PH.D. THESIS

A Long-Term Rate Scheduling Framework for Bulk Data Multicast Dissemination in Hybrid Heterogeneous Satellite-Terrestrial Networks

by Apinun Tunpan
Advisor: M. Scott Corson

CSHCN PhD 2002-5
(ISR PhD 2002-7)

ABSTRACT

Title of Dissertation: A LONG-TERM RATE SCHEDULING FRAMEWORK FOR BULK DATA MULTICAST DISSEMINATION IN HYBRID HETEROGENEOUS SATELLITE-TERRESTRIAL NETWORKS

Apinun Tunpan, Doctor of Philosophy, 2002

Dissertation directed by: Dr. M. Scott Corson
Institute for Systems Research

We study a problem of long-term rate control for multicasting bulk data to heterogeneous receivers through hybrid satellite-terrestrial networks. By setting one of the control parameters called the *'knob'*, our proposed multicast rate control systems can trade off between multicast bandwidth requirement, which is the administrative issue, and reception latency, which is the users' satisfaction issue. Through a proactive use of the forward error correction (FEC) in the form of Reed-Solomon erasure (RSE) correcting codes, our proposed multicast rate control can also probabilistically guarantee reception reliability.

In this dissertation, there are four contributions. One, our real experiments on a hybrid satellite-terrestrial internet channel that result in our proposed Gaussian approximation approach. Two, our proposed end-to-end version of the multicast rate control system which utilizes the Gaussian approximation approach. Three, our study on the issues of multicast traffic's co-existence with terrestrial TCP background traffic. Four, our proposed two-staged version of the multicast rate control system which also utilizes the Gaussian approximation approach and allows significantly improved scalability. We validate our proposed systems through simulations, some of which are trace-driven using real satellite traffic traces. We have found our proposed mechanisms to be very effective.

There are a variety of applications for our proposed multicast rate control systems. Some examples include: a military application where commands or strategic information need to be disseminated at different priorities, a financial application where a faster access to certain key information can make a difference in profitability, and a news application where news-related contents such as images, weather, headlines are to be disseminated based on their urgency.

Keywords: satellite multicast, reliable multicast, forward error correction (FEC), multicast rate control, Gaussian approximation

A LONG-TERM RATE SCHEDULING FRAMEWORK FOR BULK DATA MULTICAST DISSEMINATION IN HYBRID HETEROGENEOUS SATELLITE-TERRESTRIAL NETWORKS

by

Apinun Tunpan

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2002

Advisory Committee:

Professor A. Udaya Shankar, Chair
Dr. M. Scott Corson, Co-chair/Advisor
Associate Professor Leana Golubchik
Professor Raymond Miller
Professor Robert Newcomb

DEDICATION

To the memory of my grandmother Pratum,
who watched over my very first steps.

To my parents,
who always appreciate the great value of higher education.

And, to my violin teacher, Mr. Sutin Srinaronk,
who taught me of patience and practice.

ACKNOWLEDGEMENTS

First, this dissertation would not be possible without the guidance and support from the dissertation advisor, Dr. M. Scott Corson. To whom, the author would like very much to express his gratitude. The author also appreciates the suggestions from the members of the dissertation advisory committee.

The author thanks Go Hasegawa of the Osaka University in Japan for the discussion on DRR+ queueing and its sample codes, Professor Alan Genz of the department of Mathematics, Washington State University, for the discussion on multivariate Gaussian probabilities and the sample codes, Mr. Joseph Macker (NRL) and Mr. R. B. Adamson for the discussion on the Multicast Dissemination Protocol (MDP).

The author also thanks Ms. Diane Hicks and Mr. Jau-Ling Chou at the Center for Satellite and Hybrid Communication Networks, University of Maryland College Park, respectively for their numerous helps regarding administrative issues and technical issues.

The author thanks Ms. Darawalee Wangsa who partly helped proofread this dissertation.

Lastly, the author thanks the Institute for Systems Research (ISR) at the University of Maryland for research funding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

A satellite system is an effective medium for multicast data dissemination. A satellite multicast network can be very large, covering distant or isolated geographical areas, but rather flat. Only a short hierarchy would exist between the sender and the receivers. The large bandwidth-delay product and the likely bandwidth-limited feedback channels (i.e. from receivers back to the sender) of the asymmetric satellite network makes traditional receiver-initiated repair mechanisms (e.g. with the use of Negative Acknowledgement or NACK) inefficient, especially when the receiver population is large and heterogeneous. Considering that the cost of having and operating a satellite channel can be high, there are two competing aspects under our consideration. First, we want the most efficient use of the satellite network's bandwidth as well as the terrestrial network's bandwidth (i.e. the administrative issue). Second, we want to minimize delivery delay (i.e. the users' satisfaction issue).

This dissertation addresses rate control issues in deploying satellite multicast to heterogeneous receivers. Receiver heterogeneity arises from a number of reasons. Examples include: different link capacities, underlying background traffic or congestion in the network, and different bandwidth allocation policies. In a heterogeneous receiver population, the transmission rate affects the reception quality at each individual receiver differently. Fast receivers or those which sit behind uncongested, unrestricted routers can receive data at a relatively high transmission rate. On the other hand, a slow transmission rate would benefit slow receivers or those who are located behind congested or restricted routers.

1.1 The Long-Term Multicast Rate Control Problem

We study a multicast rate control problem of disseminating a large bulk file to heterogeneous receivers via a hybrid satellite-terrestrial network. The bulk file could contain time-sensitive information such as financial data that is available only at certain times of day and has to be disseminated to receivers as quickly as possible. On the other hand, the bulk file could contain less time-sensitive information such as weather information, news, images, or software applications. One key constraint is that **the whole bulk file must be completely delivered** to each receiver in order to be useful. Any fraction of the bulk file is considered useless. This is particularly true in the case of highly compressed files (i.e. with minimal redundancy) or in the case of application files.

We want a multicast rate control mechanism which can trade off between bandwidth consumption and delivery delay. At the same time, a high degree of delivery reliability should be guaranteed. The bandwidth/delay trade-off setting will be a controllable **knob** parameter to the system. The multicast rate control mechanism will utilize packet survival statistics collected over a long period of time, and will work in a long-term time scale, in the order of the time required for file transmission itself (i.e. this is not a short-term transmission control like TCP that potentially adjusts the rate every round-trip time).

To provide a framework to solve the multicast rate control problem, we need to understand the characteristics of the hybrid satellite-terrestrial internet network. We also need to understand how different multicast transmission rates would impact multicast packet survivals, especially among heterogeneous receivers. Also based on such packet survival information, we must establish a computational model as a state space that the rate control mechanism can work on algorithmically. The multicast traffic resulting from the rate control mechanism will have a significant impact on existing background traffic. Therefore, we also need to study the issues on traffic co-existence.

This dissertation is organized into eight chapters. This chapter describes scenario, concepts and other assumptions that will be used in later chapters. Chapter 2 presents a survey of related work. Chapter 3 investigates and analyzes the characteristics of a real hybrid satellite-terrestrial Internet channel. We establish our Gaussian approximation approach and validate it in Chapter 4. We then formulate and solve the end-to-end version of the long-term multicast rate control problem in Chapter 5. In Chapter 6 we study the impact of multicast traffic on the terrestrial background TCP traffic, and propose a simple way to allow the multicast traffic from the satellite to co-exist with terrestrial TCP. We then propose a two-staged scalable version of the long-term multicast rate control in Chapter 7. Finally, Chapter 8 summarizes our work and presents some possible future research issues generated by this dissertation.

Note that parts of the materials in Chapter 5 formerly appeared in [TC00a, TC00b] and parts of the materials in Chapter 6 formerly appeared in [TC01].

1.2 The Asymmetric Satellite Multicast Scenario

We have a single multicast sender which can transmit packets (either data or parity) at some discrete rate to a *forward satellite channel* via a gateway and an uplink. We assume that only **one** high-speed satellite forward channel is available to us. That is, an earth station will subscribe to this channel when at least one of its downstream receivers wants to receive the bulk data files.

The satellite channel simply acts as a communication pipe with a large bandwidth-delay product. Multicast packets passing through the satellite channel may be subject to certain losses. The satellite does not perform any on-board packet processing[1]. Figure 1.1 illustrates an example of the satellite multicast scenario.

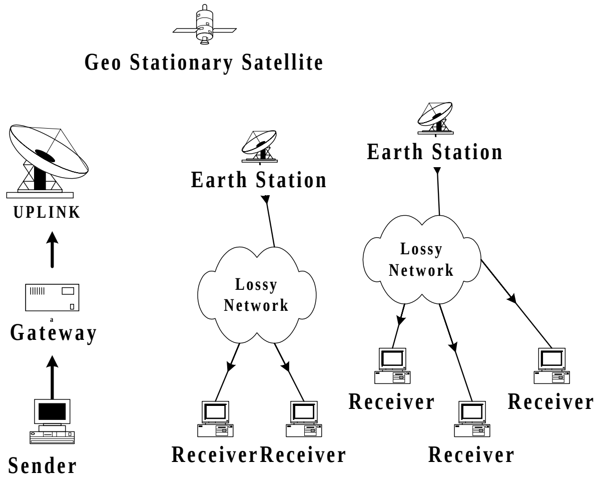


Figure 1.1: An example satellite multicast scenario

[1] It is to note that the new generation [Ale00] of satellites may have on-board packet processing capabilities, and/or can establish inter-satellite links to expand the coverage area. We still do not exploit such features at the moment. We will further study the issues in our future work.

Once the packet arrives at the earth station, it is then multicasted through a terrestrial Internet-like network to downstream receivers associated with that earth station. We assume that there is no terrestrial connectivity among these earth stations or among receivers belonging to different earth stations, and that the satellite forward channel is the only source to receive multicast traffic.

Depending on the specific scenarios that we will further study, each receiver can periodically and reliably report feedbacks (which are primarily packet survival statistics) back to either its parent earth station or to the sender. The feedback may go through terrestrial routes, or through the satellite *return channel* which has limited capacities. The notion of **receiver representatives** can be implemented to extend the scalability of the rate control mechanism. With receiver representatives, only a smaller subset of receivers may report the feedback. Generally receivers which belong to the same sub-net can have similar statistics, and it is more economical to report only one statistics (along with the number of receivers the statistics represents) per sub-net. We assume that the set of receiver representatives is pre-defined and known to our rate control mechanism. The selection (or election) and maintenance processes of receiver representatives (e.g. [DA01]) is not included in the scope of our work.

In Chapter 5, we study a case where the multicast sender is the only one who can control the multicast transmission rates. In Chapter 7, on the other hand, we study a case where we delegate most of the rate control functionalities to the earth stations.

1.3 Forward Error Correction (FEC)

The general concepts of FEC is discussed in this section. We shall further present related work on FEC in Section 2.1.

Forward error correction is a technique of transmitting redundant (or parity) information additionally to the original data to correct errors (i.e. corrupted data) or erasures (i.e. lost data) that may occur with the original data.

In coding theory, there are a number of error correcting codes that have different error correcting capabilities. We are interested in a class of **block codes** where the encoding and decoding algorithms work with a finite number of data and parity symbols at a time.

There are two broad steps in the process of error correction: the first step is to locate where the error(s) occur, the second step is to deduce, through some algebraic structures of the code, the 'correct' code word. If a code can correct errors, it can also correct erasures. Unlike in error correction, the erasure correcting algorithm knows exactly where the erasures are.

In this paper, we assume the use of FEC to perform **erasure correction** of packets only. A packet can be lost in two ways, either that a network router is congested and has to drop the packet or that the packet is corrupted due to transmission noise and is recognized as being corrupted and dropped by a data link protocol. With a good framing design of the data link protocol, we can assume that the probability that a corrupted packet is recognized as being a valid packet is negligible.

