip time (RTT) values and packet loss can signal possible congestion [3]. However, some empirical studies showed that there seems to be a weak correlation between RTT values and congestion [5] [6]. Therefore the information in RTT samples cannot on its own be used to reliably predict congestion or future packet loss and cannot be used to improve throughput. While it is not necessarily true that packet loss will result in an increase in delay, it has been observed that loss is often preceded by high delays [7]. This means that even if we have high delays, it is only when we observe loss that we can ascertain the extent of

the congestion. Moreover, when the network load is high the correlation between high delays and losses is substantial.

Studies on RTT distributions show that there is considerable spatial and temporal variation in RTT samples [8]. The distributions are dominated by the mode and are skewed with large differences between the mean and the mode. It has been shown that the distributions tend to change slowly. The distributions tend to take several forms, including Poisson's [9] but with brief periods of high delay bursts, Pareto and Gamma distribution [8], depending on the state of congestion [10]. For example, a non-congested network typically has an asymmetric delay distribution best modeled on the Pareto distribution [10]. When there is no loss, the distribution is unimodal with inherent RTT appearing often. As the level of congestion increases, the shape becomes a Gamma distribution. Link characteristics also have an effect on the RTT distribution [10].

1.3 Objectives and Problem Statement

Initially, the objective of the research was to investigate the dynamic characteristics of the Internet in South Africa in terms of delay and loss parameter distributions. In the course of the research, it became clear that any characteristics exhibited by delay and loss parameters are reflected in the congestion status. The research then focused on the study of the distributions of delay and loss parameters with a view of investigating correlations with congestion states.

Specifically, the following issues are addressed:

- Characterization and modeling of end-to-end delay and packet loss parameters
- Investigating the effects of distance and geographical spread on the end-to-end delay and loss measurements
- Inference of congestion based on the delay and loss parameters

1.4 Methodology

The procedure involved three phases. The first phase was to collect data from Pretoria using the Internet Control Message Protocol (ICMP) echo probes. The sites for data collection were selected to cover the geographic spread across the country and to reflect different usage patterns. Some overseas sites were also selected.

The data collected was analyzed offline in the second phase. It is plotted to determine the kind of distributions followed by RTT and loss parameters. The distributions' behaviors were gauged in relation to congestion. In this phase, we attempted to establish a mechanism that could be used to model delay and loss characteristics of the internet.

In the third phase a simulation model was implemented in NS2 using information gathered in previous phases.

1.5 Contribution

Various studies have shown that RTT and loss must complement each other to be used as a basis form congestion control. It is also clear tat using absolute values of RTT and loss rate is not the most effective way of inferring congestion status. This research attempts to directly link parameters of the distributions to the various congestion states. The distributions themselves are based on a three-parameter distribution – the Weibull distribution.

1.6 Outline of the dissertation

The next Chapter presents the theory related to end to end delay and loss, outlining various methods used to model these parameters. In chapter 3 we apply this theory to the sample data obtained in our study for comparison. We then move on to apply our proposed model on the data in chapter 4. In chapter 5 we give a discussion of the results as well as a conclusion to the study.

2. THEORETICAL BACKGROUND AND MODELING FRAMEWORK

2.1 Introduction

A network is said to be congested when the amount of traffic injected into the network links exceeds the available transmission capacity of the network resources. Packets that cannot be routed are queued in the router's buffer until resources become available for them to be sent, or the buffers fill up and all incoming packets are discarded [1]. This situation overloads the network and if not corrected, might result in network collapse – a situation where the network load is very high, but throughput is reduced to almost zero [11] [12]. Two of the fundamental characteristics of a packet switched network which are affected by network conditions are end-to-end delay and packet loss [13]. Since the Internet is a "best effort" network which does not provide any guarantee on delay and loss [7], transport protocols typically use these parameters to monitor the network in order to determine the congestion status. Selection of these two parameters seems appropriate, as they are greatly influenced by buffering within the network [1].

In this chapter we will look at the typical behaviours of both delay and packet loss, and how they have been modelled and characterised as a result of previous studies. We will then explain how they are used by different flavours of TCP in their congestion detection mechanisms.

2.2 Measurement methods

In order to analyse and model the characteristics, we first need to find a measurement tool to gather the data. We classify measurement tools on the internet as those that take round-trip, or two-way measurements, and those that can take one-way measurements. A brief discussion of these methods and their examples follows.

2.2.1 Two-way measurements

The most commonly used method for measuring delay and packet loss characteristics are the ICMP tools, *ping* and *traceroute*, which send probe packets to the destination and gather information about the network path from the response [14]. The main reason for the popularity of these tools, especially *ping*, is that the *ICMP Echo* and *ICMP Time Exceeded* services on

which they rely are already widely deployed on the Internet [14]. This means that they can be used across a wide spectrum of hosts on the Internet without requiring any special installation. Another advantage of these tools is that their results can easily be interpreted.

The drawbacks for the ICMP tools are as follows:

1. ICMP filtering: due to widespread abuse of the ICMP services, some operating systems, e.g. Solaris, limit the rate of ICMP responses from hosts. The effect of this is that the packet loss rates reported by *ping* are falsely overestimated [14]. A good number of sites take this action a bit further and completely bar all ICMP packets altogether, making it impossible to gather data from these sites. Another problem encountered is that some firewalls respond on behalf of the hosts behind them, giving inaccurate end-to-end measurement readings.

2. Loss Asymmetry: it is impossible for the source to deduce whether a packet was lost en route to the destination, or its corresponding response was lost in the reverse direction. Empirical studies have shown that loss of a packet and loss of the ACK in the reverse direction are sometimes independent of each other [15], so the loss rate in the forward path is not the same as the loss rate in the return path. In fact, one empirical study quantified the difference between these rates for Web traffic and found that on average the reverse loss rate (1.5%) was more than twice that in the forward direction (0.7%) [14]. The ratio grew even larger during busy periods. The explanation given for this phenomenon was that Web servers normally send significantly more data than they receive, but the connections are fully-duplex, putting the path *from* the servers under more strain and more susceptible to bottlenecks, which eventually lead to packet drops.

3. Path asymmetry: the path from the source to destination is not always the path taken in the reverse direction [5].

2.2.2 One-way measurements

One-way measurements are more valuable as they give information as to the direction in which the observed behaviour occurs, unlike two-way measurements which offer the aggregate measurements. For example, by noting the exact arrival and departure time of packets at both hosts, it is trivial to isolate the different components which contribute to the round trip delay – an exercise that is quite difficult, if not impossible to do using two-way

measurements. Knowing the direction in which the behaviour occurs makes it easier for us to determine the exact cause of that particular behaviour. Some tools have been implemented that measure the one-way network characteristics with great detail and accuracy. Among them we single out the National Internet Measurement Infrastructure (NIMI) [48] and Surveyor [16]. These initiatives put measurement software at both the source and destination, and are therefore able to correctly calculate and measure characteristics in both directions.

The problem with such approaches is that they are not widely deployed, and they require installation of software at both the source and destination, which is difficult and thus severely limit the number of potential hosts that can be used in measurements [14]. A TCP-based measurement tool called STING was developed at the University of Washington to accurately measure packet loss in both directions between two hosts [14]. To achieve this, it utilises TCP properties, such as the error control and the fast retransmit mechanisms. This effectively eliminates the problem encountered by other one-way measurement tools – the necessity to install additional software at the destination host. The tool is quite difficult to implement, however, and there are inhibiting portability issues.

2.3 Packet delay

For TCP to offer reliable data transmission between the source and destination, the source sends a packet to the destination, which in turn sends back an acknowledgement (ACK), informing the sender that the data was indeed transferred successfully. The time taken for a packet to be transmitted from the source to the destination is defined as the one-way transit time (OTT) and is given by:

$$OTT = \sum_{1}^{n} (\mathrm{T_n} + \mathrm{P_n} + \mathrm{D_n} + \mathrm{Q_n}) \qquad \textbf{(2.1)}$$

where T is the transmission delay, P is the propagation delay, D is the processing delay, Q is the queuing delay, n is the number of nodes from the source to the destination. Transmission delay is the time taken to transmit the whole packet along the path and depends on the link bandwidth and the packet size [17] [18]. Propagation delay represents the time taken to propagate a bit along a communication link [19], which depends on the link length and the speed of propagation. This delay is related to the fact that nothing can travel faster than the speed of light and is thus not dependent of the traffic along the network. Processing delay is the time taken to process each packet and prepare it for transmission, whereas queuing delay is

the time a packet spends queued in the router's buffer before being forwarded to the destination or next hop. The transmission and propagation delay are determined by the underlying network infrastructure and have minimal variation. The minimum time for the end-to-end transmission of the packet, the inherent delay, consists mainly of these two deterministic components and is normally very close to their sum [10]. This value depends more on the network topology than the network load [17]. The variation in the overall delay arises primarily from the processing and queuing delay. Of these two, queuing delay is the one that is influenced most by network conditions; hence it contributes most to this variation.

A measure of delay used in TCP and other similar transport protocols for congestion control and retransmission mechanisms is the RTT. RTT values are taken per segment and segment RTT is defined as [20]

$$segmentRTT = F_{delay} + D_{delay} + A_{delay} \qquad \textbf{(2.2)}$$

where F_{delay} is OTT arising from segment travelling from the source to the destination. D_{delay} is processing delay at the destination. A_{delay} is OTT arising from ACK travelling back from destination to source.

It has been found that delay values varied enormously across large ranges of values [21]. These delays form bursts of higher values following stretches of lower values, but the burst lengths have no common time scale and vary widely. The study showed that when the RTT is high, then there is a greater chance that the next RTT value will be high as well, which indicates some temporal dependency in network delay. An empirical study by Acharya and Saltz into Internet round trip delays also found that there are both temporal and spatial variations in RTT [8]. Delay behaviour varies significantly depending on the time of day as well as the physical or geographical distance between the communicating hosts. RTT values were seen to follow a unimodal asymmetrical distribution which is skewed towards the right. Characteristics of this distribution tended to vary significantly over time. Specifically, the distributions changed slowly over time and possessed a long heavy right tail which decayed sub-exponentially, with most of the values tightly grouped around the mode. This observation, along with the fact that the skewed distribution indicates that there is some difference between the mode and the mean, makes the mode a good characteristic value for the RTT distribution. They also observed that the inherent RTT occurs frequently, more so in cases where the mode is low. A small jitter was also noted in RTT observations. Two of their conclusions were later

shown to be correct only when the packet loss rate is small by Bi, et al [10]. Bi, et al showed that although the distribution is indeed unimodal and grouped around the mode when the loss rate was negligible, as soon as the loss rate increased the distribution becomes less clustered around the mode. As the distribution becomes scattered along the x-axis, more than one mode emerges. Also, the occurrence of inherent RTT seemed to be inversely proportional to the loss rate. This phenomenon could be due mainly to lack of capacity at bottleneck links [8] and frequent route changes as the routers try to balance the load and avoid congestion.

Another empirical study of the end-to-end delay over short time periods and multiple paths was carried out in [22]. It showed that the round trip delay could be modelled by a constant plus gamma distribution, with parameters of the gamma distribution depending on both the time of day and path taken. An analysis of the mean delays showed a diurnal cycle, which according to Bolot [13] implies the presence of "a base congestion level which changes slowly with time." The notion of delay distribution being "gamma-like" was supported by another empirical study, which found that about 84% of delay distributions studied followed a typical gamma distribution with a heavy tail [19]. 6% of the distributions displayed what is described as "gamma-like distribution with a Gaussian or triangle lob." A visual inspection of the example distribution graph shows that it has strikingly similar features to the effect described by Bi, et al [10]. This behaviour will be displayed later in Figure 3.4c. Bi, et al showed that the delay distribution is not always unimodal. Although there is no mention of packet loss or congestion in [19], it can be speculated that 6% of the distributions were taken during periods of substantial packet losses or congestion. These samples were taken between 9am and 9pm, an interval considered to be the busiest and most congested period of the day.

The effect of packet loss on the delay distribution is discussed in section 2.5.

It is also known that Internet traffic is self-similar – that is, it behaves the same way regardless of the time scale at which it is viewed. It was shown in [5] that contrary to expectations, delay values on their own do not indicate congestion. An increase in the delay could be due to other possible reasons. For example, the primary route could be broken down, forcing the packets to be rerouted via a secondary path. This new path could be longer, have more hops, or it could consist of slower links than the primary path, resulting in higher propagation and queuing delays at the intermediary nodes. In another scenario, there could simply be an influx of packets into the network, but not enough to congest it. The more packets sent by a node, the greater the transmission time, hence the greater the delay. These two situations will obviously

result in higher RTT values, without necessarily congesting the network. Therefore, mean RTT values alone do not give conclusive insight into the congestion state of the network.

2.4 Packet Loss

In practice, a loss event is assumed to have occurred when the segment RTT is infinity. Packet loss normally occurs as a result of one of two things: buffer overflows at intermediary nodes (which implies congestion), and bit errors, such as those caused by noise [23]. Except for high bandwidth and wireless networks, the latter rarely occurs. The detection of packet loss, therefore, is normally used as a feedback mechanism to indicate congestion on the network, which can be used by transport protocols to prompt the sender to adapt its sending rate [3]. TCP and its variants assume that all losses are due to congestion [6] [24], and adapt their transmission rate accordingly. When congestion is detected, not only does the sender have to reduce its sending rate in order not to perpetuate the congestion, it also has to retransmit the lost packet. Such losses are referred to as "congestive losses" [11].

In his empirical study to determine if packet losses are well-modelled as independent, Bolot used conditional probabilities [13]. He used two probabilities: the probability that the next packet will be lost, p, and the conditional probability, c, which is the probability that if a packet is lost, then the next packet will also be lost. His study of the two probabilities in relation to the time interval between the transmission times of two successive packets, δ, showed some interesting, and unexpected behaviours. Results showed that c increased as δ decreases, and that if a packet was dropped at time t due to buffer overflow, then any packet that arrives within time $t + \delta$ will also be lost if δ is less than the time required to reduce the overflowing queue in the buffers. This showed that losses are not entirely independent of each other. It was, however, also noted that c approached p as δ increases, indicating that the loss correlations are short-lived. The conclusion was that as long as packet probes used a small fraction of the available bandwidth, then loss of these probes was random. This fraction was later quantified as 10% by Paxson in [15].

In his study of packet loss, Paxson [15] analysed the differences between loss of data packets from the source to the destination, and loss of ACK segments in the reverse path. Since TCP adapts its sending rate after experiencing packet loss, it was decided that ACK losses are more representative of the overall Internet loss patterns since they do not adapt to network conditions. It was discovered that by observing a path which has losses, the future loss status

of that path can be predicted. Likewise, if a path has no loss, then it can be predicted that the path is not likely to sustain packet losses in the near future – which can be a matter of hours. Predicting the actual loss rate, however, is limited to only a short period of time, normally just minutes. Another empirical study further reduced the time during which what happens to one packet depends on what happens to another to 1 second or even less [25]. Even though the correlation is short-lived, it is enough to suggest that loss events cannot be accurately modelled as independent. Similarly, Arai, et al also found that the independent loss model did not fit their measurements and thus could not explain the characteristics very well [23].

Another empirical study showed that packet loss events are normally bursty, with single-packet loss occurring infrequently [26]. This means that there is some dependency between lost packets in that during a loss event, a burst or a group of packets is more likely to be lost than a single packet. In other words, if packet n is lost then there is a high probability that packet $n+1$ will be lost as well [21]. Studies of packet loss burst length have found that loss lengths tend to span several orders of magnitude, showing great variability [21] and their distribution graph had a heavy upper tail, closely resembling a Pareto distribution [15] [27] [26]. The tail suggests that most losses occur in short bursts which can be attributed to queue overflow at routers, whereas the longer bursts are due to infrequent events such as router outages [26].

Other than loss event occurrence, packet loss can be characterised by loss distance and loss length [28]. Loss distance refers to the number of packets between consecutive loss events, whereas loss length is the number of packets lost per event. From the loss distance, we can observe how frequently loss events occur. A small loss distance indicates that losses are close to each other and occur after short intervals, which might suggest possible congestion. The loss length, on the other hand, gives an indication of the nature of the loss event – whether it is a single-packet or a bursty loss. Single-packet losses are normally due to link or transmission errors, whereas bursty losses are likely to be congestion-related. This was also observed in [23], whereby the loss length tended to be high when the packet loss rate is also high.

2.5 Correlation between Packet loss and delay

Empirical studies have found that delays and losses share many properties and are correlated, especially when network load is high [11]. Moon, et al went even further to claim that congestive losses and packet delays are mainly due to buffering within the network [7].

Packets travelling on the network are queued at the buffers while waiting to be routed to the next hop, which is closer to their destination. Unless these packets are forwarded in time or the buffers are huge in size, the buffers might be filled to capacity, resulting in buffer overflows and the discarding of all arriving packets [49]. In this scenario, when the packets are queued in the buffer, their queuing delay increases, so the subsequent (congestive) loss is indeed preceded by high RTT values. Also, even after packets are discarded, the buffer is still full as some packets are still queued, and these will also suffer from a high delay. As congestion is alleviated, due to adaptation at the TCP sender, for example, the buffered packets begin to be popped from the queue, so the queue becomes smaller, resulting in a decrease in the RTT values. Thus, we expect to see an increase in delay prior to congestive loss; the delay remains high for a short period after the loss before decreasing.

However, the reverse is not true; when a loss burst occurs, delays just prior and after this loss burst are *likely* to be high, whereas losses do not necessarily follow when delays are high [1]. This most likely occurs because routers have large buffers that can hold all incoming packets until congestion clears, causing high delays without dropping packets. It can therefore be ventured that although a series of high delay values *might* indicate congestion, packets are not always lost, especially if routers have enough buffer space to handle the influx of arriving packets [1]. This was shown in [6], where it was also found that RTT information is not sufficient to reliably predict packet losses. Moreover, the increase in RTT prior to packet loss is usually short-lived. The level of correlation between increase in RTT and packet loss was found to be weak by empirical studies, which found that on average only 7-18% of observed loss events were preceded by a detectable increase in RTT [5]. In his study of the correlation between packet delay variation and loss, Paxson, too, concluded that packet loss is weakly correlated to rises in one-way delay [11]. Apart from routers with large buffer space, he suggested that this could be due to the fact that the end-to-end delay reflects the summing of a number of smaller variations into a single, cumulative variation.

Looking at the distributions might present a higher level of correlation between these characteristics. A study carried out using hosts situated in China into the correlation between delay and loss discovered that an increase in the loss rate has an effect on the delay distribution [10]. Bi, et al showed that when there is little or no packet loss, delay follows a Pareto distribution, with most of the distribution clustered around the mode. However, as the loss rate increases the distribution becomes more detached from the mode and the shape

changes from being unimodal, to having more than one mode. Another interesting observation is that the occurrence of inherent RTT also tends to diminish with increasing loss rate, a behaviour which can be explained if one goes back to the discussion about congestive losses. Increase in loss rate is normally due to congestive losses, which are by definition due to buffer overflows, meaning that packets are queued at the routers, hence there is a high queuing delay. This ensures that inherent RTT occurs less frequently, if at all, during congestive losses.

2.6 Weibull Distribution

As stated earlier, a number of empirical studies have shown that the packet delays follow the Pareto [27] [10] as well as the Gamma [19] distribution depending on the current network conditions. The intuition in our study was to use the Weibull distribution to model packet delay.

The Weibull distribution is a three-parameter lifetime distribution which, according to [29] is "one of the most widely-used lifetime distributions in reliability engineering." The reason that makes this particular distribution valuable is that it is versatile, i.e. its shape can vary so that it displays characteristics of other distributions. To make it even more pertinent to our study, it is noted that among the distributions that can be emulated using the Weibull we have *Pareto*, *Gamma* as well as another distribution that is not widely mentioned in literature but which we found occurs in certain cases – a *left-tailed* distribution. Without doubt, it would be more convenient to use one distribution, rather than three, to model delay and loss characteristics. Our claim is that the delay distribution is not constant – as network characteristics change, so does the shape of the distribution, which explains the presence of the three different distributions. If this is correct, then some useful information concerning the network status can be inferred from the distribution shape. The important question we need to answer, therefore, is: under what conditions does the shape follow one particular distribution?

The three parameters of the Weibull distribution are as follows:

1. β, the slope or shape parameter, which defines the shape of the distribution. This is the parameter that controls the versatility property of the distribution.

2. η, the scale parameter, which defines how stretched out the distribution is. It corresponds to the range covered by the distribution along the x-axis.

3. γ, the location parameter, which defines the starting point or origin of the distribution.

The distribution normally occurs in one of three different forms. In its most general case, the Weibull distribution's probability density function (*pdf*) contains all three parameters, and is given by:

$$f(T) = \frac{\beta}{\eta}\left(\frac{T-\gamma}{\eta}\right)^{\beta-1} e^{-\left(\frac{T-\gamma}{\eta}\right)^{\beta}} \qquad (2.3)$$

where f(T)≥0, T≥0 or γ, β>0, η > 0, -∞< γ < ∞.

The second case, the Two-parameter Weibull distribution, is obtained when the location parameter, $\gamma = 0$. Substituting $\gamma = 0$ into (2.3), the *pdf* becomes

$$f(T) = \frac{\beta}{\eta}\left(\frac{T}{\eta}\right)^{\beta-1} e^{-\left(\frac{T}{\eta}\right)^{\beta}} \qquad (2.4)$$

The third special case of the distribution, the One-parameter Weibull distribution scenario, occurs when $\gamma = 0$ and β = C, a constant which is already known. The *pdf* in such a case is given by substituting those values into (2.3) to obtain

$$f(T) = \frac{C}{\eta}\left(\frac{T}{\eta}\right)^{C-1} e^{-\left(\frac{T}{\eta}\right)^{c}} \qquad (2.5)$$

The statistical properties of the Weibull distribution are:

The Mean, $\overline{T}$ which is given by the following equation:

$$\overline{T} = \gamma + \eta \cdot \Gamma\left(\frac{1}{\beta} + 1\right) \qquad (2.6)$$

where Γ(α) denotes the gamma function given by

$$\Gamma(\alpha) = \int_0^{\infty} t^{\alpha-1} e^{-t}\, dt \qquad (2.7)$$

The median, $\breve{T}$ is given by

$$\breve{T} = \gamma + \eta(\ln 2)^{\frac{1}{\beta}} \qquad (2.8)$$

The mode, $\tilde{T}$ is given by

$$\widetilde{T} = \gamma + \eta\left(1 - \frac{1}{\beta}\right)^{\frac{1}{\beta}} \qquad (2.9)$$

And the standard deviation, σ_T, is given by

$$\sigma_T = \eta \cdot \sqrt{\Gamma\left(\frac{2}{\beta}+1\right) - \Gamma\left(\frac{1}{\beta}+1\right)^2} \qquad (2.10)$$

Of particular interest for our study is the general case of the distribution, the Three-parameter Weibull distribution. As mentioned earlier, the distribution is very versatile, and can take on characteristics of other distributions – hence it can be used to model a wide variety of behaviours. The two parameters that have an interesting effect on the shape of the probability density function (*pdf*) are the shape and scale parameters. To illustrate the effect of β on the *pdf*, we kept η constant (at η =100) and varied β (values β= 1, 3 and 5). The results are shown in Figure 2.1. Figure 2.2 illustrates the effect of varying η (values η = 10, 50 and 100) on the pdf, while β is kept constant (at β = 3).

As documented in [29], some of the characteristics of the *pdf* are that for $\beta > 1$,

1. f(T) = 0 at T = 0 or T = γ

2. f(T) increases as T approaches the mode, and then decreases thereafter.

3. for $\beta \leq 2.6$, the distribution has a right tail (i.e. it's skewed positively) and looks like the Pareto distribution, whose *pdf* is given by

$$f(T) = \frac{\beta\gamma^{\beta}}{T^{\beta+1}} \qquad (2.11)$$

Where β is the shape parameter, γ is the location parameter and $\beta > 0$, b>0 and T ≥ b. When $\beta \geq 3.7$, then the plot is skewed to the left. For the intermediary range, i.e. $2.6 < \beta < 3.7$, the level of skewness is reduced, and the shape somewhat resembles the Gamma distribution, whose pdf is given by

$$f(T) = \frac{\left(\frac{T-\gamma}{\eta}\right)^{\beta-1} \exp\left(-\frac{T-\gamma}{\eta}\right)}{\eta\Gamma(\beta)} \qquad (2.12)$$

where β is the shape parameter, η is the scaling parameter, γ is the location parameter and $T \geq \gamma, \beta > 0$ and $\eta > 0$.

After looking at the different characteristics of the Weibull distribution, we note that the Three-parameter Weibull distribution can be used to model packet delay and loss. The Weibull probability density function graph takes different shapes depending on the value of the shape parameter, β.

For $\beta \leq 2.6$, the distribution is skewed to the right. From literature [10], it was noted that the distribution for a non-congested network is skewed to the right, specifically when packet loss (and thus congestion) is negligible. For the intermediary range, $2.6 < \beta < 3.7$, the shape almost resembles the normal distribution.

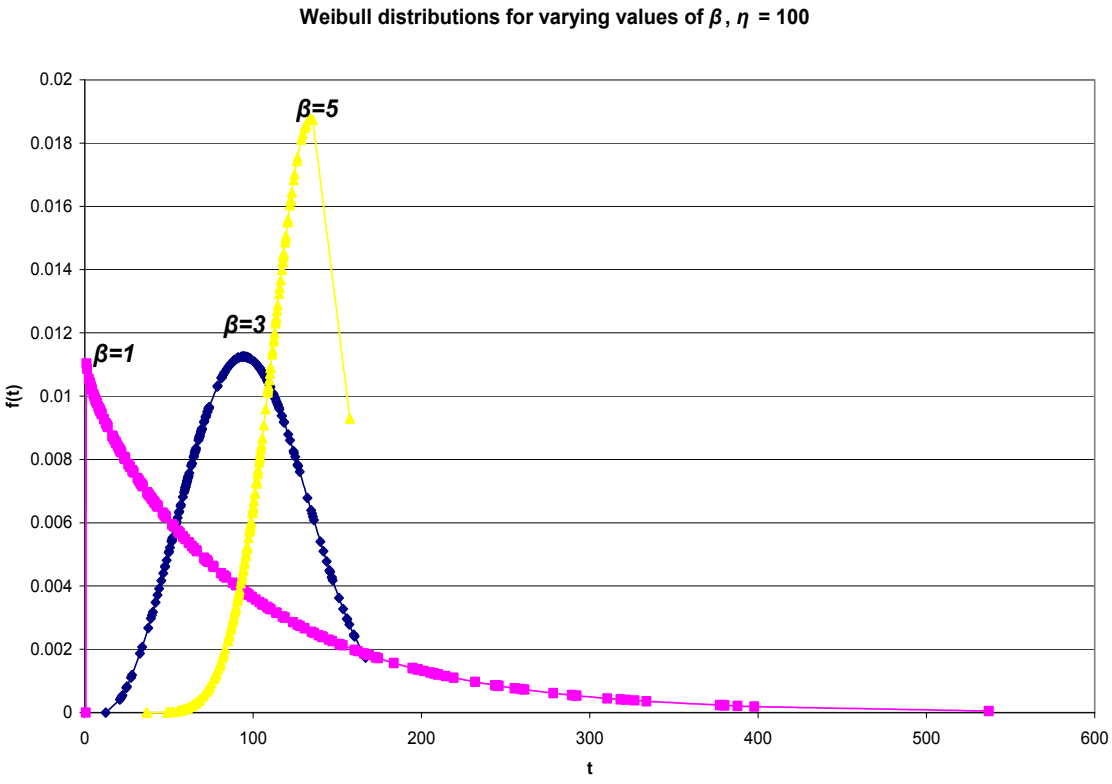


Figure 2.1: The effect of the shape parameter on the probability density function.

We note that the right tail for a non-congested state is reduced as the congestion level increases, until a Gamma (almost normal) distribution emerges for a moderately congested network. When $\beta \geq 3.7$ the probability density function plot is skewed to the left. For a highly congested network, we will show in subsequent chapters that the distribution has a distinct left tail.

From this information, and the observation that congestion plays a significant role in the distribution of RTT [30], we assert that when a path is not congested, then the Weibull shape parameter, β, for the *pdf* will be less than 2.6. If the path is moderately congested, then the value of β will be $2.6 \leq \beta < 3.7$. A value greater than or equal to 3.7 indicates an extremely congested network. This assertion, in addition to "quantifying" the congestion status of the network, allows us to define three distinct congestion states: "Not congested", "Moderately congested" and "Heavily congested."

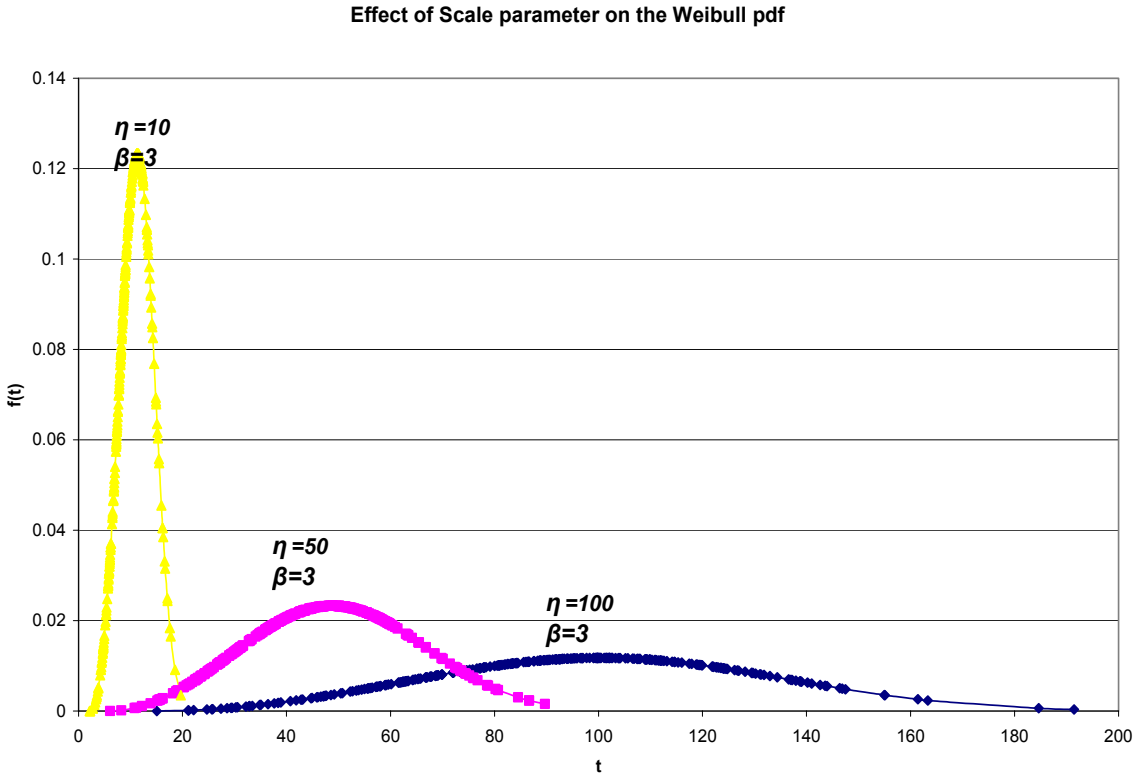


Figure 2.2: The effect of the scale parameter on the probability density function

As shown in Figure 2.3, a Markov chain can be used to specify the congestion states and transition probabilities for simulation purposes. The states S0, S1 and S2 represent non-congested, a moderately congested a heavily congested network, respectively. The transition matrix for the different congestion states is given by

$$\begin{bmatrix} p_{00} & p_{01} & p_{03} \\ p_{10} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} \end{bmatrix} \qquad (2.13)$$

where p_{ij}, $i, j \in \{0,1,2\}$, is the transition probability, which is the probability of the congestion state changing from state i to state j, and for all i, $p_{i0} + p_{i1} + p_{i2} = 1$.

The Markov chain is suitable in this scenario since the next state depends on the current state. For example, if the state is "Not congested", then there is a higher probability that the next state will be "Moderately congested" than "Highly congested."

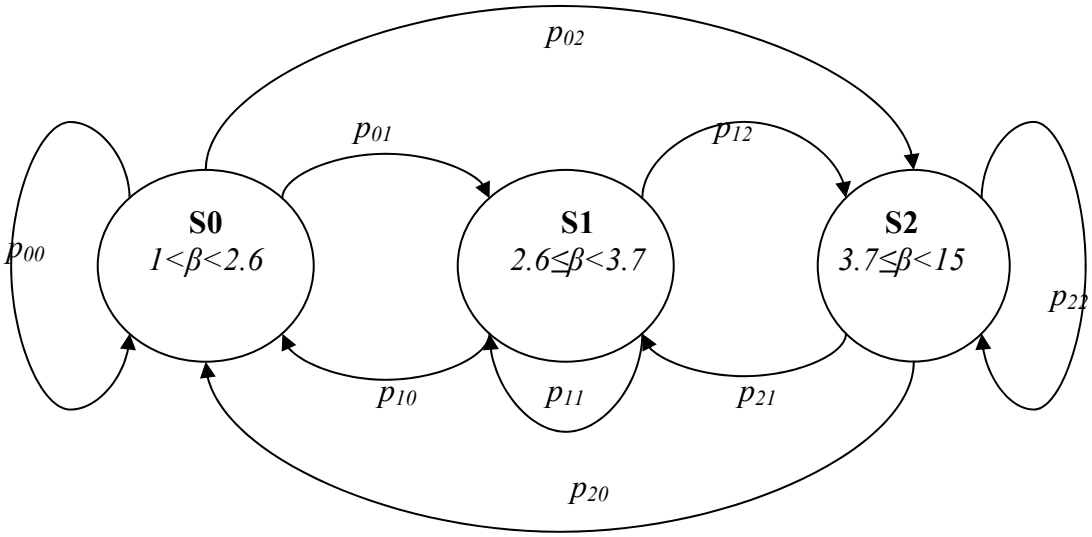


Figure 2.3: Markov chain representation of the congestion states

2.7 DELAY AND LOSS IN TCP CONGESTION CONTROL

A lot of traditional Internet applications use TCP as their transport layer protocol and depend on TCP congestion control mechanisms to detect and recover from congestion in the network [1]. TCP is a reliable, connection-oriented end-to-end transport protocol that is part of a layered hierarchy of protocols that support multi-network applications [31]. To ensure reliable delivery of packets, TCP retransmits all packets which have been lost along the network path. In this protocol, a sender increases its transmission rate additively until a packet is lost, which means that there is congestion somewhere on the network path. In response to this, the sender then adapts and decreases the transmission rate multiplicatively to counter the inferred congestion. By using this mechanism, it is ensured that the transmission rate of the TCP sender is determined and limited by the level of congestion in the network [50].

The traditional notion of TCP, with the exception of Vegas and a few others, has been that all losses are due to congestion, so loss is used as the primary indicator of congestion [4]. Whereas this may have been acceptable for relatively slow and wired networks, it has negative impact on performance when wireless and high speed links are used [47]. TCP's inability to differentiate between congestion-related loss and those caused by link errors, which are common in wireless networks, results in the protocol reducing the congestion window, cwnd, by half even when the loss was not congestion-related. This obviously leads to some performance degradation. Additionally, the traditional additive increase is too slow to fill the pipe in high speed links and links with a high bandwidth-delay product, and multiplicative decrease is too drastic, leading to underutilisation of high speed networks [32]. It is for these reasons, therefore, that more advanced variants and additions to the protocol have been introduced that look at not only packet loss, but delay characteristics as well, to cater for these types of networks [51].

Although some of the advanced implementations use both loss and delay as congestion measures, we categorise the different TCP variants as loss-based and delay-based, depending on the main measure they use. For each TCP variant mentioned, we look at how the loss and delay parameters are used and how they fit into the congestion control mechanism.

2.7.1 Loss based congestion control

2.7.1.1 Tahoe

The aim is to send as much data as possible, without overloading the destination buffers – a concept referred to as flow control [18]. When the sender transmits a packet, its copy is placed in a retransmission queue, and a timer is started. A variable called the congestion window (*cwnd*) is used to limit the amount of data that the sender is allowed to transmit at any given time. Upon receiving data, the destination sends an acknowledgement back to the sender to signal that the packet has been received and is no longer in the network. If an ACK is received before the retransmission timeout (RTO) expires, then the packet is removed from the queue and the timer restarted. It is quite crucial that a suitable value for the RTO be determined, and is calculated using RTT values [9] [6] [3] [33].

2.7.1.1.1 Slow-start

At the beginning of each transmission *cwnd* is set to one. When an ACK is received for this packet, *cwnd* is incremented by one to become 2. Since for each ACK received the *cwnd* increases by 1, *cwnd* increases by 2 in response ACKs for these two packets. This continues for each ACK received, until a packet is lost or buffers are full [3].

In this scheme, the protocol aggressively determines the available bandwidth by continually doubling the number of packets injected into the network. With each ACK received, 2 new packets are injected into the network due to the fact that a packet has been removed from the network and that *cwnd* has increased. So *cwnd* increases exponentially. It continues to do so until a packet is lost or it is limited by receiver's buffers, at which state the multiplicative decrease phase is triggered. The occurrence of packet loss can be detected if a transmission times out or the sender receives a duplicate ACK.

2.7.1.1.2 Congestion avoidance

Once a loss has been detected, available bandwidth is estimated to be half of *cwnd*, and is specified in the slow-start threshold (*ssthresh*) variable. Slow start is reinitialised, and *cwnd* continues to grow exponentially, until it reaches *ssthresh*, at which point slow-start ends and congestion avoidance phase kicks in [22]. The value of *cwnd* then increases at a more conservative rate determined as follows:

$$cwnd = cwnd + \frac{1}{cwnd} \qquad (2.14)$$

This increase, which is now linear, continues until a packet is lost. Exponential growth could be too aggressive, and worsen congestion.

2.7.1.2 TCP Reno

Unlike Tahoe, Reno does not wait for an RTO to expire to infer that a packet has been lost. In addition to slow start and congestion avoidance used in Tahoe, Reno modifies the mechanism and introduces another mechanism– *fast retransmit and fast recovery* [35] [46]. For each packet received by the destination, an acknowledgement is returned, regardless of whether the packet has been received previously [18]. Now, when a packet arrives out of order at the destination, the receiver does not acknowledge this packet. Instead it sends the same ACK it sent the last time. When the sender receives this duplicate ACK (DUPACK), it deduces that the correct packet has either been delayed or lost on the network. As soon as the sender

receives three duplicate ACKs, it immediately retransmits what seems to be the lost packet without waiting for the timeout to expire [34].

Like in Tahoe, delay is used to calculate the RTO, and loss indicates congestion. It improves on the Tahoe performance because it retransmits at early signs of loss (without waiting for RTO) and since slow-start is not used after a loss event, the protocol reaches the previous sending rate – the one it had prior to the loss event – faster. It is also able to make more intelligent guess of how much data is still outstanding [23]. However, empirical studies have shown that the algorithm is only efficient when only 1 segment is lost within a window – it does not recover well if multiple segments are lost within a single window [34].

The receipt of three duplicate ACKs indicates mild congestion, to which the protocol responds by triggering the fast retransmit/recovery algorithms. A retransmit timeout represents severe congestion, requiring the window to be reset to 1, and the slow-start algorithm to be initialised.

2.7.1.3 TCP with Selective Acknowledgements (SACK TCP)

This is optional extension to TCP Reno was introduced to help the protocol completely recover from loss of multiple segments efficiently [36]. For Reno to recover from multiple segment loss, it has to wait a full RTO, or go through multiple Fast Retransmits – possibly of data that might already have been received [34] [23]. Using SACK TCP, the receiver sends ACKs with the SACK field set to let the sender know which segments have been lost, so that it retransmits only these segments. Subsequent arrivals of missing data are ACKed normally by the receiver.

One hindrance with SACK is that for it to work well it requires that both the source and destination support it, which makes deployment difficult [15]. Moreover, it only specifies the packets that need retransmission, but does not specify when the retransmission should take place.

2.7.1.4 High Speed TCP (HSTCP)

As stated earlier, the TCP congestion control mechanisms are inefficient and constrain the bandwidth utilisation, especially when dealing with high speed networks and large congestion windows [37]. It was to solve this problem in high speed wide area links that High Speed TCP (HS TCP) was introduced [38]. Like Reno, HS TCP uses the *additive increase, multiplicative decrease* (AIMD) approach, but it was designed to quickly reach high throughput at

reasonable loss rates and to promptly recover from losses in high-speed network conditions [37].

The TCP response function was modified such that it relates loss to throughput. This response function introduces three parameters: *Low_window*, *High_window*, and *High_P*. When the congestion window is less than the *Low_window* threshold, HSTCP behaves exactly like Reno. Otherwise during the congestion avoidance phase, the congestion window increases or decreased using new parameters, *a(cwnd)* and *b(cwnd)*, which depend on the current value of the congestion window and are obtained from a static look-up table [32]. When an ACK is received, the congestion window increases as follows

$$cwnd = cwnd + \frac{a(cwnd)}{cwnd} \tag{2.15}$$

and reduced as follows when a packet is dropped

$$cwnd = cwnd - b(cwnd) \times cwnd \tag{2.16}$$

A large value of *a(cwnd)* for a large *cwnd* ensures that it will take a shorter period for *cwnd* to return to its previous value prior to a loss occurring. If *b(cwnd)* becomes smaller for a large *cwnd*, then when a loss occurs, the *cwnd* reduction is not too drastic, so not much is lost by way of bandwidth usage.

2.7.1.5 Scalable TCP (STCP)

This variant works well for high speed WANs, and efficiently utilises the bandwidth. It requires only minimal modification to the congestion window update algorithm, and is backward compatible with the Reno and Tahoe TCP [39]. Using this scheme, when an ACK is received during an RTT when there is no congestion detected on the network, the congestion window is updated as

$$cwnd = cwnd + 0.01 \tag{2.17}$$

If congestion has been detected, the congestion window is updated as follows

$$cwnd = cwnd - \left(0.125 \times cwnd\right) \tag{2.18}$$

It follows the multiple increase, multiple decrease approach when updating the congestion window.

2.7.1.6 Delayed Congestion Response TCP (TCP-DCR)

This variant is well suited for wireless networks, which are prone to channel errors. Whereas traditional TCP (Tahoe, Reno) takes all losses to be due to congestion, DCR assumes that all losses are due to channel errors for a certain period, τ [40].

If the packet was lost due to channel error, there is a high likelihood that the network's link-level retransmission mechanism will recover that packet. The main thinking behind TCP-DCR is to let the link level mechanisms to deal with transmission error losses, and the protocol to deal with congestion related losses. Thus, when a duplicate acknowledgement (DUPACK) is received, the protocol delays its response, the start of congestion control mechanisms, for a limited time τ. If the packet is recovered before τ elapses, then the loss was due to channel error, so the protocol proceeds as if the loss didn't occur. If the packet is not recovered at the end of τ, however, then the assumption is that the loss was congestion-related, so fast retransmit and fast recovery are triggered, and the congestion window is reduced.

2.7.1.7 TCP Jersey

This variant of the protocol was designed to work on heterogeneous wired-wireless networks, and has the ability to distinguish between wireless error losses and congestion packet losses [41]. It consists of two mechanisms: the available bandwidth estimator (ABE), and the congestion warning (CW) mechanism.

2.7.1.7.1 Available Bandwidth Estimator (ABE)

This component monitors the rate of receiving ACKs to establish the available bandwidth on the network. The optimum congestion window is then calculated based on this bandwidth estimation.

2.7.1.7.2 Congestion Warning (CW)

This component is based on the Explicit Congestion Notification (ECN) mechanism, which will be discussed shortly. A threshold is set, and once the queue length exceeds this, then the network is assumed to be congested. The mechanism then marks packets by setting the CW bit on the ACK to warn the sender when congestion is detected in the network.

When the sender gets a duplicate ACK with the CW bit set, then it is positive that the network is congested, and assumes that the loss was due to congestion. In response to this, the sender

reduces the congestion window. If, on the other hand, the DUPACK received does not have the CW mark, it is assumed that the loss was due to transmission errors, and so there is no need to reduce the congestion window.

However, only one threshold is defined, and once the average queue length exceeds this threshold all packets are marked.

2.7.2 Delay-based congestion control

2.7.2.1 TCP Vegas

TCP Vegas uses fast retransmit/recovery algorithm, but differs from Reno in the way it detects losses, its congestion avoidance mechanism, and its slow-start mechanism [42]. Reno's congestion control mechanism uses segment loss as a means to detect congestion and adjust *cwnd* – it has no way to predict and prevent loss, i.e. it is reactive. It must create losses for it to determine current network congestion status. Vegas, on the other hand tries to proactively predict congestion and adjust *cwnd* before any segment losses, which reduces the number of retransmissions. It keeps measuring the throughput rate and compares it to expected throughput rate to determine whether to adjust *cwnd* downward/upward so as to decrease/increase data in flight. The discrepancy between the measured and expected throughput, especially a decline in measured throughput, indicates that network congestion has occurred.

2.7.2.2 Random Early Detection (RED)

This is one of the Active Queue Management (AQM) protocols, which uses the queue length at the router as an indication of congestion and drops packets accordingly [42] [43]. It maintains three variables, the minimum threshold, min_t, maximum threshold, max_t, and the average queue length, *avg*. Now, if the average queue length is less than the minimum threshold, then no packets are dropped. Once the length is greater than the maximum threshold then all incoming packets are dropped. When $min_t < avg < max_t$, then packets are dropped using some probability function.

The packet losses prompt the sender to retransmit and reduce its window, thus lower the congestion levels. An advantage to this approach is that dropping the packets keeps the queues short, thus there is less queue delay, which translates to increased throughput.

2.7.2.3 Explicit Congestion Notification (ECN)

This is an extension to RED [43], but instead of dropping the packets if there is congestion, the router marks the packets to inform the sender that there is congestion [44] [45]. The router marks the packet by setting the congestion experienced (CE) bit on the packet's IP header. On receiving this packet, the receiver in turn sets the explicit congestion echo (ECE) bit in the TCP header of the ACK. In response to this, the sender then reduces it congestion window and sets the congestion window reduced (CWR) bit in the TCP header of the next packet sent to acknowledge.

Using ECN mechanism has been shown to reduce unnecessary packet losses and delays, making it well-suited for low-bandwidth, delay-sensitive connections [44].

2.7.2.4 FAST TCP (FTCP)

Fast TCP, which was designed for high speed links with large bandwidth delay products and large congestion windows, uses both queuing delay and packet loss as congestion measures [42]. It paces the sender to reduce burst and massive losses by manipulating the sending of ACKs in light of imminent congestion.

When the occupancy of a buffer is less than a fixed threshold, traffic is transmitted normally in the forward direction, and the ACKs in the backward direction. Once the amount of data buffered exceeds the threshold, FTCP delays the transmission of ACKs in the backward flow by halving the ACK sending rate, which increases the queuing delay and the sender's measured RTT for the packets. This in turn results in the congestion window being reduced, prompting the sender to slow down its sending rate, thus alleviating the congestion detected in the network.

3. ANALYSIS OF RESULTS

After careful consideration of alternative methods of collecting the data, *ping* was selected as the preferred method, mainly because it did not require any installations (or user accounts) at the destination hosts. This was useful because one of the criteria for selecting sites to be used for data collection was that they should cover a wide geographical breadth across South Africa, as well as a few overseas-based sites. The sites were selected based on perceived usage patterns, ranging from the very busy commercial sites to those with relatively low traffic. However, as mentioned in chapter 2, a lot of sites, especially large commercial and academic sites, have ICMP filtering to protect themselves from denial of service attacks, so our choice was confined to sites that do not filter ICMP packets. This obviously created some limitation in our selection exercise since some of the sites we were interested in could no be accessed. Of the selected sites, 22 provided us with sufficient data for the study. **Table 3.1** shows a list of these sites.

Table 3.1: LIST OF PINGED SITES

Site	Address
1	www.etv.co.za
2	www.vodacom.co.za
3	www.dstv.co.za
4	www.natal.co.za
5	www.ofm.co.za
6	www.webmail.co.za
7	www.mobilemagic.co.za
8	www.mweb.co.za
9	www.sabc.co.za
10	www.computicket.co.za
11	www.itweb.co.za
12	www.worldonline.co.za
13	www.kulula.com
14	www.mg.co.za
15	www.iol.co.za
16	www.capetourism.org
17	www.pe.org.za
18	www.google.com
19	www.yahoo.com
20	www.bbc.com
21	www.nokia.com
22	www.altavista.com

The data we were interested in was RTT values and loss events – which were characterized as loss distance and loss length as discussed in section 3.2. The time intervals for the sample gathering were medium-term to long-term, ranging from a few minutes to several hours, and was carried out at different times of day, from as early as 8 o'clock in the morning to around 10 late at night. Analysis was done for frequency distributions exhibited, and the results are outlined in the following sections. Results showing similar patterns were not repeated.

3.1 Packet delay (RTT)

On analysis of the RTT data, three distinct frequency distribution patterns emerged, with invariably all local sites falling under one of these categories. Distribution graphs for only three sites, each representing one of the three patterns, are shown below. **Figure 3.1** is a distribution of a path that was not congested (site 3), for which the loss rate was negligible (≈ 0%). The peak represents inherent RTT.

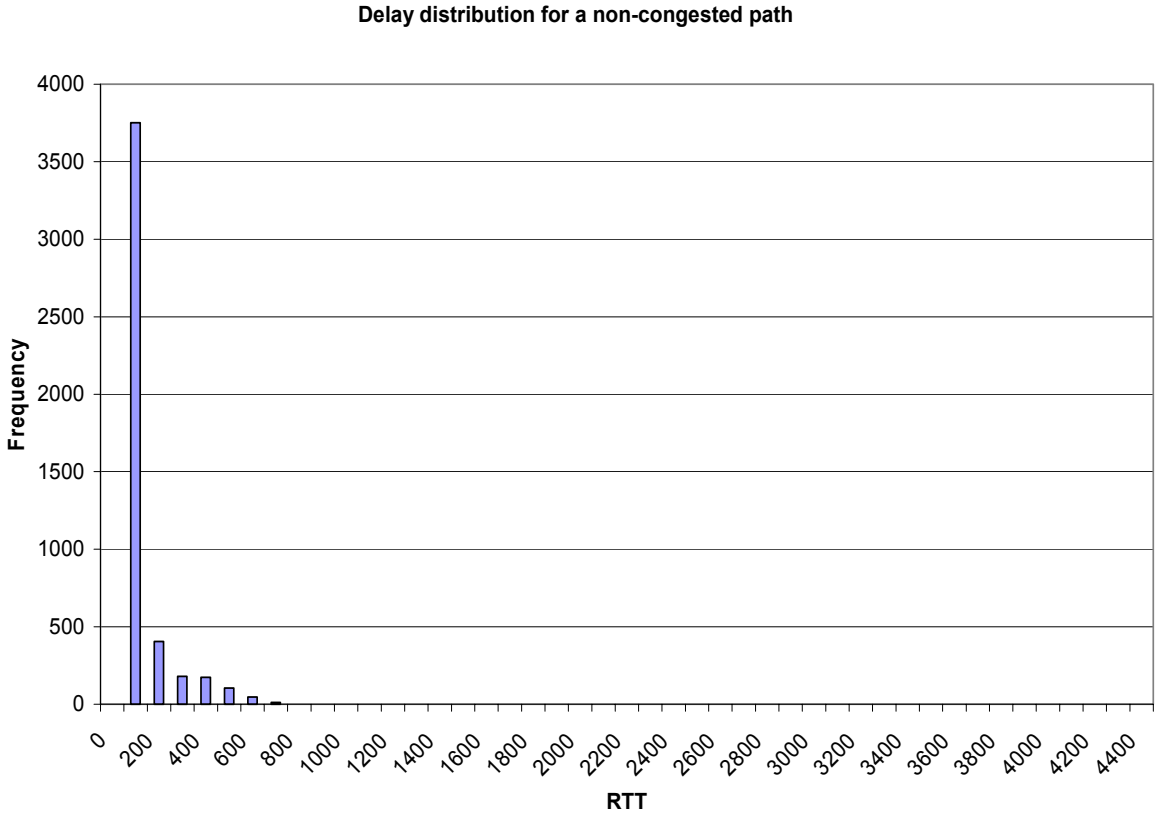


Figure 3.1: The RTT frequency distribution of a non-congested path

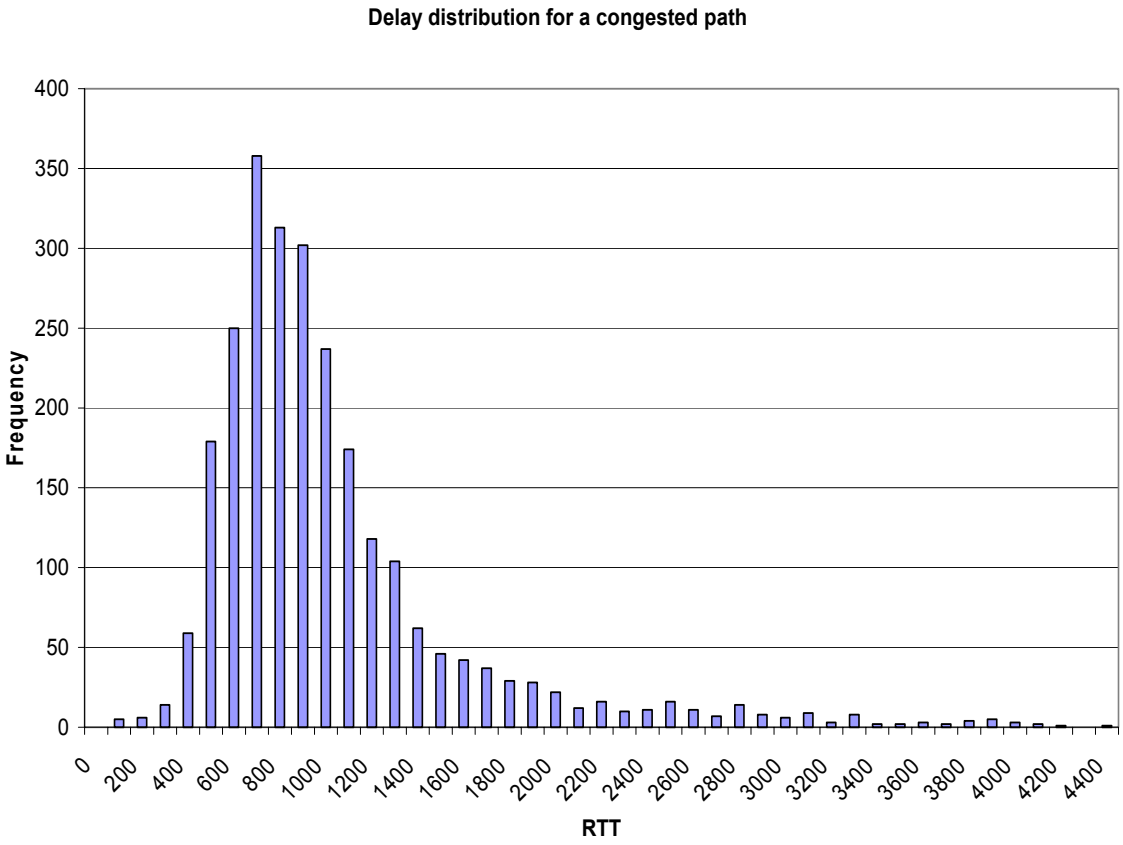


Figure 3.2: Delay distribution for a congested path

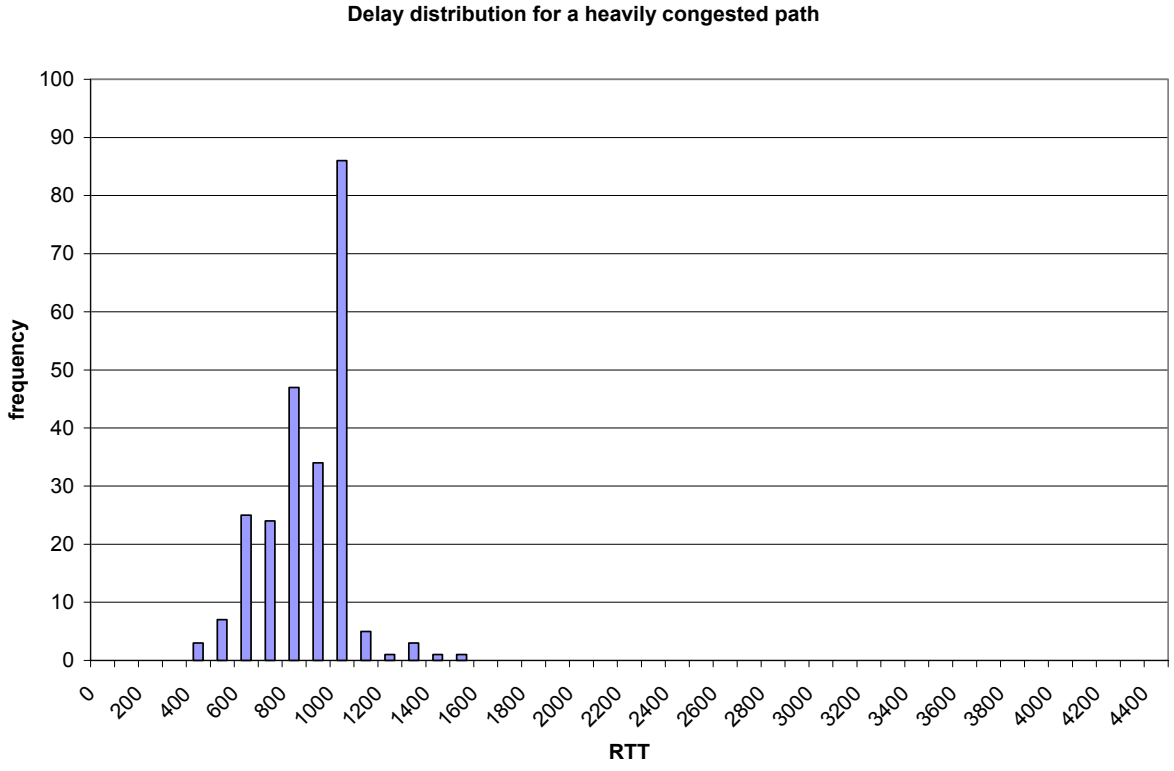


Figure 3.3: The RTT distribution for an extremely congested path

Since there is always some propagation delay as well as transmission/processing delay, every packet sent will always be delayed by a value equal to or greater than the inherent RTT. Thus, when there is little or no congestion, the most frequently occurring RTT is the inherent RTT.

A distribution for a moderately congested site, site 1 with a loss rate 1%, is shown in **Figure 3.2**. As a result of congestion along the path, queues at the routers or other intermediary nodes increase, causing packets to be delayed for longer periods. The mean or mode of RTT starts increasing, and the occurrence of inherent RTT is greatly reduced. The shape of the frequency distribution changes to a shape that can be modeled using a Gamma distribution. The local maxima in this distribution are close to the average RTT [15].

Figure 3.3 shows RTT distribution for site 1, for data obtained using a bottlenecked path (dialup connection from Bloemfontein) with a loss rate of 88%. It shows that as congestion increases to critical levels, the delay distribution shape tends to have a left tail.

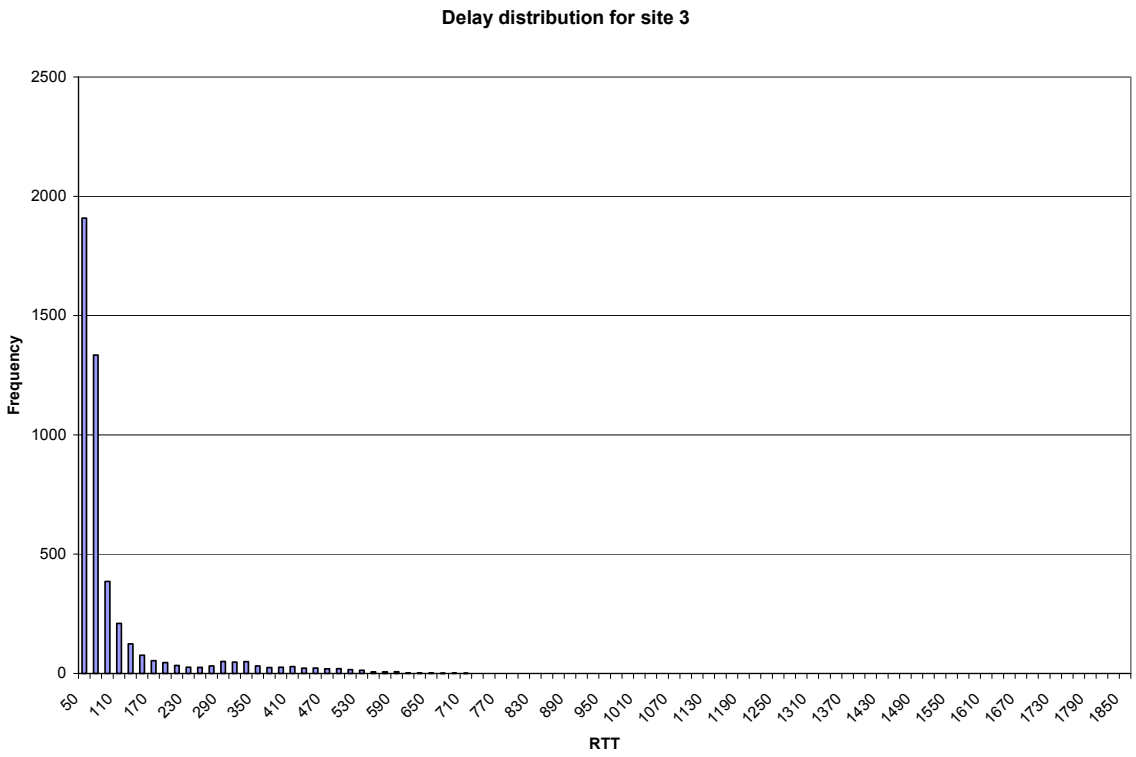
3.1.1 Effects of distance on Delay distribution

To investigate the effect of geographical distance on the delay distribution, we look at the distributions from sites at various geographic locations. We look at site 3, located in Gauteng, site 5 in Free State, site 4 in Kwazulu Natal and site 16 in Cape Town.

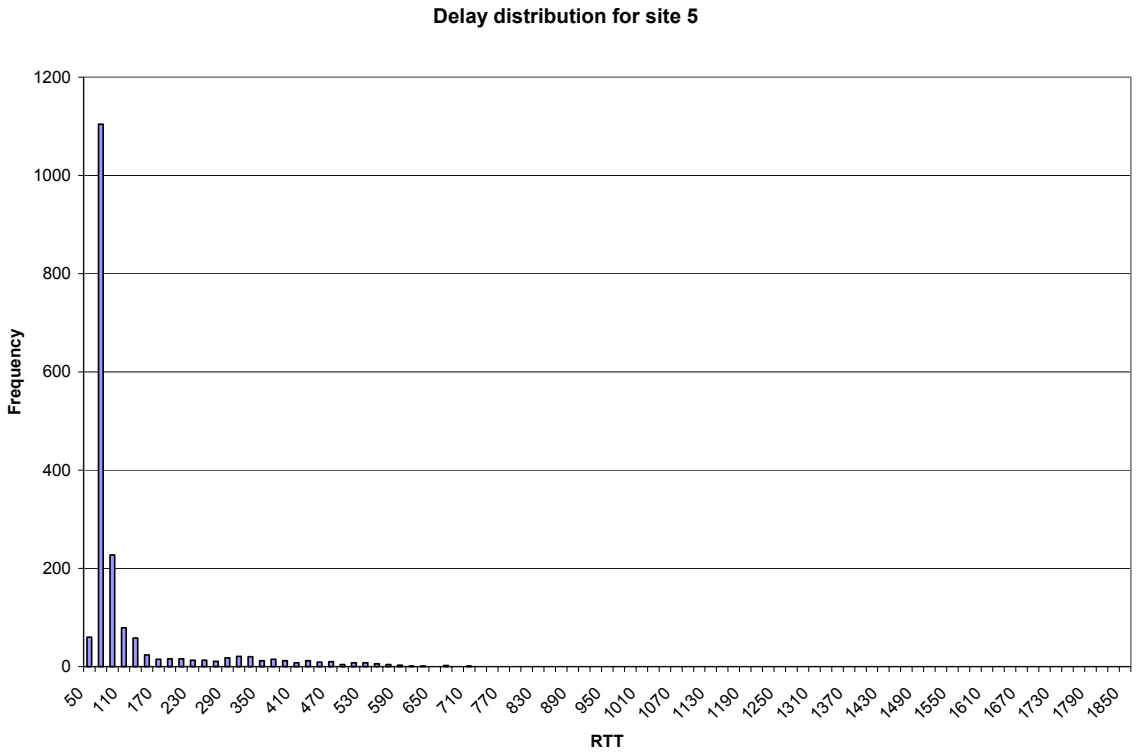
We note in **Figure 3.4** that for the nearest site, site 3, the distribution is a Pareto with inherent RTT frequently occurring. The frequency of inherent RTT starts decreasing as we move further. For site 5, in Orange Free State, the distribution still has a right tail but inherent RTT is no longer occurring frequently. For site 6, in KZN, the distribution is more spread out and, appears to be multi-modal. Site 16, which is the furthest, has a left tailed distribution. Most of the overseas sites, which are even farther away, display a distribution that is clustered together and appears to be normal, as shown in **Figure 3.5**. Further analysis in chapter 4, however, will show with certainty that these distributions are actually left-tailed.

From the observations, we can deduce that distance can also affect the distribution. This could be due to congestion as well as routing mechanisms in the network. Since routing mechanisms also perform based on congestion, it can be assumed that congestion still plays a big role. This is especially true when we consider that overseas sites still follow the trend observed for local sites and exhibit a left-tailed distribution as they are far away. It is probably safer to say that more information is required in order to be reasonably certain about the real cause of the shape

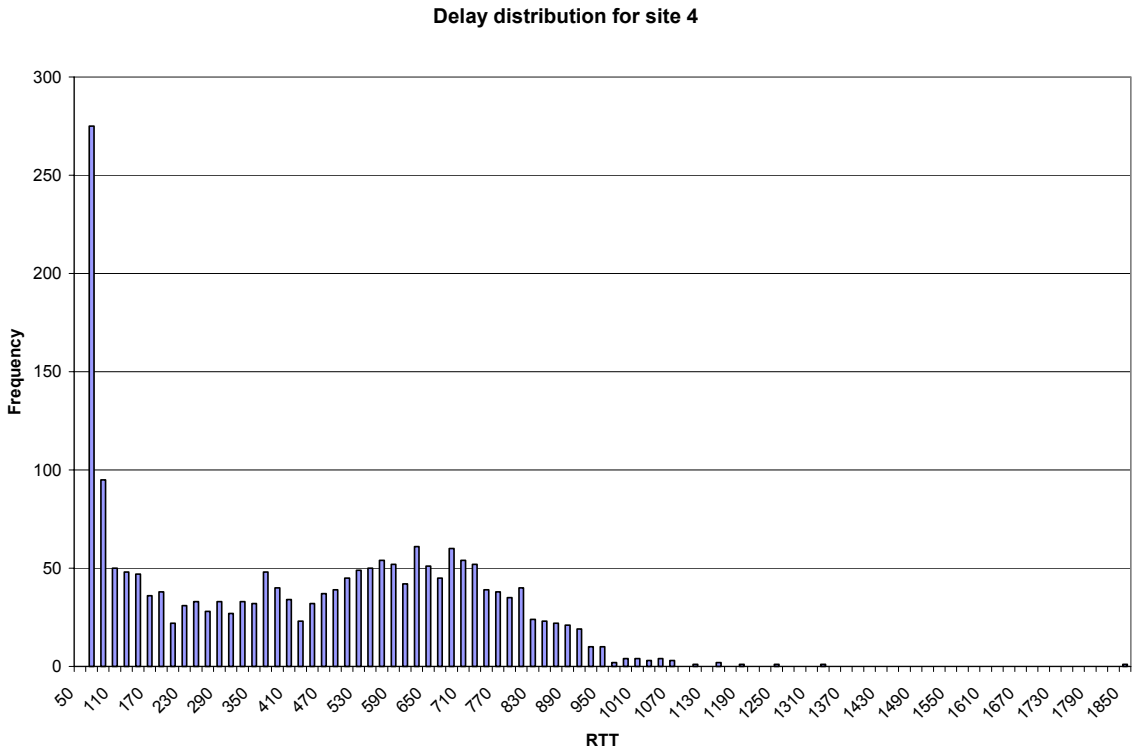
of the distribution. In this research it is asserted that results about the delay distribution should be augmented by loss observations.



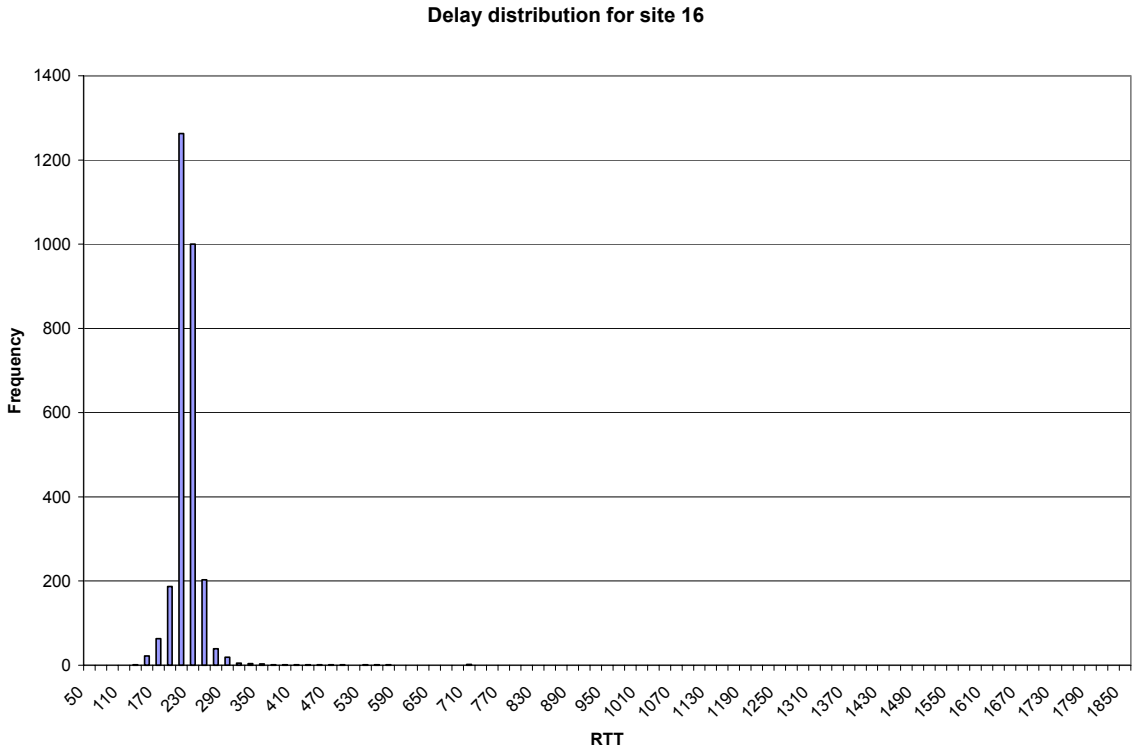
a) Delay distribution for site 3



b) Delay distribution for site 5

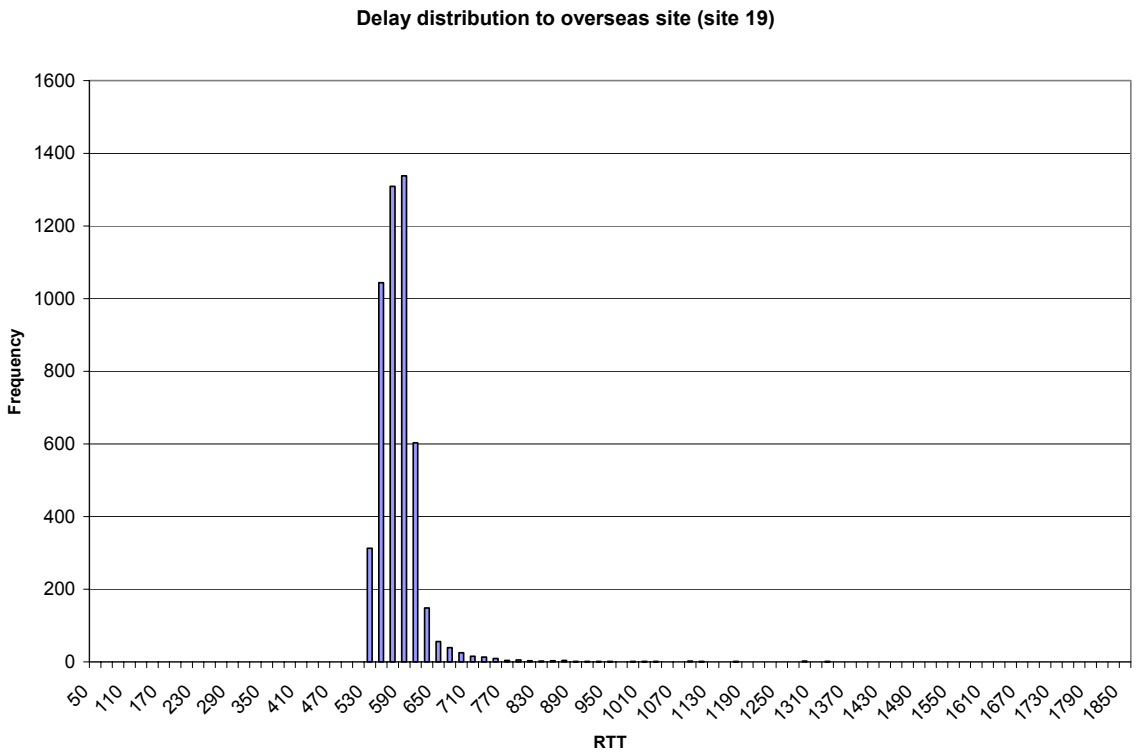


c) Delay distribution for site 4

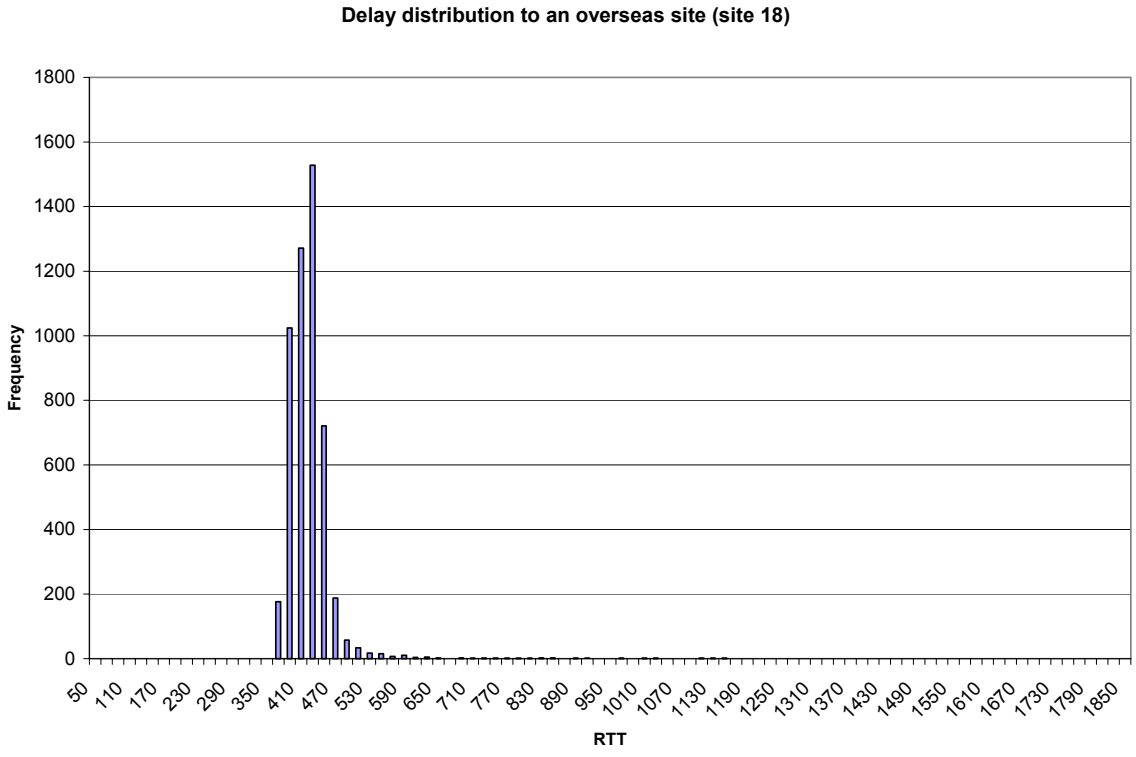


d) Delay distribution for site 16

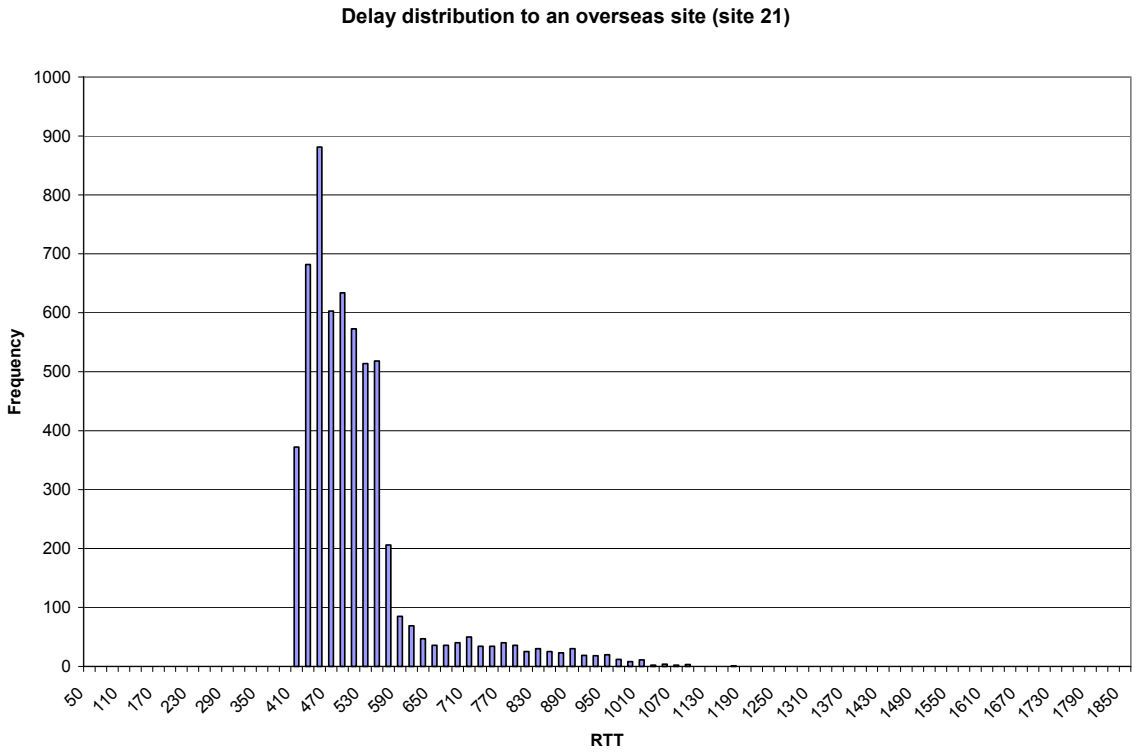
Figure 3.4: Effects of distance on the delay distribution



a) Delay distribution for site 19



b) Delay distribution for site 18



c) Delay distribution for site 21

Figure 3.5: Delay distribution for overseas sites

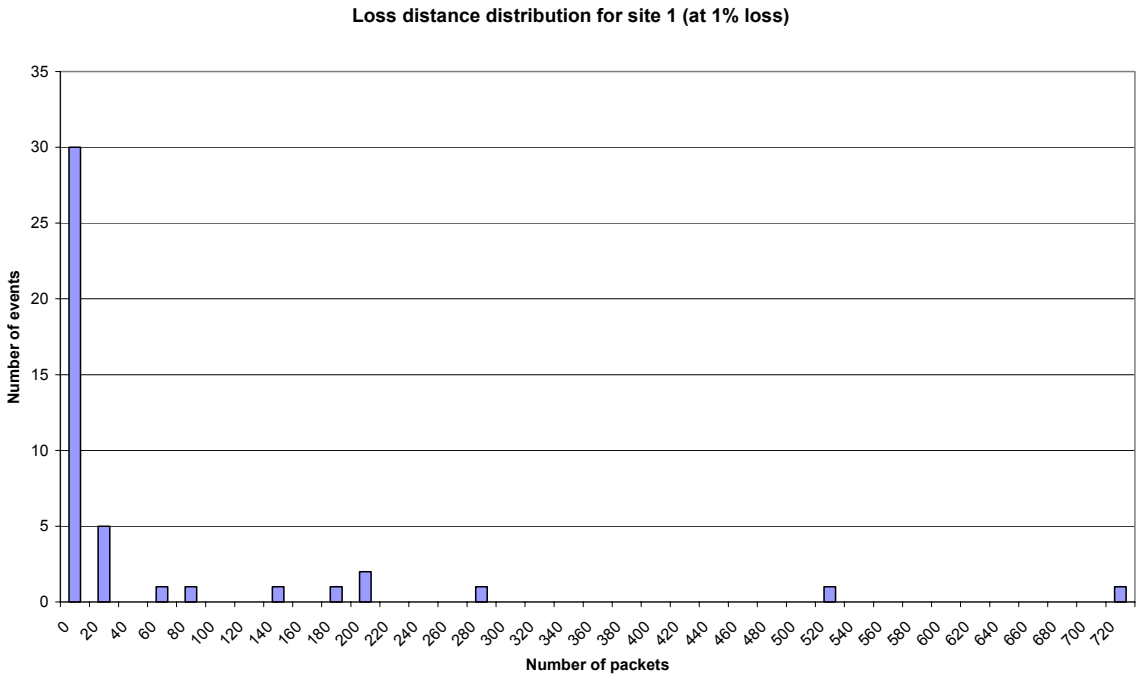
3.2 Packet loss

The analysis of packet loss characteristics was separated into two sections, the first dealing with the loss distance (number of packets successfully transmitted before a loss) and the second dealing with loss length (number of packet lost in a loss event).