There is a powerful class of algebraic block codes, namely the Reed-Solomon erasure correcting code (RSE). Assuming that a coding symbol is β bits long and a RSE-based FEC block of size B consists of h data symbols, and $c = B - h$

parity symbols. The decoding algorithm can re-construct all the h original data symbols whenever the algorithm knows at least h distinct symbols, which can be either data or parity symbols, out of the total $B = h + c$ symbols. McAuley [McA90] described how the RSE encoding and decoding are done. There is a bound $h + c < 2^{\beta}$.

Figure 1.2 shows how FEC blocks are arranged into packets for transmission. Basically, a number of parallel FEC blocks are put into the packets 'vertically' as shown in the figure, while the packets appear 'horizontally'. A collection of these $B = h + c$ data and parity packets will be called a **data block**. When a j-th packet is lost, $j = 1, \ldots, h + c$, it will result in the j-th code word erasure of every parallel FEC blocks. The key to a successful data block decoding is simply that a receiver needs to receive at least h distinct packets either data or parity. The same bound $h + c < 2^{\beta}$ also applies here.

FEC is very useful in multicast communications. When multiple multicast receivers lose different data packets from one specific data block, the repair mechanism can transmit the same parity packets from the data block to correct such different erasures at multiple receivers.

A large bulk file may not fit into one data block. Without loss of generality, we assume that the bulk file is encoded into $W \geq 1$ data blocks, each distinctly labeled by a number $w \in \mathcal{W} = \{1, 2, \ldots, W\}$. A reliable reception at a specific receiver means that the receiver has received sufficient packets (either data or parity) to decode all the W blocks (i.e. the receiver has received at least h distinct packets from each and every W blocks).

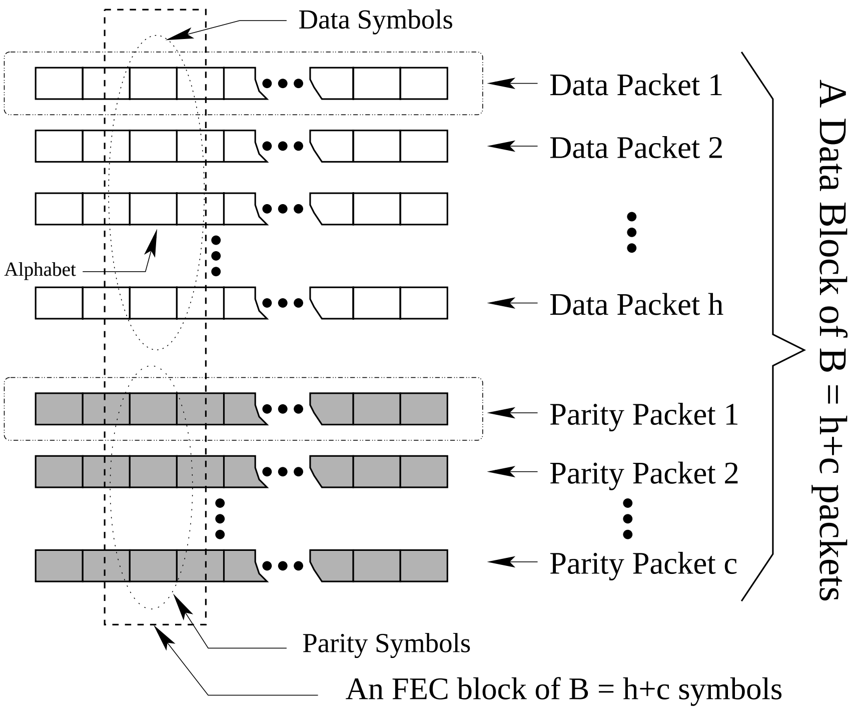


Figure 1.2: Arrangement of FEC code words within a data block.

Chapter 2

A Survey of Related Work

We present a survey of related work in this chapter. Diot et al [DDC97] addressed a number of issues in multicast communication (which can be one-to-many or many-to-many communication). We focus, however, only on the particular areas of **traffic control**, **reliability**, and **scalability** (i.e. Sections II.C, III.A, and III.C of [DDC97] respectively), especially in the one-to-many scenario.

Among the traffic control issues are the issues of

1. congestion control (e.g. how to avoid or deal with arising network congestion)
2. bandwidth/delay control (e.g. how to provide bandwidth or delay guarantee)
3. fairness (e.g. how to define fairness, how to enforce fairness)

Traffic control can become more complicated when multicast receivers are *heterogeneous* in their reception rates (i.e. when some receivers receive better at one particular transmission rate but worse at the other transmission rates).

To guarantee reliable reception, we need some form of **transmission control protocol** and **error recovery protocol**. The transmission control protocol deals with the progress of the transmission. In the event of a packet loss, the error recovery protocol dictates how the sender should transmit a **repair packet** (which can be either a retransmission of the lost data packet or a transmission of a FEC parity if the erasure-correcting capability is assumed). In a unicast environment, a receiver can respond to the sender with a positive acknowledgement, or **ACK** for short, upon the reception of each data packet. The sender can then keep track of the delivery status in progress. When the sender does not see an ACK for a particular data packet (i.e. because either the packet is missing or the ACK is missing), the sender transmits a repair packet. Such approach is generally referred to as a **sender-reliable** or **sender-initiated** error recovery mechanism. In multicast scenarios, however, a multicast sender using the sender-initiated approach can suffer from an **ACK implosion problem** where there are too many ACKs for the sender to process and from a **scalability problem** that the sender cannot keep track of every receiver's delivery status (e.g. protocol *A* of [TKP97]). An alternative is the **receiver-reliable** or **receiver-initiated** error recovery mechanism (e.g. protocols *N1* and *N2* of [TKP97]). In such approach, each receiver is responsible for keeping track of its own reception status. If a receiver sees a missing data packet, the receiver issues a negative acknowledgement, or **NACK**. The sender only transmits a repair packet upon seeing the NACK. This approach reduces the computational load at the sender. However, there is still a possibility of a **NACK implosion** problem when there is a commonly missing packet and a large number of receivers issue the NACKs. A clever use of a timer backoff algorithm, along with NACK multicasting, and NACK suppression (e.g. in SRM [FJL$^+$97]) can help reduce the number of NACKs transmitted by receivers that experience packet losses. In practice, it is preferable to have only a few receivers who are located near to the points of packet loss (and thus are able to detect the loss earlier) to be active receivers who transmit NACKs. This notion also directly leads to the idea of having **receiver representatives** (e.g. in [DO97, DA01]).

Towsley et al [TKP97] analyzed and compared the sender-initiated and the receiver-initiated approaches in non-FEC scenarios (i.e. the repairs are the retransmitted data packets only). Three generic protocols were studied. In all of these protocols, only the sender transmits the data and repair packets.

- Protocol *A* is a sender-initiated protocol. In *A*, the sender needs to maintain an ACK list. When a receiver receives a packet, the receiver transmits an ACK for the received packet. The sender uses timers to detect packet losses and selectively repeats the transmission of only the packets that are suspected to be lost.

- Protocol *N1* is the first variation of a receiver-initiated protocol. When a receiver detects a lost packet, the receiver *unicasts* a NACK for the missing packet to the sender, and initiates a random timer. If the timer expires prior to receiving the requested packet, it is an indicator of packet loss and another NACK is sent and another timer cycle starts. No collaborating NACK suppression mechanism can be deployed in *N1*.

- Protocol *N2* is the second variation of a receiver-initiated protocol. A receiver who detects a packet loss initiates a random delay timer. If the timer expires without receiving the missing packet and without receiving the NACK for that missing packet, the receiver *multicasts* a NACK for the missing packet and again starts a timer. If another receiver receives the NACK for a packet the receiver has not received, but for which it has initiated a random delay timer before sending a NACK, the receiver suppresses its own NACK and starts a timer as if it were the one who sent the NACK. Protocol *N2* tries to ensure that there is at most one NACK for each missing packet. The NACK generated by the receiver whose delay timer expires first generally suppresses the other receivers' NACKs.

In their work [TKP97], one-to-many multicast and many-to-many multicast scenarios were studied. In the one-to-many scenario, only one fixed sender exists. In the many-to-many scenario, each participating multicast member is equally likely to be a sender. They studied the processing rates (packets/sec) at both the sender and receivers and tried to determine who is the bottleneck (i.e. whose processing rate is the lowest) in the two mentioned scenarios. In the **one-to-many scenario**, the sender of protocol *A* quickly becomes a system bottleneck as the number of receivers increases. The sender in protocol *N1* performs slightly better than the sender of protocol *A*, but eventually becomes the system bottleneck as well. Protocol *N2* turns out to be the best in terms of throughput. In the **many-to-many scenario**, protocols *N1* and *N2* still outperform protocol *A* but are not as significantly different as those of the one-to-many scenario. This is due to the fact that, in the many-to-many scenario, the receiver processing rates now dominate the system throughput (unlike the sender processing rate in the one-to-many scenario). It was shown that protocol *N1* outperforms protocol *N2* in the many-to-many scenarios because the way *N2* receivers multicast their NACKs increases the complexity of receiver processing.

2.1 Forward Error Correction (FEC)

We described the basic concepts of Reed-Solomon FEC repair mechanism earlier in Section 1.3. The RSE-based FEC repair mechanism can efficiently supplement a variety of transmission control and error recovery protocols. A single parity packet can correct different packet losses (as long as they are in the same block as the parity packet's) at multiple receivers. FEC can therefore help reduce the repair bandwidth required in a multicast environment.

FEC can be used as either a **layered FEC** approach or an **integrated FEC** approach. The layered[1] FEC approach (e.g. Section 3.1 of [NBT97]) takes an aggressive step of transmitting *all* of the FEC parity packets (*before* they are requested) along with original data packets. In an integrated FEC approach (e.g. Section 3.2 of [NBT97] or [Mac97]), data packets are transmitted first, along with *none* or *some* (but not all) of the FEC parity packets. Additional FEC parity packets will be subsequently transmitted upon a request (i.e. through NACK). If none of the parity packets are transmitted along with the original data packets, it is sometimes referred to as "**reactive FEC**". However, if some of the parity packets are transmitted along with the original data packets, it is sometimes called "**proactive FEC**" (e.g. [RKT98]).

Nonnenmacher et al [NBT97] showed that the RSE-based FEC repair mechanism is effective in a wide variety of multicast scenarios. These include: temporally independent packet losses (e.g. assuming packets are lost according to some i.i.d. Bernoulli process), bursty losses (e.g. assuming the loss processes are governed by a two-state Markov chain), spatially uncorrelated losses (i.e. losses

[1] The term "layered" used here is closely related to a term "layered" used in the context of the management of the multicast channels or groups to be described later

that occur independently), spatially correlated losses (i.e. shared losses that occurs at some shared internal link of a multicast tree), and from homogeneous losses to heterogeneous losses. Nonnenmacher et al also arrived at the following conclusions. First, the integrated FEC approach generally outperforms the layered FEC approach in terms of the bandwidth requirement in comparable scenarios. This is due to the fact that additional parity packets are transmitted only when needed. Second, in a scenario with burst losses, the layered FEC approach requires even more bandwidth than a non-FEC counterpart. The integrated FEC with interleaved transmission of packets from different FEC blocks helps reduce the bandwidth by subjecting each FEC block to a relatively shorter length of burst losses. Using larger FEC blocks can help reduce the impact of burst losses as well. A moderate FEC block size ($h = 20$ in their settings), however, can close the bandwidth performance gap between the interleaved and non-interleaved integrated FEC transmissions. A large FEC block size ($h = 100$ in their settings) can even eliminate the need for interleaved transmission. However using a very large FEC block is not desirable because of the encoding/decoding complexity. Third, when losses are spatially correlated (i.e. shared losses), one can model shared loss by a reduced number of receivers that lose packets independently. This has an important implication that if a transport protocol assumes independent losses when coping with shared losses, the protocol can overestimate the number of FEC repair packets needed to guarantee ‘global reliability’. Fourth, when using the integrated FEC, the sender will become the performance bottleneck. This problem can be alleviated by pre-encoding the data offline or using a faster machine for the sender. Fifth, in the case of heterogeneous receiver, high-loss receivers have a significant impact on the performance even there are a few of them.