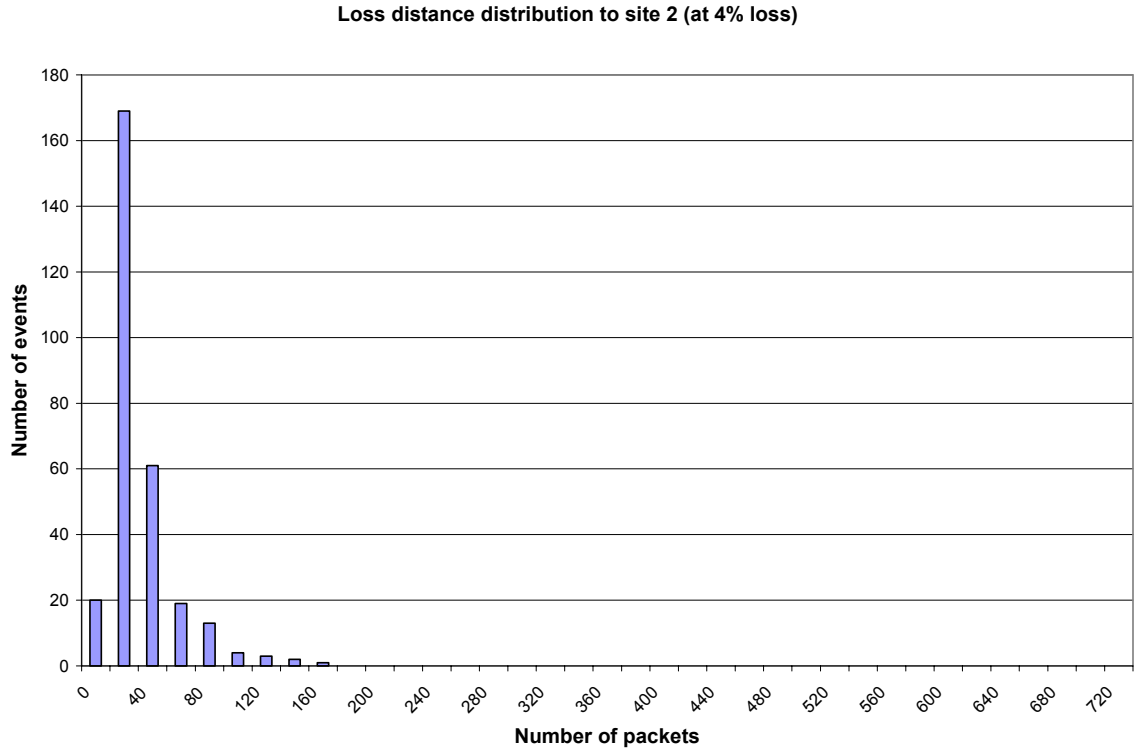
3.2.1 Loss distance

The loss distance was analyzed for various sites and various loss rates. **Figure 3.6** shows that loss distance distributions are always right-tailed – either a Pareto or a gamma distribution. For 1% loss rate for site 1, which was somehow moderately congested, the most frequently occurring loss distance was 0, suggesting bursty losses. A similar pattern is observed for site 5 (19% loss rate) and site 1 (29%, 43% and 88% loss rates). For site 4 (11% loss rate) the 0 loss distance is still prominent even though it is not the mode. For site 2 (4% loss rate) and site 6 (6% loss rate) the mode moved away from zero.

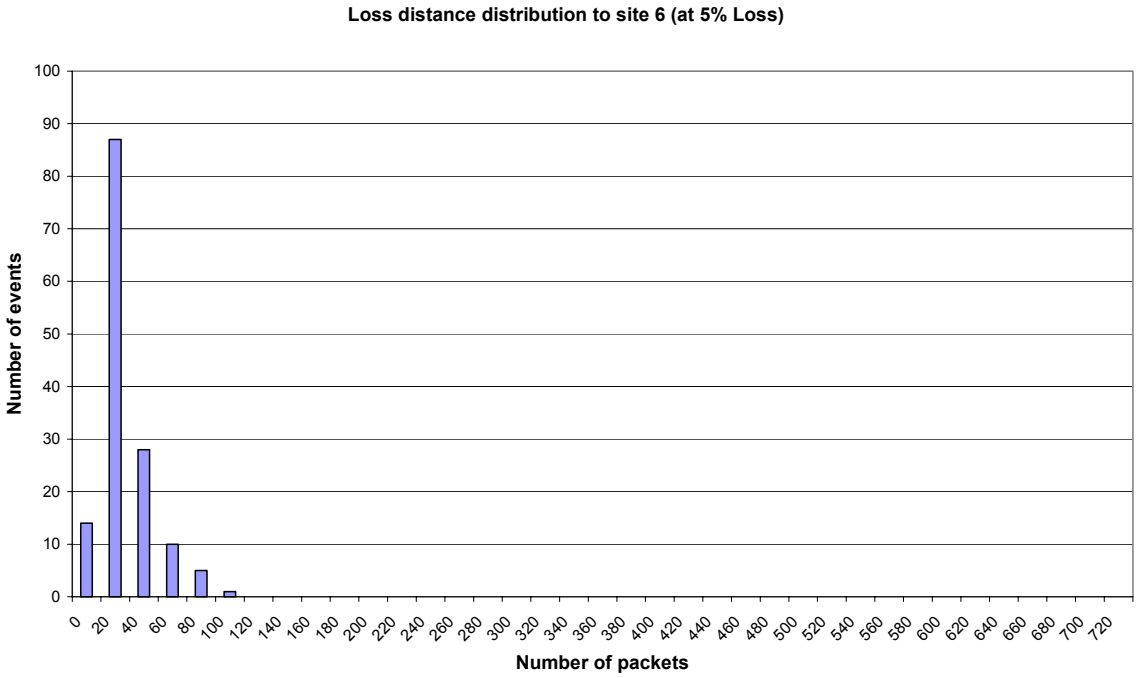
Analysis of the loss distance distributions and RTT distributions suggests that packet losses are frequently bursty for congested paths.



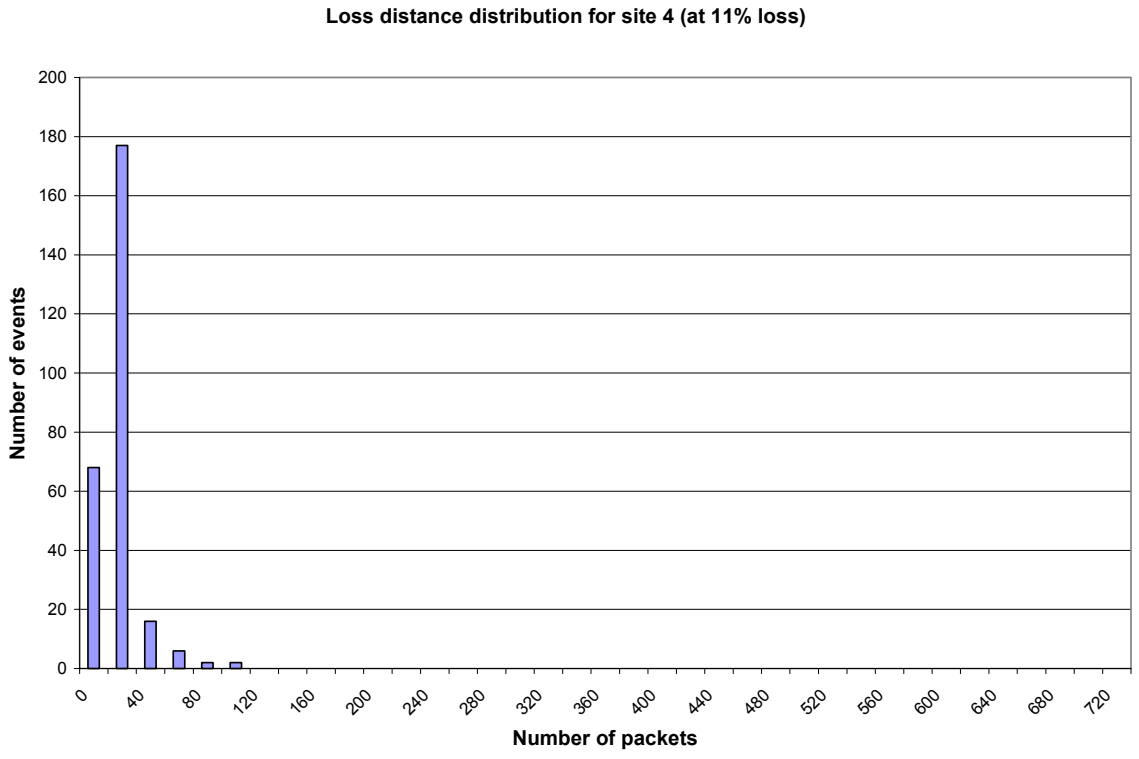
a) Loss distance distribution for site 1 at 1% loss rate



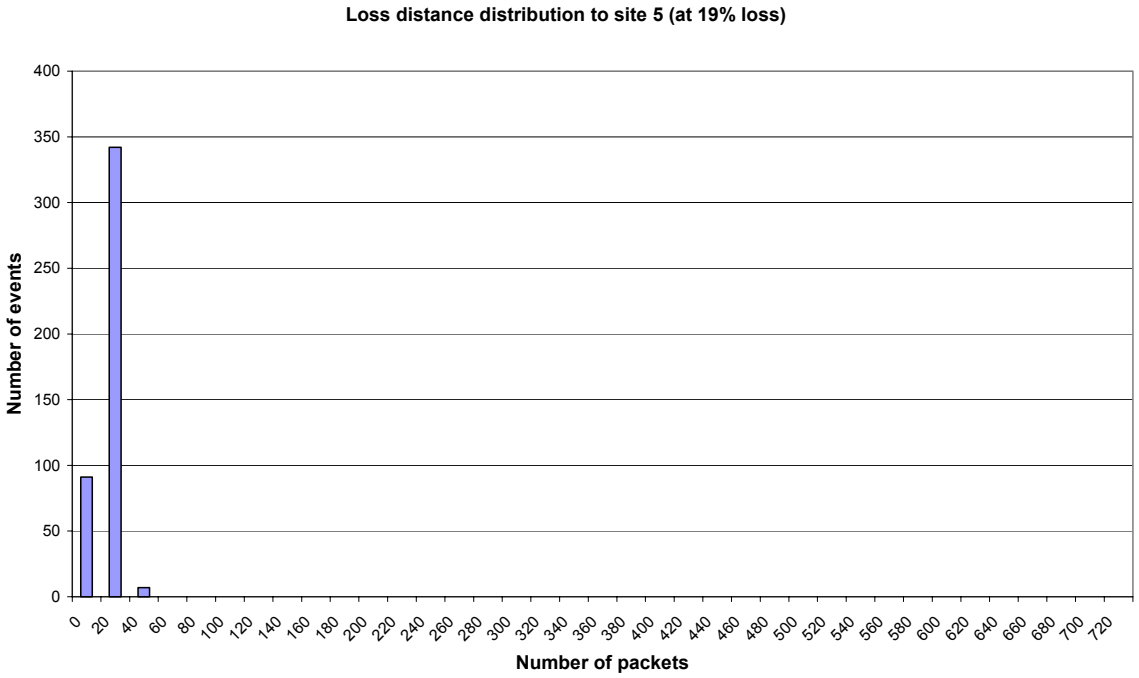
b) Loss distance distribution for site 2 at 4% loss rate



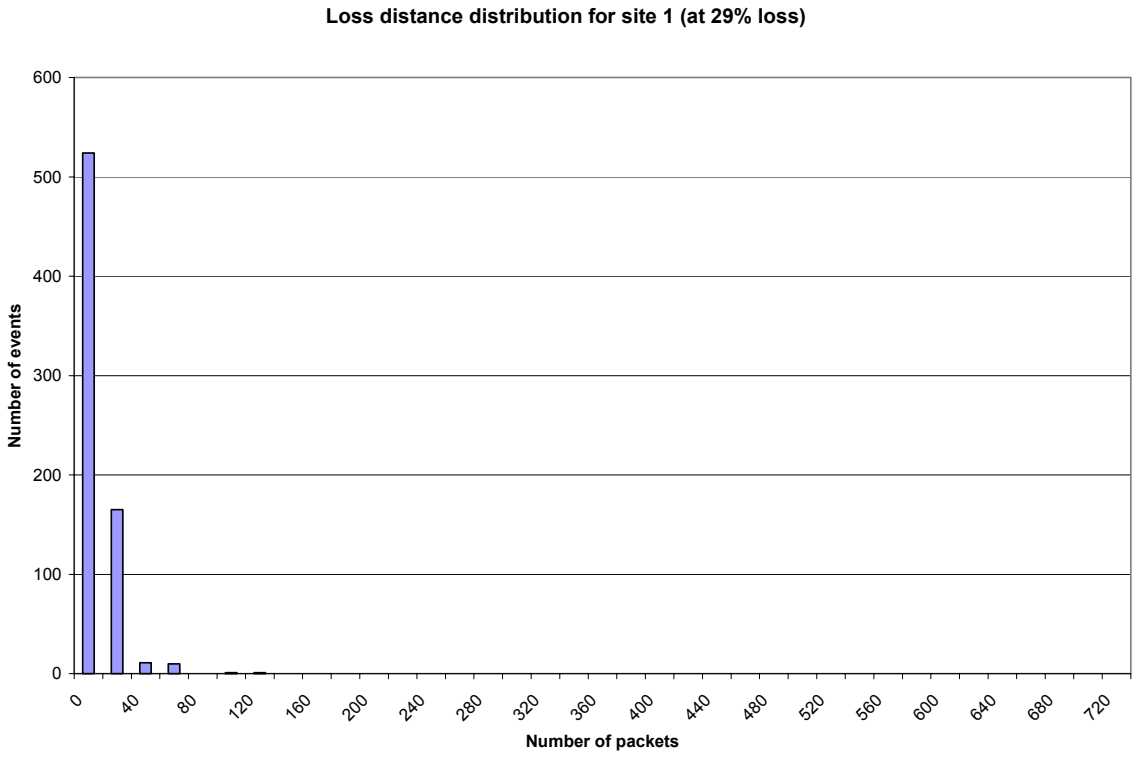
c) Loss distance distribution for site 6 at 5% loss rate



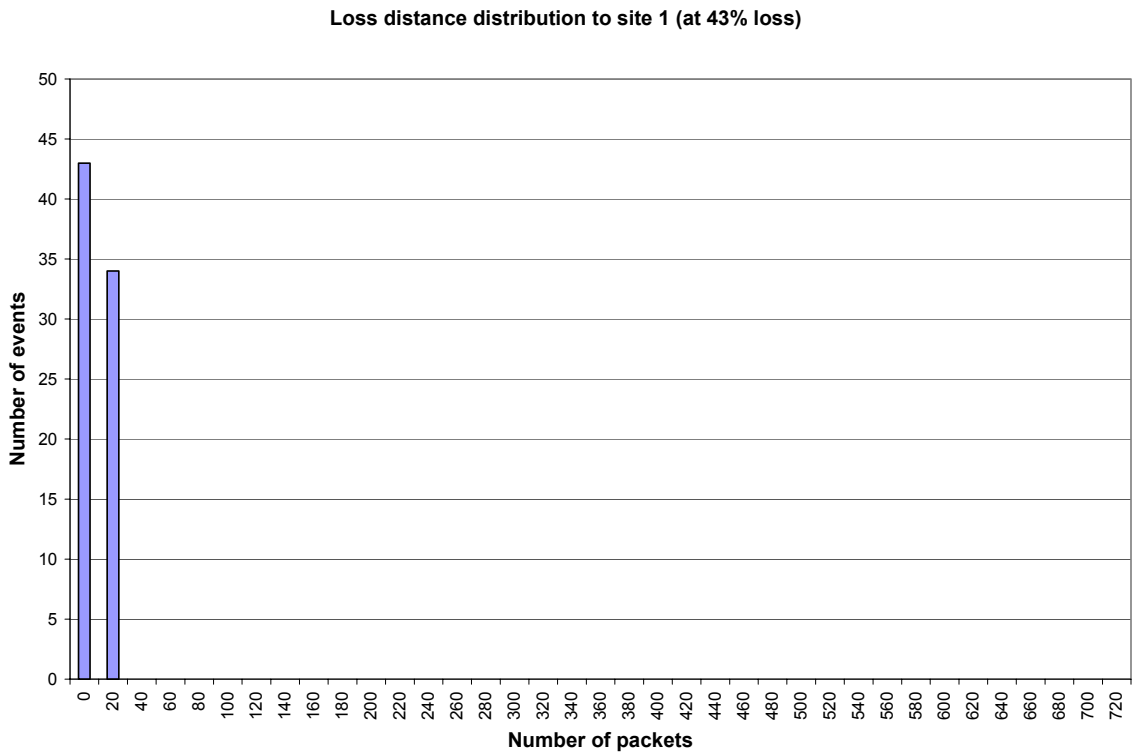
d) Loss distance distribution for site 4 at 11% loss rate



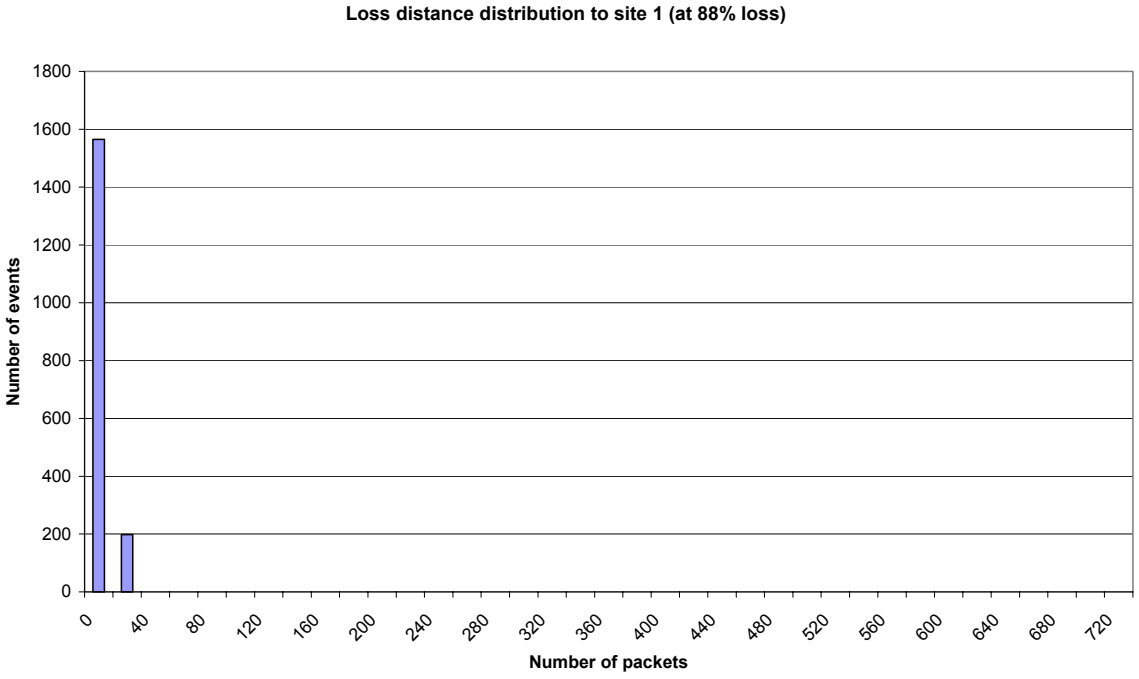
e) Loss distance distribution for site 5 at 19% loss rate



f) Loss distance distribution for site 1 at 29% loss rate



g) Loss distance distribution for site 1 at 43% loss rate



h) Loss distance distribution for site 1 at 88% loss rate

Figure 3.6: Loss distance distributions for different loss rates

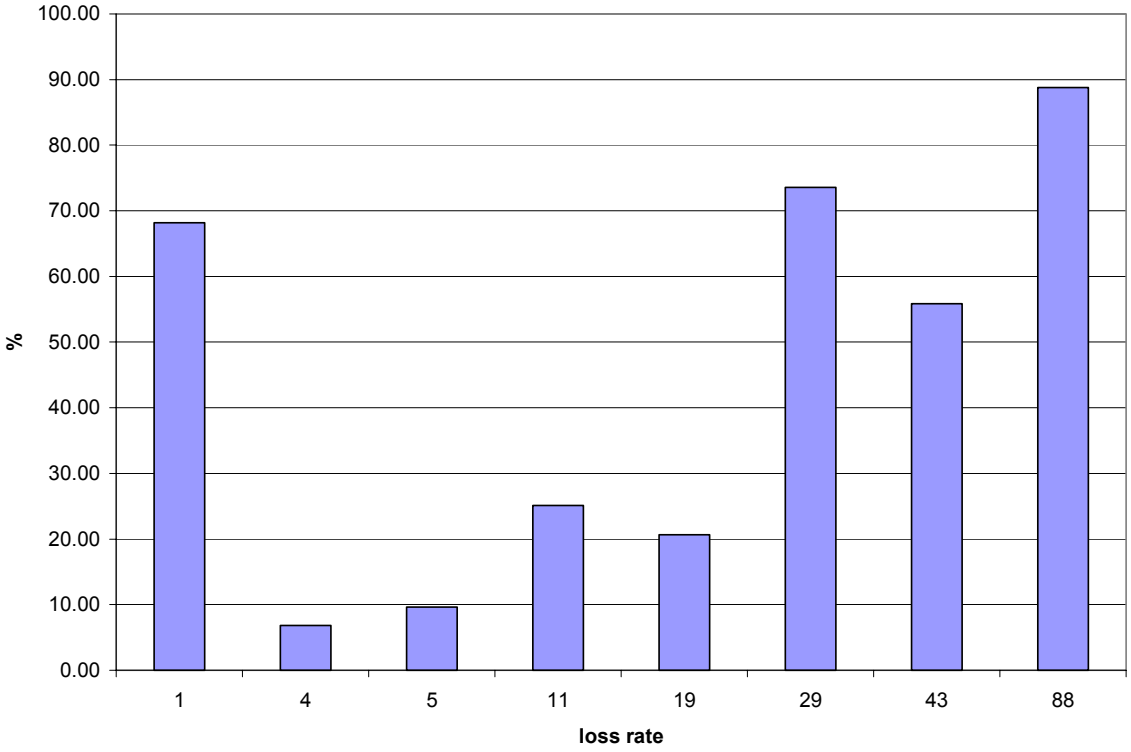


Figure 3.7: Comparison of bursty losses with loss rates

Figure 3.7 shows a comparison of bursty losses and overall loss rates. It seems that bursty losses increase with an increase in congestion levels. Although the loss rate for site 1 is small, it has a high percentage of bursts – suggesting congestion.

To study the effect of loss rate on the loss distance, the average loss distance at a given loss rate was computed as shown in **Table 3.2**. The graph is shown in **Figure 3.8**.

Table 3.2: Comparison of loss rate and loss distance

Loss rate (%)	Average Loss distance	Standard deviation
1	52.82	139.96
3	32.07	34.10
4	21.42	24.78
5	16.51	17.57
19	4.11	5.05
29	2.48	8.85
43	1.29	2.21
88	0.13	0.44

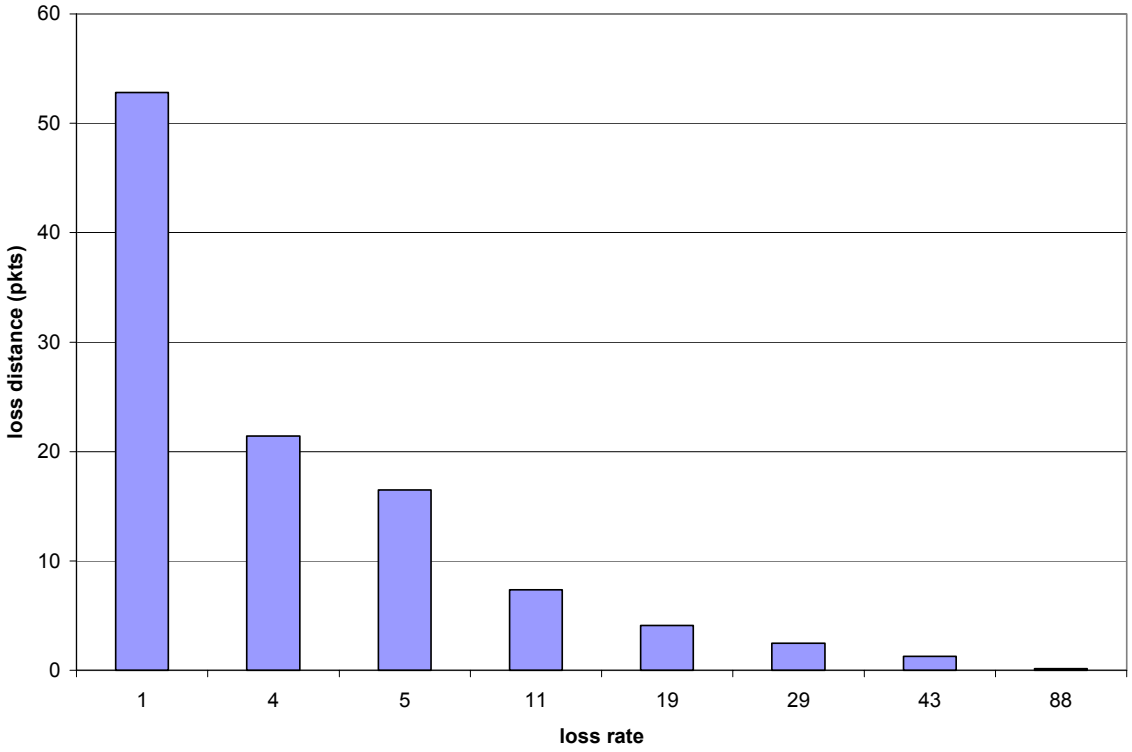


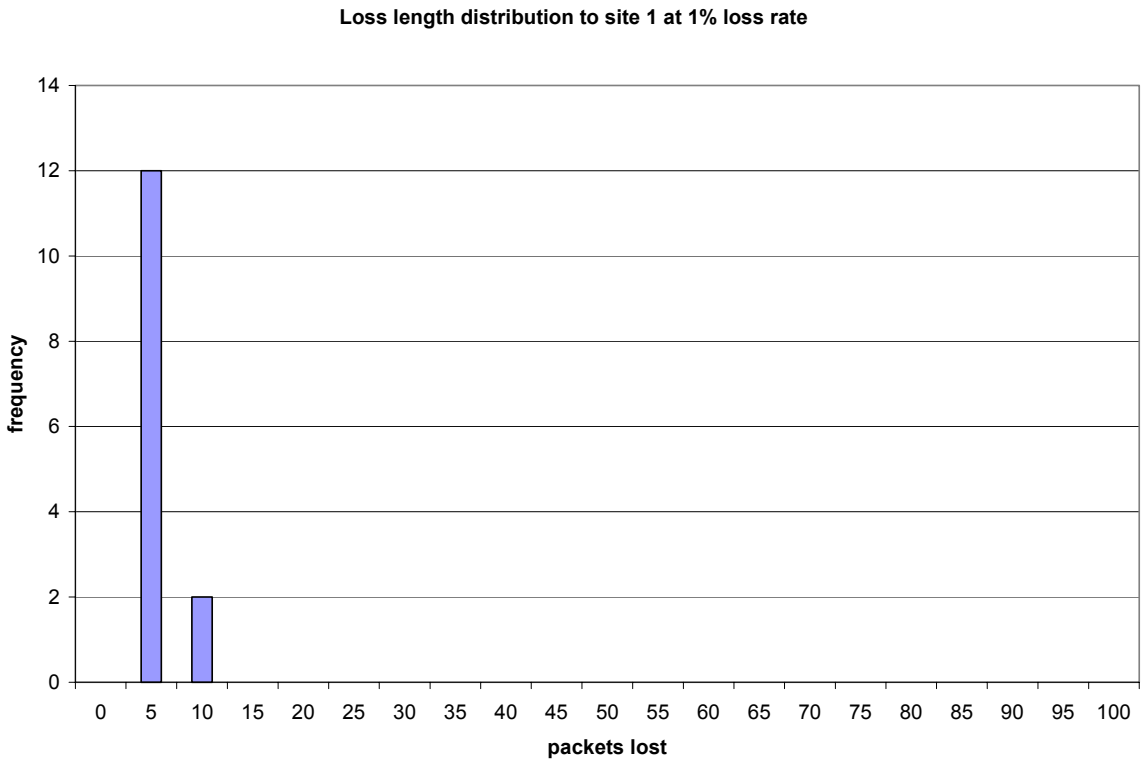
Figure 3.8: The effect of increasing loss on the loss distance

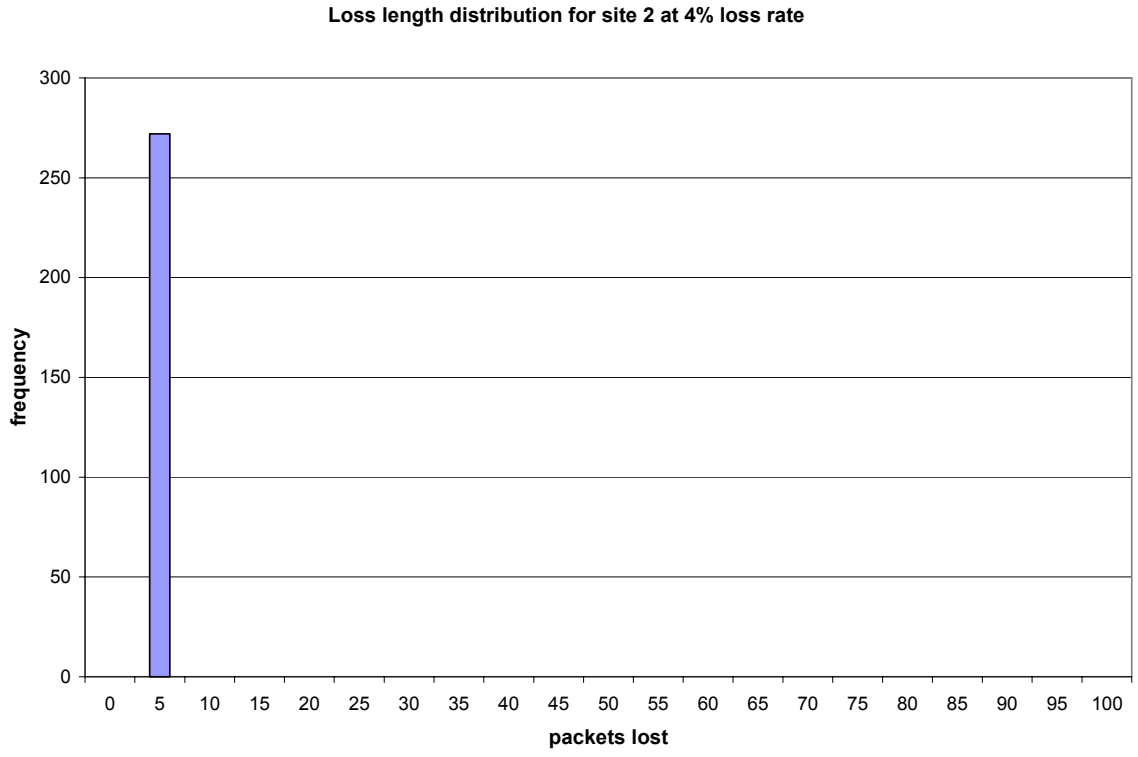
It is noted that while loss rate might not always be due to congestion, it almost always has the effect of reducing the throughput – due to the reduction of average loss distance.

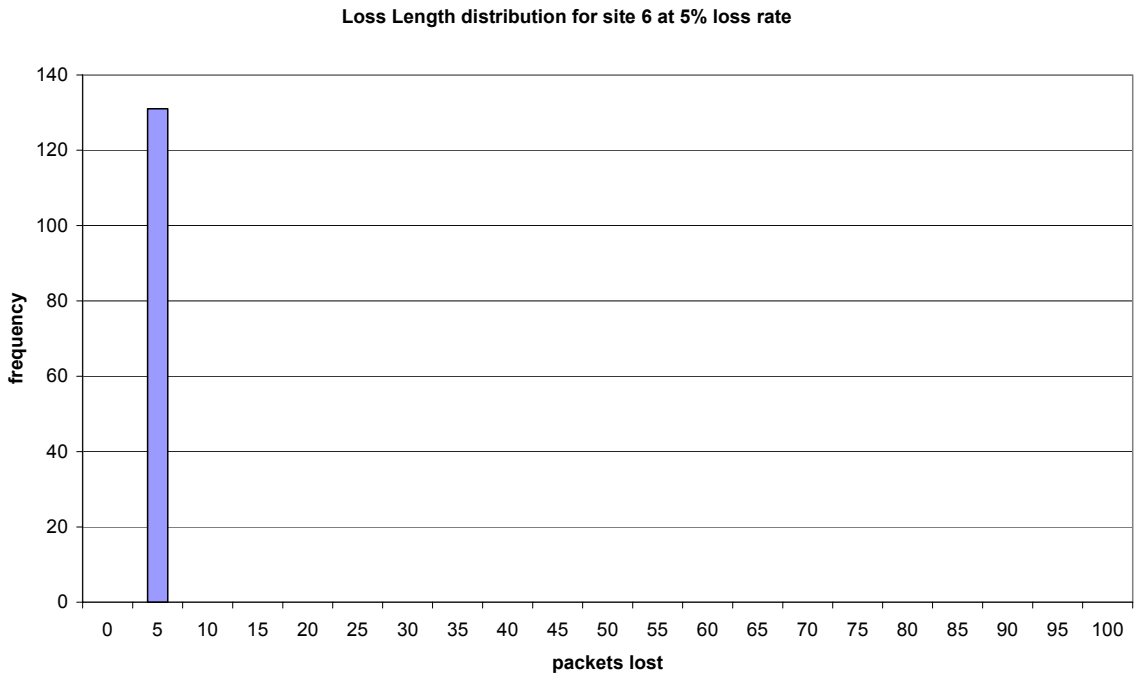
3.2.2 Loss length

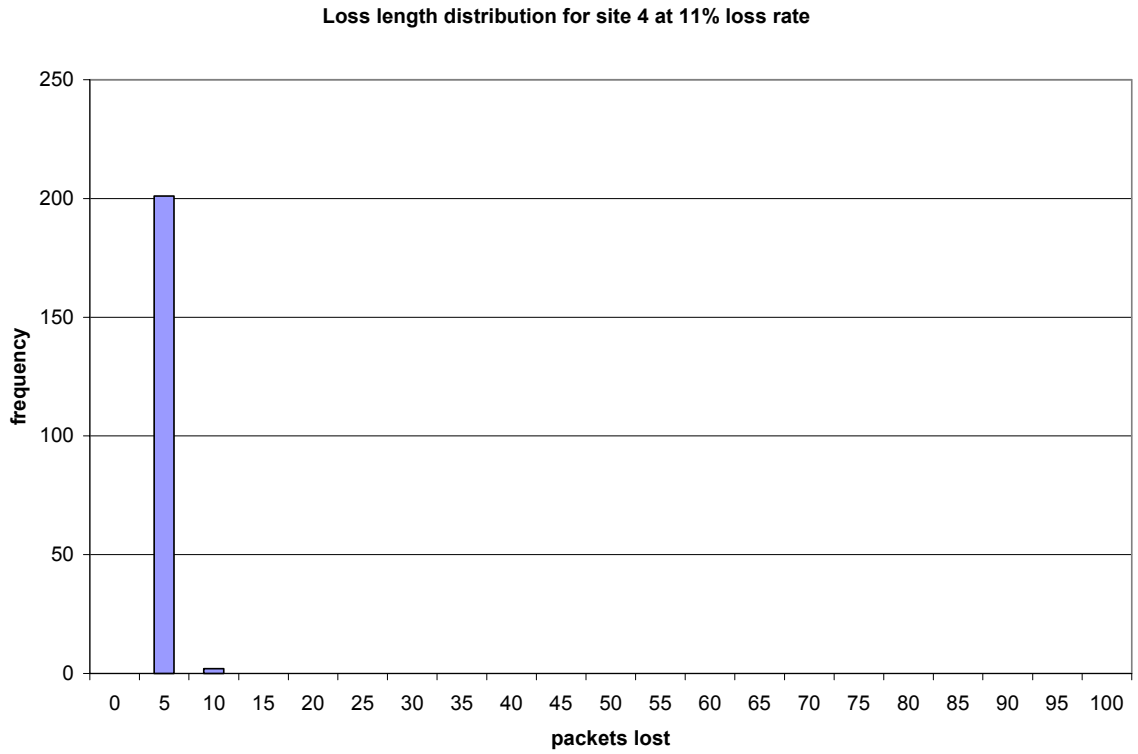
It has been established that losses, especially those that are congestion-related, occur in bursts rather than in isolated single packets. The loss length – the number of packets lost in a loss event – helps in understanding the behavior of the bursts. Various loss length distributions are shown for different loss rates in **Figure 3.9**.

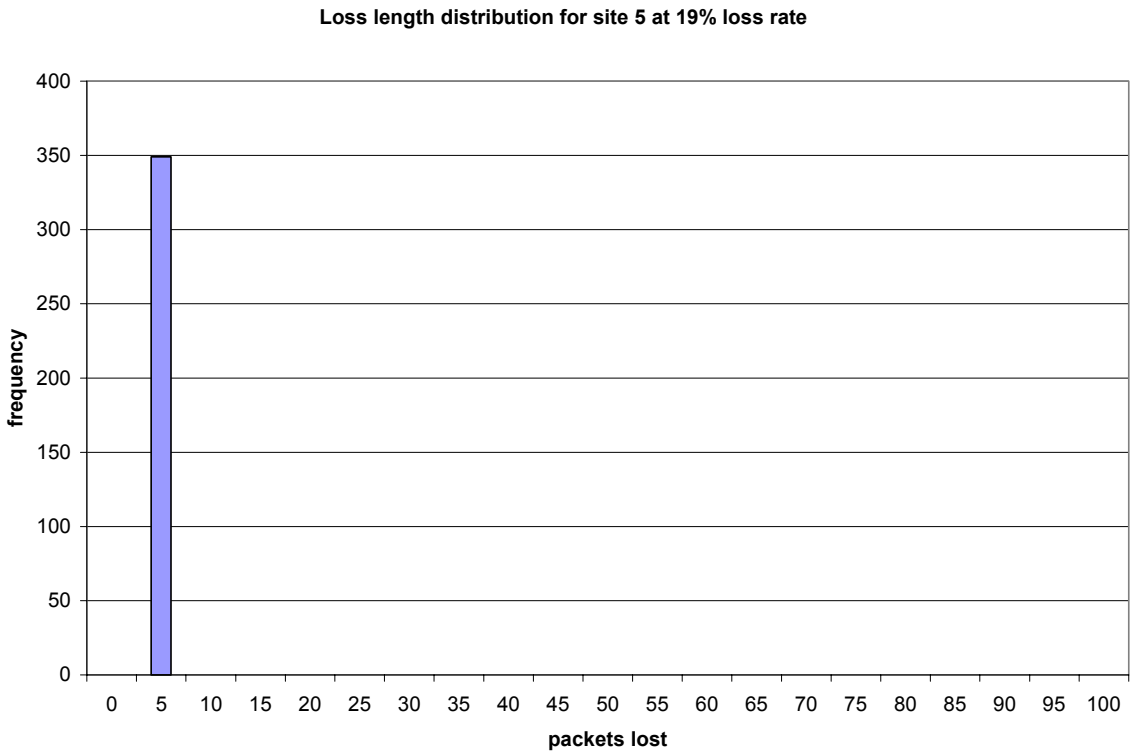
The distributions are right-tailed – similar to loss distance distributions. What can be observed is that even though packet losses occur in bursts, the bursts are generally of a considerably small number of packets. Bursts of a large number of packets occur infrequently, especially for small loss rates. The frequency of losing a large number of packets in a burst seems to increase with increasing loss rates. It is therefore inferred that when congestion increases, the number of losses per loss event increases. The inference is somehow validated by **Figure 3.10**.

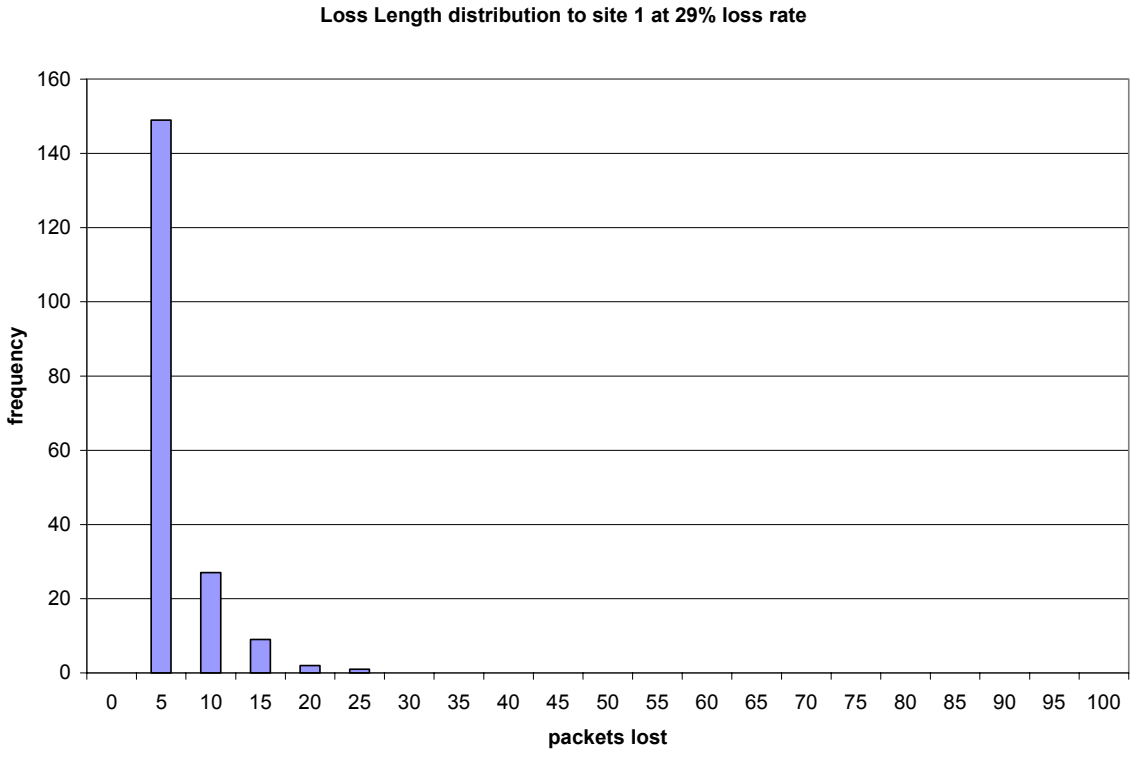

Loss length distribution to site 1 at 1% loss rate
frequency
0
2
4
6
8
10
12
14
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
packets lost


Loss length distribution for site 2 at 4% loss rate
frequency
0
50
100
150
200
250
300
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
packets lost


Loss Length distribution for site 6 at 5% loss rate
frequency
140
120
100
80
60
40
20
0
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
packets lost


Loss length distribution for site 4 at 11% loss rate
frequency
250
200
150
100
50
0
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
packets lost


Loss length distribution for site 5 at 19% loss rate
frequency
400
350
300
250
200
150
100
50
0
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
packets lost


Loss Length distribution to site 1 at 29% loss rate
frequency
160
140
120
100
80
60
40
20
0
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
packets lost

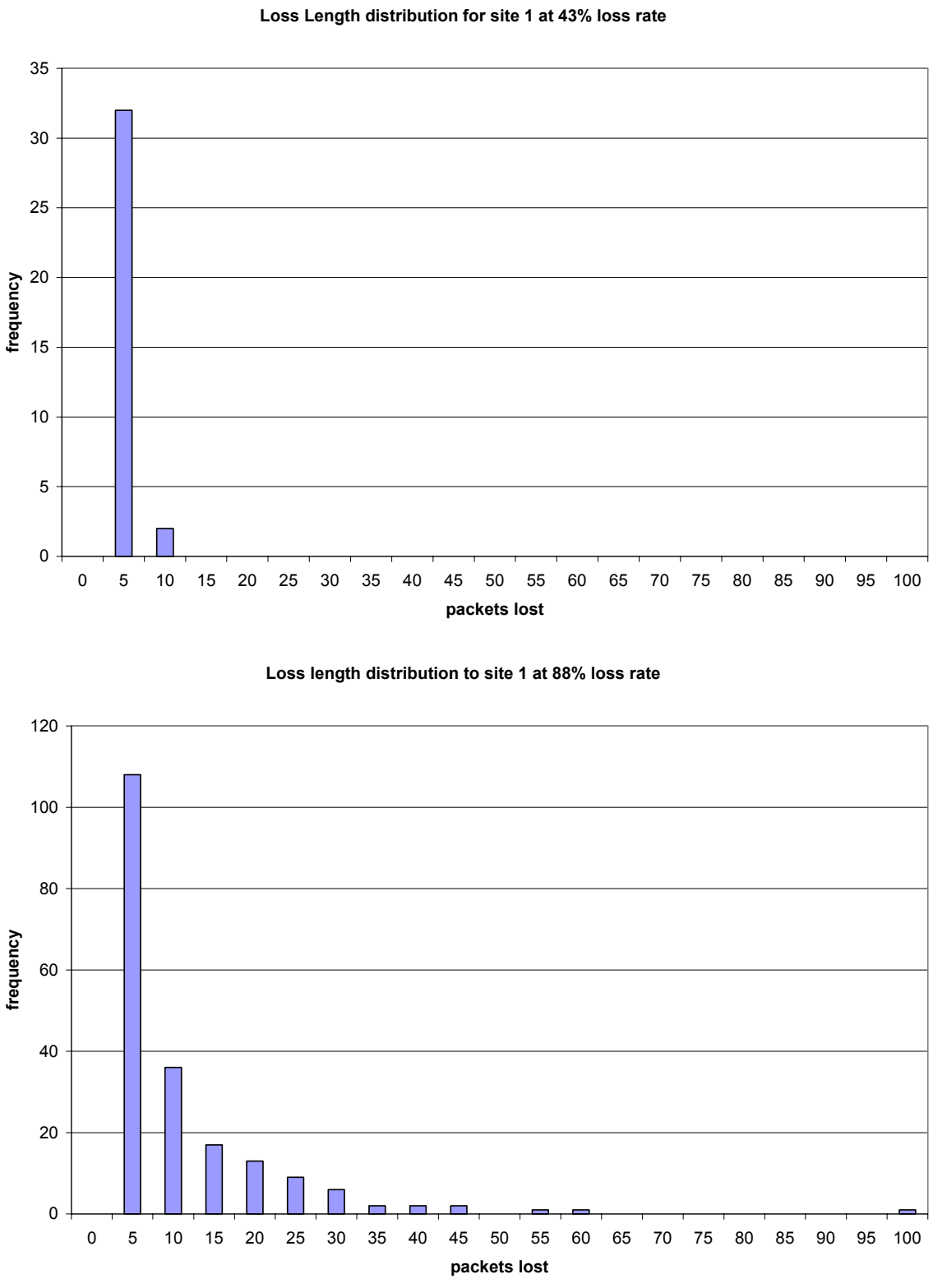


Figure 3.9: Loss length distribution for different loss rates

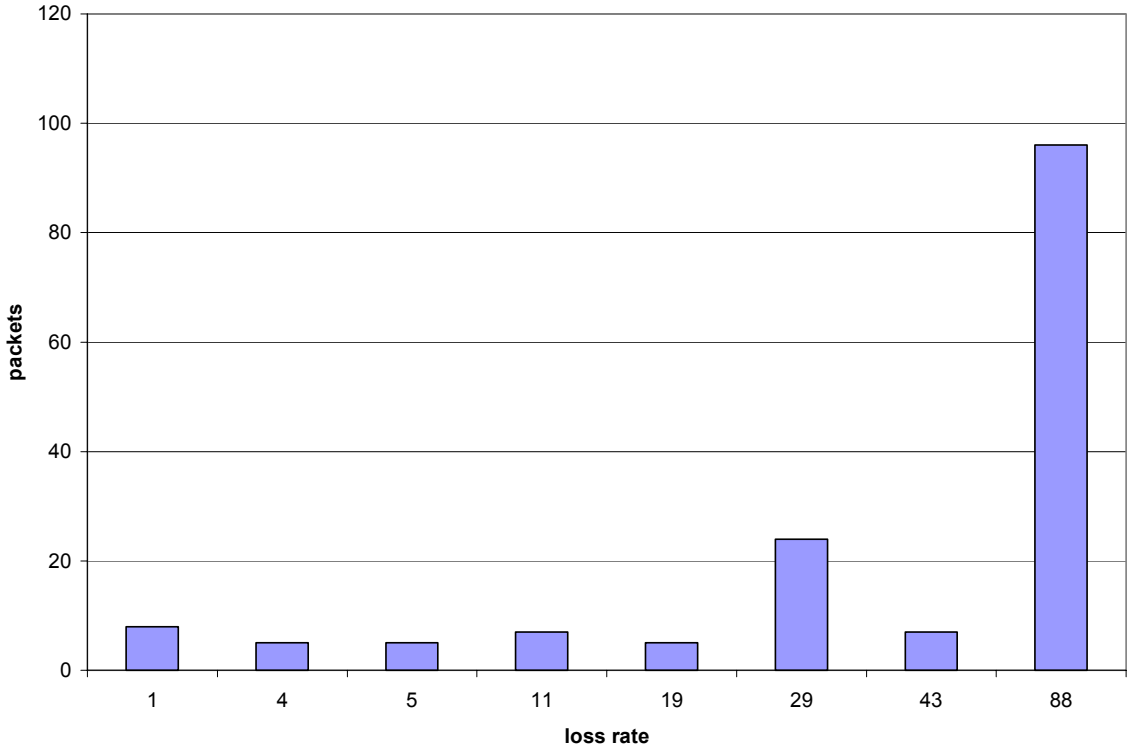


Figure 3.10: The effect of loss rate on burst size

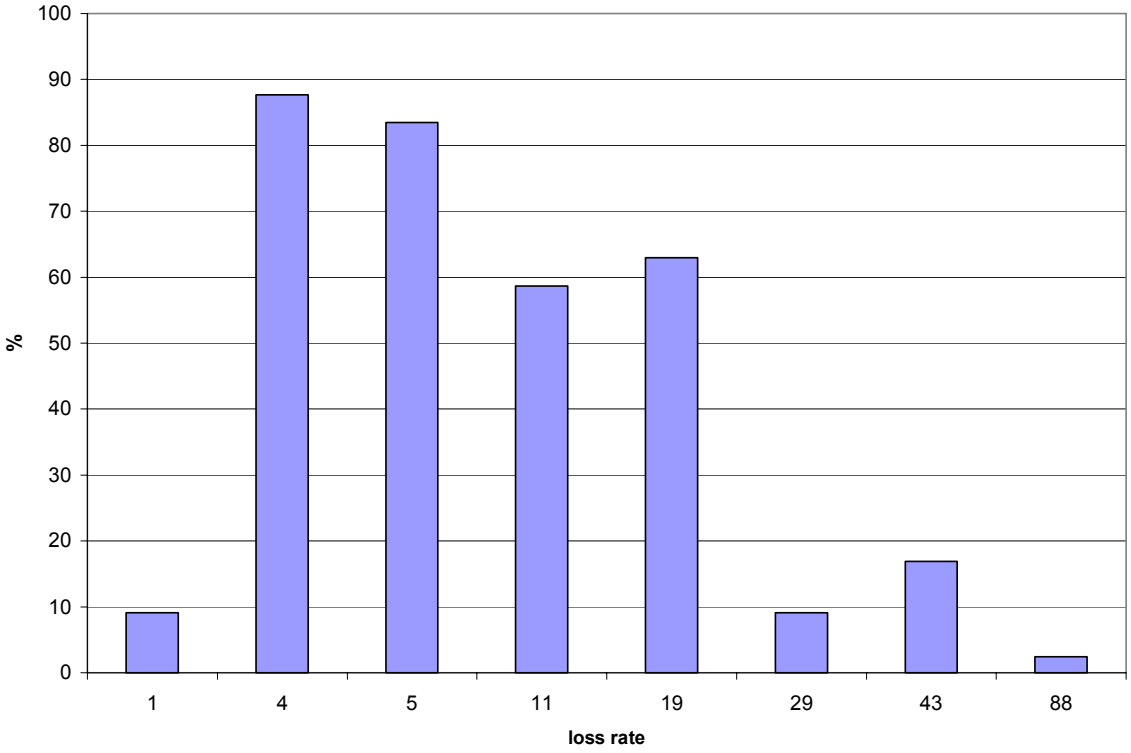
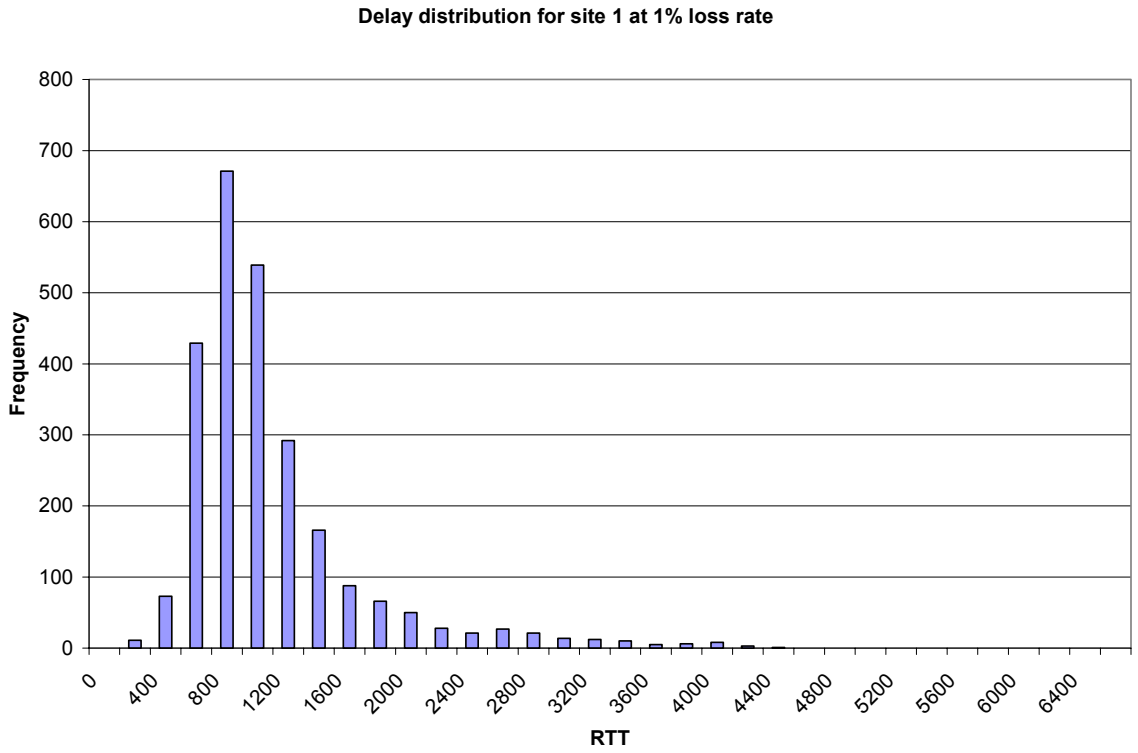


Figure 3.11: Graph of single-packet losses per loss rate

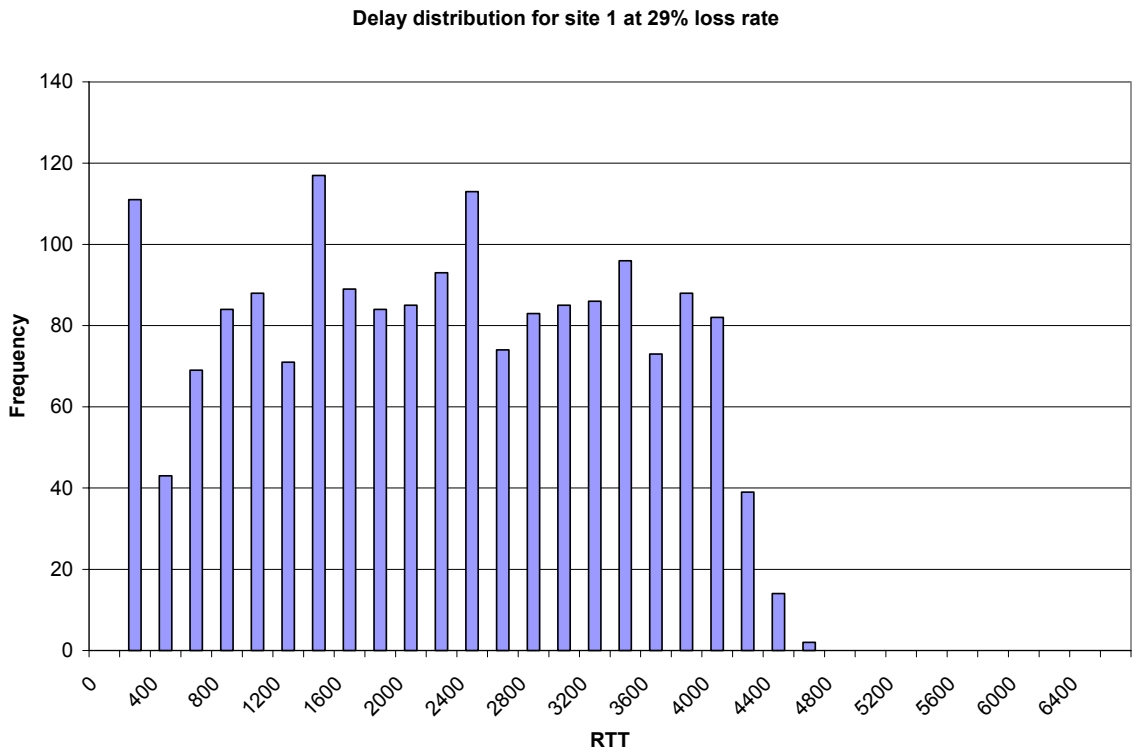
It is also noticed from **Figure 3.11** that the occurrence of single-packet losses decreases as the loss rate increases. From this observation we further infer that when there is no congestion, the packet losses are due to errors, which do not necessarily affect consecutive packets. These losses are isolated, and not due to buffers being full at the intermediary routers. As the congestion increases, more packets are queued at the routers and there is a greater likelihood that consecutive packets will be discarded.

3.3 Effect of Loss on Delay distribution

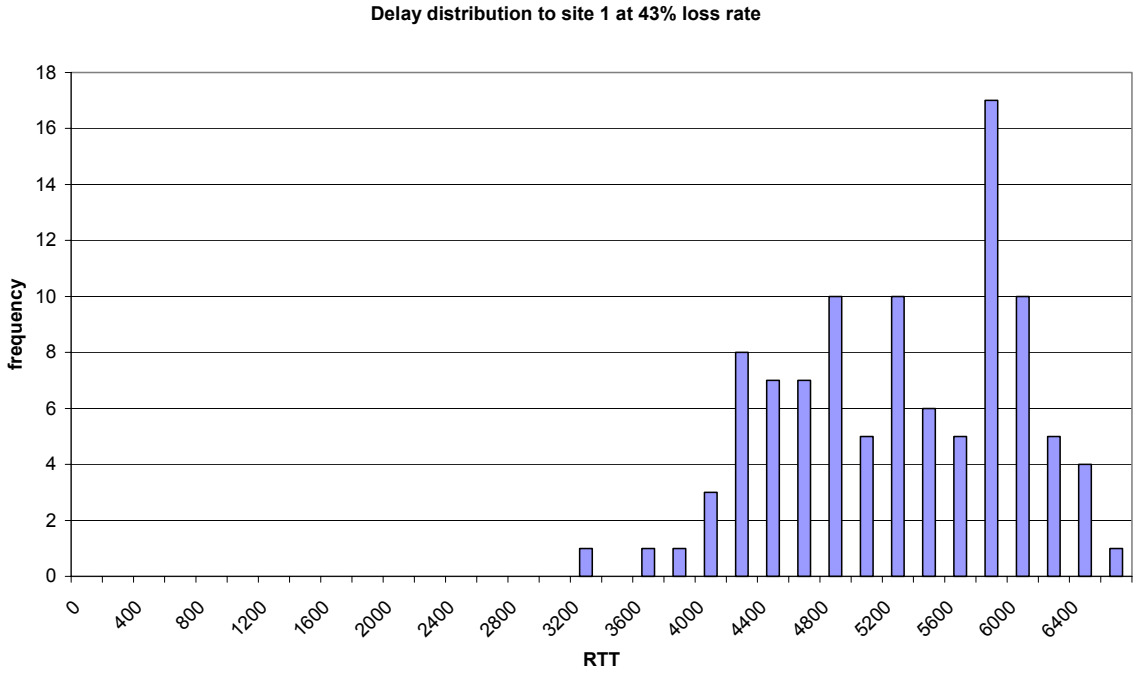
To investigate the effect of increasing packet loss on the delay distribution, RTT traces from one site were observed at different times of day to obtain different loss rates. Site 1 was chosen for this purpose as it had a tendency to have different loss rates at different times of day. **Figure 3.12** shows the various delay distributions and how an increase in the loss rate affects the shape.



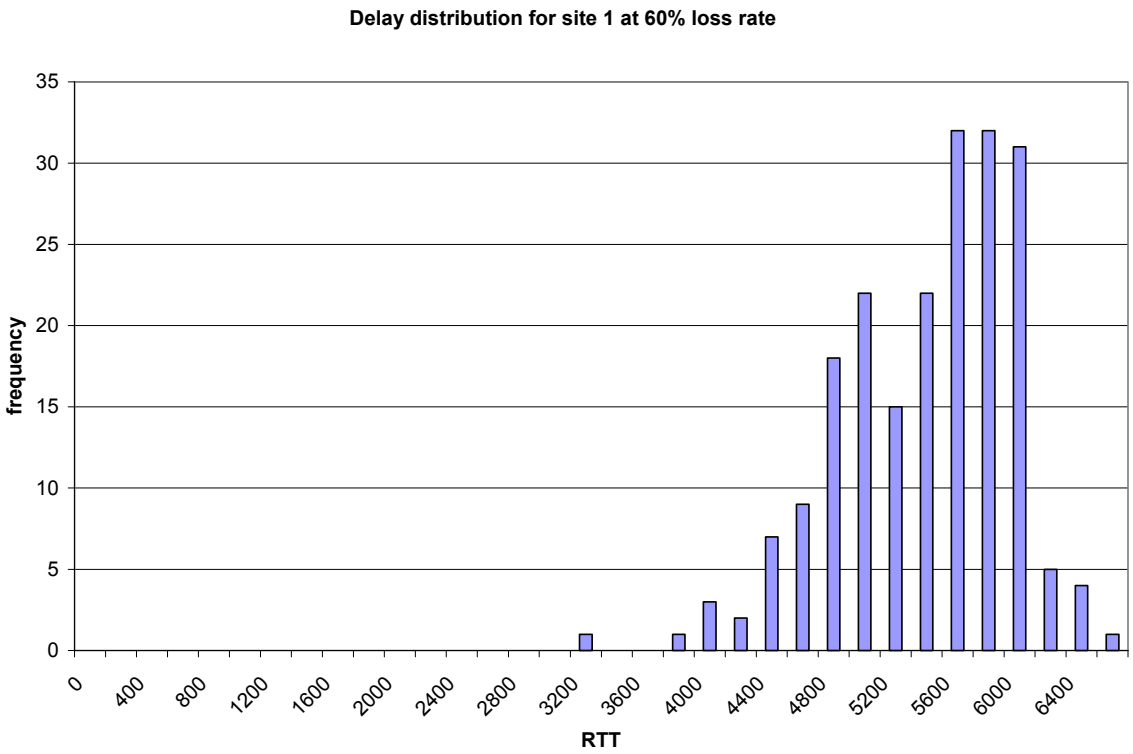
a) Delay distribution for site 1 at 1% loss rate



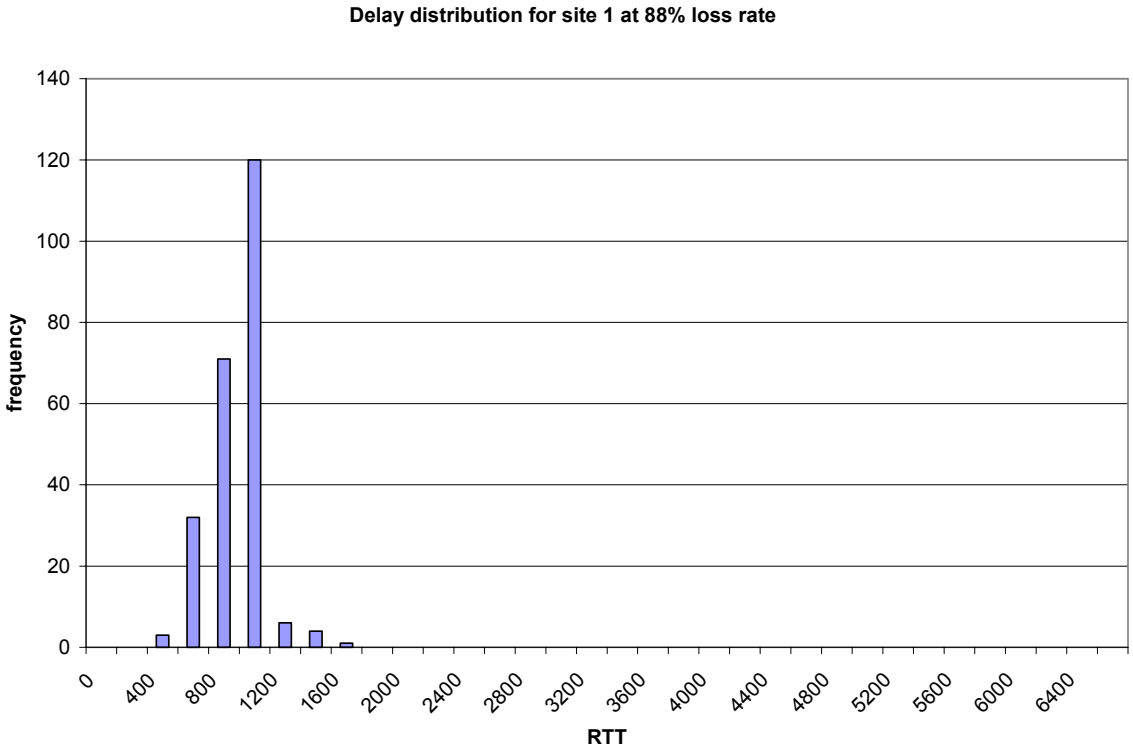
b) Delay distribution for site 1 at 29% loss rate



c) Delay distribution for site 1 at 43% loss rate



d) Delay distribution for site 1 at 60% loss rate



e) Delay distribution for site 1 at 88% loss rate

Figure 3.12: Effects of packet loss on the delay distribution

In the case where the loss rate is 1%, the delay follows a Gamma distribution, with most of the distribution clustered around the mode. As the loss rate increases to 29% (shown in **Figure 3.12**b), the distribution becomes less clustered around the mode, is more dispersed along the axis, starts to appear multi-modal, and does not seem to follow any specific known distribution. Moreover, it is no longer right shaped as in the previous case. As the loss rate increases to 43% and upwards, the distribution shapes become more and more distinctly left-tailed, as shown in **Figure 3.12**c) to e).

4. MODELING THE DELAY AND LOSS PARAMETERS

As discussed in section 2.7, the delay and loss distributions can be modeled using the Weibull. In this chapter, we model the data acquired in chapter 3 and compare the Weibull parameters with the network status. Sections 4.1 and 4.2 of this chapter model the gathered data for delay and loss. In section 4.3, a simulation framework is developed and implemented using ns2.

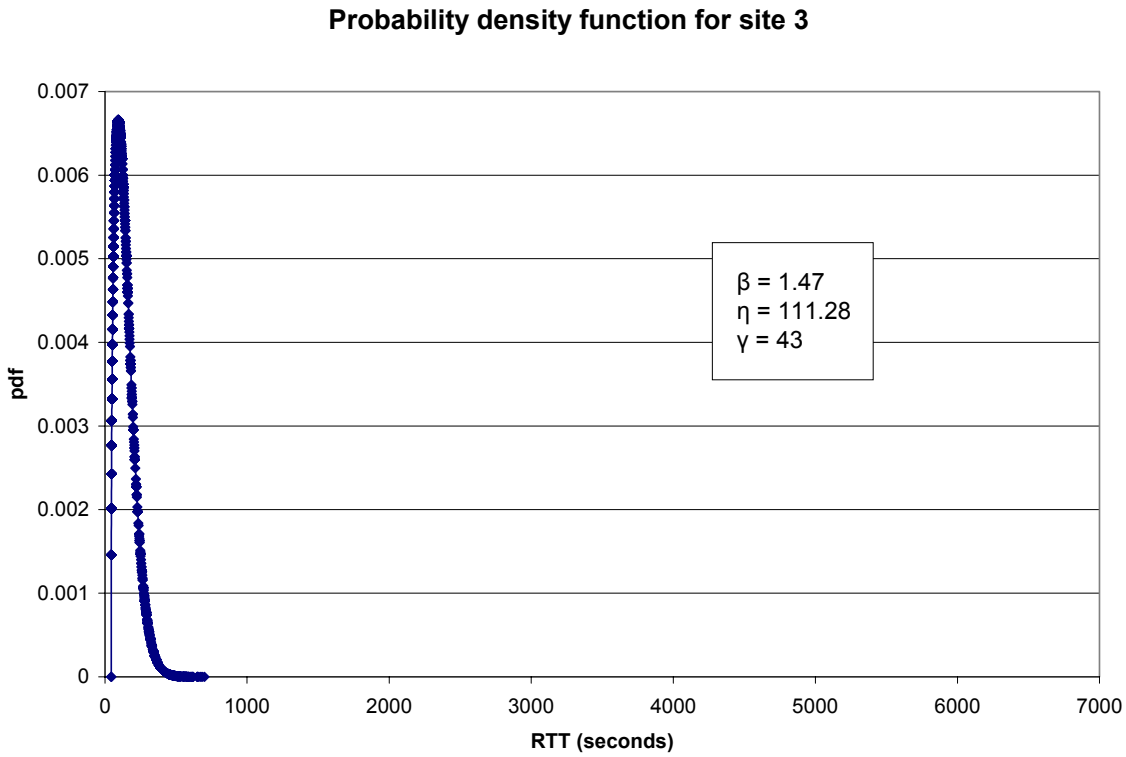
4.1 Modeling delay

The delay is expected to follow a Pareto distribution when the network is not congested, a gamma distribution for a moderately congested network, and a left-tailed distribution for highly congested network. Using the Weibull, this translates to $\beta<2.6$ for a non-congested, $2.6<\beta<3.7$ for a moderately congested network, and $\beta>3.7$ for a highly congested network. Equation 2.3 is used to compute the parameters of the Weibull, includingβ, and relevant graphs for the probability function are drawn. The results are compared to those found in chapter 3, especially the histograms.

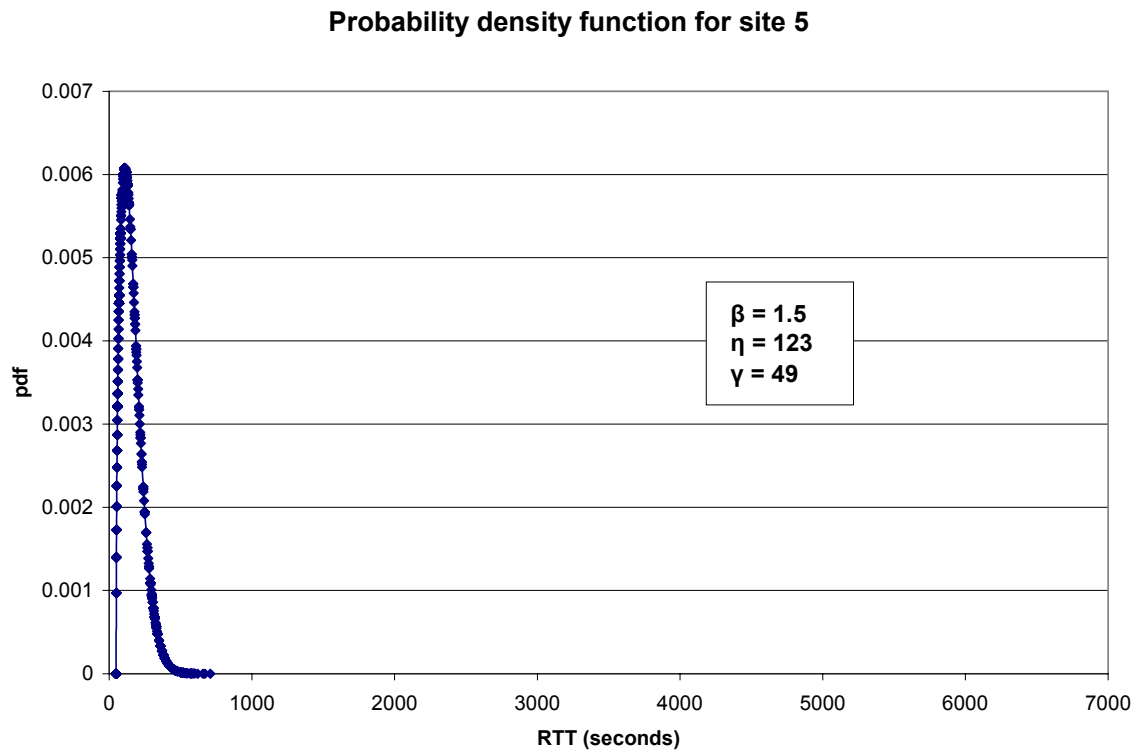
Table 4.1: Weibull distribution parameters for the different delays

Site	Loss (%)	β	η	γ
1	29	1.31	2,425.47	59
1	1	2.46	1,091.68	60
1	88	5.25	876.64	321
1	43	8.10	5,430.77	3,138
3	0	1.47	111.28	43
4	11	1.29	482.12	52
5	19	1.53	123.28	49
7	0	1.93	151.18	59
16	3	10.27	221.97	123
18	0	12.29	432.60	360
19	1	15.34	592.40	513
20	0	13.10	390.63	333
22	0	7.13	663.68	506

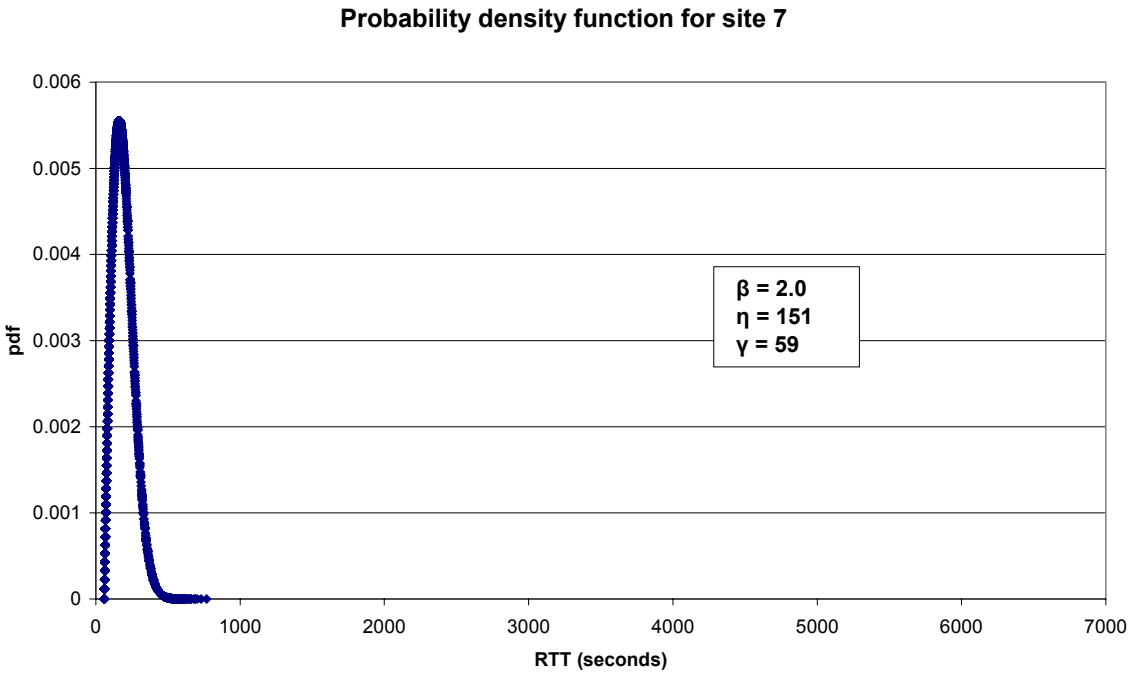
The Weibull parameters are shown in **Table 4.1**. The main interest is in the shape parameter, β. The sites that were determined to be non-congested, such as site 3 and 5, show a β value less than 2.6 (β=1.47 and β=1.53 respectively). For Site 1, when it was not congested, a β value of 1.31 is realized. When it was moderately congested, a β value of 2.46 (close to 2.6) is realized. When it was highly congested, a β value of 5.25 is realized.