A proactive repair technique is capable of providing bandwidth-delay tradeoff in a multicast transmission because the parity packets that are transmitted along with original data packets can compensate the losses, and thus reduce the need of sending a NACK (which in turns reducing the NACK implosion problem) and the waiting for a repair packet to arrive. **Our work in Chapter 5 ([TC00a, TC00b]) and Chapter 7 enhances the proactive FEC technique by further providing a tunable tradeoff between bandwidth and delay through the adjustments of the transmission rate sequence.**

Other applications of the RSE-based FEC repair mechanism can be seen in [JNB99, RKTK99, RV98]. One problem that arises from using the RSE-base FEC repair mechanism is the **decoding complexity** [BLR98]. Suppose only $h - c$ packets out of the h original data packets arrive, a receiver will need c parity packets to reconstruct the lost packets. The reconstruction process requires solving a system of c equations (i.e. for c unknowns). The Reed-Solomon construction allows this system of equations to be solved in $O(c^2)$ time via matrix inversion and matrix multiplication. This corresponds to a system of *dense* linear equations. In an ideal situation, one would want to encoded the whole bulk file into one single RSE-based data block. In practice, to circumvent decoding complexity we have to divide a large bulk into smaller blocks and transmit these blocks interleavingly. **RSE decoding inefficiency due to interleaving** will arise because a packet from one RSE-based FEC block is not useful to the other RSE-based FEC blocks.

Byers et al [BLR98] suggested the use of **Tornado codes** which are constructed from *sparse* random equations (each equation, on average, has a smaller number of variables). Unlike the RSE-based FEC codes which use finite field operations, the Tornado-based FEC codes use 'XOR' operations. To recover h original data packets, a Tornado-based FEC decoder needs slightly more than h distinct packets (either data or parity). The **Tornado decoding inefficiency due to the use of sparse equation system** arises. This decoding inefficiency is a random variable due to the structure of the sparse equation systems. A special case (based on the structure of the equation system) called Tornado-Z codes was studied. Their results show that on average the Tornado-Z decoding inefficiency is 1.054 (i.e. they would require on average 1.054 $h/2$ original data packets and 1.054 $h/2$ parity packets to decode one block in their settings). For the RSE-based codes, they would be able to reconstruct a data block from exactly h/2 data and h/2 parity packets. Their results also reveal that for the same FEC block size (i.e. non-interleaved RSE-based FEC v.s. Tornado-Z FEC), the Tornado-Z encoding/decoding needs only a very small fraction of the times required by the RSE encoding/decoding. Thus, **a large bulk file can be encoded into one Tornado-based FEC block with a slight increase in the decoding inefficiency, but receiver can decode the file at much greater speed than the interleaved RSE counterpart.** This makes Tornado codes a very interesting choice for ongoing research in bulk data multicasting.

2.2 Error Recovery Techniques for Reliable Multicast

Error recovery techniques in multicast can be divided into two broad categories [LNB00]: **centralized error recovery (CER)** and **distributed error recovery (DER)**. In a CER approach, only the sender transmits the repairs (either the data or parity packets) for error recovery. In a DER approach, some or any multicast member may transmit the repairs (either data or parity packets). The DER approach can be further divided into two subcategories. If the multicast members are partitioned into smaller local groups and the retransmissions are limited to each group, it is called **Grouped DER**. In **Ungrouped DER**, any multicast member can transmit the repairs. Grouped DER is sometimes also referred to as **local recovery**.

Protocols *A*, *N1*, and *N2* of [TKP97] described earlier are CER by the above definition since only the sender transmits repair packets (which are data packets). RMTP [PSLB97] is a Grouped DER protocol which utilizes a receiver hierarchy and designated receivers to perform retransmission within a local group. SRM [FJL$^+$97] is an Ungrouped DER protocol. SHARQFEC [Ker98] is a Grouped DER protocol.

Kasera et al [KKT98] studied two non-FEC DER approaches: the **server-based local recovery** and the **receiver-based local recovery**. In the server-based local recovery approach (i.e. protocol *L1* of [KKT98]) uses designated hosts called **repair servers** which are attached to routers (i.e. this is equivalent to having multicast-enhanced routers which can serve as repair servers). Each repair server responds to retransmission request within a specific area by retransmitting the requested data packets. The receiver-based local recovery approach (i.e. protocol *L2* of [KKT98]) does not utilize designated repair servers; only end-hosts (sender or receivers) get involved in the repair process. The receiver-based local recovery approach also includes the dynamic selection of a receiver within a specific area to supply repairs when attempting a local recovery of a lost packet.

- In protocol *L1*, the sender multicasts data packets to a global multicast address which is subscribed by every receiver and repair server. When a *receiver* detects a packet loss, it waits for a random period of time before multicasting a NACK to the receiver's local neighborhood and the local repair server and starting a NACK retransmission timer. If, however, while waiting, the receiver receives a NACK from another receiver for this missing packet, the receiver suppresses its own NACK and starts a NACK retransmission timer as if the receiver had sent that NACK. Similarly, when a *repair server* detects a packet loss, it waits for a random period of time before multicasting a NACK to the sender (or the upstream repair servers) as well as to the other repair servers which reside at the same multicast tree level. If, however, while waiting, the repair server hears a NACK from another repair server, it suppresses its own NACK and starts a NACK retransmission timer as if the repair server had transmitted that NACK. When receiving a NACK from the group a repair server is associated with, if the repair server can deliver the requested data packet, it multicasts the packet to the group. Otherwise the repair server starts a packet request cycle from the sender (or the upstream repair server).

- In protocol *L2*, the sender multicasts data packets to a global multicast address which is subscribed by every receiver. When a receiver detects a packet loss, it waits for a random period of time before starting a *local recovery mode* by multicasting a NACK to the local neighborhood and starting a local NACK

> retransmission timer. If the receiver does not receive the lost packet before the local retransmission timer expires, it waits for a random amount of time before starting a *global recovery mode* by multicasting a global NACK and starting a global NACK retransmission timer. If the receiver still does not receive the lost packet before the global NACK retransmission timer expires, the receiver switches into the local recovery mode again. Upon receiving a local NACK, if the receiver has a requested data packet, the receiver waits for a random period of time and if it does not receive the same repair from the other receiver during such period, it multicasts the packet into the local group; otherwise, the receiver suppresses its own repair. When the sender receives a global NACK, the sender always multicasts the repair packet to a global address. Only the *L2* sender can multicast global repairs.

Both the protocol throughput and the bandwidth requirement were analyzed. Protocols *L1* and *L2* here were compared to protocol *N2* (which has a NACK suppression mechanism but does not use local recovery) [TKP97]. The analyses in [KKT98] assumed that the repair server of protocol *L1* has sufficient processing power and never becomes a bottleneck. In terms of throughput, *L2* consistently delivers about twice the throughput of protocol *N2*, while protocol *L1* performs prominently better than protocols *N2* and *L2* as either the packet loss probability or the number of tail links increases. In terms of bandwidth usage, protocol *L1* is also significantly better. Their analysis also shows that the repair servers used in *L1* have to be slightly faster than the receivers and only have some small amount of buffer to keep the repair packets when the loss probability is in a reasonable range. In summary, [KKT98] shows that using local recovery (DER) significantly benefits the non-FEC multicast scenarios and that the server-based local recovery approach (i.e. with dedicated repair servers) is better than the receiver-based local recovery approach.

Rubenstein et al [RKTK99] further studied the case of a server-based DER approach utilizing FEC. Their approach, called the **"Active Parity Encoding Service" (APES)**, combines the repair server with FEC parity encoding. APES partitions receivers into smaller repairs domains; each has an associated repair server. Three APES approaches based on how the repair servers encode, decode, store and forward repair packets were studied. The three approaches are

1. **"Store-Data-Build-Repairs" protocol (SDBR)**,
2. **"Build-Repairs-Store-Repairs" protocol (BRSR)**, and
3. **"Get-Repairs-Store-Repairs" protocol (GRSR)**.

In all APES approaches, a sender multicasts *source* packets (which can be either data or parity) to receivers. Each repair server is located on the multicast tree at a point between the sender and the corresponding receiver domain. This allows the repair servers to receive the source packets and later use them for local recovery.

- In the SDBR protocol, when the repair server receives enough source packets (either data or parity), the repair server decodes the packets to reconstruct the h original data packets, and buffers these data packets. When additional FEC parity packets are needed by receivers, the repair server generates distinct repairs by FEC encoding.

- In the BRSR protocol, the repair server generates a pre-determined number a ($0 \leq a \leq h$) of FEC parity packets "on-the-fly", as the source packets are passing through the repair server. All the a parity packets will become useful after at least h source packets are received at the repair server. The repair server stores these a parity packets and multicasts some to the repair domain as needed. If any receiver in the repair domain experiences more than a packets, the receiver will need to obtain additional repairs from the sender.

- In the GRSR protocol, the repair servers do not have FEC capability. They simply request a repair packets from the sender and after that operates exactly the same way that the BRSR protocol does. Since the repair server in GRSR does not perform FEC encoding/decoding, it will require additional bandwidth on the links between the sender to the repair server.

The bandwidth and buffer requirements, but not the throughput, were analyzed. BRSR and GRSR requires the same amount of bandwidth between a repair server and the receivers. SDBR is most bandwidth efficient, since SDBR always transmits the minimum number of distinct repairs. Under the domain sizes and loss rates expected in practice, the BRSR's bandwidth requirements between repair server and receivers are similar to SDBR's. When considering the bandwidth between a sender and a repair server, their analysis shows that for sufficiently large a, the additional bandwidth needed from the sender by BRSR or GRSR is negligible. Since SDBR stores original data packets, the buffer requirement of SDBR at a repair server is equivalent to that of a non-FEC local recovery protocol. The buffer requirements of GRSR are very small compared to the buffer requirements of the non-FEC or SDBR protocols. BRSR requires more buffer to accommodate a fixed miss probability at slow data rate; but as the data rate increases, BRSR's buffer requirements reduce and approach those of GRSR. APES is also shown to effectively help reduce bandwidth in inter-heterogeneous scenarios, i.e. where different domains have different loss rates. The use of repair servers helps limit the bandwidth usage to the high-loss domains.

Recently, Lacher et al [LNB00] analyzed generic CER and DER protocols, both using FEC and not using FEC as the repairs. Both CER and DER protocols can benefit from the use of FEC. Four generic protocols were compared in [LNB00]. Protocol C is a CER protocol utilizing FEC while protocol C_{noFEC} is a CER protocol not utilizing FEC. Protocol *D1* is a grouped DER without FEC whereas protocol *D2* is a grouped DER with FEC:

- In protocol C, the sender multicasts h original data packets of a FEC data block to receivers. If the receiver's feedback indicates that any receiver needs repair packets, the sender transmits the FEC parity packets according to the maximum number of required FEC packets among all the receivers. Receivers suppress their feedback as follows: a receiver multicasts feedback (NACK) only either if the receiver's timer expires before the receiver receives another receiver's feedback which indicates at least the same number of required packets, or if the feedback from the other receiver(s) indicates a smaller number of required packets.

- Protocol C_{noFEC} is similar to protocol C, except that C_{noFEC} uses data retransmission instead of FEC.

- In protocol *D1*, a sender is a group leader for all DER nodes which form an extra multicast group. The sender first multicasts data to all the nodes in the multicast tree, then the error recovery cycle for only the DER nodes begins. Once a DER node has received all the data packets successfully, the DER node then starts serving local repair requests (i.e. for receivers that have the DER node as a group leader). Repairs are strictly limited within corresponding groups (i.e. from a sender to DERs or from a DER to receivers). In their work, however, a DER or a receiver *unicast* its NACK indicating the missing packet sequences to the group leader (which is either the sender or the DER respectively).

- Protocol *D2* is similar to protocol *D1* except that *D2* uses FEC parity retransmissions.