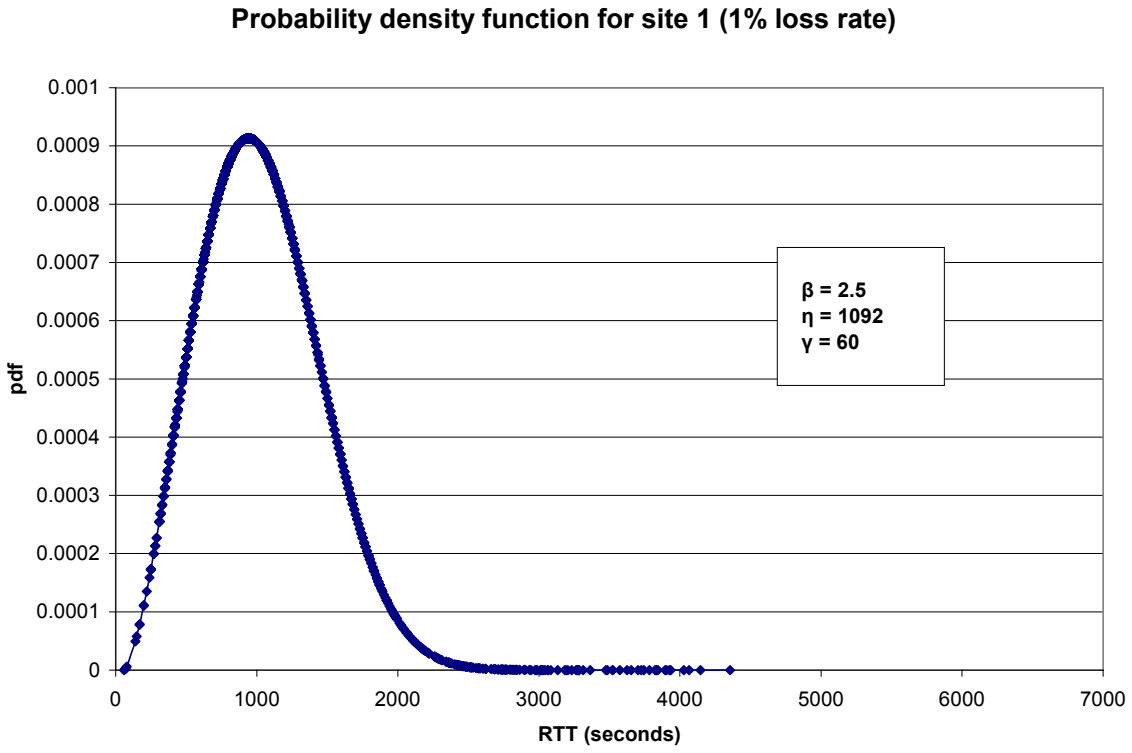
a) pdf for non-congested site at 0% loss rate



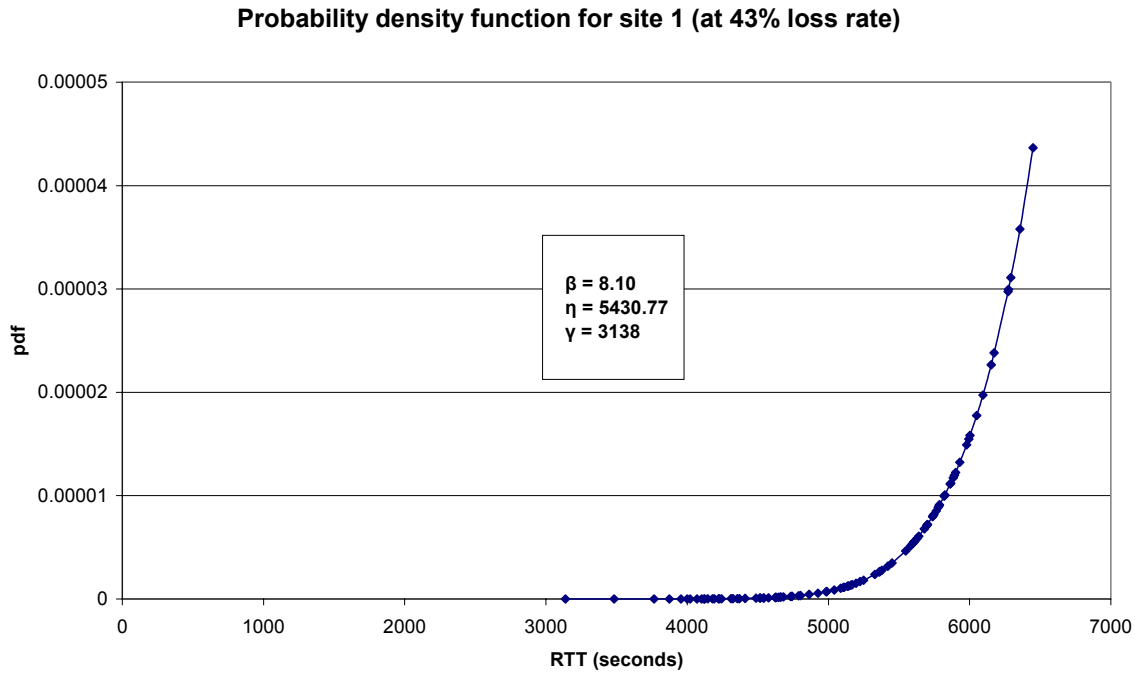
b) pdf for non-congested site at 19% loss rate



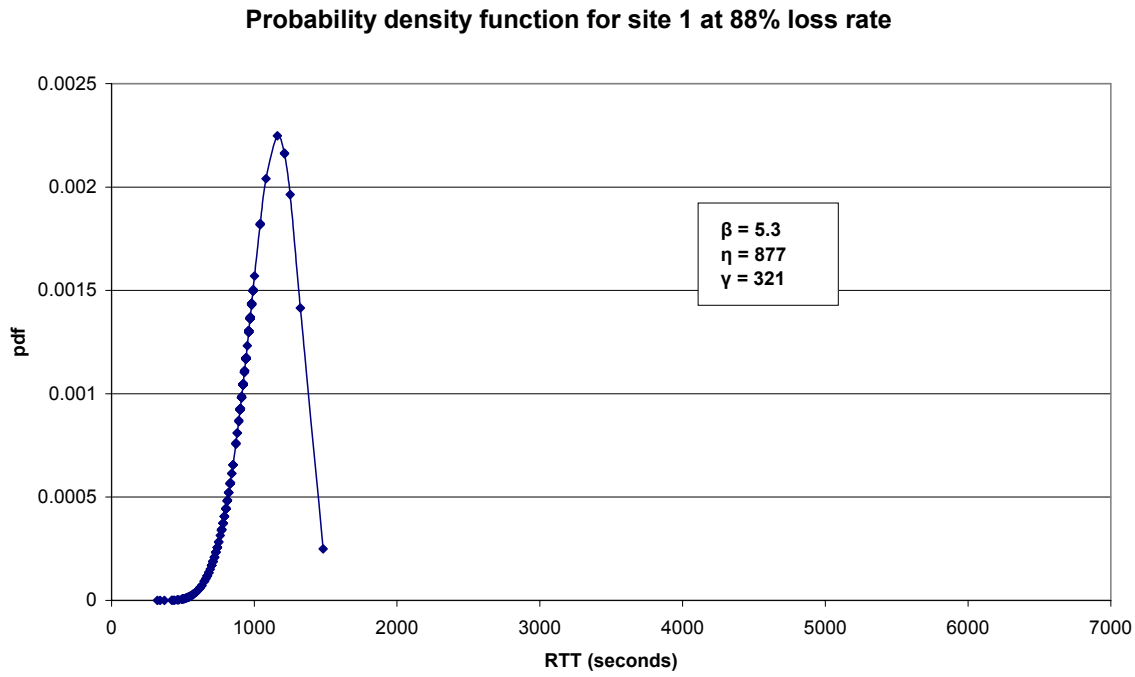
c) pdf for moderately congested site



d) pdf for moderately congested site at 1% loss rate



e) pdf for a heavily congested site at 43% loss rate

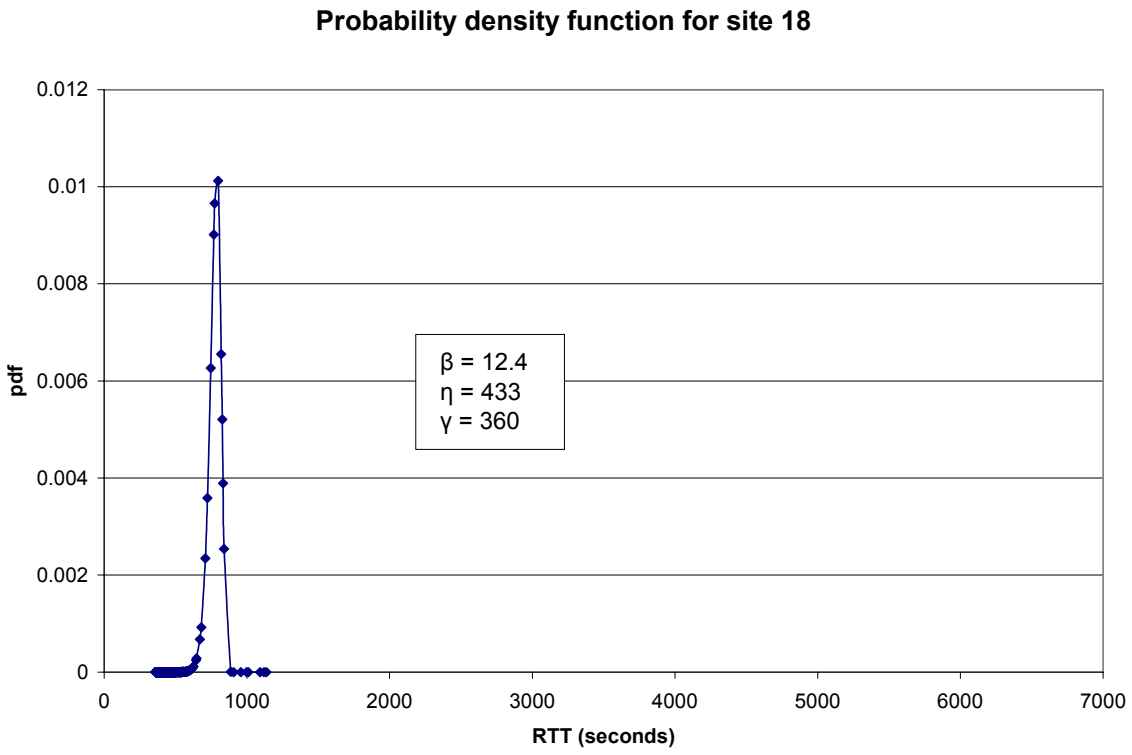


f) pdf for a heavily congested site at 88% loss rate

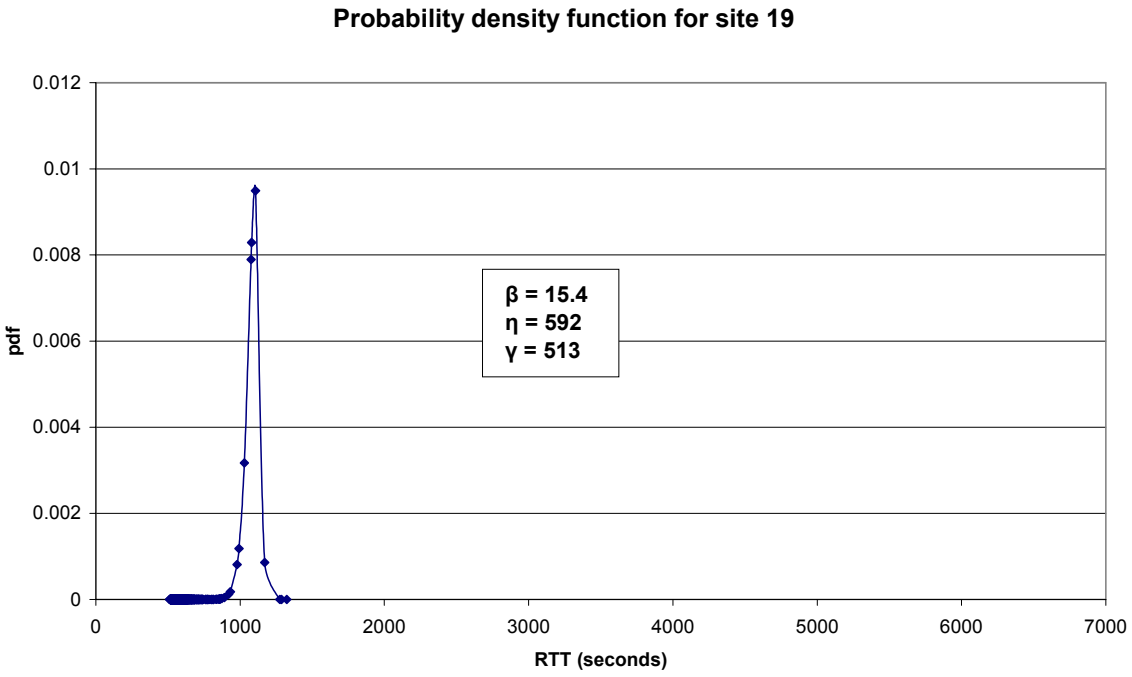
Figure 4.1: Probability density function plots for delay

Figure 4.1 shows the *pdf* plots for the computed parameters. The results resemble the histograms in Section 3.1. A right-tailed distribution, signaling a non-congested or a moderately congested network, is manifested by Figure 4.1a) similar to Figure 3.1 in chapter 3. A left-tailed distribution, signaling a highly congested network, is manifested by Figure 4.1e) and it is similar to Figure 3.3 in chapter 3.

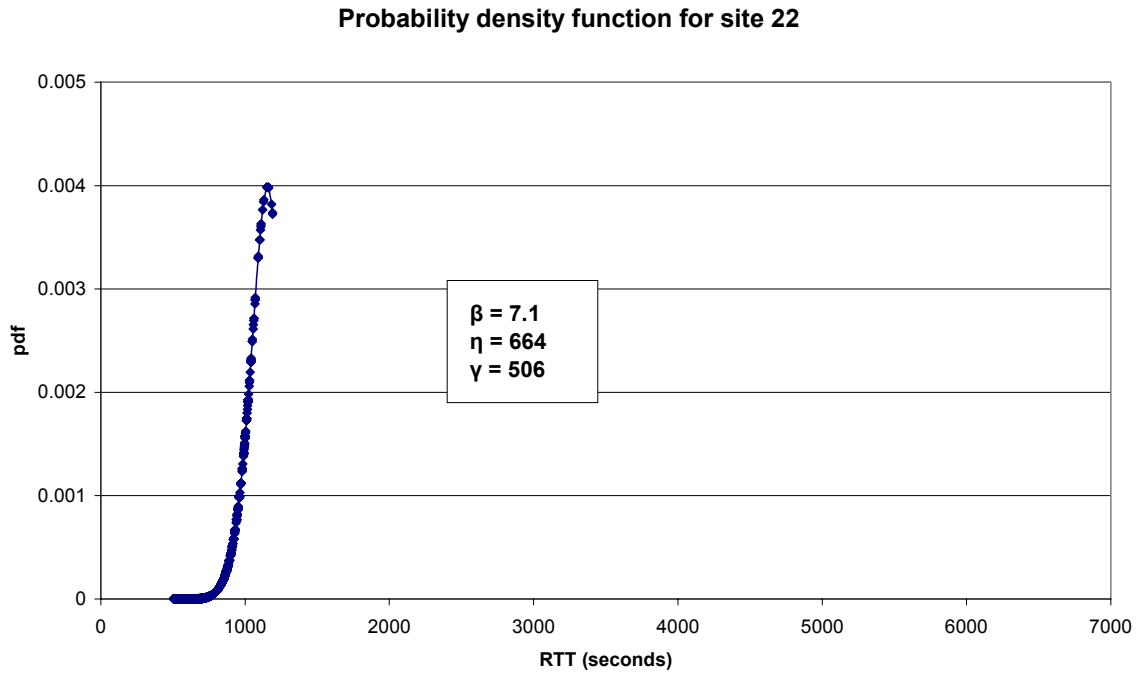
With respect to the overseas sites, it is quite clear that they have left-tail distribution (as shown in **Figure 4.2**), which can be attributed to distance.



a) pdf for delay to site 18



b) pdf for delay to site 19



c) pdf for delay to site 22

Figure 4.2: Probability density function plots for delay to overseas sites

4.1.1 Hypothesis testing

Now we must establish that the observed differences in values of the shape parameter are indeed due to the congestion status of the network, and are not a result of chance. To do this, we will resample the values of beta, given in Table 4.2 below.

Table 4.2: BETA VALUES

Not congested	Congested	Heavily congested
1.29	1.93	5.25
1.31	2.46	7.13
1.43		8.10
1.47		10.27
1.53		12.29
1.41		13.10
		13.80
		15.34

For the test, we compared the values for the two extreme columns with the most number of instances – the “not congested” and “heavily congested” columns. Since the two columns have different number of elements in them, we used the re-sampling tool [52] to add two more numbers to the first column. After the procedure, the two columns looked as follows:

Table 4.3: COMPARISON FOR BETA VALUES

Not congested	Heavily congested
1.47	5.25
1.53	7.13
1.29	8.10
1.29	10.27
1.29	12.29
1.47	13.10
1.53	13.80
1.31	15.34

The averages for these two columns are 1.40 and 10.66, for the not congested and heavily congested paths, respectively, with a difference of -9.26. The null hypothesis in this case is that these average values are a product of chance, not congestion, and if a sufficient number of samples is taken, it is just as likely that the difference will be greater than or less than the observed value. In other words, it is likely that we will find many cases whereby the difference is smaller than or equal to the one observed above. To test whether the null hypothesis is true, we took another sample from the small set of possibilities (consisting of values in Table 4.3), calculated the averages and then noted the difference between the two. This procedure was repeated 2,000 times and the differences were noted each time. From the resulting 2,000 values, we note that *none* of them were less than -9.26, meaning that the null hypothesis is not true! The difference in the beta values was not due to chance.

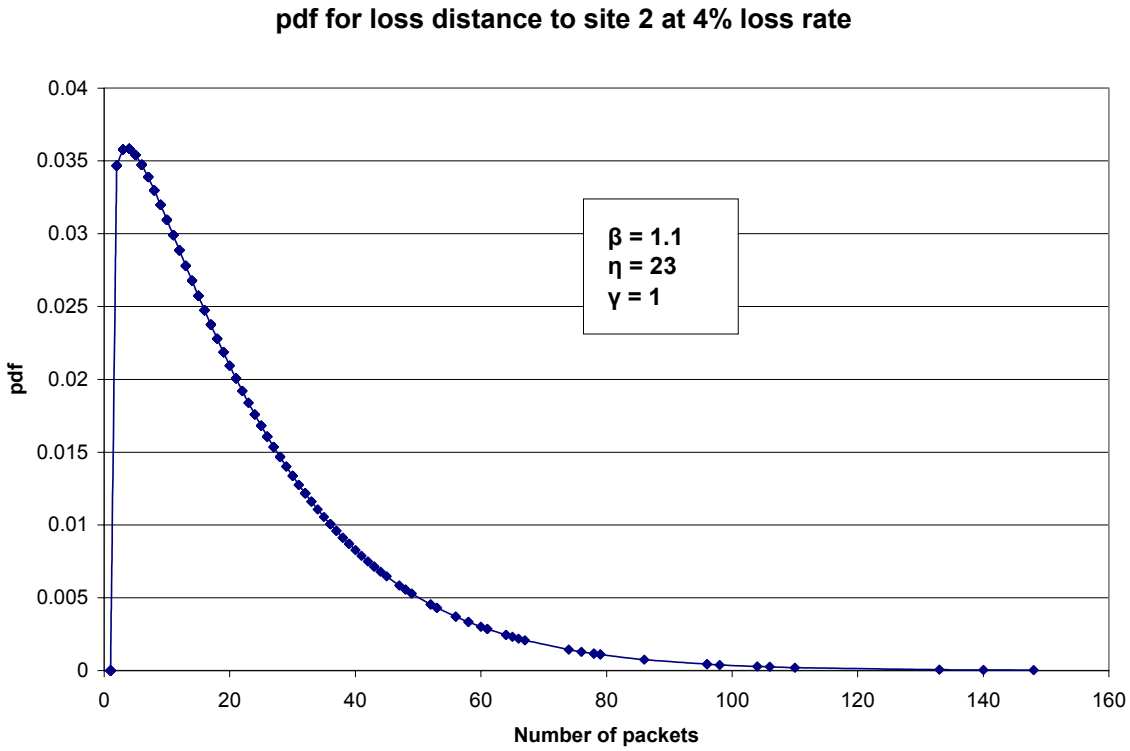
4.2 Modeling packet loss

4.2.1 Loss Distance

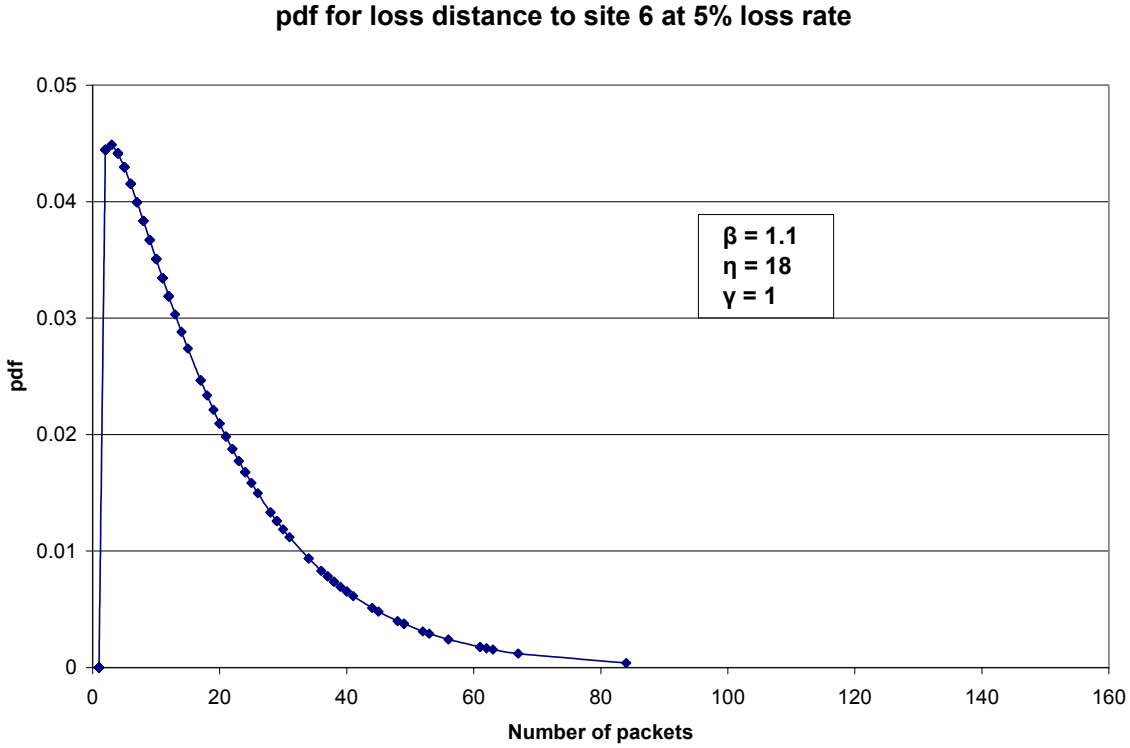
In chapter 3 it was observed that a loss distance of zero becomes more prominent when there is congestion. This requires that the shape parameter, β, be approximately 1. When observing **Figure 4.3**, it is observed that the relationship between the shape of the graph, in other words β, and the congestion state of the path is not apparent. Although the value of β seems to be increasing with the loss rate (from 1.1 for 4% loss rate to 1.8 for 88%), it is always less than 2.6. This agrees with our assertion in chapter 3 that loss distance tends to follow a right tailed distribution – either a Pareto or a Gamma distribution.

A more useful parameter is the scaling parameter, η, which tends to decrease as congestion increases. Looking at the graphs, we note that as the loss rate increases, the graphs become less spread out across the x-axis. A graph of the scaling parameter versus loss rate, shown in **Figure 4.4**, further confirms this point. An explanation for this behavior is that as the

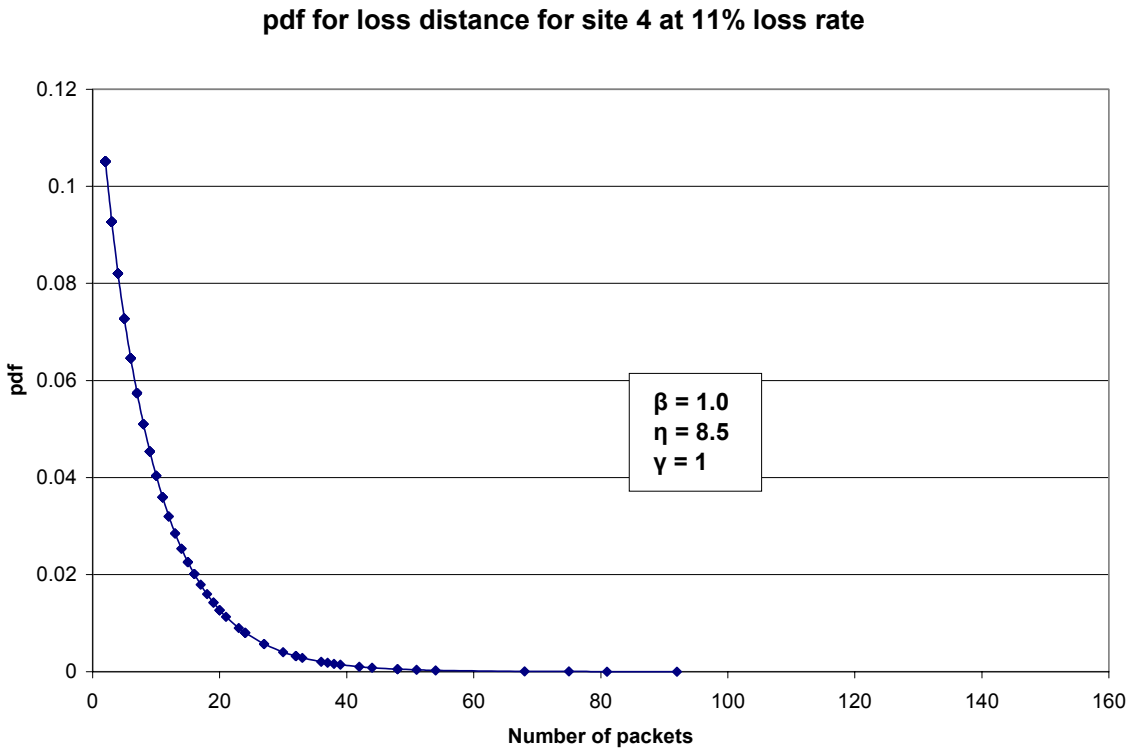
level of congestion increases, we expect that more packets will be discarded, which decreases the number of successfully delivered packets. This means that the mean loss distance value will be decreased, since fewer consecutive packets are successfully delivered.



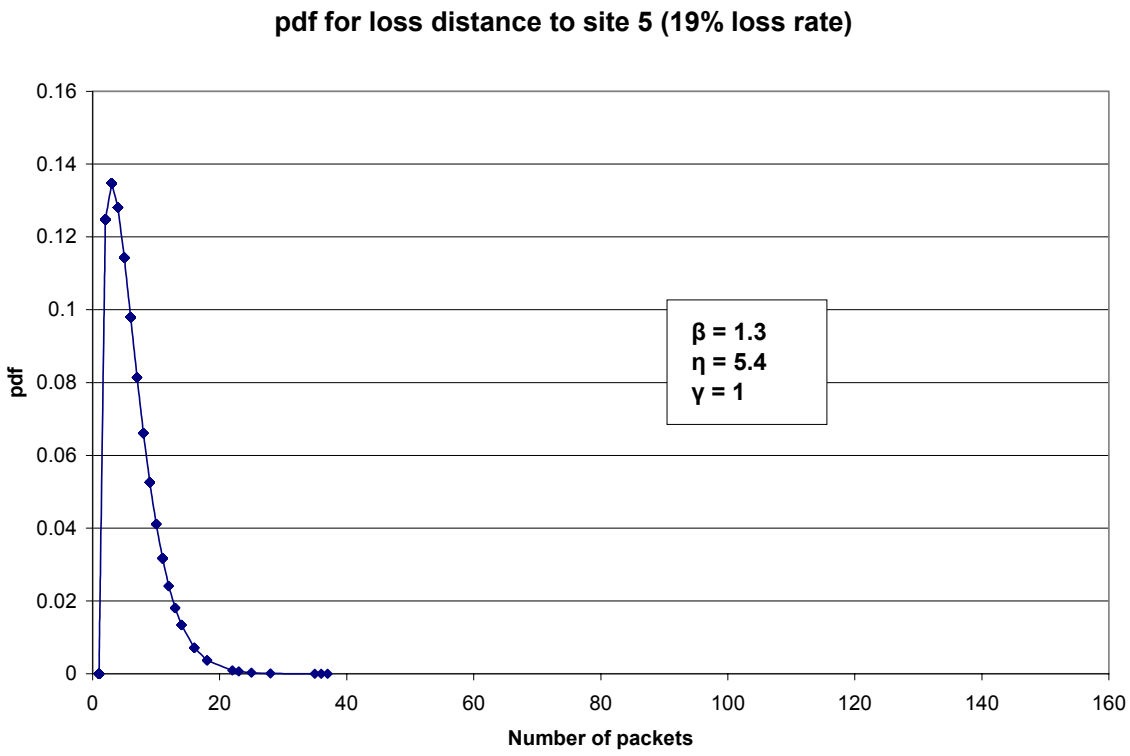
a) pdf for site 2 at 4% loss rate



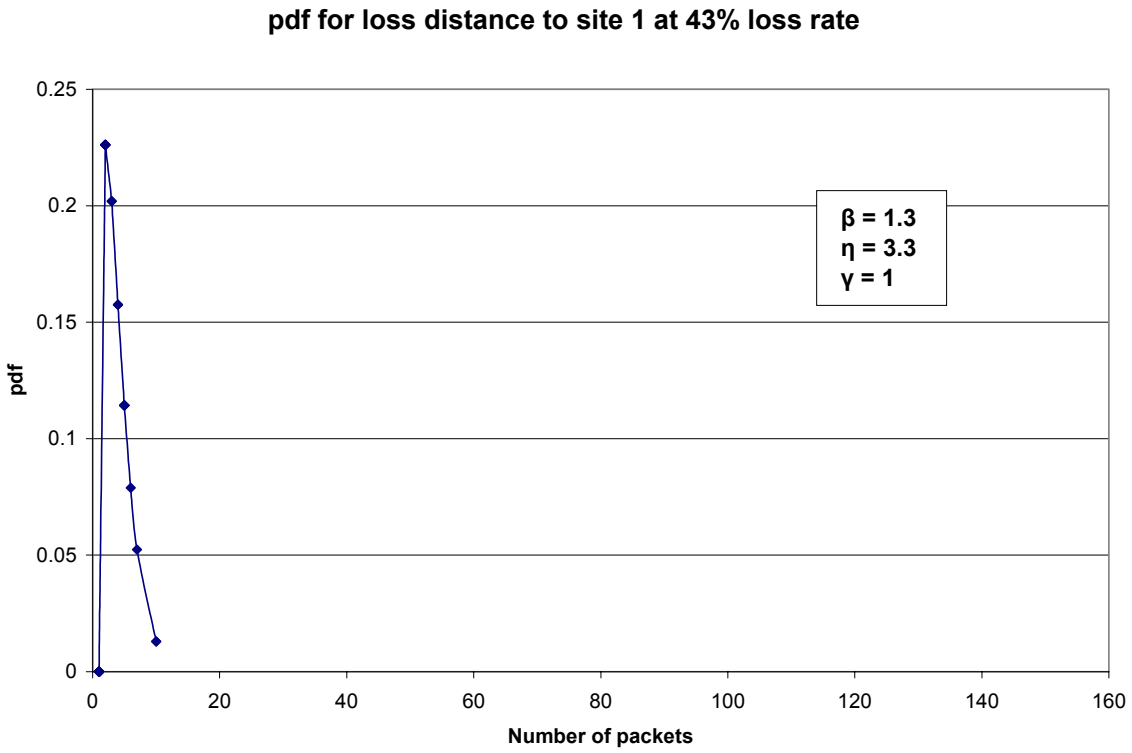
b) pdf for site 6 at 5% loss rate



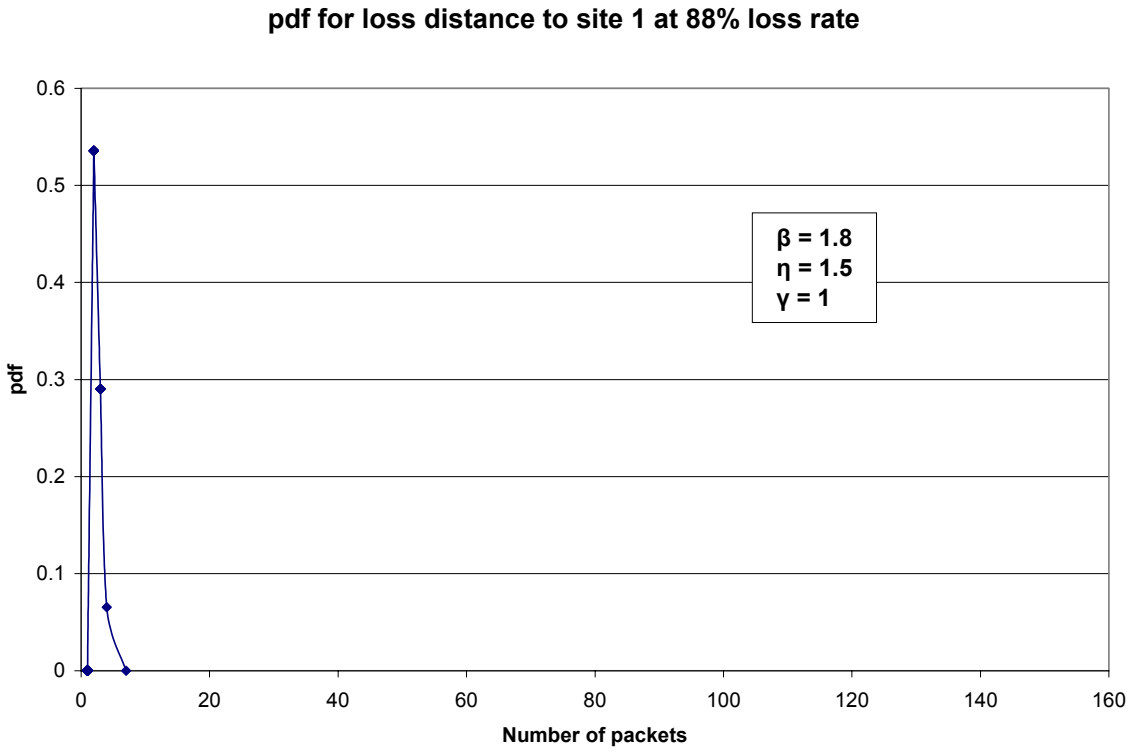
c) pdf for site 4 at 11% loss rate



d) pdf for site 5 at 19% loss rate



e) pdf for site 1 at 43% loss rate



f) pdf for site 1 at 88% loss rate

Figure 4.3: Probability density function for loss distance at different loss rates

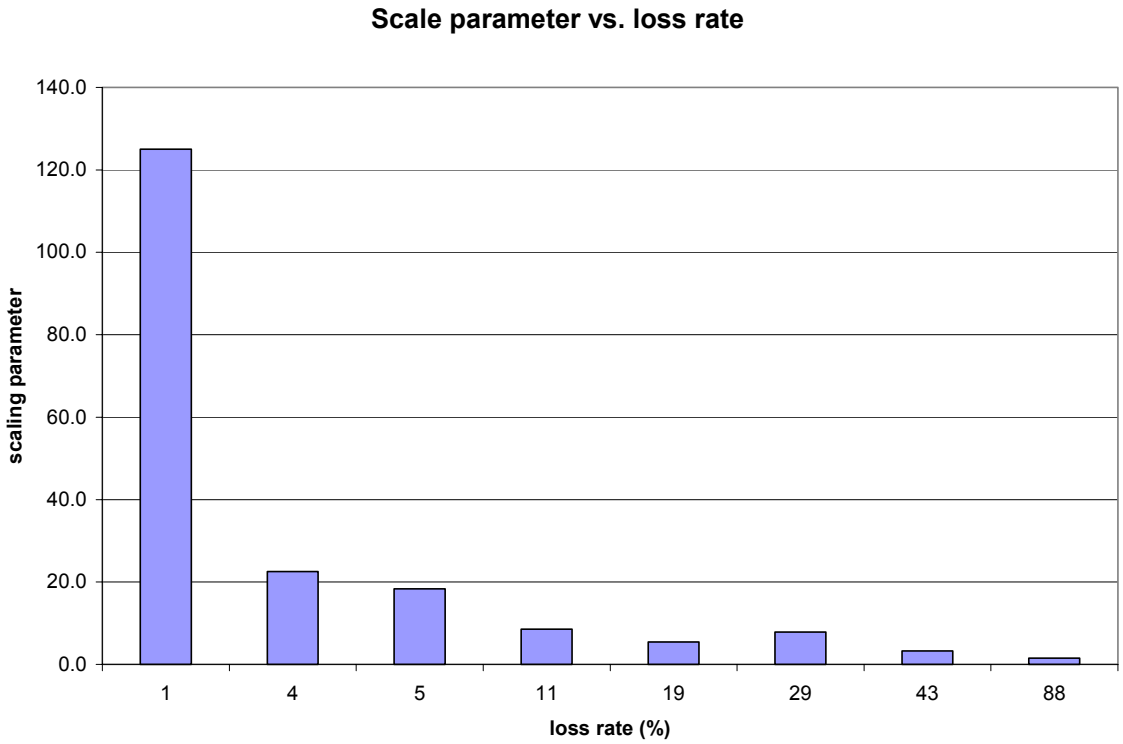
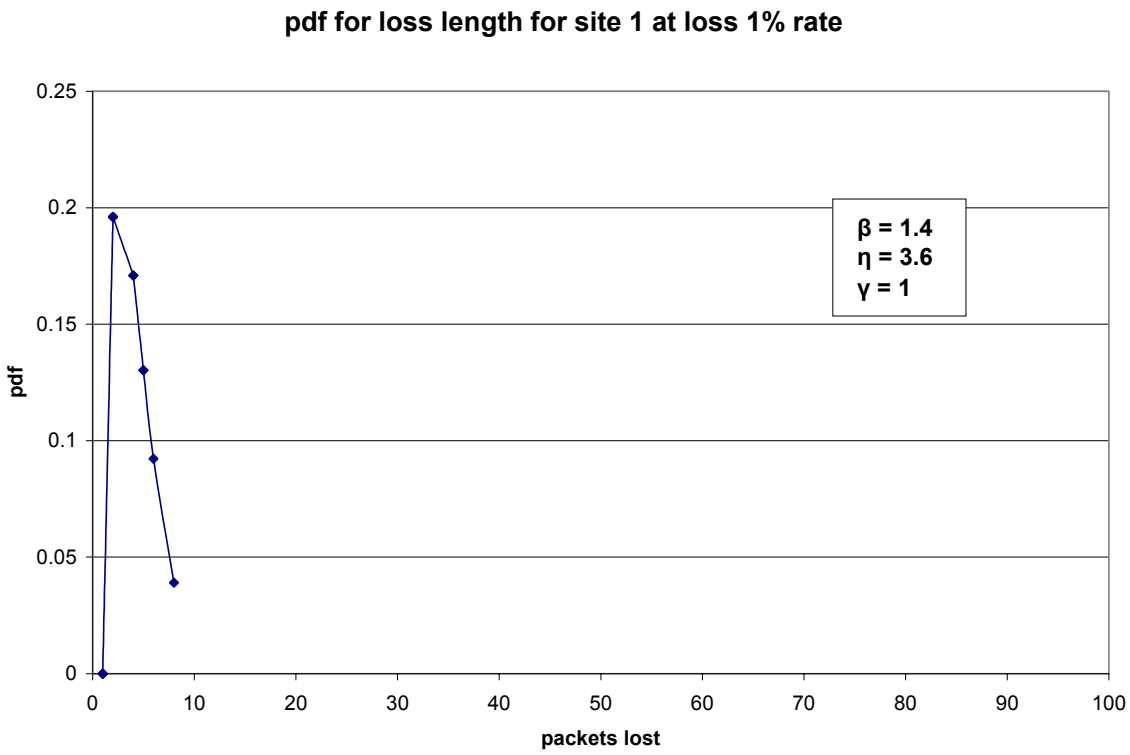


Figure 4.4: The scaling parameter vs. loss rate

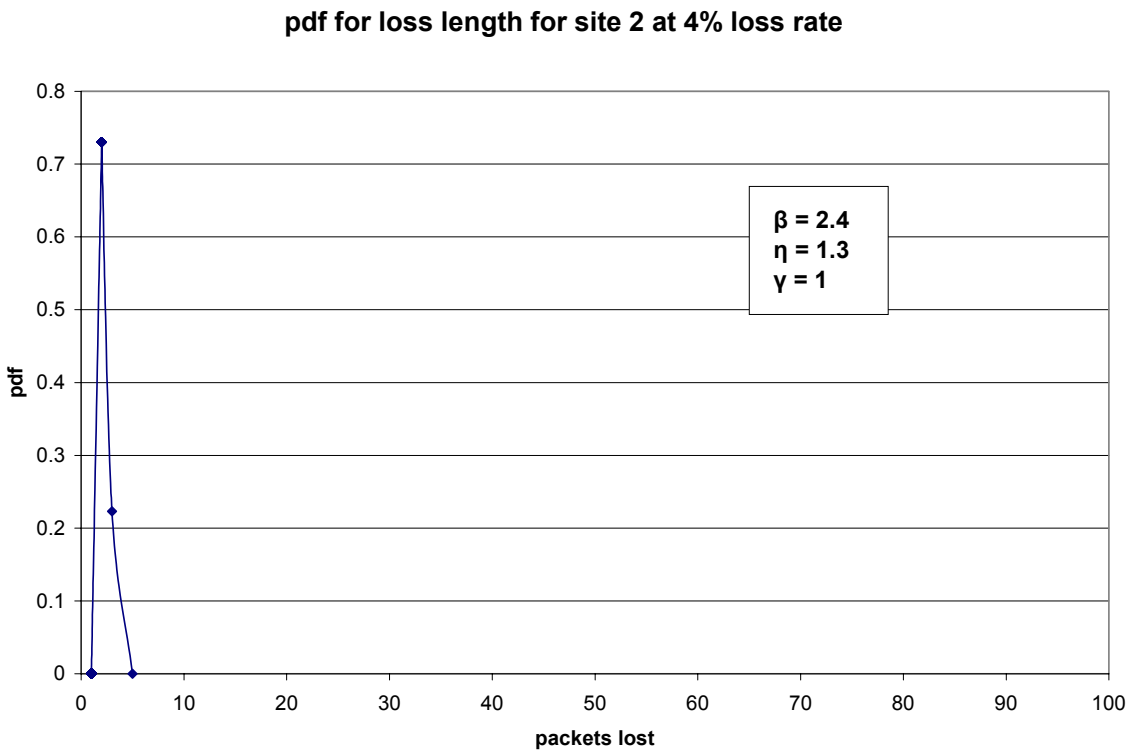
4.2.2 Loss length

Similar to loss distance, the probability density functions for loss length show a right tailed distribution and the shape parameter, β, is always less than 2.6 regardless of the loss rate, as shown in **Figure 4.5**. This supports the assertion from chapter 3 that loss length follows a Pareto or Gamma distribution. The scale parameter, η, tends to increase with increasing congestion, as shown in **Figure 4.6**. This makes sense since it was shown in Fig 3.7 (section 3.2.1) that as congestion increases, then packet losses become burstier, i.e. more packets are lost per loss event. A wider range of packets lost is thus covered, hence the increase in scaling parameter.