These protocols were compared in terms of bandwidth requirements and latency. Their results show that FEC-based error recovery improves the performance in both CER and DER. With a 'fast' FEC scheme like the Tornado-Z codes [BLR98], encoding/decoding FEC will no longer post a major performance concern, but the other hand, will greatly help scalability of a multicast system. The results in [LNB00] also confirms the findings in [RKTK99] that DER performs better than CER in terms of bandwidth if the receivers' loss patterns are bursty or very heterogeneous. CER can, however, have a close bandwidth performance to DER if large FEC data block ($h \gg 0$) is used. In terms of latency, various loss patterns do not have a significant impact on the relative performance of DER and CER. Similar to the case of bandwidth performance, using large FEC data block ($h \gg 0$) also makes CER perform close to DER in terms of latency. The authors of [LNB00] also cite that CER is more attractive to deploy due to its simplicity and its minimal requirement from the network.

Our work in Chapter 5 ([TC00a, TC00b]) can be classified as a CER approach with parity repairs. Our new work in Chapter 7 can be classified as a variant of the grouped DER approach, having the helps from the earth stations in transmitting sender-constructed parity repairs. Unlike the other researchers' work described above, our work is based on a different point of view. Most of the analyses discussed above assume some fixed (although can be heterogeneous) sets of loss probabilities and derive bandwidth requirements and/or delay/throughput. Our work is based on the notion that the source or earth station's transmission rates can affect packet survival probabilities, and thus we can *control* either bandwidth consumptions or latency by a proper choice of the transmission rate sequence. We also primarily depend on the proactive FEC repair technique, but not the timer mechanism, for error recovery.

2.3 Traffic Control Issues for Heterogeneous Receivers

There are at least two research problems in multicasting to heterogeneous receivers. One, how to deal with receiver heterogeneity. Two, how to achieve fair bandwidth allocations between multicast traffic and other background traffic. Traffic control problems are non-trivial and can be aimed to specific goals (e.g. achieving fairness, alleviating congestion, or providing bandwidth-delay trade off).

There have been a number of proposed methods to deal with receiver heterogeneity. Some researches focus on a **multiple-channel** or **layered** approach. Others rely on partitioning heterogeneous receivers into a number of more homogeneous sets. The APES [RKTK99] approach described above, for example, may also be applied to this inter-heterogeneous, intra-homogeneous scenario.

In a layered approach (e.g. [BKTN98, DAZ99, VRC98, RJL$^+$00]), each receiver subscribes to a subset of channels (or layers) in such a way that the total capacity of the subscribed channels does not exceed the receiver's capacity. The layered approach is very suitable for the transmission of continuous data stream (e.g. video) in which receivers can tolerate losses and the reception quality depends on the available capacity. The layered approach can also be applied to reliable multicast (e.g. bulk files) in which a complete reception of the whole file is a key requirement. When applying the layered approach to reliable multicast, we generally see a larger transmission redundancy (e.g. in [BKTN98]) as a tradeoff to smaller reception delays, or face with a channel synchronization problem unless the data is wisely partitioned and transmitted (e.g. [DAZ99]).

Chawarthe et al [CMB00] proposed an approach using **"Reliable Multicast ProXies" (RMXs)** which are intelligent agents. Their approach is based on a

divide-and-conquer idea and departs from traditional end-to-end congestion control. In their approach, a large wide-area heterogeneous multicast network is partitioned into smaller clusters of more-likely homogeneous receivers. Each cluster is associated with a locally-scoped multicast group and has a RMX which serves as the cluster's representative. Within each cluster, the problem of local multicast to homogeneous receivers is more simple. For the communications among different clusters, a RMX communicates to its peers by a reliable unicast protocol (such as TCP). Conceptually, the connection among RMXs can be seen as an overlay network at an application level. Their approach also relies on a notion of **semantic reliability** (i.e. the reliability of information), rather than **data reliability** (i.e. the reliability of the information representation). This means that the multicasted 'information' can be presented at many detail levels. When the 'information' is propagated in in among heterogeneous RMX clusters, a component inside RMX, which understands what the information is, may transform the representation of such information (e.g. lowering the JPEG image resolution) so that the new representation suits the bandwidth available on a congested link.

Our work in Chapters 5 and 7 can be categorized as a single-channel (or non-layered) approach. However, we effectively deal with receiver heterogeneity through the adjusted sequence of transmission rates at the source or the earth stations, and through the use of forward error corrections (FEC).

2.3.1 Fairness Issues

There are many different ways to define, compute and enforce fairness. We present only some recent relevant work in this section.

Studying one-to-many multicast congestion control in the Internet, Golestani et al [GS99] showed a three-way relationship between the choice of traffic regulation parameter (e.g. rate or window size), the need for the traffic regulation algorithm to know receiver round-trip times and the type of fairness that can be achieved through such traffic regulation algorithm. Two commonly used **traffic regulation parameters** for Internet congestion control are **rate** and **window size**. A **traffic regulation algorithm** adjusts the traffic regulation parameter (e.g. in TCP, window size is adjusted) according to the network condition. A rate-based regulation mechanism can be either instantaneous, where instantaneous transmission rate is regulated, or on average (which corresponds to a leaky bucket scheme) where the average transmission rate is kept below an allocated bandwidth. A window-based regulation mechanism keeps the number of outstanding packets (i.e. those who are considered 'traveling' somewhere in the network) below the maximum-allowable window size. Their work [GS99] reveals that to maintain the *same average transmission rate* for every multicast receiver, a window-based traffic regulation mechanism requires a separate window size for each multicast receiver. Using a single global window size would have resulted in a lower transmission rate at a receiver with higher round trip time. Note that using a separate window size for each receiver implies that the task of regulating window sizes should be done at receivers; otherwise a severe scalability problem will occur. In a rate-based traffic regulation mechanism, every receiver can easily share the same transmission rate which is dictated by the slowest receiver. Two fairness orientations were defined for any rate-based or window-based traffic regulation algorithm. Regardless of rate-based or window-based, a regulation algorithm has **"rate-oriented fairness"** (respectively **"window-oriented**

fairness") if the throughput (respectively the outstanding data amount) of each session governed by such the algorithm at the statistical equilibrium depends *solely* on packet loss probability and *not* on the round trip time. Their work also shows that when either providing *rate-oriented fairness* through a *window-based* regulation algorithm, or providing *window-oriented fairness* through a *rate-based* regulation algorithm, the regulation algorithm must depend on the round trip time. **TCP can be described as having *window-oriented fairness***. Thus, to make any rate-based regulation algorithm TCP-compatible, the rate-based regulation algorithm must depend on the round trip time. On the other hand, one would not need the round trip time if one adopts a different fairness orientation from that of TCP *or* uses a window-based regulation instead. The rest of [GS99] describes a window-based regulation approach for multicast traffic along with a hierarchical approach to **aggregate receivers' congestion control feedback** in a multicast tree and an algorithm to estimate round trip times in an environment where sender and receivers' clocks are synchronized.

Bhattacharyya et al [BTK99] addressed the **"loss path multiplicity problem" (LPM)** in an end-to-end multicast congestion control. LPM results from the fact that a packet can be lost at many different points in the network. If for every loss, each receiver experiencing loss transmits a "loss indication[2]" (LI) back to the source and the source transmission rate is reduced every time upon receiving the LI, the multicast traffic rate will soon suffer from a **"drop-to-zero"** problem. As an example, if there are n independent receivers; each can experience a packet loss with probability p. The probability that a transmitted packet is lost

[2] Unlike NACK, LI's is used primarily for rate adjustment or congestion control – the sender may or may not retransmit the packet upon receiving a LI.

somewhere (and thus a LI is transmitted) is $1 - (1 - p)^n$. When n gets very large, this probability approaches 1. To avoid LPM, [BTK99] suggests the use of a "loss indication filter" (LIF), which allows only a (tentatively smaller) number of LIs to be passed to the rate adjustment algorithm as a "congestion signal" (CS). The LIF can be implemented in many forms. The simplest version of LIF is a "pass-all" LIF which passes every received LI as a CS. A more useful "pass-K-of-N" LIF passes only the LI's from K receivers chosen as representatives out of the total of N receivers (K $\leq$ N). The most stringent version is a "pass-worst" LIF where only the LI's from the lossiest receiver is allowed to pass as a CS. When LIF is used with a form of additive-increase-multiplicative-decrease (AIMD) algorithm, it forms a class of "Filtered Loss Indication-based Congestion Avoidance" (FLICA) algorithms. Assuming the use of FLICA, and a modified star network topology (i.e. the source is connected to one of the branches, and each of the rest branches connects to a receiver), their work shows that the "pass-all" LIF results in a quick throughput degradation as the number of receivers increases and thus renders the approach unscalable. The "pass-K-of-N" LIF shows significant bandwidth improvement over the "pass-all" LIF. They also show that, when every session (either multicast or unicast) uses FLICA, if the multicast session uses the "pass-worst" LIF, every session will automatically achieve a (single-rate) **max-min fair share** [BG92][3] of the network bandwidth. To achieve max-min

[3] An intuitive definition of the **max-min fairness** in a unicast scenario is that if a receiver, say x, receives at a rate above its max-min fair rate and all the receivers' reception rates still yield a feasible bandwidth allocation (i.e. not overutilizing any network link), then there must be some other receiver y: 1) whose max-min fair rate is no larger than that of x, and 2) the reception rate of y has to be reduced to accommodate the increase of x's reception rate. In max-min fairness, if the receiver capacity is not the limit, it can be shown that there exists at least one fully-utilized

fairness, it is important that the multicast session keep track of its 'worst' receiver; this requires the multicast source to monitor all of its end-to-end paths.

Rubenstein et al [RKT99] studied the **fairness properties in multi-rate multicast** scenarios (i.e. when receivers can receive at different rates). Unlike unicast scenarios, in multicast scenarios there are two different perspectives of fairness: **receiver perspective** which states that a receiver should receive at a largest possible rate that does not *"steal"* bandwidth from slower receivers, and **session perspective** which states that each session that share a link should be able to get a 'fair' share of the link's capacity. These two fairness perspectives can contradict each other. In **single-rate multicast max-min fairness**: 1) each and every receiver in a session receives at the same rate, 2) the notion of 'fairness' is applied to session rates, and 3) a session's bandwidth share is the same on all of the links utilized by the session. In **multi-rate multicast max-min fairness**: 1) receivers in a session may receive at different rates, 2) the goal is shifted towards finding "fair" receiving rates, and 3) the bandwidth used by a session on a certain link is the maximum bandwidth used on downstream link (i.e. similar to a layered protocol). Their work identified four additional fairness properties that a single-rate max-min fair allocation policy might fail to satisfy, but a **multi-rate max-min fair** allocation policy can satisfy:

- **"Same-Path-Receiver-Fairness"**: If two or more *receivers* share an *identical path* from their respective sources, the reception rates of these receivers should be the same.

- **"Fully-Utilized-Receiver-Fairness"**: On some (fully-utilized) links, a ses-

network link. The rest of the max-min fairness issues deal with the bandwidth allocation on such fully-utilized link(s).

sion's *receiver* should not any receive worse bandwidth share than another session's receiver who shares those links. Each and every session should have all of its receivers satisfying such condition.

- **"Per-Session-Link-Fairness"**: A *session* should not receive any worse bandwidth share than the other sessions on some (fully-utilized) links over *some* branches of the session's multicast tree.

- **"Per-Receiver-Link-Fairness"**: A *session* should not receive any worse bandwidth share than the other sessions on some (fully-utilized) links over *all* branches of the session's multicast tree.

Their work shows that the multi-rate max-min fairness can be realized by a layered (i.e. multiple-channel) multicast approach. In practice, the number of available layers may be limited and may not match the multi-rate max-min fair bandwidth. To get a multi-rate max-min fair rate *on average*, a receiver might need to periodically leave/join certain layers. This also leads to another problem of redundancy, because when receivers uncoordinately leave or join a layer, the link usage will increase and that can affect fairness property. A *coordinated* leave/join can alleviate such a problem.

Based on the notion of **receiver satisfaction**, Legout et al [LNB99] proposed that the bandwidth share of a (unicast or multicast) flow passing through a network link should be some function of 1) the number of flow's receivers downstream from that link, and 2) the overall number of receivers of every flow passing through the link. Assuming no packet loss, R receivers, and one-to-R transmission, the **bandwidth cost** of a delivery scheme (either unicast or multicast) is the sum across every link of the number of packets crossing each link in order for the delivery scheme to get one sender packet to all the R receivers. For example, suppose there are R receivers as the R leaves of a full m-ary multicast tree (i.e. $R = m^d$ where $d > 0$ is the depth of the tree) where a sender is located at the tree root. The unicast bandwidth cost would be $R \cdot \log_m(R)$. That is, for R receivers to receive a particular packet, the source must transmit R times and each time the packet passes through $d = \log_m(R)$ links. The multicast bandwidth cost would be $m + m^2 + \ldots + m^d = \frac{m}{m-1}(R-1)$. That is, a packet is duplicated to m copies each time it passes a node in the tree. The **multicast gain** is the ratio of the unicast bandwidth cost to the multicast bandwidth cost. In the full m-ary tree case, this gain is: $\log_m(R) \cdot \frac{R}{R-1} \cdot \frac{m-1}{m}$. Three bandwidth allocation methods were studied in [LNB99]:

- **"Receiver Independent" (RI) Allocation**: Every flow passing through a link receives equal share of the link's available bandwidth, regardless of the number of the flow's downstream receivers.

- **"Linear Receiver Dependent" (LinRD) Allocation**: The fraction of link bandwidth for a flow passing through a link is the ratio of the number of flow's downstream receivers to the total number of receivers in every flow sharing the link. The LinRD method would allocate, for a multicast flow of R receivers, the bandwidth corresponding to the total of R separate *unicast* flows.

- **"Logarithmic Receiver Dependent" (LogRD) Allocation**: The fraction of link bandwidth for a flow f passing through a link l is:

$$\frac{1 + \ln R(f,l)}{\sum_{\hat{f} \in F_l}(1 + \ln R(\hat{f},l))} \tag{2.1}$$

> where $R(f, l)$ is the number of receivers that both belong to flow f and are located downstream from link l, F_l is the set of all active flows crossing link l. The LogRD allocation simply rewards multicast receivers by a would-be amount of *multicast gain* described earlier.

Their results show that the LinRD allocation tends to oversatisfy multicast flows (i.e. giving them too much bandwidth) while likely starves unicast flows on a link from which there many multicast receivers downstream. The RI allocation, on the other hand, undersatisfies multicast flow with a large number of receivers, but is generally fair to unicast flow. The LogRD allocation seems to provide the best trade off between multicast receiver satisfaction and fairness to unicast flows. The main reason is that the LogRD's multicast traffic fraction (2.1) does not increase too fast when $R(f, l)$ increases.

Our multicast rate control approach does not adjust the rate every time that a packet is lost but rather adjusts the length of each transmission rate based on the long-term packet survival statistics. We are thus shielded from the LPM or 'drop-to-zero' problem. Our multicast rate control approach does not explicitly deal with the fairness issue. Obviously, since our approach does not rely on receiver round trip times, it will not achieve window-oriented fairness with TCP, as defined in [GS99]. Our approach is intended to work in a longer time scale (i.e. many round trip times when compared to TCP) which is not fully compatible with the existing TCP fairness notion. However, in Chapter 6 we propose a way for our multicast rate control approach to be able to coexist with TCP, and especially not to starve TCP flows or any other background traffic that may run within the terrestrial network. To that goal, we will need some support from the network. Related work on network support for multicast is described in the next subsection.

2.3.2 Network Support for Multicast

Grossglauser et al [GB00] investigated the tradeoff between **application complexity**, **network complexity**, and **network efficiency** as the result of different **service models**[4]. Traditional Internet provides end-to-end best-effort unicast service model which pushes complexity towards application while having simple network architecture (e.g. stateless FIFO queueing). When adopting a multicast/multilayer service model, the problems of channel estimation, error control for application become significantly more difficult than the unicast counterpart. The current Internet architecture already requires that some *per-flow state* (e.g. group information) be kept at the routers for multicast routing protocols. If the network becomes more aware of resource dependency and provides a high degree of predictability (e.g. in CBR), the tasks of multicast application such as error control can become easier (e.g. packets are rarely lost if sent below the allocated CBR rate). In terms of network efficiency, the use of FEC redundancy can hide certain resource waste, if the channel loss probability is *fixed*. However, if all the applications assume that their packet generation rates do not affect the channel loss probability, and tend to use higher transmission rates, then the ”goodput” of the network will be in fact negatively affected. Their work shows that by pushing some complexity back into the network (i.e. to provide some guaranteed services instead of the best-effort service) when adopting multicast, it can be more cost effective than the unicast case as well as help reducing application complexity and increasing network efficiency.

Lehman et al [LGT98] proposed three possible router enhancements in an

[4] The service models can range from no-guarantee service (best effort), to partially guaranteed service (Diffserv, variable bit rate VBR), to fully guaranteed service (constant bit rate CBR).

approach called **Active Reliable Multicast (ARM)**:

- **Data caching for local retransmission**. This allows the router to cache incoming data packets for retransmission. When a router receives a NACK, it can immediately responds to the request and thus cuts down the delay.
- **NACK aggregation/suppression**. When there are two or more NACKs for the same packet (which is not in the router's cache) coming in from downstream router links, the router may drop the duplicate NACKs (or 'fuse' them in some other way) before sending a single NACK upstream.
- **Partial and Scoped multicast retransmission**. When a router transmits a repair, the router scopes the transmission to only those the receivers who requested the repair (i.e. by recording the branches that NACKs came in).

Their approach has three significant performance impacts. First, the average recovery delay is reduced (can be less than one round trip time) when the caching is used. Second, NACK implosion is well controlled. Third, the bandwidth requirement is reduced when repair scoping is use. In fact, all of these impacts can be seen when only a fraction of routers are strategically-placed ARM routers. However, Cain et al [CST99] argues that the buffer mechanism required to help local repairs in [LGT98] would be too complicated to implement in today's high-speed routers. Some other scoping technique can be implemented without any router modification, e.g. setting the packet's 'time-to-live' (TTL) to limit the number of hops that multicast traffic would travel.

Luby et al [LVS99], preliminarily proposed a network-assisted multicast scheme where multicast source transmits at some fixed rate (chosen from the maximum rate a receiver can receive) and let the routers filter out excessive multicast traffic from congested links.

Even with a single-channel approach, routers can play an active important role in choosing which fraction of the multicast traffic to be dropped or forwarded based on current link load and *fair* share. **Fair queueing algorithms** can be deployed at the routers to ensure a certain degree of *local* 'fair' share of bandwidth over a specific network link. This notion of local fairness is obviously different from that of the max-min fairness. There have been some studies of how TCP traffic interacts with some fair queueing algorithms (e.g. [HMMM00]). **In Chapter 6 (also in [TC01]), we study how our long-term multicast rate control system can potentially co-exist with terrestrial traffic through the use of fair queueing algorithms**.

2.4 Satellite Multicast

Jung et al [JNB99] analyzed the bandwidth requirements in a number of satellite multicast scenarios, with and without the use of a feedback channel. They came to an obvious conclusion that the feedback channel is required to either guarantee reliability or be more bandwidth efficient. However, they did not consider the impact of their protocols on receivers' reception latency while we do.

Payne [Pay99] compared delay and bandwidth requirements in two major scenarios of satellite multicast: **unconnected clusters scenario** and **connected clusters scenario**. The delay is defined in terms of the multiple of transmission rounds needed to transmit a data block reliably to receivers. The major difference between the two scenarios above is the connectivity *among* clusters. In the unconnected clusters scenario, error recovery is limited to each local cluster, while in the connected clusters scenario, error recovery can be collaborated among neighboring clusters. Payne assumed that FEC is used, that losses are homogeneous and that the error recovery protocol works on a data block basis. In the unconnected clusters scenario, multicast transmission is done in two stages. The first stage involves reliable delivery of a data block from the source to a set of **"privileged receivers" (PR)**. The second stage is a reliable delivery of a data block from each PR to its terrestrial cluster which starts only *after* the PR completely received the data block. The delay and bandwidth analysis technique for the two stages are the same (albeit with a different number of 'receivers'). In the connected clusters scenario, when PR detects a loss, it will first start a local recovery cycle in which the PR requests repair packet(s) from neighboring clusters and starts a local recovery timer. If the timer expires and the recovery is not successful, the PR will then start a global recovery cycle in which the PR transmits a global NACK via the satellite to the source and starts a global recovery timer. In fact, the PR can also try multiple local recovery cycles before starting a global recovery cycle. If the global recovery cycle is not successful, the PR will then repeat the local and global recovery cycles so on and so forth, until the PR receives a complete data block. Payne's results show that proactive parity transmission benefits the unconnected clusters scenario by reducing the expected number of transmission rounds needed to reliably deliver a data block, and that with a channel estimation technique (i.e. estimating the maximal channel loss probability) one may estimate the amount of the proactive parity packets. In the connected clusters scenario, local recovery (among PRs) can help reduce both the number of satellite rounds and the satellite bandwidth; the reduction in satellite bandwidth was indeed transfered to the local network. Using dynamic proactive parity (with channel estimation) along with having multiple local recovery rounds prior to using a global recovery round shows the lowest number of transmission rounds and the lowest bandwidth requirement.

Our scenario would be similar to Payne's unconnected clusters scenario. However, unlike Payne's work, we assume heterogeneous receivers and aim at the construction of the rate control mechanisms that can trade off between multicast bandwidth consumption and reception delay in a heterogeneous receiver population.

Wong et al [WHK00] studied a satellite multicast problem for *"periodic"* information. They proposed a **"Tunable quasi-reliable multicast protocol (TUNA)**". The periodic information is updated after some regular interval. The notion of *quasi*-reliability is that only the *newest* version of the information is of interest to the receivers. Old information can get expired and therefore is no longer demanded. TUNA protocol takes the frequency of information updates into account and decides if a receiver should either wait for the next update or send NACKs to retrieve the current data. TUNA is suitable for frequently changing information such as the most recent stock quote (i.e. without history). There is no benefit when applying TUNA to relatively 'static' information. Their notion of reliability and approach are fundamentally different from ours.

Clerget et al [CD98] studied a data partitioning technique for layered multicast in satellite. However, the cost of having multiple satellite channels can be high, and these multiple-channel techniques might not be cost-effective. Unlike their approach, we shall assume a non-layered (single-channel) multicast.

Chapter 3

A Hybrid Satellite-Terrestrial Internet Channel

In this chapter, we present the results from a set of real hybrid satellite-terrestrial internet experiments which we conducted to analyze the channel characteristics. The results shown here will be used again in Chapter 4 to establish an approximation model for the multicast rate control to work on.

We use a commercially available Hughes Network's two-way DirecPC(tm) Direct Broadcast Satellite (DBS) system. A two-way[1] DirecPC can be conceptually illustrated in Figure 3.1 and consists of the following key components:

1. A geo-stationary satellite which orbits the earth in a unique circular geo-stationary orbit at an altitude of approximately 35,786 kilometers above the equator [Rod01]. In this particular geo-stationary orbit, the satellite will appear relatively stationary with respect to the surface of the earth[2] Such a satellite can thus provide a communication channel.

[1] Sometimes it is refered to as a *satellite-return* system.

[2] There is another term *geo-synchronous orbit* which simply means that the orbital period of the satellite and the rotation period of the earth are synchronized. However, the geo-synchronous orbit needs not necessarily be geo-stationary or circular.