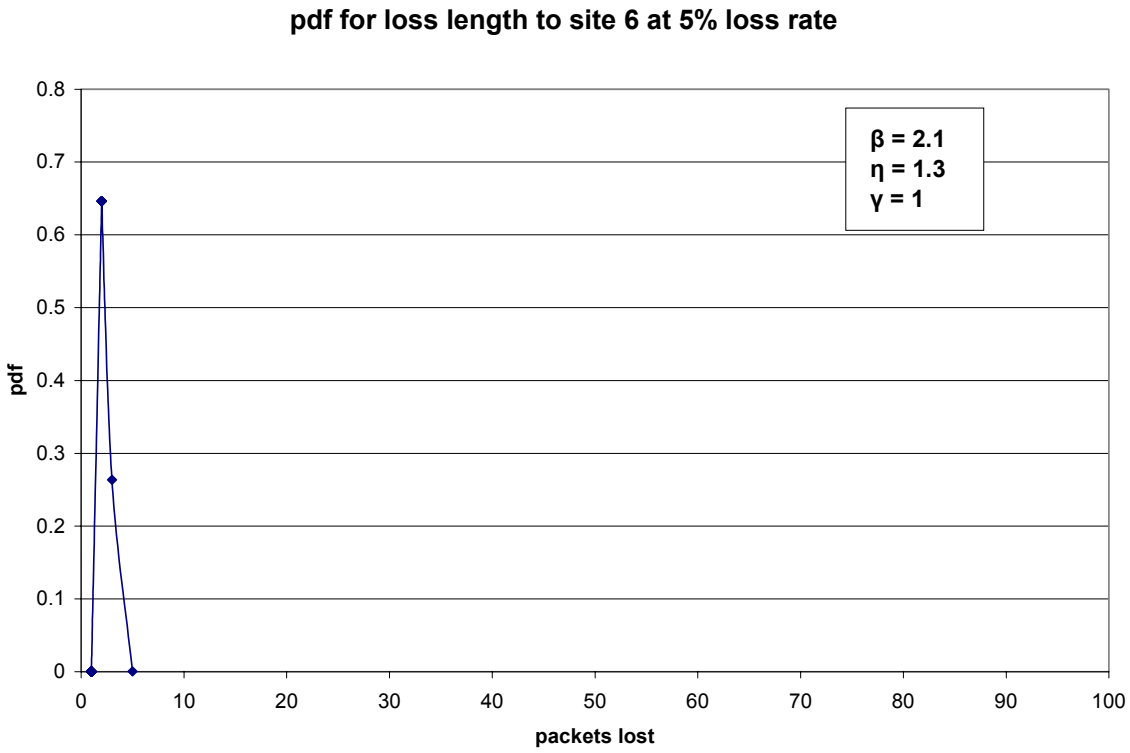
We notice that for site 1 (with 1% loss rate) the scale parameter is higher than sites 2, 6, 4 and 5 even though they have higher loss rates (4%, 5%, 11% and 19%, respectively). This is not surprising since it has already been established in section 3.2.1 that site 1 was in fact moderately congested. Since for every loss event the minimum number of packets that can be lost is 1, the location parameter, γ, is always 1. This parameter does not contribute much to the discussions, hence it is largely ignored.



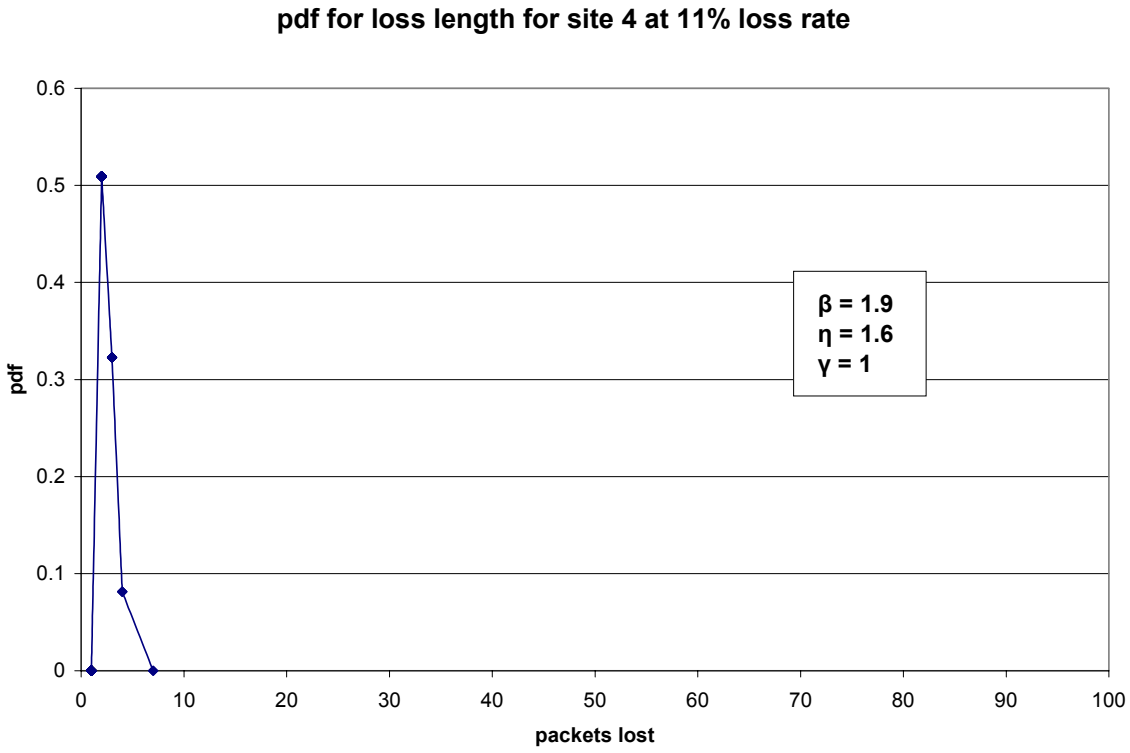
a) pdf for loss length to site 1 at 1% loss rate



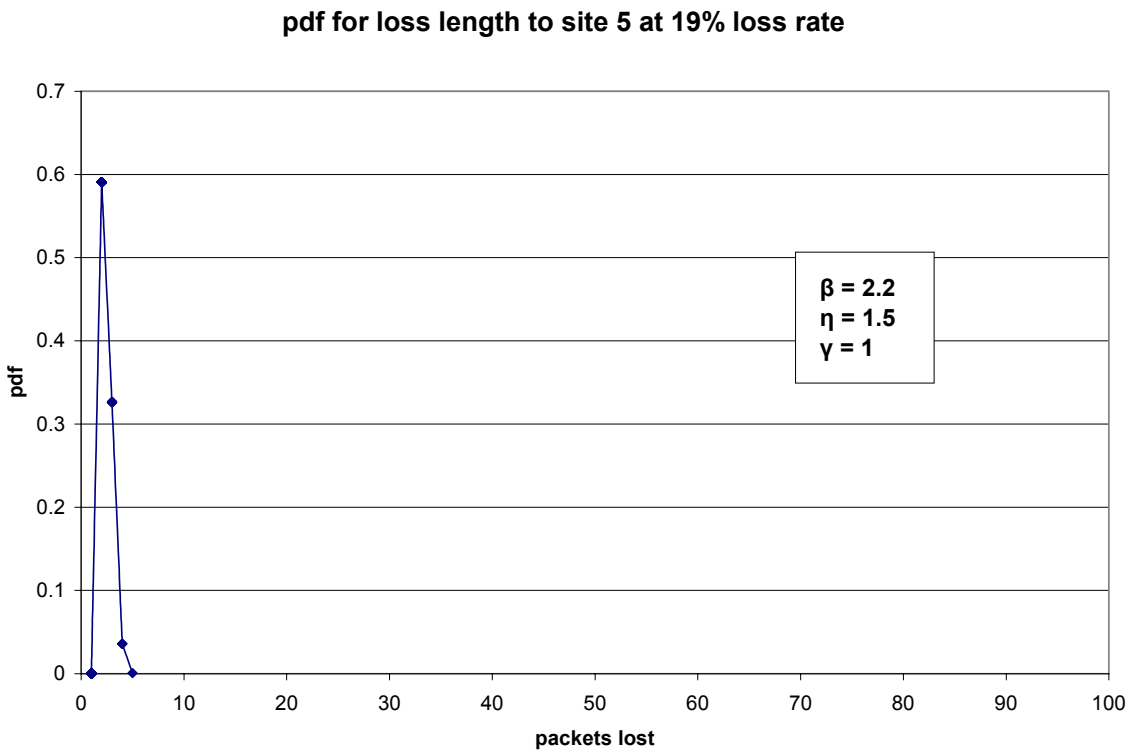
b) pdf for loss length to site 2 at 4% loss rate



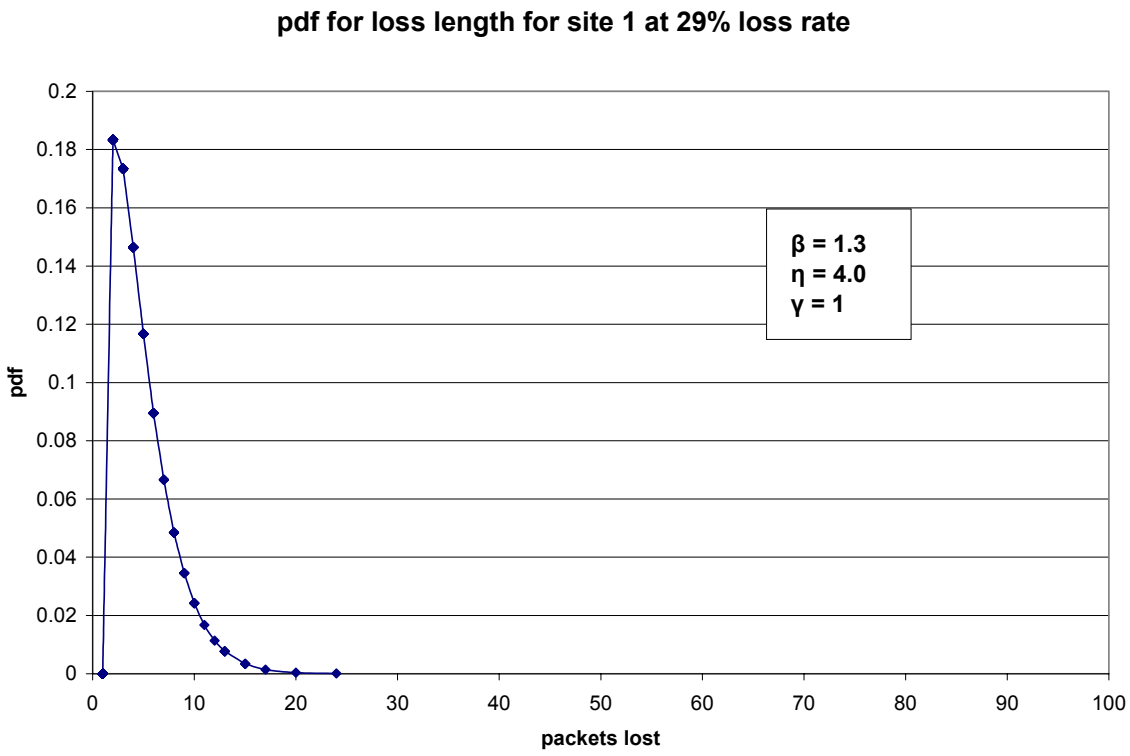
c) pdf for loss length to site 6 at 5% loss rate



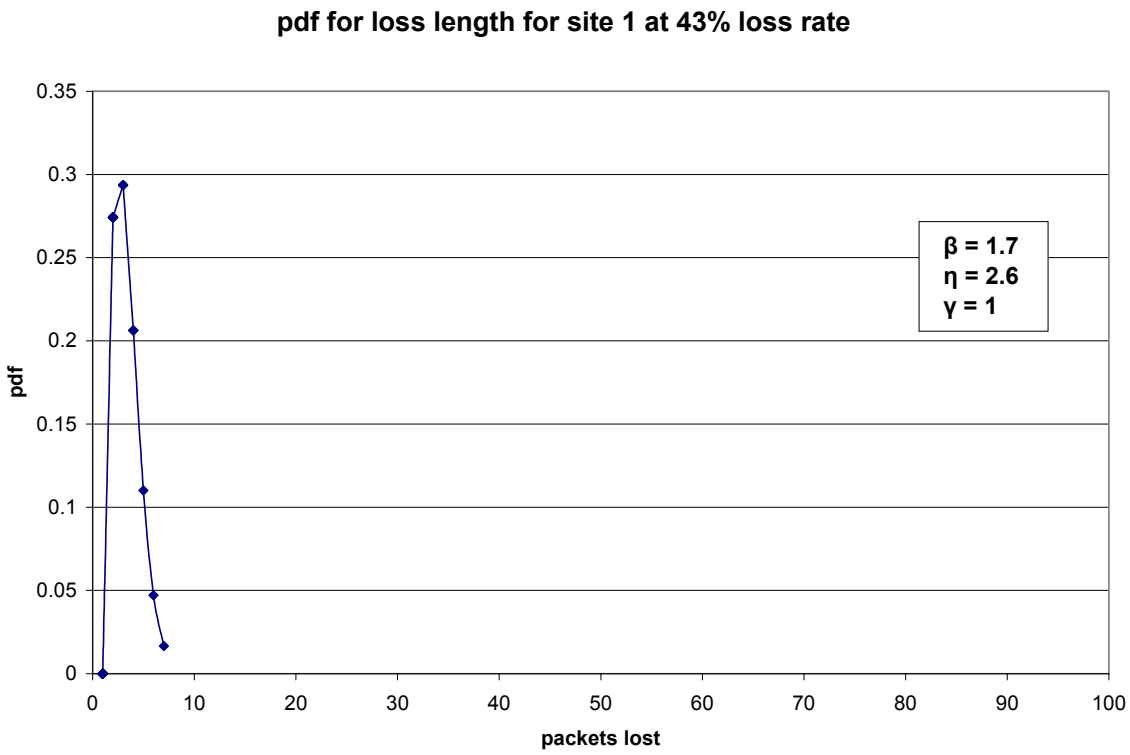
d) pdf for loss length to site 4 at 11% loss rate



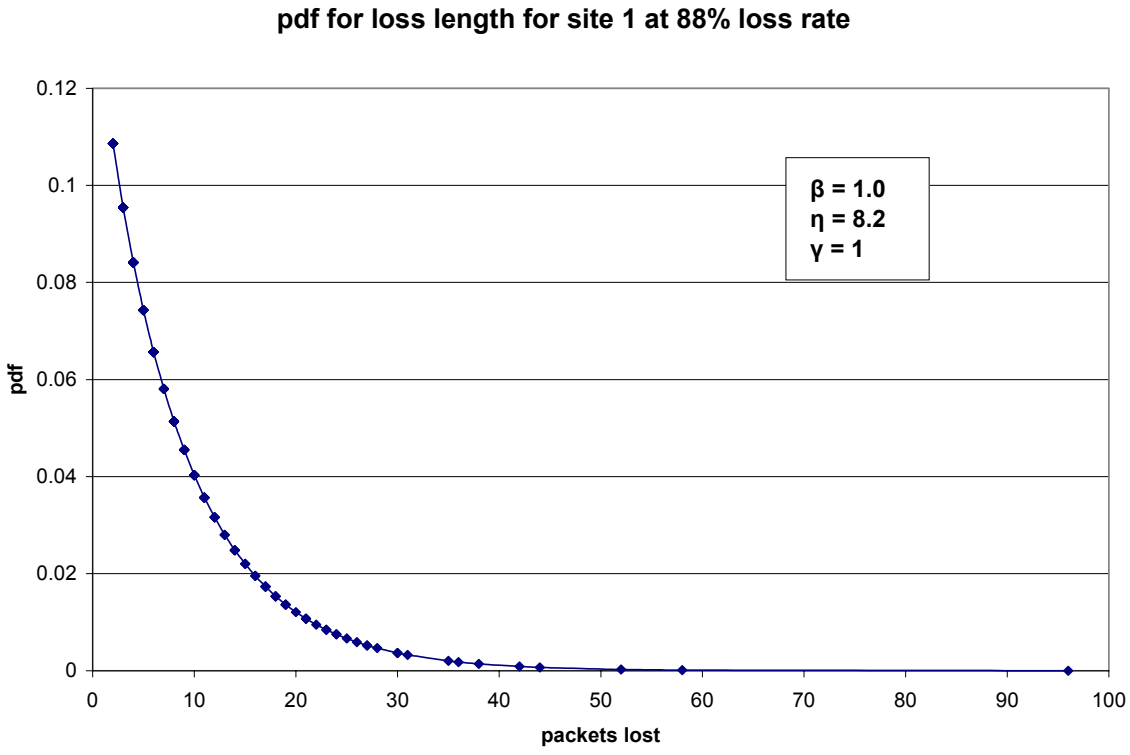
e) pdf for loss length to site 5 at 19% loss rate



f) pdf for loss length to site 1 at 29% loss rate



g) pdf for loss length to site 1 at 43% loss rate



h) pdf for loss length to site 1 at 88% loss rate

Figure 4.5: Probability density functions for loss lengths at different loss rates

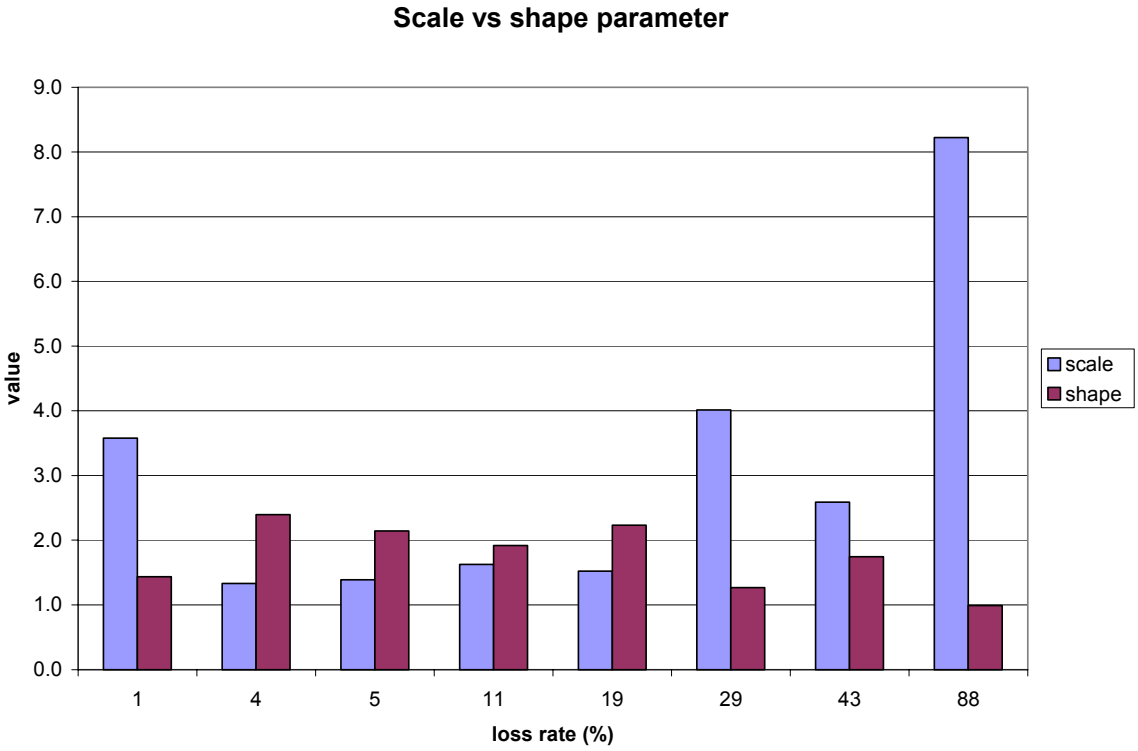


Figure 4.6: Graph of the shape and the scale parameters for different loss rates.

4.3 Simulation

The model for delay and loss was implemented in ns2 using the "DelayBox" node []. The "DelayBox" is a tool developed by the Distributed and Real-Time systems research group at the University of North Carolina at Chapel Hill for ns2 to implement a transport layer delay following a specified distribution in a link [53]. It is placed on the link to introduce a delay to each packet based on a specified distribution. For this project it has been modified to implement the Weibull distribution.

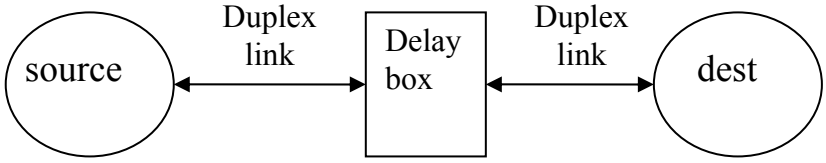


Figure 4.7: Simulation setup

The basic setup for the simulation is made up two end nodes, the source and the destination nodes, with "DelayBox" node inserted in between as shown in **Figure 4.7**. The actual setup for the simulation is shown in **Figure 4.8**. It consisted of a group of source nodes

(*n_src0* and *n_src1)* and destination nodes (*n_sink0* and *n_sink1*) connected by two delay box nodes, *db0* and *db1*, which were connected by a 100Mb link. TCP was selected as the protocol since it is the dominant protocol on the internet [27]. Multiple sources and destinations were used so that there would be more than one flow in the bottleneck link, as opposed to having a dedicated link. This is more representative of real-life networks, especially the internet. Two DelayBox nodes are used to present propagation delay along the link.

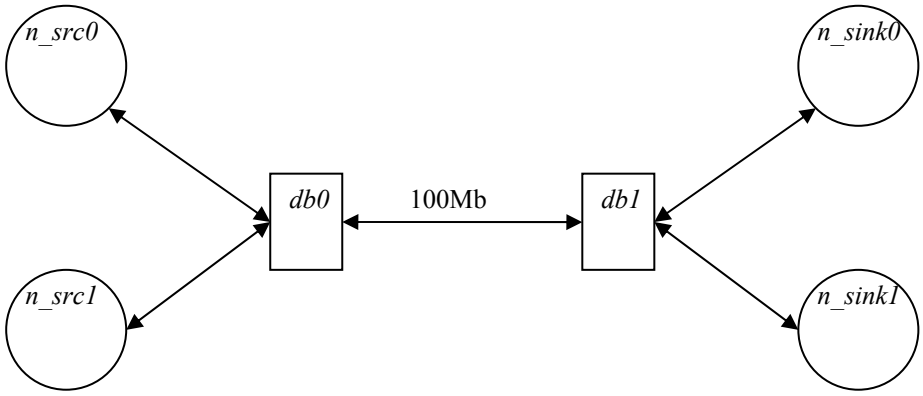


Figure 4.8: Layout of the simulation network

4.3.1 Delay simulation results

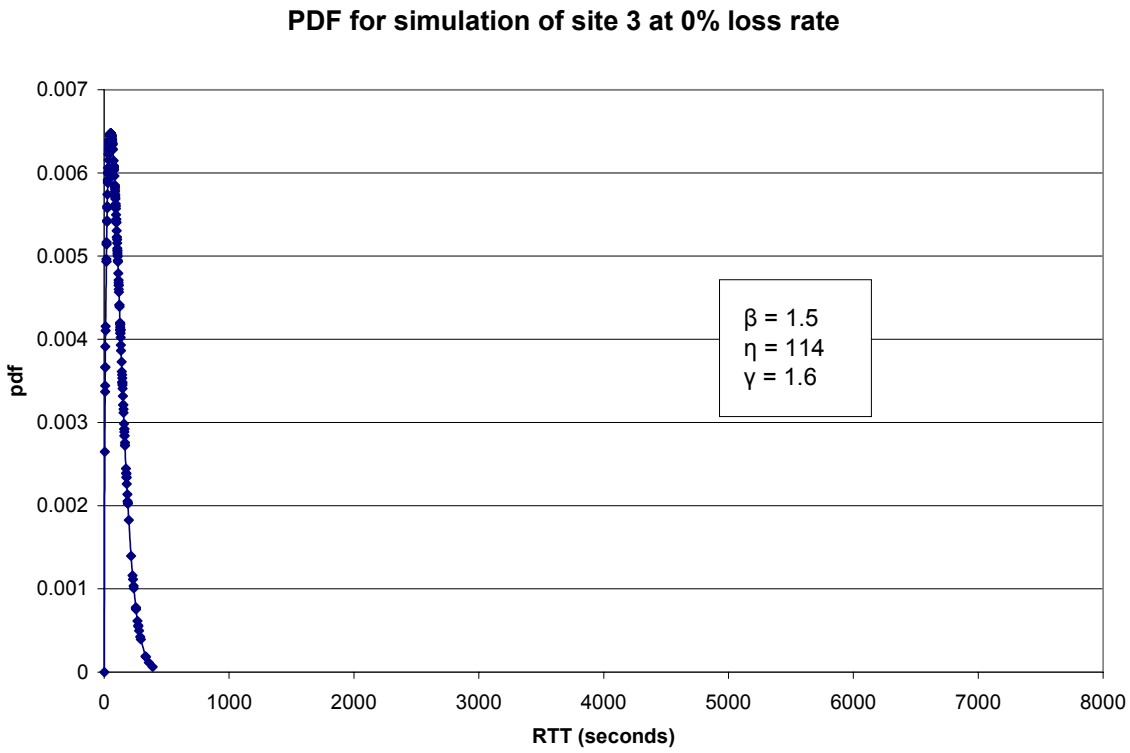
The shape and scale parameters obtained in Chapter 3 for different congestion levels were used as input to the simulator, and the outputs were compared. **Figure 4.9** shows the *pdf* plots for simulation results. Although they are more spread out along the x-axis, the shapes are generally similar to those exhibited in **Figure 4.1**.

Table 4.4: comparison of inputs and outputs in the simulation

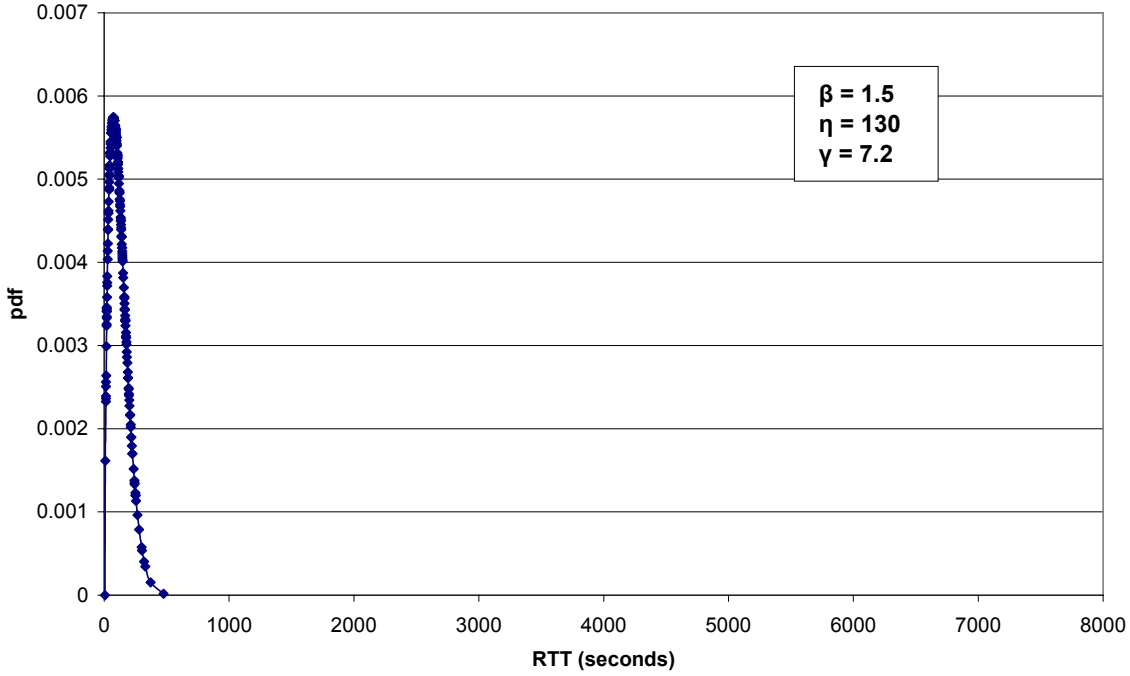
Site	Input		Output	
	β	η	β	η
1	2.46	1092	2.4	1016
1	8.1	5431	8.6	5354
1	5.25	877	5.6	860
3	1.47	114	1.5	111
5	1.53	123	1.5	130
7	1.93	151	1.8	146

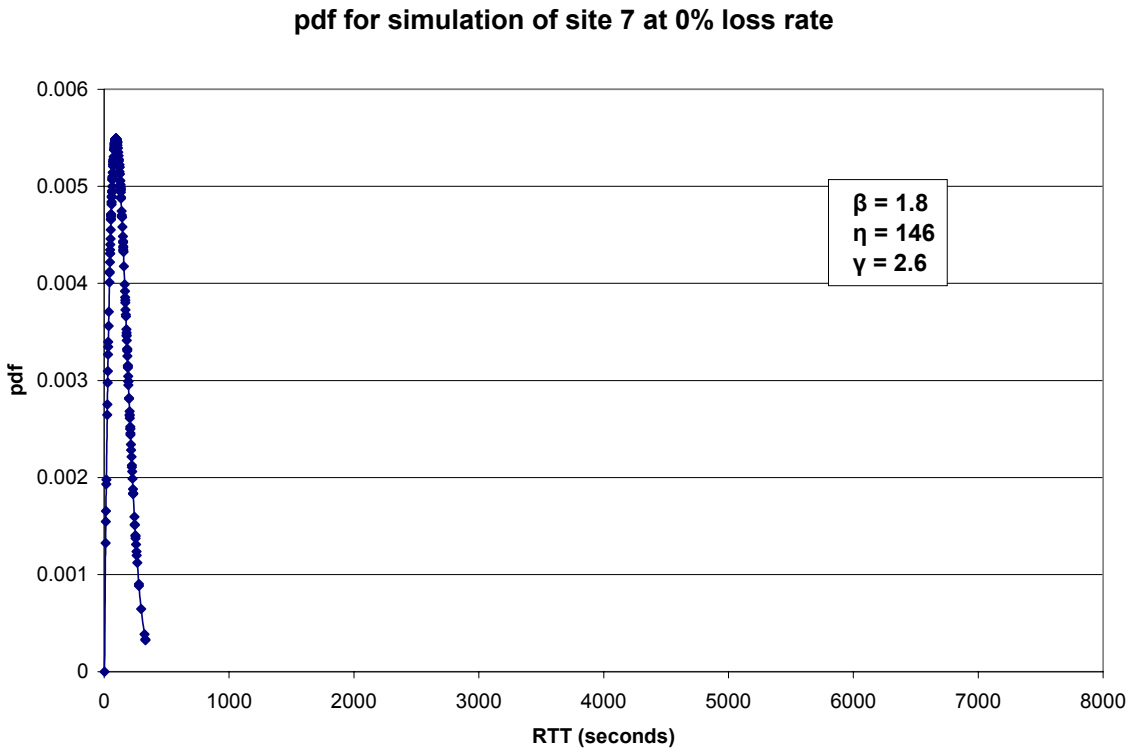
The parameters are the ones that differ slightly, however, as shown in **Table 4.4**. The β and η input values to simulate site 3 in a) were 1.47 and 111, respectively, which differed

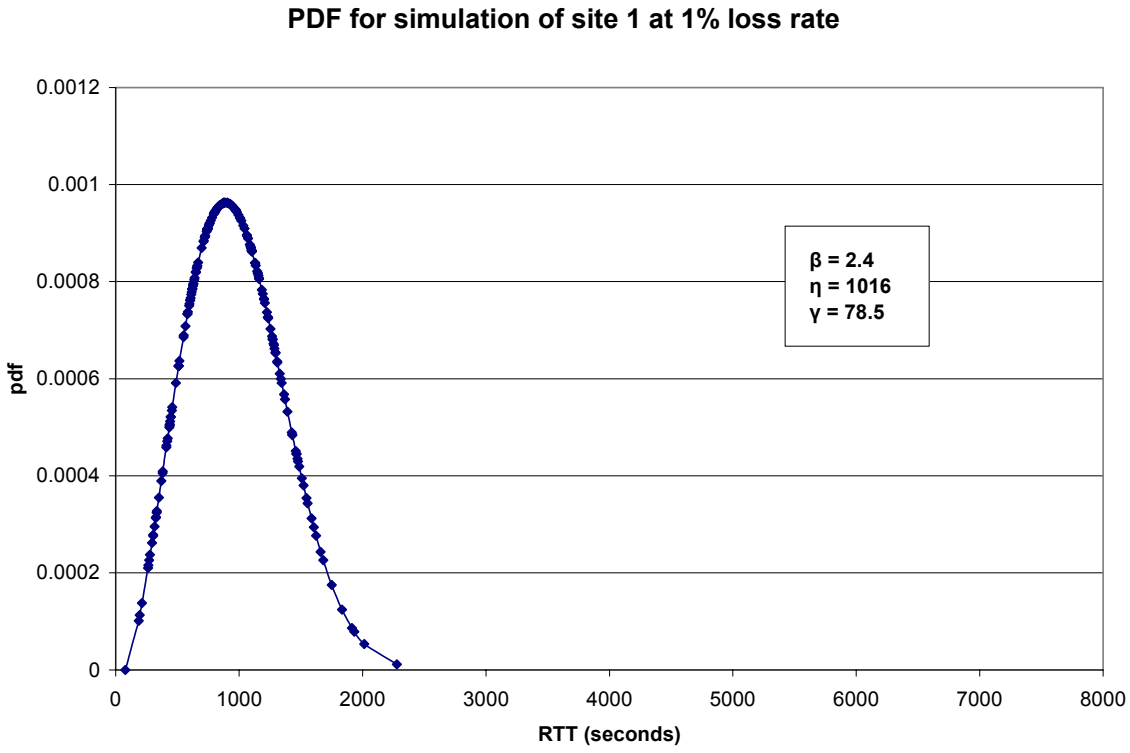
slightly to the 1.5 and 114 outputs. Similar discrepancies in inputs and outputs were observed in simulations for sites 5, 7 and 1 at various loss rates.



a) pdf for delay simulation on a non-congested site




pdf for simulation of site 7 at 0% loss rate
0.006
0.005
0.004
0.003
0.002
0.001
0
pdf
β = 1.8
η = 146
γ = 2.6
0
1000
2000
3000
4000
5000
6000
7000
8000
RTT (seconds)


PDF for simulation of site 1 at 1% loss rate
0.0012
0.001
0.0008
0.0006
0.0004
0.0002
0
pdf
β = 2.4
η = 1016
γ = 78.5
0
1000
2000
3000
4000
5000
6000
7000
8000
RTT (seconds)

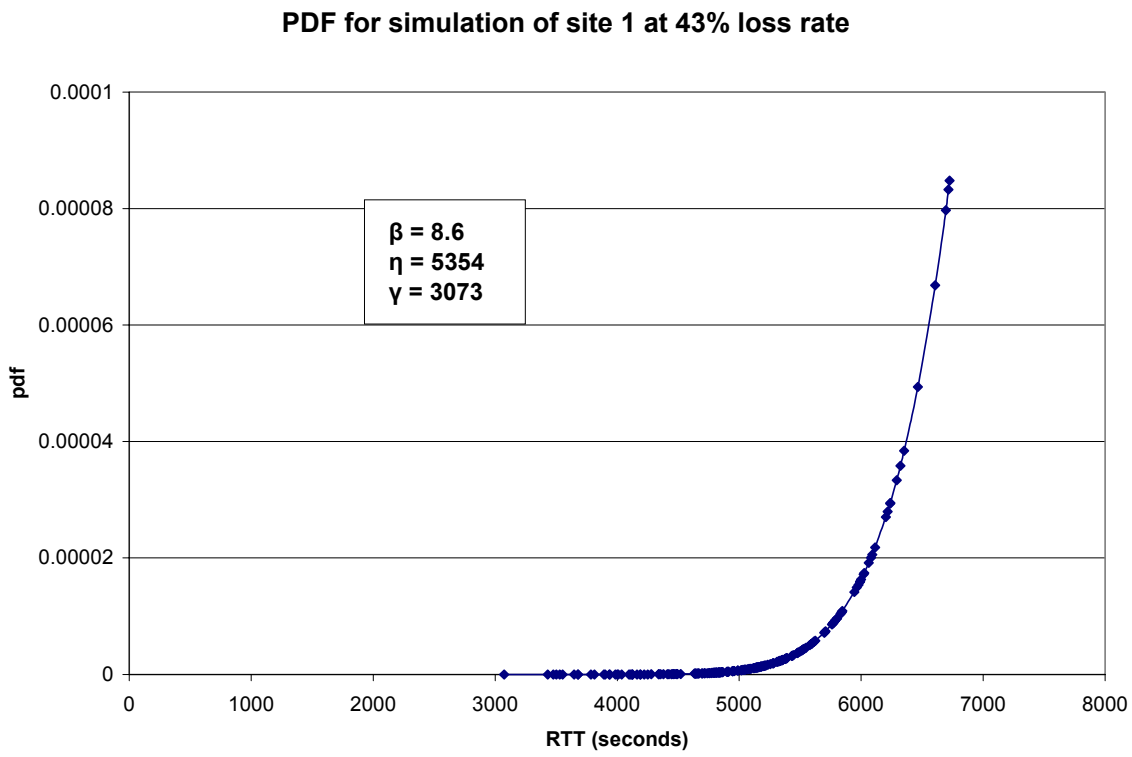
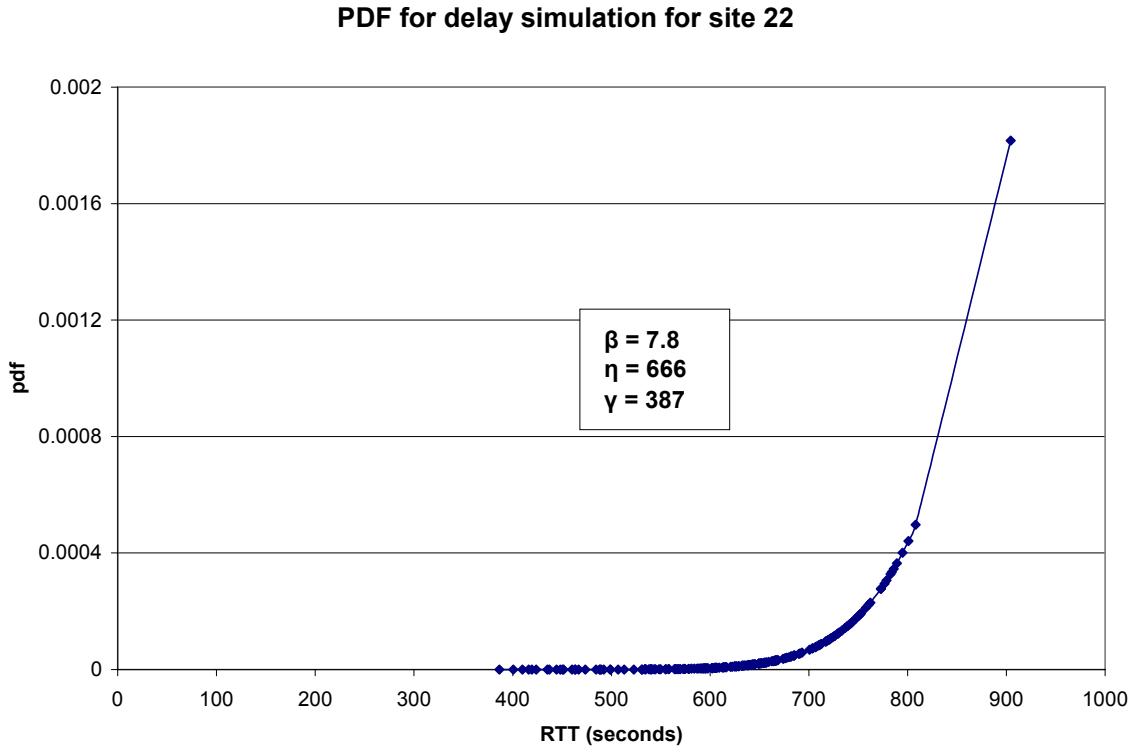
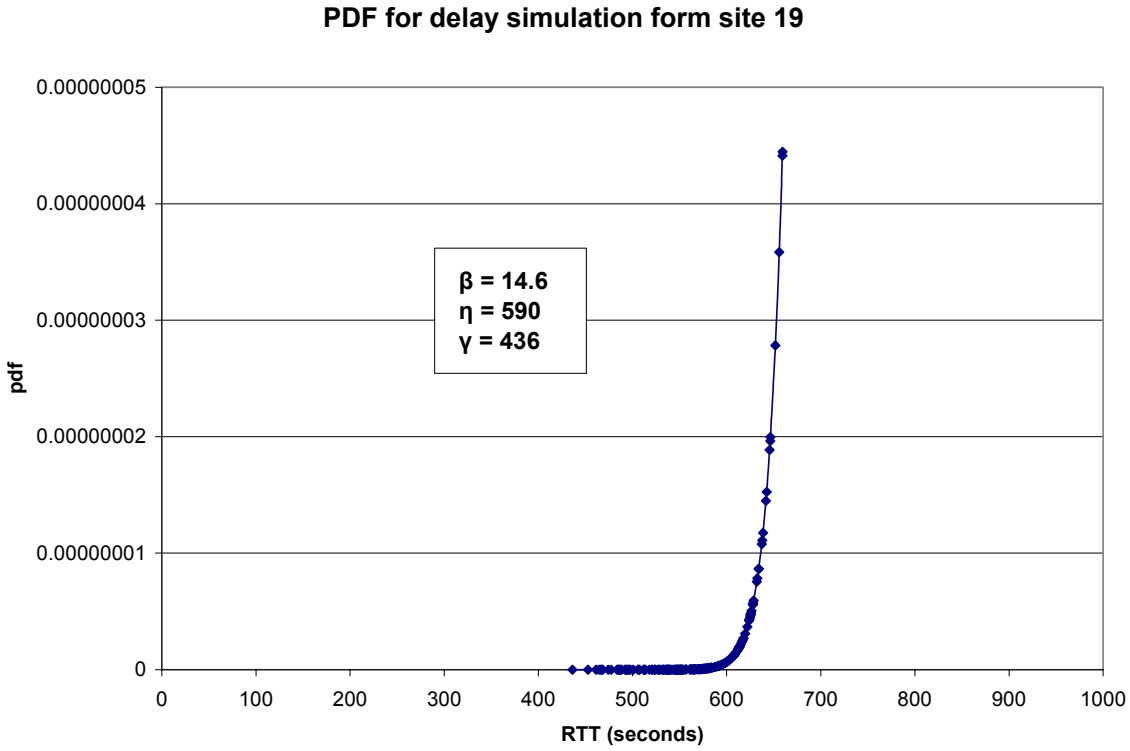


Figure 4.9: Probability density functions for delay simulations



a) pdf for the delay simulation of site 22 at 0% loss rate



b) pdf for the delay simulation of site 19 at 1% loss rate

Figure 4.10: pdf for the delay simulations of overseas sites.