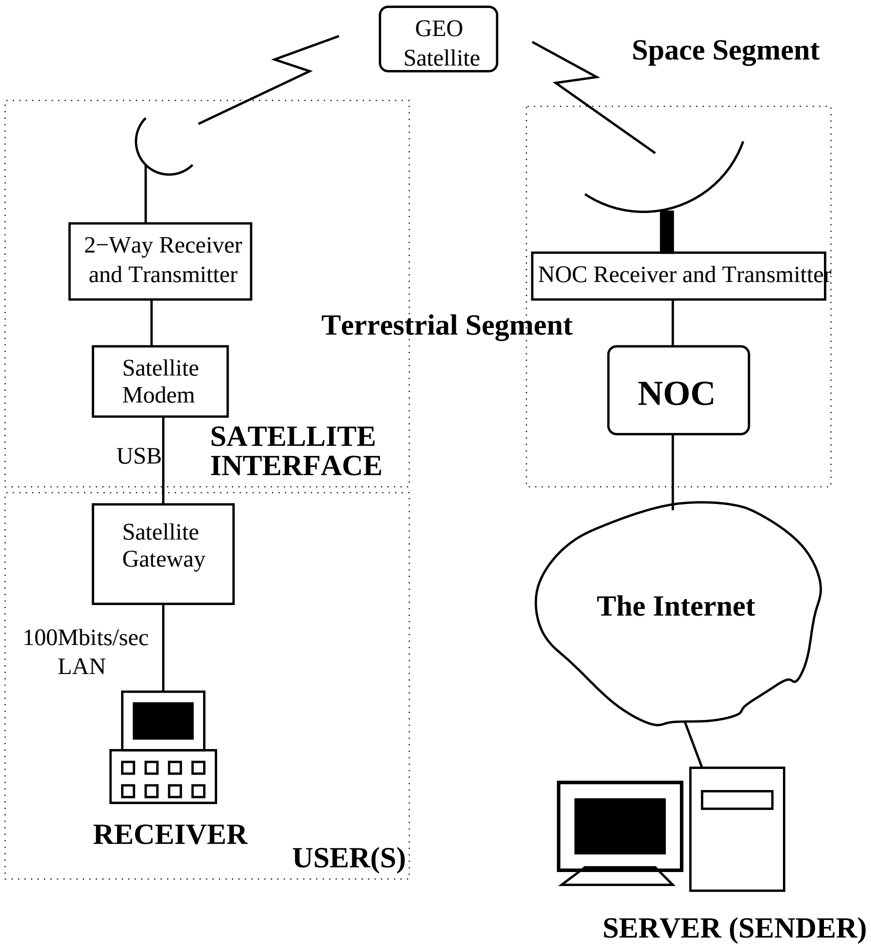


Figure 3.1: A hybrid satellite-terrestrial (HST) internet system

2. An earth-based satellite interface unit which consists of a two-way satellite receiver-transmitter and a satellite modem. The satellite interface unit respectively encodes and decodes data symbols to be transmitted and received via the satellite channel (in an analogous way to what a telephony modem does for a terrestrial telephone line).

3. A network operation center (NOC) which relays requests from a user to a server. The NOC first receives requests from the user via the satellite chan-

nel, then establishes corresponding terrestrial internet requests to a terrestrial internet server site. Once the server's responds arrive, the NOC relays the responds back to the user also via the satellite channel. The NOC may implement some satellite access policies, such as specifying those who can utilize more satellite bandwidth than the others, and at what time of day.

We use a topology that also appears in Figure 3.1 for our experiments. We summarize the setup of our experiments as follows:

On user's side, the satellite interface unit is connected to a satellite gateway. In our experiments, our satellite gateway is a desktop computer with an Intel Pentium II(tm) 300Mhz processor and 64 MB main memory, running the Microsoft Windows 2000 Professional(tm) operating system and Internet Connection Sharing (ICS). Our receiver is a notebook computer with an AMD K6-2(tm) 475Mhz processor and 128MB memory. The receiver is connected to the satellite gateway via a 100 Mbits/second Ethernet local area network (LAN). Our server, which acts as a sender of probe traffic, is a workstation with a SUN UltraSparc(tm) processor and 128MB main memory, running the Solaris(tm) operating system. We use the SUN workstation as the sender because it handles timer expirations more accurately than a Linux-based workstation counterpart.

Because the satellite channel is a broadcast medium in nature, we may look specifically into only one end-to-end characteristic (i.e. unicast or sender-to-receiver) . To setup a one-way, sender-to-receiver (i.e. forward only) satellite multicast scenario, multiple satellite interface units can theoretically be programmed to filter and receive packets from a single specific address.

There are two objectives in our experiments. First, we want to analyze the end-to-end packet survival characteristics of this commercially available hybrid satellite-terrestrial internet channel when we transmit a large number of User Datagram Protocol (UDP) probe packets through the channel at some constant rate $r \in \mathcal{R}$ (where $\mathcal{R}$ is a finite, discrete set of the sender's transmission rates). Second, we want to establish a simple approximation model that the multicast rate control can rely upon. In contrast with the previous work of Yajnik et. al [YMKT99] who studied the temporal dependence of packet losses in a terrestrial-only internet, we study a similar problem in the context of the hybrid satellite-terrestrial internet with an explicit emphasis towards a rate control problem for satellite multicast. We cover the first objective in the remaining of this chapter and the second objective in the next.

In our experiments, each sender's packet is 1024 bytes long. Each byte consists of 8 bits. The first 43 bytes contained our transmission protocol specific data. The rest of the bytes are payload and filled with uniformly and randomly generated data to maximize the entropy of the payload. **From Information theory, this uniformly and randomly filled data will ensure that no compression algorithm can effectively reduce the size of the packet payload**. In computing the time between consecutive packet transmissions, each packet is assumed to additionally have a 20-byte IPv4 header and a 8-byte UDP header. For example, if the sender's transmission rate r is 512Kbits/second, the time between any two consecutive sender's packets will be $\frac{(1024+20+8)\times 8}{r} \approx 0.01605$ seconds.

The receiver logs the information regarding all the packets that the receiver has received. In terms of notations, let each $I(i)$ be a binary (0,1) random variable which represents the reception indicator of the i-th packet, $i = 0, 1, \ldots, N-1$, that has been transmitted by the sender. If $I(i) = 0$, it indicates that the i-th transmitted packet is lost, whereas $I(i) = 1$ indicates that the i-th packet survives and is received by the receiver. The sequence of $\{I(0), I(1), I(2), \ldots\}$ forms a **time series** which we shall represent concisely as $\{I(i)\}$ or sometimes as $\{I(i); i = 0, 1, \ldots\}$ when we need to emphasize the range of the indices more explicitly. Note that this time series needs not be stationary in the wide sense (i.e. wide sense stationarity means that every $I(i)$ in the time series has the same constant mean and the same constant variance). Also any two distinct outcomes $I(i_1)$ and $I(i_2)$ where $i_1 \neq i_2$ need not be independent of each other. We also consider out-of-order packets to be received successfully because these out-of-order packets may still contain useful information and can be handled properly by a transport protocol. The receiver notes the reception of each packet and subsequent duplicates, if any, are discarded.

3.1 Throughput Dynamics

Here we explore one aspect of the time series $\{I(i)\}$. We look at the throughput dynamics of the hybrid satellite-terrestrial internet channel. In doing so, we can identify portions of the time series $\{I(i)\}$ that are stationary (or almost stationary) in the mean, and where we can apply further analysis.

In general, we use the following set of sender's transmission rates: $\mathcal{R} = \{512, 256, 128, 64\}$ Kbits/second. Occasionally, we also explore a different set of rates : $\mathcal{R}' = \{400, 200, 100, 50\}$ Kbits/second.

The configurations of our experiments are summarized in Table 3.1. Different sender's transmission rates are used and for each rate, we record the reception outcomes $I(i)$. Plots are made in such a way that each point in the plot

Date	Network Configs.	Sender's Transmission Rates (Kbits/sec)	Results in Appendix Figure
Jul 27, 2001	HST	512,256,128,64	A.1
Jul 27, 2001	HST	400,200,100,50	A.2
Jul 30, 2001	HST	512,256,128,64	A.3
Jul 30, 2001	HST	400,200,100,50	A.4
Aug 02, 2001	HST	512,256,128,64	A.5
Aug 02, 2001	HST	400,200,100,50	A.6
Aug 06, 2001	HST	512,256,128,64	A.7
Aug 08, 2001	HST	512,256,128,64	A.8
Aug 09, 2001	HST	512,256,128,64	A.9
Aug 10, 2001	HST	512,256,128	A.10
Aug 17, 2001	HST	512,256	A.11
Aug 17, 2001	HST	512,256,128,64	A.12
Aug 22, 2001	HST	512,256,128,64	A.13
Aug 27, 2001	HST	512	A.14
Aug 19, 2001	DSL	512,256	A.15
Aug 23, 2001	CAN	512	A.16

Table 3.1: Experiment configurations for throughput dynamics (Network Configurations: HST=Hybrid Satellite-Terrestrial internet Network, DSL=Digital Subscriber Line,and CAN=Campus Area Network)

represents the percentage of packet survivals in a 10-second non-overlapping time interval (this can also be interpreted as a 10-second, instantaneous, non-overlapping, normalized throughput). We present the throughput dynamics results in Appendix A.

We also have configurations which use only the terrestrial internet network for comparisons. From Table 3.1, in the two configurations of August 19, 2001 and August 23, 2001, we respectively connected the sender to the receiver via a commercial Digital Subscriber Line (DSL) Internet service provider and the University of Maryland's campus local area network (i.e. the Ethernet network inside the Institute for Systems Research, A.V. Williams building). In either case, the sender is the same workstation used in all other experiments.

In Table 3.2, we report the weather conditions as they happened during our experiments. Past experiences have indicated that a severe storm can disrupt the satellite signal and temporarily bring down the satellite data link (e.g. on August 10, 2001). But as long as the satellite signal strength, as seen from the satellite diagnostics utility, is at least 31 out of 100, then the satellite data link can be established.

We have the following observations. There are **two distinct patterns of packet losses in the hybrid satellite-terrestrial internet channel**. The first packet loss pattern appears to result from a natural phenomenon (e.g. cosmic or thermal noises in the satellite system) or from a terrestrial network congestion. This, in turns, makes the packet losses in the hybrid satellite-terrestrial internet channel appear bursty. We summarize where readers can clearly observe this type of packet loss pattern in Table 3.3 below. Note that this packet loss pattern also appears in *all other* hybrid satellite-terrestrial internet results, but it appears most distinctly in these figures.

The second packet loss pattern likely results from an explicit intervention of the NOC. The NOC may implement a fair access policy to the satellite channel.

Date	Time	Weather Condition
Jul 27, 2001	14:31pm	Mostly Sunny, 77F, UVindex 8, Relative Humidity 47%
Jul 30, 2001	13:55pm	Mostly Cloudy, 75F, Relative Humidity 73%
Aug 02, 2001	13:55pm	Mostly Sunny, 81F, UVindex 8, Relative Humidity 62%
Aug 06, 2001	15:45pm	Cloudy Hazy
Aug 08, 2001	18:08pm	Partly Cloudy, 94F, Relative Humidity 49%
Aug 09, 2001	13:45pm	Hazy, 93F, UVindex 8, Relative Humidity 56%
Aug 09, 2001	15:51pm	Partly Cloudy, 95F, UVindex 5, Relative Humidity 56%
Aug 10, 2001	17:00pm	Raining, Heavy at times
Aug 17, 2001	12:51pm	Hazy, 79F
Aug 22, 2001	15:30pm	Mostly Sunny, 86F, UVindex 6, Relative Humidity 38%
Aug 27, 2001	15:51pm	Mostly cloudy, 83F, UVindex 2, Relative Humidity 69%

Table 3.2: Weather conditions during our experiments

Occasionally we see that our UDP traffic stream has a significant shift of its packet survival mean. Before the shift, the UDP traffic survival rate appears stabilized at one particular value, and once the shift occurs, the UDP traffic

Date	Appendix Figure(s)
Jul 27, 2001	A.1
Jul 30, 2001	A.3, A.4
Aug 02, 2001	A.5
Aug 06, 2001	A.7

Table 3.3: Examples of the first type of the packet loss pattern

survival rate then becomes re-stabilized at another particular value. We notice that this type of shift often occurs after the sender have consistently transmitted data packets for some time. Readers can use Table 3.4 below to find results which distinctively display the packet loss pattern of the second type.