Because of TCP filtering and other algorithms, we expected some of the outputs to differ slightly from the inputs. It is further noted that these discrepancies increase with increasing congestion, as TCP attempts to control that congestion, thus deviating further from what is expected. Similar results were obtained for overseas sites, whose pdf's are shown in **Figure 4.10**.

4.3.2 Loss simulation results

Table 4.5: Parameters for loss distance simulation

Site	Loss (%)	Input		Output	
		β	η	β	η
1	1	0.5	125.0	0.5	109.3
1	29	0.9	7.9	0.8	7.5
1	43	1.3	3.3	1.2	3.3
1	88	1.8	1.5	1.9	1.4
2	4	1.1	22.5	1.1	21.3
4	11	1.0	8.6	0.9	8.4
5	19	1.3	5.4	1.3	5.5
6	5	1.1	18.3	1.1	18.4

Table 4.6: PARAMETERS FOR LOSS LENGTH SIMULATION

site	Loss (%)	Input		Output	
		β	η	B	η
1	1	1.4	3.6	1.4	3.5
1	29	1.3	4.0	1.3	3.7
1	43	1.7	2.6	1.6	2.5
1	88	1.0	8.2	1.0	8.8
2	4	2.4	1.3	2.3	1.3
4	11	1.9	1.6	1.8	1.7
5	19	2.2	1.5	2.2	1.6
6	5	2.1	1.4	2.0	1.4

Parameters for loss distance and loss length, shown in **Table 4.5** and **Table 4.6** were not much different from those obtained earlier. The value of β varies by 0.1 in some cases, otherwise it remains the same. For site 1 the difference is seen for the 29%, 43% and 88% loss rates. Site 4 also showed 0.9 as opposed to 1.0. The differences in the η value were more noticeable, the greatest being for site 1 with 1% loss rate (109.3 vs. 125.0). **Figure 4.11** shows that the actual and simulation loss rates differed. As stated in section 4.3.1, the discrepancies were mainly due to TCP filtering and other algorithms. **Figure 4.12** and **Figure 4.13** show the simulation results for the loss distance and loss length, respectively.

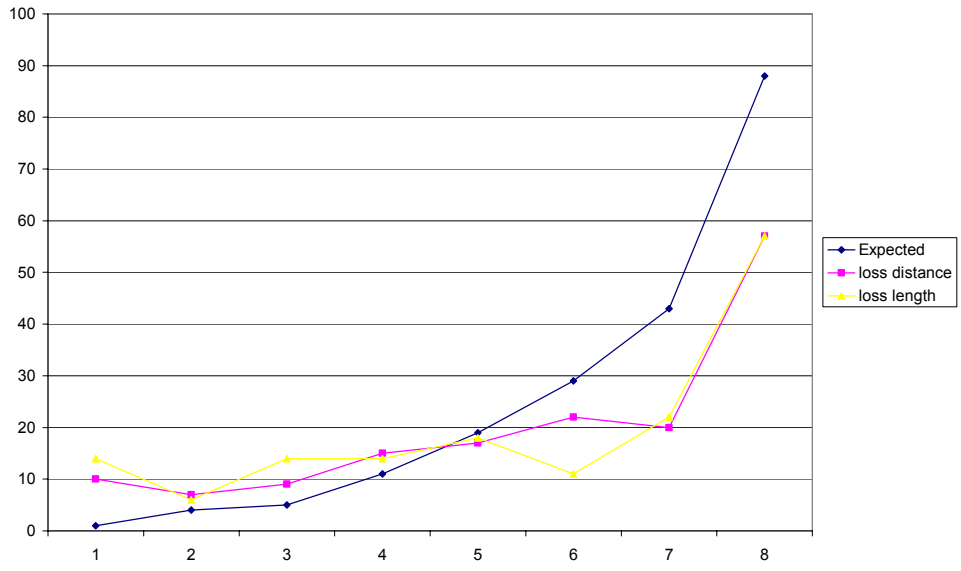
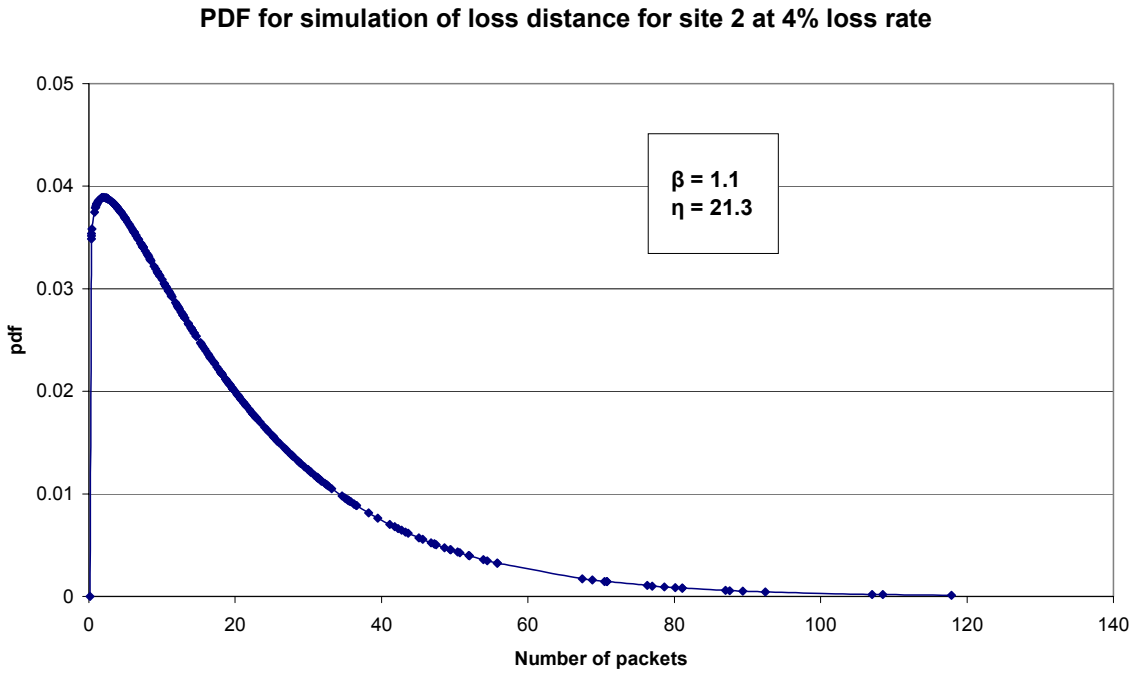
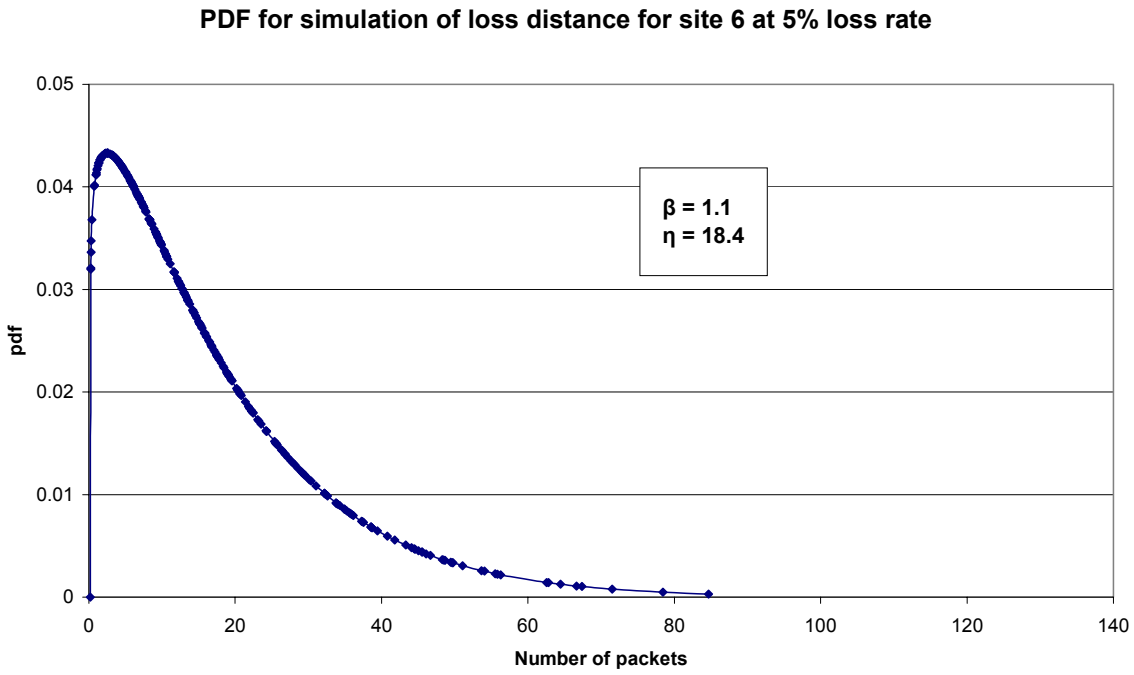


Figure 4.11: Differences between actual and simulation loss rates

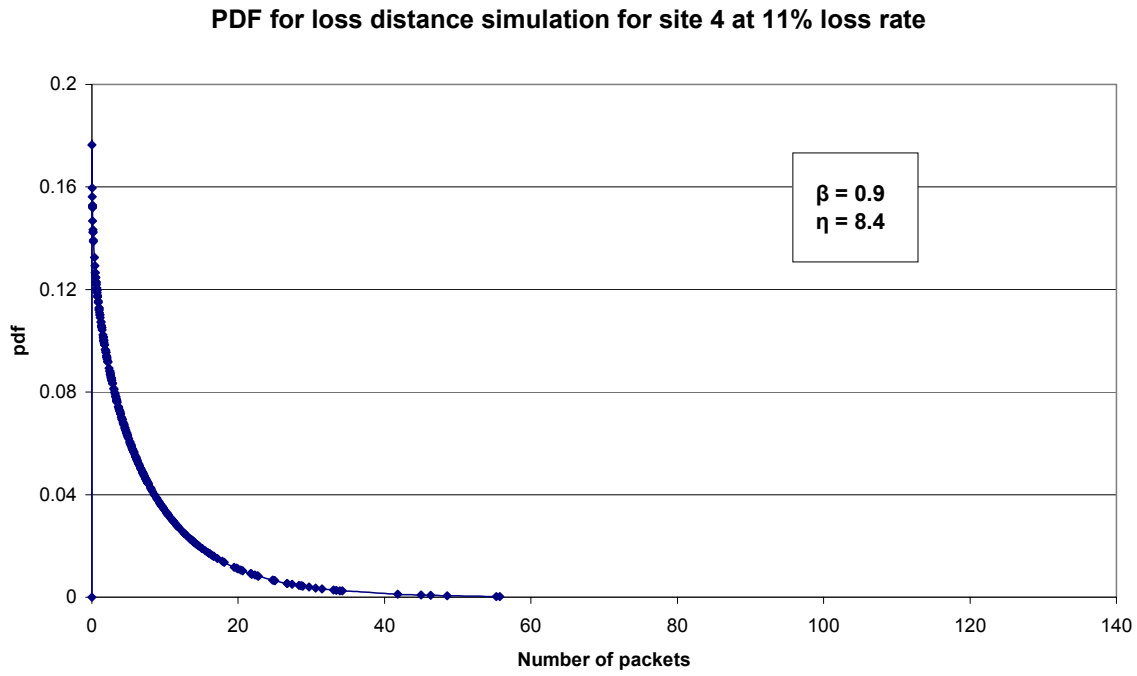
The graphs in **Figure 4.12** show a similar trend to those in **Figure 4.3** in that the scaling parameter diminishes with increasing loss rate, whereas the shape parameter never exceeds 2.6. The opposite behavior, which was displayed in **Figure 4.5**, whereby the scaling parameter increases with increasing loss rate is somewhat repeated in **Figure 4.13**.



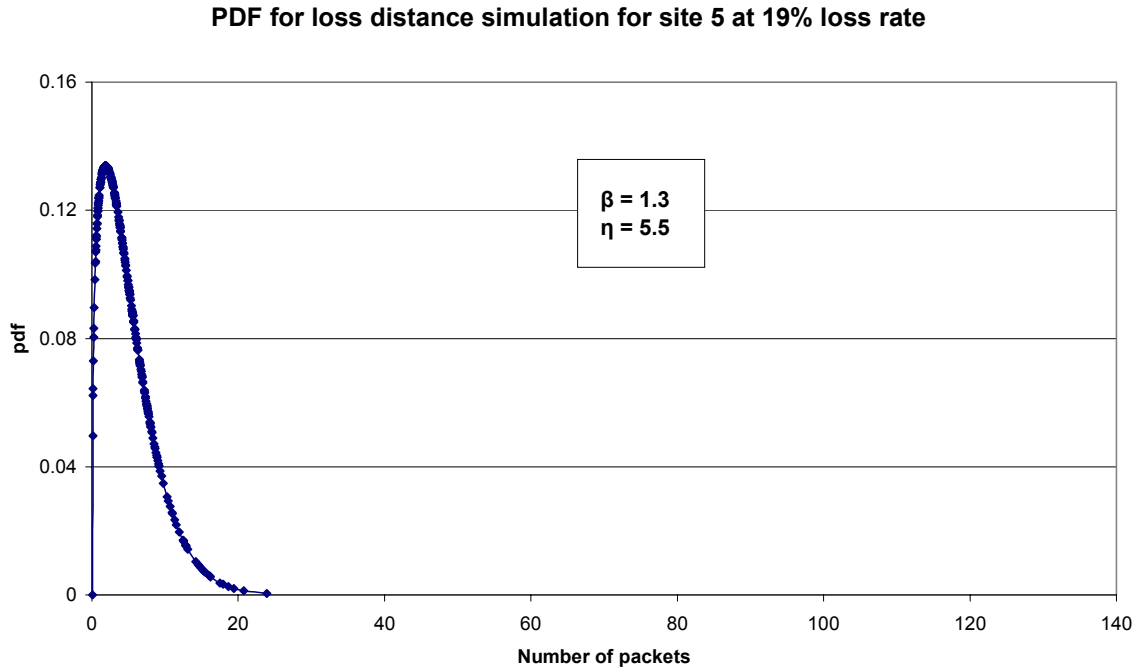
a) pdf for loss distance simulation for site 2



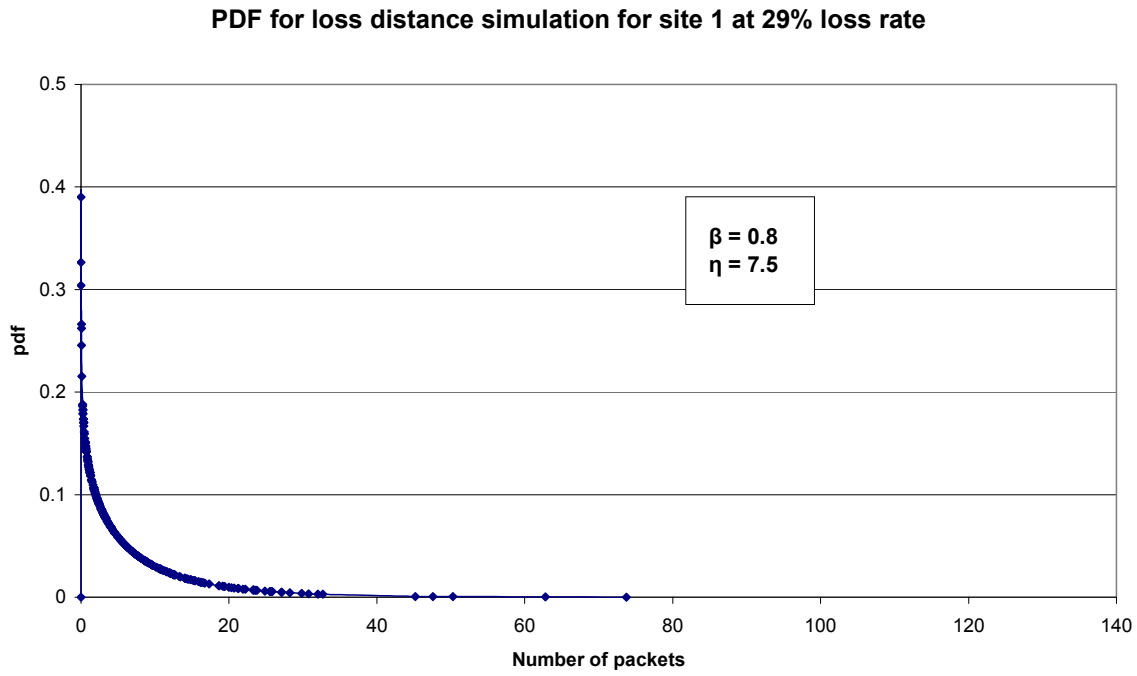
b) pdf for loss distance simulation for site 6



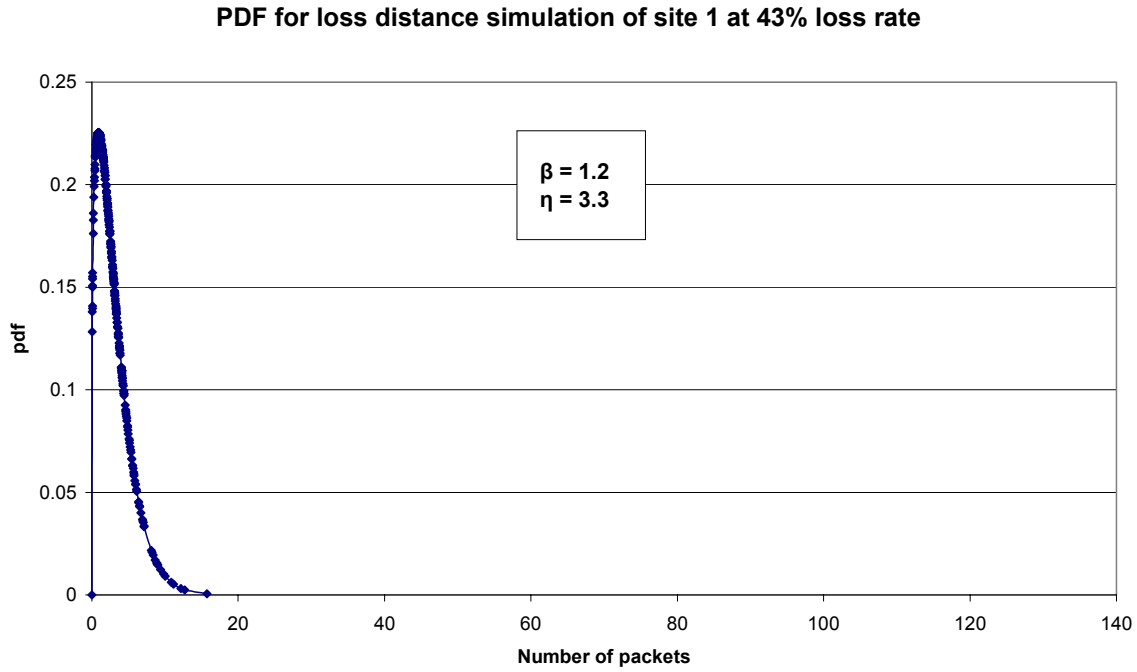
c) pdf for loss distance simulation for site 4



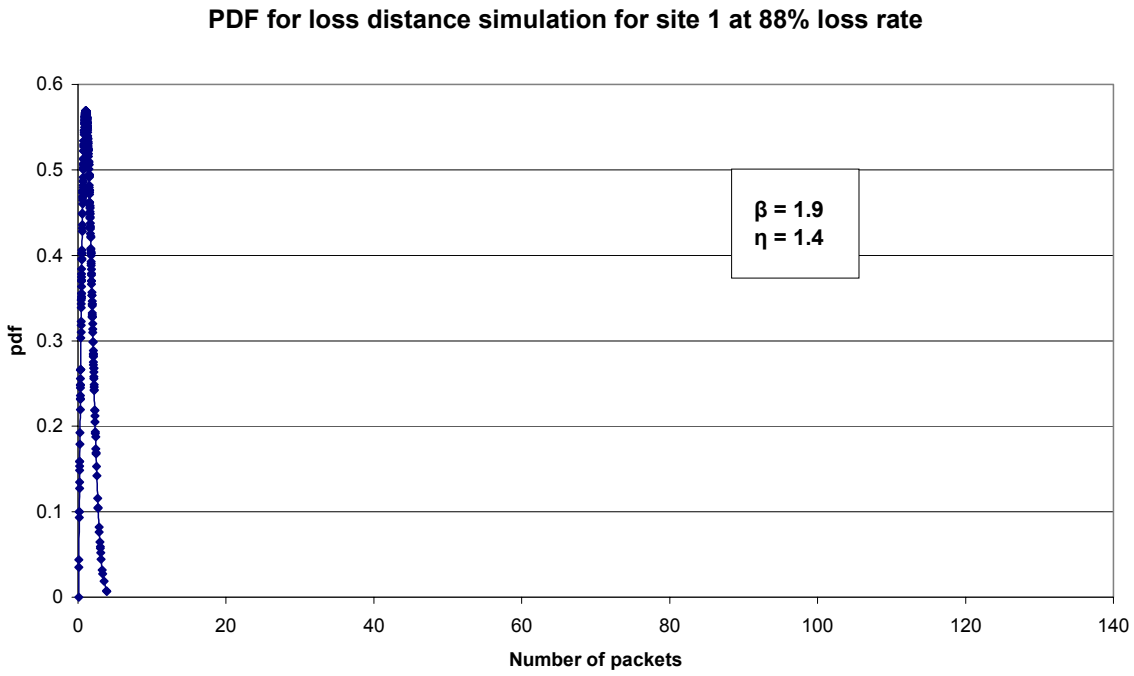
d) pdf for loss distance simulation for site 5



e) pdf for loss distance simulation for site 1 at 29% loss rate

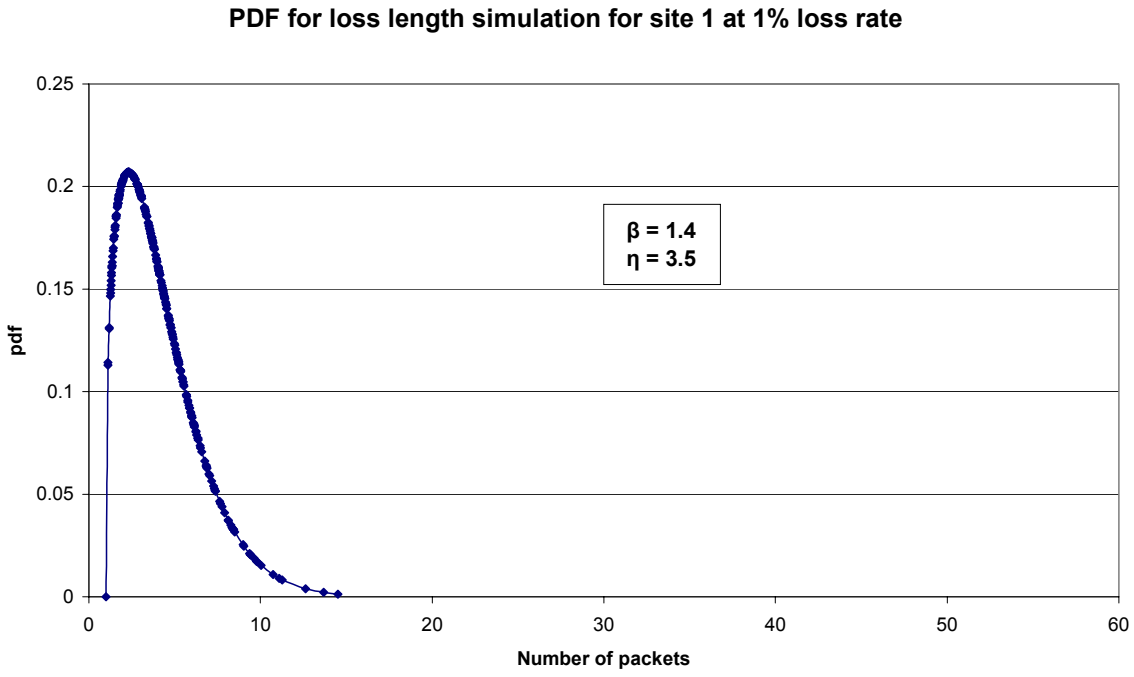


f) pdf for loss distance simulation for site 1 at 43% loss rate

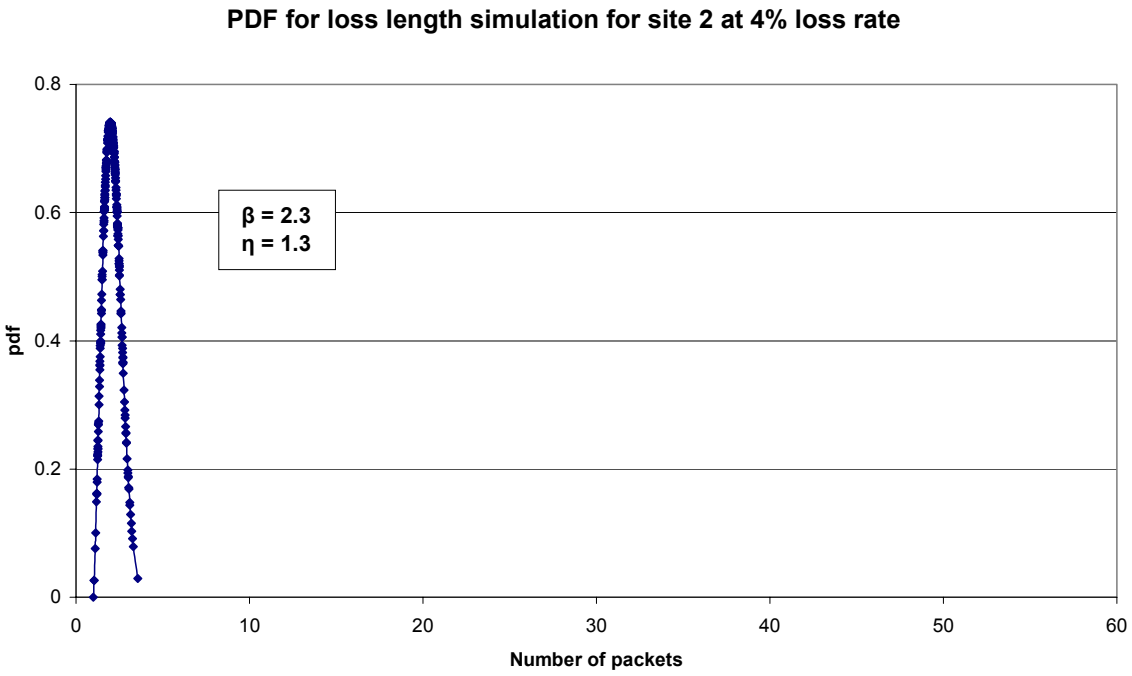


g) pdf for loss distance simulation for site 1 at 88% loss rate

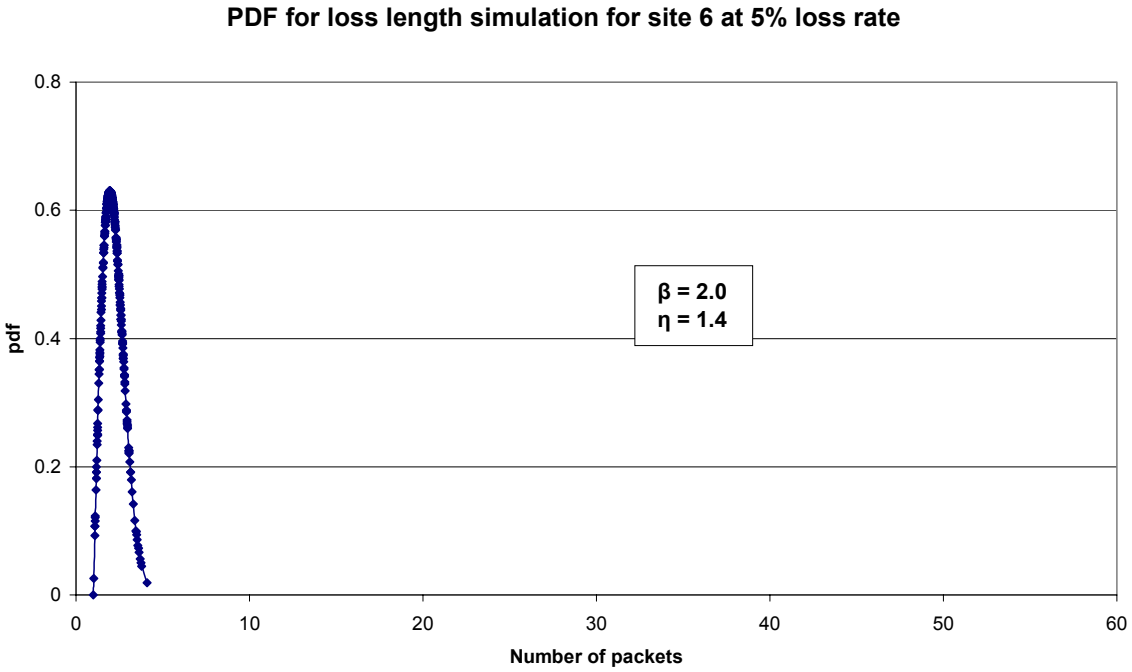
Figure 4.12: Probability density functions for loss distance simulations



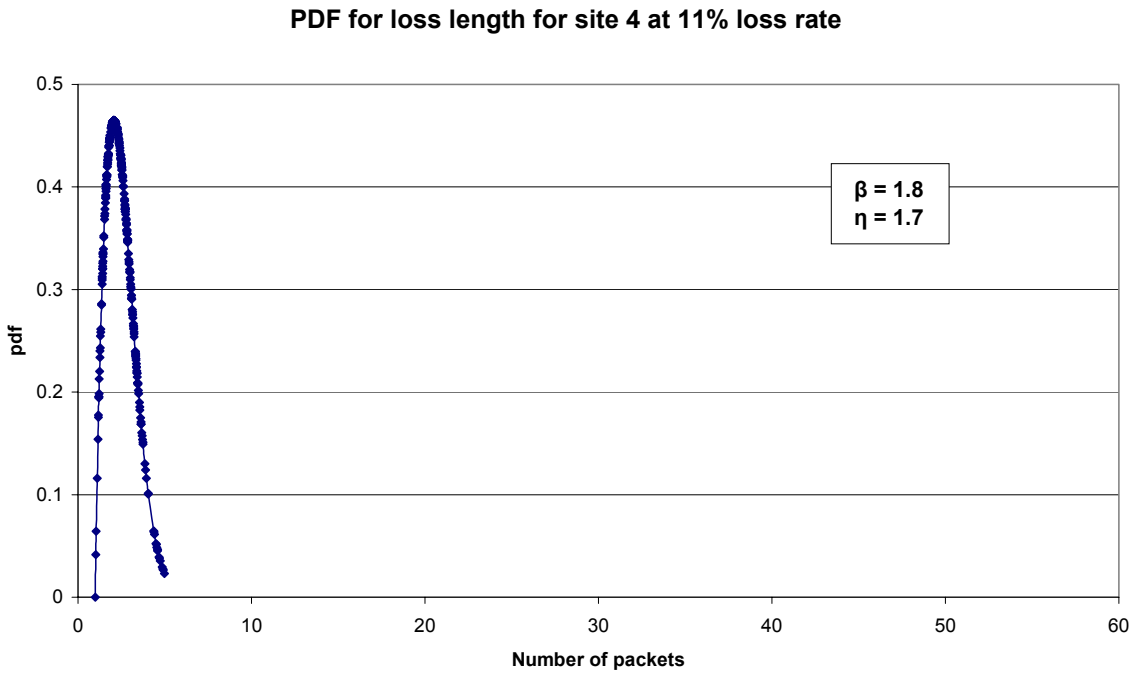
a) pdf for loss length simulation for site 1 at 1% loss rate



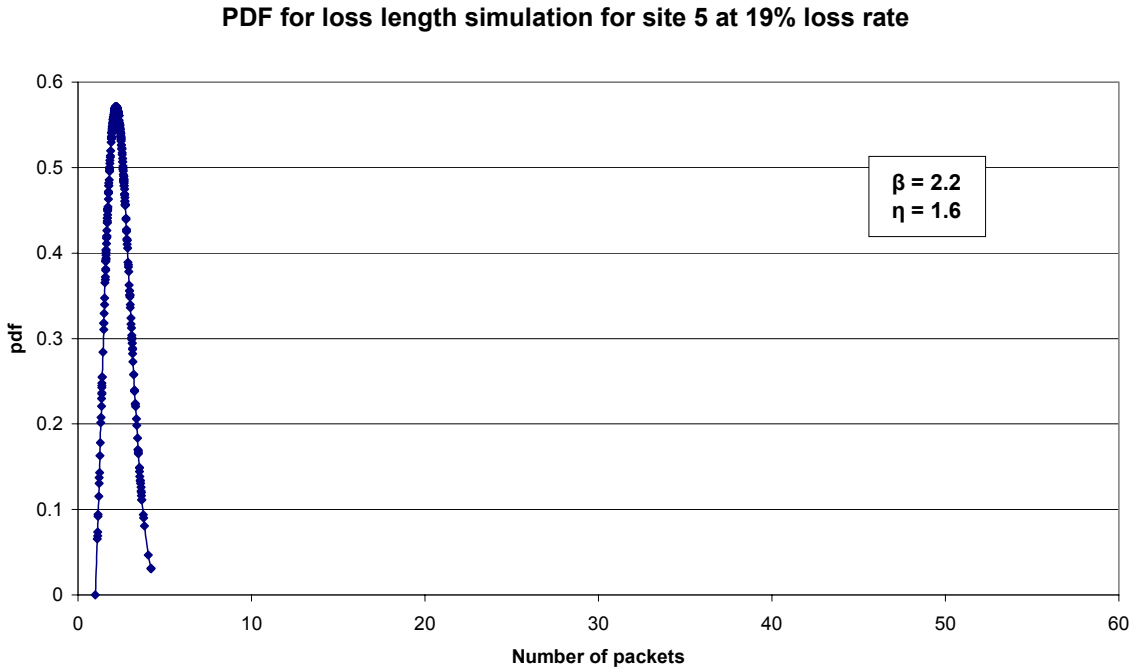
b) pdf for loss length simulation for site 2 at 4% loss rate



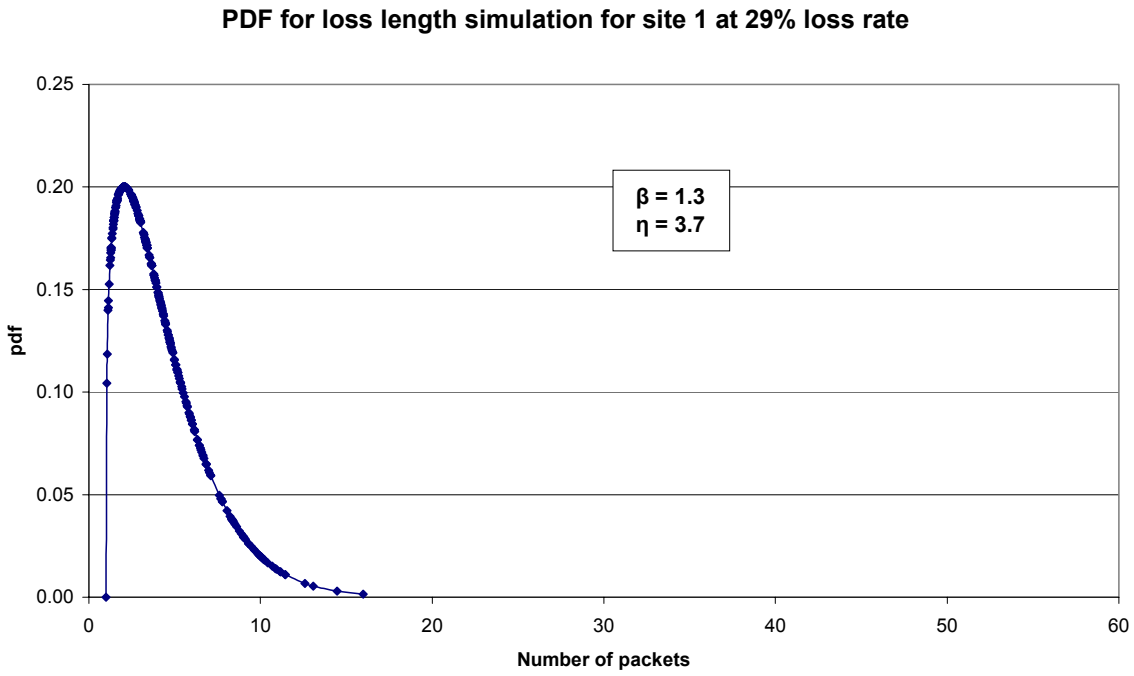
c) pdf for loss length simulation for site 6 at 5% loss rate



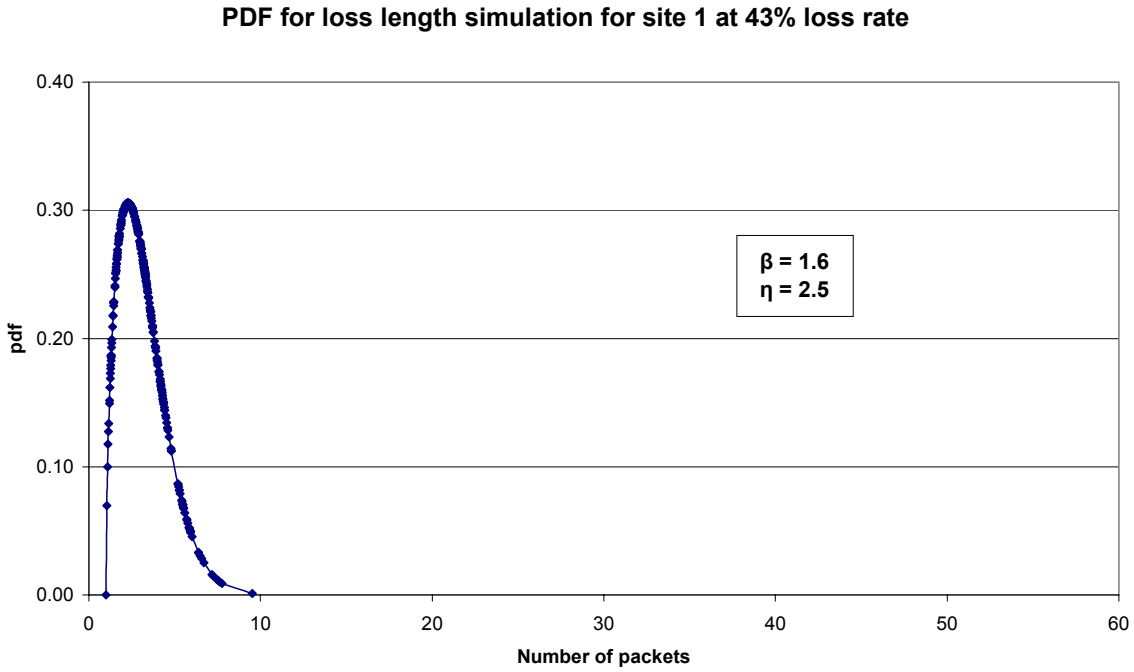
d) pdf for loss length simulation for site 4 at 11% loss rate



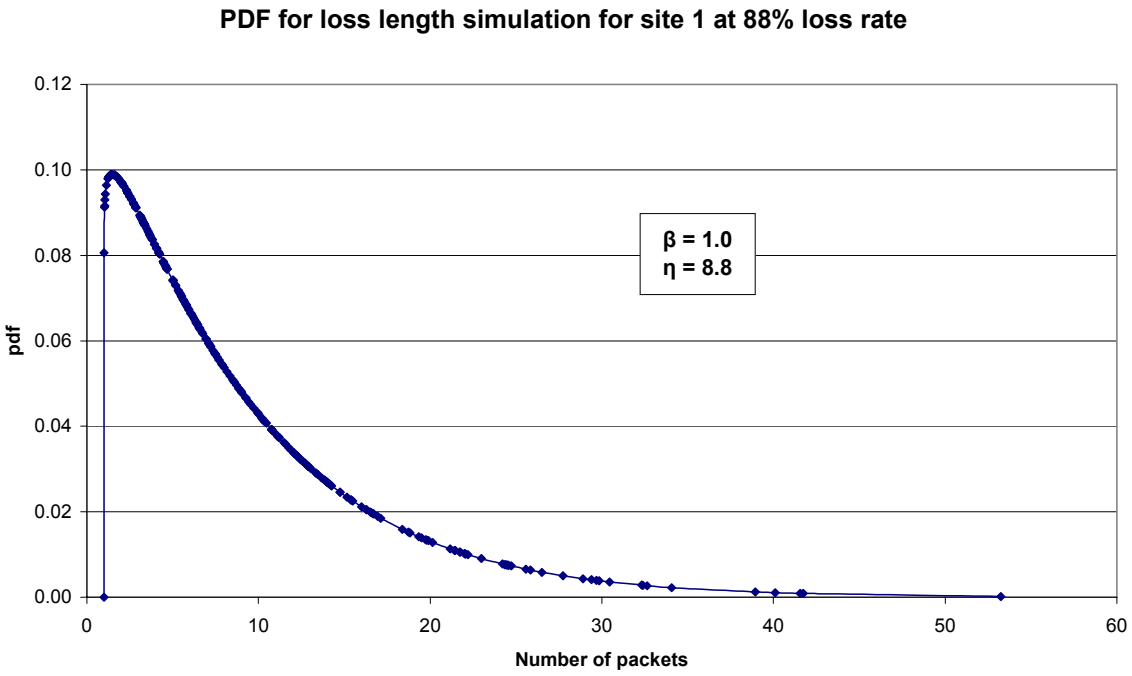
e) pdf for loss length simulation for site 5 at 19% loss rate



f) pdf for loss length simulation for site 1 at 29% loss rate



g) pdf for loss length simulation for site 1 at 43% loss rate



h) pdf for loss length simulation for site 1 at 88% loss rate

Figure 4.13: Probability density functions for loss length simulations

4.3.3 Generic simulation framework

In this section we show that the parameters of the Weibull can be used to develop a generic simulation framework for various states of a network. In particular, we show a technique for simulation of the delay distribution for a network that continuously changes its congestion states.