Date	Appendix Figure	Notes
Jul 27, 2001	A.2	at 840, 1580 seconds of 400 Kbits/s
Aug 02, 2001	A.6	at 580, 1000 seconds of 400 Kbits/s at 800 seconds of 200 Kbits/s
Aug 08, 2001	A.8	at 600 seconds of 512 Kbits/s
Aug 09, 2001	A.9	at 600 seconds of 512 Kbits/s
Aug 17, 2001	A.11	at 300 seconds of 256 Kbits/s
Aug 22, 2001	A.13	at 550 seconds of 512 Kbits/s

Table 3.4: Examples of the second type of the packet loss pattern

We observe that the second type of packet losses occurs very infrequently and once it stabilizes, it appears as if the system simply makes a transition to operate at a new different stationary point. Because all the stationary portions last long enough to be useful, such as for transmitting a large bulk file (i.e. up to hundreds of megabytes), **we shall focus primarily on the stationary, end-to-end characteristics of the hybrid satellite-terrestrial internet channel**.

Also we observe that a higher transmission rate (e.g. at 512 Kbits/s, or 256 Kbits/s) tends to result in more out-of-order packets. When a *data link layer* utilizes the *link-level forward error correction* (called *link-level FEC*), the out-of-order packets may indicate a re-construction of lost or corrupted packets at the data link layer. However, there are other possible explanations, and we are not concerned with the issues specific to the data link layer in our work.

3.2 Time Series Analyses

We present a short overview of time series analysis and then analyze the portions of the reception history that appear to be stationary or almost- stationary in the mean. We use only well-known analysis methods, such as the ones found in [BD91].

When we have a discrete-time, stationary, time series $\{X(i); i = 0, 1, \ldots\}$, the **autocorrelation function** of the time series $\{X(i)\}$ is defined at time lag k, as:

$$\gamma_X(k) = \frac{E[(X(i) - \bar{X})(X(i+k) - \bar{X})]}{\sigma_X^2}, \quad k = 0, 1, 2, \ldots \tag{3.1}$$

where the two constants: $\bar{X}$ and σ_X^2 are respectively the mean and the variance of the stationary time series $\{X(i)\}$. The correlation function (3.1) has the following properties:

- $-1 \leq \gamma_X(k) \leq 1$. The magnitude $|\gamma_X(k)|$ intuitively represents the strength of correlation or how linearly predictable $X(i+k)$ is in terms of $X(i)$ or vise versa.
- $\gamma_X(0) = 1$. The self-correlation is always the strongest.
- If at some lag $k \neq 0$ we have $\gamma_X(k) = 0$, then $X(i)$ and $X(i+k)$ are **uncorrelated**. Note that $X(i)$ and $X(i+k)$ may or may not be independent.

It is practical to derive a good estimate of $\gamma_X(k)$ from experiment samples. Let $\{x(i); i = 0, 1, \ldots, N-1\}$ be a finite realization of a discrete-time, stationary, time series $\{X(i); i = 0, 1, \ldots\}$, then the **sample autocorrelation function** $\hat{\gamma}_X(k)$ at time lag k is defined as:

$$\hat{\gamma}_X(k) = \frac{\sum_{j=0}^{N-k-1} (x(j) - \bar{x})(x(j+k) - \bar{x})}{\sum_{j=0}^{N-1} (x(j) - \bar{x})^2}, \quad k = 0, 1, 2, \ldots, N-1 \tag{3.2}$$

where $\bar{x} = \frac{\sum_{j=0}^{N-1} x(j)}{N}$ is the sample mean of $\{x(i)\}$. Note that $\hat{\gamma}_X(k)$ has similar properties to $\gamma_X(k)$ as we already described. In practice, the sample autocorrelation function $\hat{\gamma}_X(k)$ provides reliable estimates of the autocorrelation function $\gamma_X(k)$ for time lag upto $k \approx N/4$. For time lag $k \to N$, there are fewer samples available and thus $\hat{\gamma}_X(k)$ is less accurate.

When we have two time series $\{X(i)\}$ and $\{Y(i)\}$, the **cross-correlation function** $\varrho_{XY}(k)$ between $X(i)$ and $Y(i)$ at time lag $+k$ is defined as:

$$\varrho_{XY}(k) = \frac{E[(X(i) - \bar{X})(Y(i+k) - \bar{Y})]}{\sqrt{\sigma_X^2 \sigma_Y^2}}, \quad k = 0, 1, 2, \ldots \tag{3.3}$$

and at time lag $-k$ is defined as:

$$\varrho_{XY}(-k) = \frac{E[(X(i) - \bar{X})(Y(i-k) - \bar{Y})]}{\sqrt{\sigma_X^2 \sigma_Y^2}}, \quad -k = 0, -1, -2, \ldots \tag{3.4}$$

The cross-correlation function has the following properties:

- In general $\varrho_{XY}(k) \neq \varrho_{YX}(k)$ (but not always). However, $\varrho_{XY}(k) = \varrho_{YX}(-k)$. Therefore, we require only one of the representations.

- Unlike $\gamma_X(0)$, $\varrho_{XY}(0)$ needs not be 1.

When we respectively have the realization pairs $\{x(i), y(i); i = 0, 1, \ldots, N-1\}$ of the time series pairs $\{X(i), Y(i)\}$, the **sample cross-correlation function** $\hat{\varrho}_{XY}(k)$ provides a good estimate of $\varrho_{XY}(k)$:

$$\hat{\varrho}_{XY}(k) = \begin{cases} \frac{\sum_{i=0}^{N-k-1}(x(i)-\bar{x})(y(i+k)-\bar{y})}{\sqrt{s_x^2 s_y^2}} & k = 0, 1, 2, \ldots \\ \frac{\sum_{i=0}^{N+k-1}(y(i)-\bar{y})(x(i-k)-\bar{x})}{\sqrt{s_x^2 s_y^2}} & k = 0, -1, -2, \ldots \end{cases} \tag{3.5}$$

where

$$\bar{x} = \frac{\sum_{i=0}^{N-1} x(i)}{N}, \qquad \bar{y} = \frac{\sum_{i=0}^{N-1} y(i)}{N}$$

$$s_x^2 = \sum_{i=0}^{N-1}(x(i) - \bar{x})^2, \quad s_y^2 = \sum_{i=0}^{N-1}(y(i) - \bar{y})^2$$

There is an equivalent representation of the binary time series $\{I(i)\}$:

$$\{I(i); i = 0, 1, \ldots\} \equiv \{I(0), A_0(a_0), A_1(a_1); a_0, a_1 = 0, 1, \ldots\} \tag{3.6}$$

where $\{A_0(a_0)\}$ and $\{A_1(a_1)\}$ are the two non-zero positive-integer time series that respectively, and alternatingly, represent the number of contiguous zeros and the number of contiguous ones that occur in $\{I(i)\}$. $I(0)$ is the starting binary value. For example, if $\{I(i); i = 0, 1, \ldots, 7\} = \{1, 1, 1, 0, 1, 1, 0, 1\}$ it is equivalent to having

$$\begin{aligned} \{A_0(a_0); a_0 = 0, 1\} &= \{1, 1\}, \\ \{A_1(a_1); a_1 = 0, 1, 2\} &= \{3, 2, 1\} \end{aligned}$$

Yajnik et. al [YMKT99] used a similar representation to analyze the distributions of $\{A_0(a_0)\}$ and $\{A_1(a_1)\}$.

Now we analyze the data from our experiments. Table 3.5 summarizes the traces that we use for the time series analysis and Table 3.6 summarizes the presentation format of the results that we refer to in Table 3.5. We present one example of the time series analysis results in Figure 3.2 and all of the time series analysis results in Appendix A.

Date	Network Config.	Sender's Transmission Rates (Kbits/second)	Appendix Figure(s)
Aug 02, 2001	HST	512,256,128,64	A.17, A.18, A.19, A.20
Aug 02, 2001	HST	400	A.21
Aug 06, 2001	HST	512,256,128,64	A.23, A.24, A.25, A.26
Aug 08, 2001	HST	512,256,128,64	A.27, A.28, A.29, A.30
Aug 09, 2001	HST	512,256,128,64	A.31, A.32, A.33, A.34
Aug 10, 2001	HST	512,256,128	A.35, A.36, A.37
Aug 17, 2001	HST	512,256,128	A.38, A.39, A.40
Aug 22, 2001	HST	512,256,128,64	A.41, A.42, A.43, A.44
Aug 27, 2001	HST	512	A.45
Aug 19, 2001	DSL	512	A.46
Aug 19, 2001	CAN	512	A.47

Table 3.5: Time series analysis of the experiment data

(Network Configurations: HST=Hybrid Satellite-Terrestrial Internet Network, DSL=Digital Subscriber Line, and CAN=Campus Area Network)

(A) Packet Survival Data: Time Series $\{I(i)\}$ using 10-second, non-overlapping window average.	(B) Sample Auto Correlation Function: $\hat{\gamma}_I(k)$ (using time-based representation).
(C) Length of Contiguous Packet Losses: Time Series $\{A_0(a_0)\}$	(D) Length of Contiguous Packet Survivals: Time Series $\{A_1(a_1)\}$
(E) Sample Auto Correlation Functions: $\hat{\gamma}_{A_0}(k)$ and $\hat{\gamma}_{A_1}(k)$	(F) Sample Cross- Correlation Function: $\hat{\varrho}_{A_0 A_1}(k)$

Table 3.6: Presentation format of the time series analysis of the experiment data

Note: See Figure 3.2 for an example. See Appendix A for all the time series analysis results.

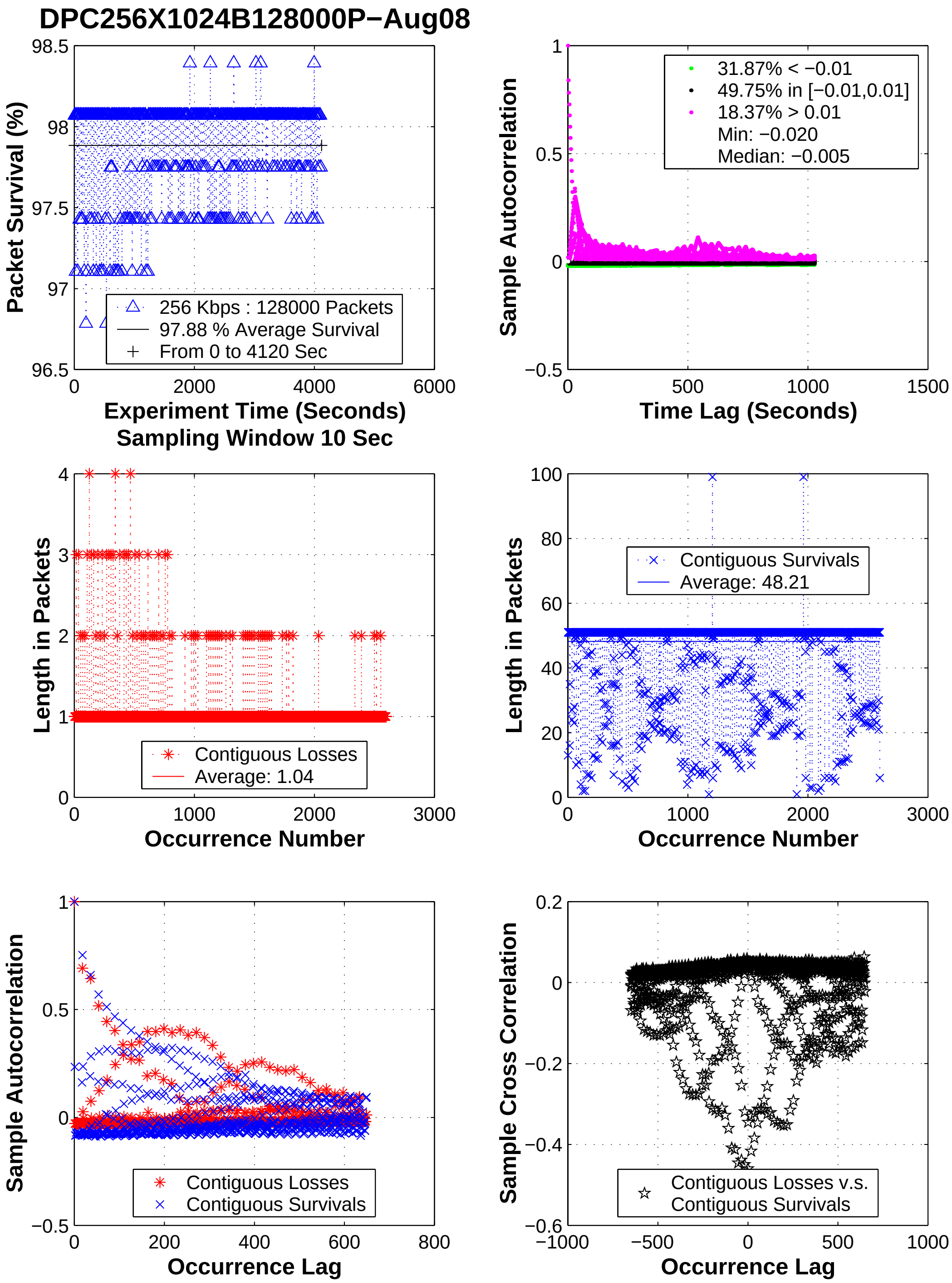


Figure 3.2: Time series analysis: August 08, 2001: HST 256 Kbits/sec transmission rate