A generic simulation framework is based on Markov chain as discussed in chapter 2. We consider three congestion states discussed in section 2.7: not congested (S0), moderately congested (S1), and highly congested (S2), characterized by $\beta < 2.6$, $2.6 \leq \beta < 3.7$, and $\beta \geq 3.7$, respectively.

An example of a generic simulation scenario was defined by the transition matrix:

$$\begin{bmatrix} 0.2 & 0.7 & 0.1 \\ 0.5 & 0.3 & 0.2 \\ 0.1 & 0.6 & 0.3 \end{bmatrix} \qquad (4.1)$$

The Markov chain is shown in **Figure 2.3**. The initial state is set to be S0. Using random numbers and the pre-defined transition matrix in (4.1), new values for β are selected randomly.

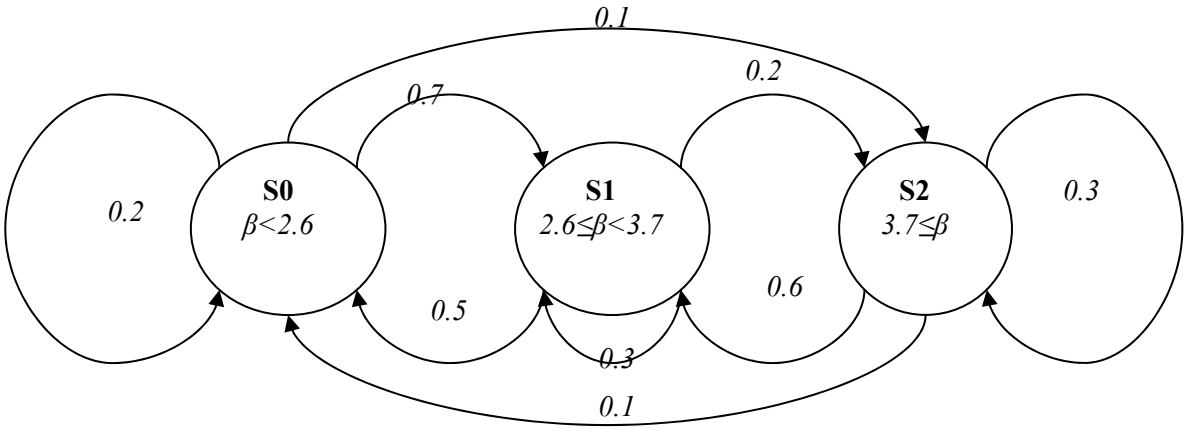


Figure 4.14: Markov chain representation of the congestion states

ns2 was again used as a simulation tool and results are shown in **Figure 4.15**. For the simulation η was fixed at 10 and γ at 1.

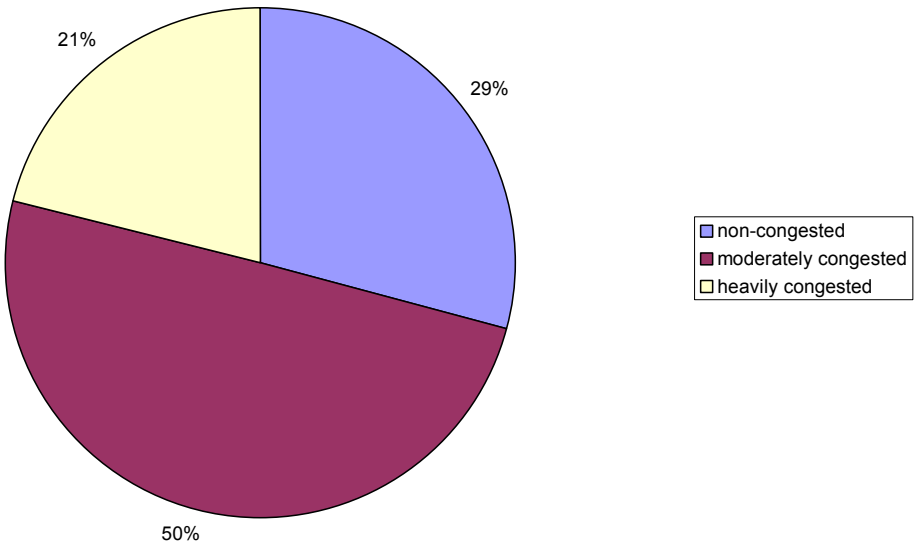


Figure 4.15: Proportion of time spent in each congestion states

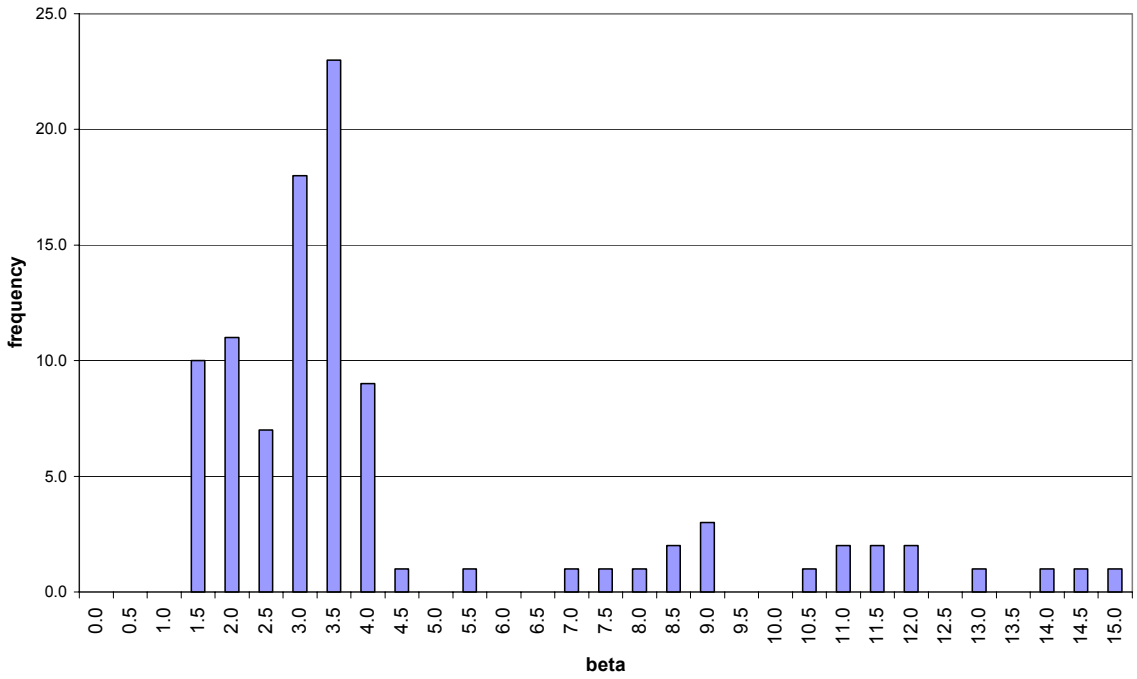


Figure 4.16: Distribution of β values

The chance of changing from S0 to S1 is 70%, and from S2 to S1 is 60%. With the given transition probabilities, the network spent most of the time in S1 as expected. The least of the time was spent in S2. The aggregate value of β was computed and found to be 3.1. This value is greater than, but closer to, 2.6 as expected. The overall distribution is given in **Figure 4.16**, which verifies that most of the values are clustered in the $2.6 \leq \beta < 3.7$ region.

5. DISCUSSIONS AND CONCLUSION

Due to the somewhat limited scope of this study, we cannot claim for certain that the framework presented here is representative of the whole Internet, as some of the behavior could be site-specific [15]. Another problem is that since we do not know the exact paths taken by the packets, it is not easy to tell whether certain behavior is due to congestion or route changes [8]. An attempt to trace the backbone structure of the Internet in South Africa proved fruitless as ISPs and companies view their infrastructure as proprietary and therefore not available for public use. This mission was therefore promptly abandoned in the very early stages of this study. The study does, however, highlight some useful information regarding delay, loss and congestion which warrants some consideration. Our findings are summarized below.

5.1 Delay

It was shown in chapter 3 that the congestion state of a network path affects the delay distribution. When there is no congestion, delay tends to follow a Pareto distribution, whereas for a moderately congested path a Gamma distribution is displayed. Delay distribution for an extremely congested network path is typically left-tailed. Using the parameters of the Weibull distribution, this translates to the shape parameter $\beta < 2.6$ for a non-congested, $2.6 \leq \beta < 3.7$ for a moderately congested, and $\beta \geq 3.7$ for a highly congested network.

However, it was also observed that the delay distribution is affected by the distance between the communicating hosts. For sites which are near each other, delay follows a Pareto distribution, with inherent RTT being dominant. As the distance increases, the distribution shifts to the right, becoming less clustered together, and multiple modes appear. This shift continues as distance increases, until the distribution is left-tailed for vast distances, including overseas sites. This effect can be attributed to the routing algorithms and characteristics of the alternative paths used by these algorithms. Since most routing algorithms take into account congestion states of the network, it can be inferred that congestion does play a role in the shape of the distribution. However, this shows that the delay distribution cannot on its own be used to conclusively determine the congestion state of the network. Loss characteristics have to be taken into account.

5.2 Loss

The loss rate, too, may not on its own be used to detect or measure congestion. To illustrate this point, site 1, with a loss rate of 1%, was shown to be moderately congested, whereas site 5, which had a loss rate of 19%, was shown to be non-congested. Two loss metrics, loss distance and loss length, give more information regarding the congestion status of the network.

For both loss distance and loss length, the shape of the distribution does not offer any useful information, as it is always right-tailed ($\beta < 2.6$). A more significant parameter is the scaling parameter, η, which changes according to the congestion state of the network. In the case of loss distance, it decreases with increasing congestion, but it increases with congestion for loss length. We try to explain this behavior below.

By definition, loss distance decreases as congestion increases, meaning that when there is no congestion, the loss distance approaches some large number, N, and approaches zero as congestion increases. Looking at the statistical properties of the two-parameter Weibull distribution, particularly the mean, we note that when $\beta = 1$, equation 2.6 becomes

$$\overline{T} = \eta \cdot \Gamma(2) = \eta \qquad (5.1)$$

and the probability density function in equation 2.5 translates to

$$f(T) = \frac{1}{\eta} e^{-\left(\frac{T}{\eta}\right)} \qquad (5.2)$$

From the probability density function graph for $\beta = 1$ in Figure 2.1, we note that as the loss distance (on the x-axis) approaches 0, the probability density function, *f(T)*, increases. In equation 5.2, it is evident that when *f(T)* increases, η – which also happens to be the mean, as shown in equation 5.1 – decreases.

Loss length, on the other hand, has been shown to increase with congestion. Following a similar approach to that used for the loss distance, we observe the behavior for *f(T)* when $\beta = 1$ in Figure 2.1. We see that as loss length increases, *f(T)* decreases. When *f(T)* decreases, as seen in equation 5.2, the value of η increases.

Packet losses have been shown to become burstier with increasing congestion, even though the burst sizes are relatively small. An increase in congestion means more packets are queued at

the routers, increasing the likelihood of a group of consecutive packets being discarded due to buffer overflows. This phenomenon is well captured by the scaling parameter, η, for loss distance and loss length.

5.3 Significance of the results

This study has described and shown how delay and loss characteristics tend to change according to the congestion. None of these characteristics can be used in isolation to reliably detect congestion. They can, however, be used to augment each other in order to come up with a more accurate congestion measure. The following factors could be used together to determine whether there is congestion or not:

- The value of the shape parameter for delay
- The increase or decrease in the scaling parameter for loss distance and loss length
- Whether packet losses occur as burst or are isolated

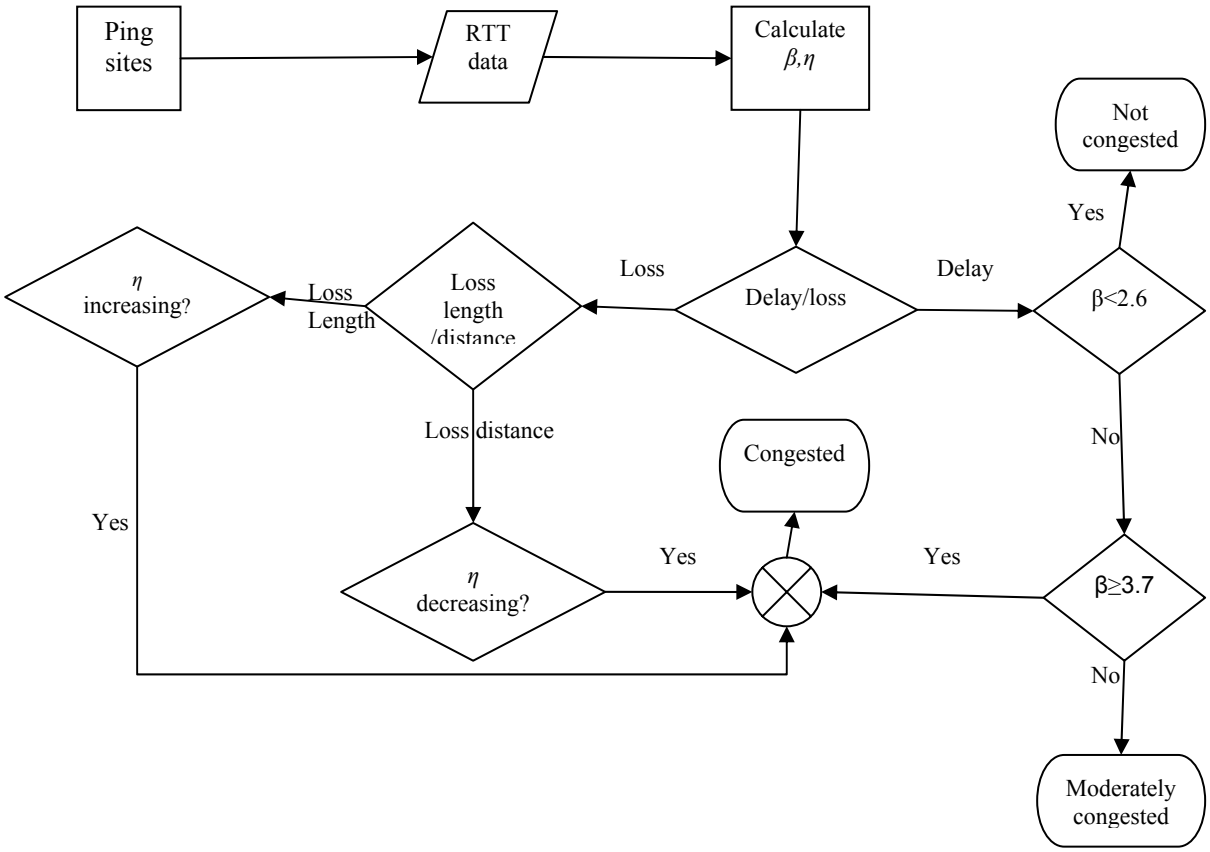


Figure 5.1: Framework for congestion detection

How these factors can be used to suggest the congestion state of a path is illustrated in **Figure 5.1**. Although not conclusive on their own, they offer alternative method of congestion detection which, if patched onto the implementation of protocols, such as TCP, could enhance the congestion detection ability of the protocol. Benefits of this would include increased efficiency as the number of false retransmits would be reduced.

Based on the outcomes of this study, a simulation framework was created to model the congestion states. This framework is based on three congestion states, which depend on the value of the Weibull shape parameter based on packet delay. This framework was specified using a Markov chain and was validated in simulation exercises.

We have shown that delay and loss characteristics contain useful information that can be used in congestion detection. We successfully linked delay and loss characteristics to the congestion state of the network using the parameters of the Weibull distribution, and then validated these findings in simulations.

5.4 Possible Future work

The framework suggested here is based almost exclusively on a study of a small number of sites hosted in South Africa. For the framework to be more representative (and valuable), we have to apply it across a wider selection of hosts to see if it holds. Prior knowledge of possible routes to be taken and their capacities will prove indispensable as path characteristics do affect delay and loss behavior – a point not pursued in this study. If the results are still positive and measures up to our study, the next step from there would be to investigate how this framework can be incorporated into a transport protocol, for example TCP, in order to further strengthen the protocol's already existing congestion detection mechanisms. Performance comparison between the modified TCP and other flavors would then be possible. Perhaps crucial to this would be determining the size of a sample such that it is big enough to make accurate calculations, while not so big as to adversely affect the performance and efficiency of the protocol.

The study was aimed at studying end to end characteristics, not exact measurements of RTT, so it was taken for granted that the measurements are correct, despite some well-known shortcomings of ICMP. To verify, comparison could be made using samples obtained using UDP or TCP traces. In our analysis we concentrated only on a coarse time scale of minutes

and hours, ignoring finer timescale of milliseconds. In the future, the study should be done on a smaller time scale to see if our results still hold.

REFERENCES

[1] S. Moon, "Measurement and analysis of end to end delay and loss in the Internet", PhD thesis, University of Massachusetts Armhest, 2000.

[2] M. Allman & V. Paxson, "On estimating end-to-end network path properties," *ACM SIGCOMM*, Cambridge, MA, September 1999.

[3] S. Biaz & N.H. Vaidya, "Is the round trip time correlated with the number of packets in flight?" *Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement*, pp 273-279, October 27-29.

[4] D.Saikia, M. Dahal, "Packet loss free congestion control in TCP for controlled packet latency and optimal throughput" *Conference on Convergent Technologies for Asia-Pacific Region*, vol. 1, pp 313-317, 15-17 Oct. 2003.

[5] J. Martin, et al, "The incremental Deployability of RTT-Based congestion avoidance for high speed TCP Internet connections**",** *Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, Santa Clara, California, United States, pp134 - 144, 2000.

[6] J. Martin, et al, "Delay-Based congestion avoidance for TCP", *IEEE/ACM Transaction on Networking*, Vol. 11, No. 3, pp 356-369, June 2003.

[7] S. Moon, J. Kurose, et al., "Correlation of packet delay and loss in the internet," Technical Report, University of Massachusetts, January 1998.

[8] A. Acharya & J. Saltz, "A Study of internet round-trip delay," Technical Report CS-TR-3736, University of Maryland, December 1996.

[9] P. Karn & C. Partridge, "Improving round-trip time estimates in reliable transport protocols," *ACM SIGCOMM*, 1987, pp 2-7.

[10] J. Bi, Q. Wu, Z. Li, "Packet delay and packet loss in the Internet," *Proceedings of the Seventh International Symposium on Computers and Communications*, ISCC 2002, pp. 3 - 8, 1 July 2002.

[11] V. Paxson, "Measurements and Analysis of End–to–End Internet Dynamics", Ph.D. thesis, UC Berkeley, 1997

[12] V. Jacobson, "Congestion Avoidance and Control," *Computer Communication Review*, vol. 18, no. 4, pp. 314-329, August 1988.

[13] J. Bolot, "Characterizing end-to-end packet delay and loss in the Internet," *Journal of High Speed Networks*, vol. 2, no. 3, pp. 289-298, 1993.

[14] S. Savage, "Sting: a TCP-based Network Measurement Tool," In *Proceedings of the 1999 USENIX Symposium on Internet Technologies and Systems*, October 1999.

[15] V. Paxson, "End-to-end Internet Packet Dynamics," *ACM/IEEE Transactions on Networking*, pp: 277-292, June 1999.

[16] G. Almes, "Metrics and infrastructure for IP performance," 1997, http://www.advanced.org/surveyor/presentations.html, Last accessed on 20 April 2005.

[17] Z. Wang, A. Zeitoun, and S. Jamin. "Challenges and Lessons Learned in Measuring Path RTT for Proximity-based Applications." *Proceedings of the Passive and Active Measurement Workshop*, La Jolla, CA, USA, pp. 181-191, Apr. 2003.

[18] L. Petersen, B. Davie, "Computer networks: A systems approach," 2nd edition, Morgan Kaufmann Publishers, San Francisco, 2000.

[19] C.J. Bovy, et al, "Analysis of End to end Delay Measurements in Internet", *Proceedings of PAM 2002*, March 2002.

[20] R. Szeto, "Transmission Control Protocol (TCP) Flow Control Simulation and Visualization," BSc Honors Thesis, Monash University, 2000. http://www.csse.monash.edu.au/hons/projects/2002/Siuwing.Szeto/thesis_word_format.zip, Last accessed on 12 January, 2005.

[21] W. Jiang, H. Schulzrinne, "Modeling of packet loss and delay and their effect on real-time multimedia service quality," *Proceedings of NOSSDAV*, Chapel Hill, June 2000.

[22] R. Stine, "FYI on a Network Management Tool Catalog: Tools for Monitoring and Debugging TCP/IP Internets and Interconnected Devices", *RFC 1147*, April 1990.

[23] M. Arai, A. Chiba, K. Iwasaki, "Measurement and modeling of burst packet losses in Internet end-to-end communications," *Proceedings of the Pacific Rim International Symposium on Dependable Computing*, pp 260-267, 16-17 Dec.1999.

[24] M. Allman, V. Paxson, W. Stevens, "TCP Congestion Control" *RFC 2581,* April 1999.

[25] M. Yajnik, S. Moon, J. Kurose, D. Towsley, "Measurement and modeling of the temporal dependence in packet loss," *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, IEEE INFOCOM '99, Vol. 1, pp: 345 – 352, 21-25 March 1999.

[26] M. Borella, D. Swider, S. Uludag, G. Brewster, "Internet Packet loss: measurement and implication for end-to-end QoS," *Proceedings of the 1998 ICPP workshop on architectural and OS Support for Multimedia Applications/Flexible communication systems/Wireless networks and mobile computing*, pp 3-12, 14 Aug. 1998.

[27] T. Altunbasak, "A Trade–off Analysis for Quality of Service in Real–Time Voice over IP", Thesis, Turkish Air Force Institute of Technology, March 1999.

[28] R. Koodli, R. Ravikanth, "One-way Loss pattern Sample Metrics," RFC 3357, August 2002.

[29] ReliaSoft's Research & Development Group's, "Introduction to Life Data Analysis," http://www.weibull.com/LifeDataWeb/lifedataweb.htm, Last accessed on 20 April 2005.

[30] M. Crovella, R. Carter, "Dynamic server selection in the Internet," *Proceedings of the 3rd IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems*, pp 158-162, 23-25 August, 1995.

[31] J. Postel, "Transmission control protocol". *RFC 793*, September 1981.

[32] C. Jin, D. Wei, S. Low, "The case for Delay-based Congestion Control," *IEEE Computer Communication Workshop*, Laguna, CA, 2003.

[33] D.D. Clark "Window and Acknowledgement Strategy in TCP", *RFC0813*, July 1, 1982.

[34] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP", *Computer Communications Review*, Vol. 26, no. 3, pp. 5–21, July 1996.

[35] W. Stevens "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms" *RFC2001,* January 1997.

[36] M. Mathis, J. Mahdavi, S. Floyd, A. Romanov, "TCP Selective Acknowledgement Options" *RFC 2018*, October 1996.

[37] E. de Souza, D. Agarwal, "A High Speed TCP Study: Characteristics and deployment issues," Technical Report LBNL-53215, http://dsd.lbl.gov/~evandro/hstcp/hstcp-lbnl-53215.pdf, Last accessed 1 April 2005.

[38] S. Fall, "High Speed TCP for Large congestion windows," *RFC 3649*, December 2003.

[39] T. Kelly, "Scalable TCP: Improving performance in high speed wide area networks," *Submitted for publication*, 2002.

[40] S. Bhandarkar, et al., "TCP-DCR: a novel protocol for tolerating wireless channel errors," *Submitted for publication,* http://students.cs.tamu.edu/sumitha/research/TMC_paper.pdf**,** Last accessed 20 April 2005.

[41] K. Xu, et al, "TCP Jersey for wireless IP communications," *IEEE Journal on selected areas of communication*, vol. 22, no. 4, pp 747-756, May 2004.

[42] L. Brakmo, S. O'Malley, and L. Peterson. "TCP Vegas: New techniques for congestion detection and avoidance" *In Proceedings of the SIGCOMM '94 Symposium*, pp 24-35, Aug. 1994.

[43] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, August 1993, pp 397–413.

[44] S. Floyd, "TCP and Explicit Congestion Notification," *ACM Computer Communication Review*, vol. 24, no. 5, pp 10–23, October 1994.

[45] K. Ramakrishnan, S. Floyd, D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", *RFC3168*, September 2001

[46] S. Floyd, T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm" *RFC 2582*, April 1999.

[47] C. Parsa, et al, "Improving TCP congestion control over Internets with heterogeneous transmission media," *Seventh international conference on network protocols*, Santa Cruz, CA, pp 213-221, 31 Oct – 3 Nov 1999

[48] V. Paxson, et al, "An Architecture for large-scale Internet measurements," *IEEE communications*, vol. 36, no. 8, pp 48-54, August 1998.

[49] J. Nagle, "Congestion control in IP/TCP Internetworks", *RFC0896*, Jan 6, 1984.

[50] R. Branden, "Requirements for Internet Hosts - Communication Layers," *RFC1122*, October 1989.

[51] V. Jacobson, R. Braden, D. Borman, "TCP Extensions for High Performance", *RFC1323*, May 1992.

[52] S. Blank, "Resampling Stats," http://www.resample.com/content/software/excel/index.shtml, Last accessed 12 March 2005.

[53] K. Jeffay, "Per-Flow delay and loss in NS-2 with DelayBox," The Distributed and Real-Time systems research group, University of North Carolina at Chapel Hill, http://dirt.cs.unc.edu/delaybox/, Last accessed 20 March 2005.

[54] M. Weigle, "Random variables in ns," *ns-users mailing list*, http://www.isi.edu/nsnam/archive/ns-users/webarch/2001/msg04976.html, Last accessed 20 April 2005.

APPENDIX A: WEIBULL PARAMETERS CALCULATION

The Weibull probability density function was calculated as

$$f(t) = \frac{\beta}{\eta}\left(\frac{t-\gamma}{\eta}\right)^{\beta-1} e^{-\left(\frac{t-\gamma}{\eta}\right)^{\beta}} \tag{A.1}$$

Where the shape parameter, β, is given by

$$\beta = \frac{k\sum_{j=1}^{k} x_j y_j - \left(\sum_{j=1}^{k} x_j\right)\left(\sum_{j=1}^{k} y_j\right)}{k\sum_{j=1}^{k} x_j{}^2 - \left(\sum_{j=1}^{k} x_j\right)^2} \tag{A.2}$$

And the scaling parameter, η, by

$$\eta = \exp\left[\frac{\left(\sum_{j=1}^{k} y_j\right)\left(\sum_{j=1}^{k} x_j{}^2\right) - \left(\sum_{j=1}^{k} x_j\right)\left(\sum_{j=1}^{k} x_j y_j\right)}{-\beta\left(k\sum_{j=1}^{k} x_j{}^2 - \left(\sum_{j=1}^{k} x_j\right)^2\right)}\right] \tag{A.3}$$

Where

$$x_j = \ln\left(t_j\right) \tag{A.4}$$

$$y_j = \ln\left\{\ln\left[\frac{1}{1-\left(\frac{j-0.3}{n+0.4}\right)}\right]\right\} \tag{A.5}$$

And the location parameter, γ, is the zero point given by,

$$\gamma = t_1 \tag{A.6}$$

APPENDIX B: SIMULATION CODE

```
# useful constants
set PCKTS 10
set FLOWS 200
set NODES 2

# setup ns
remove-all-packet-headers;            # removes all packet headers
add-packet-header IP TCP;             # adds TCP/IP headers
set ns [new Simulator];               # instantiate the simulator

global defaultRNG
$defaultRNG seed 999

# let user input the beta
puts "Please enter the shape parameter: "
gets stdin shape
puts "Please enter the scale parameter: "
gets stdin scl
puts "beta = $shape, scale = $scl."


# create src nodes
for {set i 0} {$i < $NODES} {incr i} {
    set n_src($i) [$ns node]
}
# create DelayBox nodes
set db(0) [$ns DelayBox]
set db(1) [$ns DelayBox]
# create sink nodes
for {set i 0} {$i < $NODES} {incr i} {
    set n_sink($i) [$ns node]
}

# setup links
$ns duplex-link $db(0) $db(1) 100Mb 1ms DropTail
for {set i 0} {$i < $NODES} {incr i} {
    $ns duplex-link $n_src($i) $db(0) 100Mb 1ms DropTail
}
for {set i 0} {$i < $NODES} {incr i} {
    $ns duplex-link $n_sink($i) $db(1) 100Mb 1ms DropTail
}

# setup tracing
set trace_file [open "tracefile.trq" w]
for {set i 0} {$i < $NODES} {incr i} {
    $ns trace-queue $n_src($i) $db(0) $trace_file
    $ns trace-queue $n_sink($i) $db(1) $trace_file
}
$ns trace-queue $db(0) $db(1) $trace_file
$ns trace-queue $db(1) $db(0) $trace_file

# setup TCP Agents
for {set i 0} {$i < $FLOWS} {incr i} {
    set src($i) [new Agent/TCP/FullTcp]
    set sink($i) [new Agent/TCP/FullTcp]
```

```
    $src($i) set fid_ $i
    $sink($i) set fid_ $i
}

# attach agents to nodes
set node_ind 0
for {set i 0} {$i < $FLOWS} {incr i} {
    $ns attach-agent $n_src($node_ind) $src($i)
    $ns attach-agent $n_sink($node_ind) $sink($i)
    incr node_ind
    if {$node_ind >= $NODES} {
      set node_ind 0
    }
}

# make the connections
for {set i 0} {$i < $FLOWS} {incr i} {
    $ns connect $src($i) $sink($i)
    $sink($i) listen
}

# create random variables
set recvr_delay [new RandomVariable/Weibull];     #
$recvr_delay set shape_ $shape
$recvr_delay set scale_ $scl
set sender_delay [new RandomVariable/Weibull];    #
$sender_delay set shape_ $shape
$sender_delay set scale_ $scl
set recvr_bw [new RandomVariable/Constant];       # bw 100 Mbps
$recvr_bw set val_ 100
set sender_bw [new RandomVariable/Uniform];       # bw 1-20 Mbps
$sender_bw set min_ 1
$sender_bw set max_ 20
set loss_rate [new RandomVariable/Uniform];       # loss 0-1% loss
$loss_rate set min_ 0
$loss_rate set max_ 0.15

# setup rules for DelayBoxes
for {set i 0} {$i < $NODES} {incr i} {
    $db(0) add-rule [$n_src($i) id] [$n_sink($i) id] $recvr_delay
$loss_rate \
          $recvr_bw
    $db(1) add-rule [$n_src($i) id] [$n_sink($i) id] $sender_delay
$loss_rate \
          $sender_bw
}

$db(0) set-delay-file "db0.txt"
$db(1) set-delay-file "db1.txt"

proc finish {} {
    global ns
    $ns flush-trace
    $ns halt
}

# schedule traffic
for {set i 0} {$i < $FLOWS} {incr i} {
    $ns at [expr $i + 0.5] "$src($i) advance $PCKTS"
```

```
}

$ns at 1000.0 "finish"

# start the simulation
$ns run
```

APPENDIX C: Simulation Code for state transitions

```
#
# A script that
# 1. determines the current congestion state
# 2. change to next state based on transition matrix
# 3. determine the new shape parameter (beta)
#

# useful constants
set beta 1

# setup ns
remove-all-packet-headers;            # removes all packet headers
add-packet-header IP TCP;             # adds TCP/IP headers
set ns [new Simulator];               # instantiate the simulator


#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

global defaultRNG
$defaultRNG seed 999

# define the states
#set NO_CONG "not_congested"
#set LOW_CONG "low_congestion"
#set HIGH_CONG "high_congestion"

set NO_CONG 0
set LOW_CONG 1
set HIGH_CONG 2

set state_idx 0

# put states in an array
set states [ list $NO_CONG $LOW_CONG $HIGH_CONG ]

# default state --> not_congested
set curr_state [ lindex $states $state_idx ]
set old_state $curr_state

# transition matrix for the states
set trans_mtrx { {0.2 0.9 1.0}
      {0.5 0.8 1.0}
      {0.1 0.7 1.0} }


# procedure to determine if there's a change in state
proc get_state { st } {

   global curr_state NO_CONG LOW_CONG HIGH_CONG trans_mtrx states
defaultRNG beta  state_idx
   global old_state

   $defaultRNG seed time
```

```
set curr_state [lindex $states $st]

#
for {set j 0} {$j < 100} {incr j} {
   set ran  [uniform 0 1]

   # Part 0: Select row to use from the transition matrix,
   # depending on current state
   if { $curr_state == $NO_CONG } {

         #set ran  [uniform 0 1]
         set s0 [lindex [lindex $trans_mtrx 0] 0]
         set s1 [lindex [lindex $trans_mtrx 0] 1]
         set s2 [lindex [lindex $trans_mtrx 0] 2]

   } elseif { $curr_state == $LOW_CONG } {

         #set ran  [uniform 0 1]
         set s0 [lindex [lindex $trans_mtrx 1] 0]
         set s1 [lindex [lindex $trans_mtrx 1] 1]
         set s2 [lindex [lindex $trans_mtrx 1] 2]

   } else {

         #set ran  [uniform 0 1]
         set s0 [lindex [lindex $trans_mtrx 2] 0]
         set s1 [lindex [lindex $trans_mtrx 2] 1]
         set s2 [lindex [lindex $trans_mtrx 2] 2]

   }


   set old_state $curr_state

   #PART 2: Do the state transition & select a random beta
   #
   if {$ran < $s0} {

         set curr_state $NO_CONG
         set beta [uniform 1.0 2.6]
         set state_idx 0

   } elseif {$ran >= $s0 && $ran < $s1} {
         set curr_state $LOW_CONG
         set beta [uniform 2.6 3.7]
         set state_idx 1

   } else {
         set curr_state $HIGH_CONG
         set beta [uniform 3.7 15.0]
         set state_idx 2

   }

   # display the state change
   #if {$old_state == $curr_state} {
   #     puts "No change, beta: $beta"

   #} else {
```

```
            puts "$old_state, $curr_state, $beta"

      #}

   }

}

#Define a 'finish' procedure
proc finish {} {
        global ns nf
        $ns flush-trace
      #Close the trace file
        close $nf

      exit 0
}

$ns at 0.1 "get_state $state_idx"
$ns at 50.0 "finish"

#Run the simulation
$ns run
```

APPENDIX D: Weibull Random Number Generator for ns2

The following changes were made to various ns2 files to implement the Weibull random number generator, adapted from [54].

a) The following code was added to `ranvar.cc` to set up `RandomVariable/Weibull`, which can be invoked from a TCL script.

```
static class WeibullRandomVariableClass : public TclClass {
 public:
       WeibullRandomVariableClass():TclClass("RandomVariable/Weibull") {}
        TclObject* create(int argc, const char*const* argv) {
                if (argc >= 6) {
                        return (new WeibullRandomVariable (
                               (double) atof(argv[4]),
                               (double) atof(argv[5])));
                }
                else {
                        return (new WeibullRandomVariable());
                }
        }
} class_weibullranvar;

WeibullRandomVariable::WeibullRandomVariable()
{
        bind("shape_", &shape_);
        bind("scale_", &scale_);
}

double WeibullRandomVariable::value()
{
        return(rng_->rweibull(scale_, shape_));
}
WeibullRandomVariable::WeibullRandomVariable(double shape, double scale)
{
        shape_ = shape;
        scale_ = scale;
}
```

b) Definitions for these functions were added to `ranvar.h`.

```
class WeibullRandomVariable : public RandomVariable {
public:
        virtual double value();
        virtual double avg() {return value(); }
        WeibullRandomVariable();
        WeibullRandomVariable(double shape, double scale);
        double* shapep() { return &shape_; };
        double* scalep() { return &scale_; };
        double shape()   { return shape_; };
        double scale()   { return scale_; };
        void setshape(double d)  { shape_ = d; };
        void setscale(double d)  { scale_ = d; };
private:
        double shape_;
        double scale_;
};
```

c) The following line inserted in `rng.h`.

```
   inline double rweibull(double shape, double scale) {
         return (pow (-log (uniform()), 1/shape) * scale);
     }
```

d) The following line was added to `random.h`.

```
static double rweibull(double shape, double scale) {return
        rng()->rweibull(shape, scale);}
```