(Note: Also listed in the Appendix as Figure A.28)

From Table 3.6 and the results in Appendix A, we may interpret the results as follows:

- It is obvious from the plots of the sample autocorrelation function (B) that most of the realizations of the packet survival time series $\{I(i)\}$ exhibit correlation. Therefore, the time series $\{I(i)\}$ cannot fully consist of independent random variables $I(i)$. The *degrees* of such correlation, however, vary significantly from one experiment to the other. Also note that some of the measured correlation was induced by temporary non-stability in the packet survival process (e.g. in Figure A.17 and Figure A.18 where a large burst of losses appears).

- The plots of the contiguous losses and contiguous survivals, respectively (C) and (D), show the nature of packet losses and survivals from another different perspective. Contiguous packet losses of length ten or more appear very less frequently. In one case, they are associated with *perceivable*[3] severe weather that disrupted our satellite link. We visually observe a larger degree of randomness (*entropy*) from the contiguous packet survival lengths.

- The plots of the sample autocorrelation functions of both the contiguous packet loss lengths and the contiguous packet survival lengths (E) also reveal dependence. The plots of the cross-correlation function (F) also reveal that the contiguous packet loss lengths and the contiguous packet survival lengths are not entirely independent.

[3] Note that we use the word *perceivable* here because we do not have a way of monitoring weather conditions that may happen in upper atmosphere or at far away locations along the data traffic path

3.3 Chapter Summary

In this chapter, we have studied the characteristics of a commercially available hybrid satellite-terrestrial internet channel by using constant-rate, synthetically randomized content UDP traffic. As expected, the packet survivals in the hybrid satellite-terrestrial internet channel exhibit certain degrees of correlation. We have also found that for a significant amount of the time, the channel exhibits stationarity in the mean for packet survivals. However, the sender transmission's rate does not appear to significantly affect the packet survival rates in this portion of the network. This suggests that the actual available bandwidth of the hybrid satellite-terrestrial backbone is significantly higher than the maximum transmission rate that we tested.

In our work we do not establish a detailed probabilistic model that accurately describes the packet survival process in the channel for a few reasons. One, the detailed model would indeed deal with the channel's short-term and transient behaviors and would lead to more complexity as we intend to work on a long-term time scale. Two, the detailed model may not be really necessary because of a reason in which the reader shall see in Chapter 4 where we can effectively work with a much simpler approximation model.

Chapter 4

The Gaussian Approximation Approach

A general question arises from the time series $\{I(i)\}$ which represents the reception outcomes at a specific receiver in the previous chapter. If this time series were to represent the packet survivals in a bulk file transmission, what can we do about it ? In this chapter we investigate such an issue.

4.1 An Introduction to the Gaussian Approximation Approach

Suppose we have a large bulk file to be multicasted through the hybrid satellite-terrestrial internet channel, the use of the RSE-based forward error correcting codes (FEC) mentioned in Section 1.3 has been proven to be useful for multicasting. The large bulk file can be divided into smaller data blocks, say that we have W data blocks and each data block consists of h data packets. If the RSE symbols have β bits, then in each data block, we may have upto $c \geq 0$ parity packets per data block where $h + c < 2^{\beta}$. A choice of $\beta = 8$ is generally preferred because it easily fits into a *byte* boundary on modern computers.

A technique of **interleaved transmission** as illustrated in Figure 4.1 is well-known to reduce the impact of burst (i.e. temporally contiguous) losses experienced by any specific data block. The sender's transmission, shown on top of Figure 4.1, results in the time series $\{I(i)\}$ at a receiver. At the bottom, these packets belong interleavedly to W data blocks, shown respectively in W vertical columns. When an incidence of burst losses occurs, the actual packet losses will be spreaded across many data blocks. In this study, we will look more specifically at the **impact of the interleaved transmission on the structures of the packet survival correlation**.

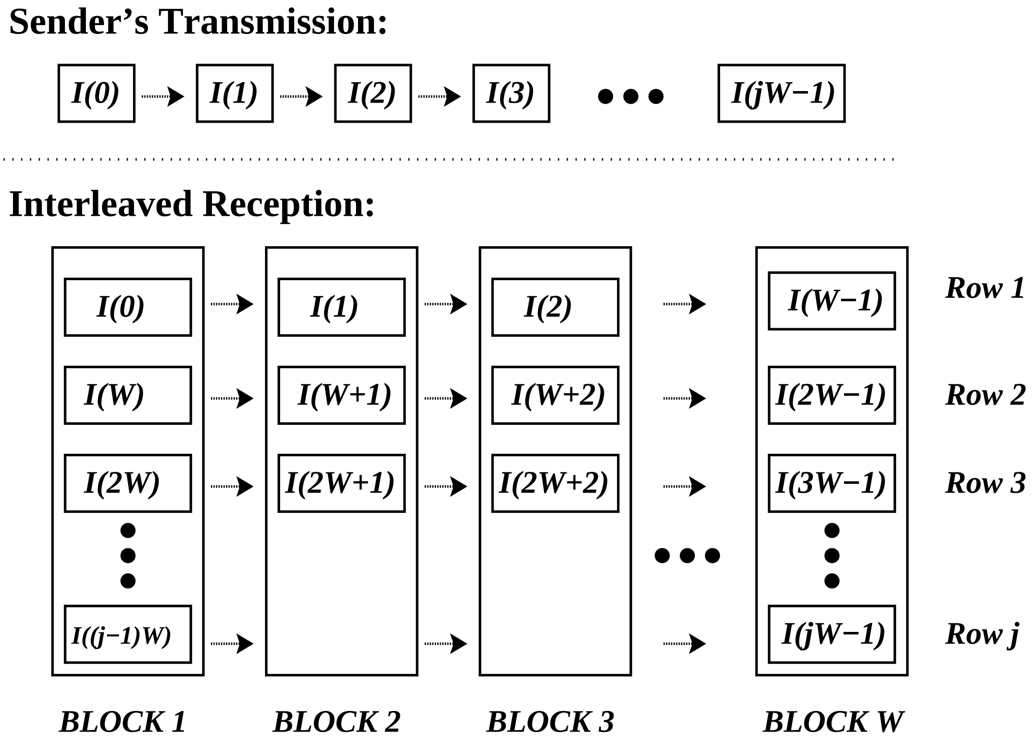


Figure 4.1: Interleaved transmission

We have seen earlier that the time series $\{I(i)\}$ generally does not consist of independent random variables. However, it is easier to work from the theoretical perspective if we use a **hypothetical binary time series** $\{I_{Ber}(i); i = 0, 1, \ldots\}$ where each $I_{Ber}(i)$ is an independently and identically distributed (i.i.d) Bernoulli (0,1) random variable. We need to test the robustness of the approach assuming the time series $\{I_{Ber}(i)\}$ in place of the actual time series $\{I(i)\}$. In doing so, we are concerned only with the portion of the time series $\{I(i)\}$ that is reasonably stationary in the mean.

We start as follows. First, assuming that when the sender transmits packets at some constant rate $r \in \mathcal{R}$, the underlying packet survival model to a specific receiver is governed by the hypothetical time series $\{I_{Ber}(i)\}$. Let $E[I_{Ber}(i)] = q$ be referred to as the **packet survival probability**. We immediately have that the variance $Var[I_{Ber}(i)] = q(1-q)$. The maximum-likelihood, minimum-variance, unbiased estimate $\hat{q}$ of the packet survival probability can be obtained from the sample mean.

Next, when the sender has transmitted upto j rows of packets (i.e. a total of jW packets), and when replacing the time series $\{I(i); i = 0, 1, \ldots, jW-1\}$ of Figure 4.1 by the hypothetical time series $\{I_{Ber}(i); i = 0, 1, \ldots, jW-1\}$, the number of packets received in each data block w (shown in vertical column $w = 1, 2, \ldots, W$) is an independent random variable $X_w(j)$. We know that $X_w(j)$ is *equivalent in distribution* to a random variable $Y(j)$ which has a Binomial distribution:

$$X_w(j) \overset{\mathcal{D}}{=} Y(j), \quad w = 1, 2, \ldots, W \tag{4.1}$$

$$Y(j) \sim Binomial(j, q). \tag{4.2}$$

We shall call $Y(j)$ the **master block**'s reception status.

Using the RSE-based FEC, the receiver can decode and receive the bulk file reliably if for each and every of the W data blocks, there are at least h distinct packets (i.e. when at least h of the binary outcomes in the block have value one). We define the notion of the **bulk file local**[1] **unreliability (BFLU)** after the sender has transmitted upto packet $I_{Ber}(\ell = jW - 1)$ as:

$$\begin{aligned} \upsilon(\ell) &= P[\text{Receiver still cannot reconstruct the bulk file}] \\ &= 1 - P[\bigcap_{w=1}^{W} X_w(j) \geq h] \qquad (4.3) \\ &= 1 - P[Y(j) \geq h]^W \qquad (4.4) \\ &= 1 - (1 - P[Y(j) < h])^W . \qquad (4.5) \end{aligned}$$

From the right hand side of (4.5), we define the notion of the **Master Block Local Unreliability (MBLU)** as:

$$\psi(j) = P[Y(j) < h] \qquad (4.6)$$

It is obvious that we may simply control the MBLU $\psi(j)$ in order to control the BFLU $\upsilon(\ell)$. From (4.5) and for a maximum allowable BFLU called υ_{max} such that $0 < \upsilon_{max} \ll 1$, we may accomplish the following probabilistic reception guarantee bound:

$$\upsilon(\ell) \leq \upsilon_{max}, \qquad (4.7)$$

by requiring

$$\psi(j) \leq 1 - \sqrt[W]{1 - \upsilon_{max}} = \psi_{max} \qquad (4.8)$$

where $\ell = jW - 1$.

Since $\psi(j) = 1$ if $j < h$, we only need to compute $\psi(j)$ when $j = h, h+1, \ldots, h+c$ for some integer $c \geq 0$. For a sufficiently large, but finite value of h, $\psi(j), j = h, h+1, \ldots, h+c$ can be approximated by using a standard

[1] We use the word *local* here to emphasize the association with only one receiver.

Gaussian distribution:

$$\frac{Y(j) - jq}{\sqrt{jq(1-q)}} \xrightarrow{\mathcal{D}} \mathcal{N}(0,1), \quad j \gg 0 \tag{4.9}$$

according to the **Central Limit Theorem** [Ash72]. This leads to our **Gaussian approximation approach** in its simplest form. We shall extend this Gaussian approximation approach in later chapters.

Because there exist some fast algorithms to compute the cumulative distribution function (CDF) of the standard Gaussian distribution $\mathcal{N}(0,1)$ (e.g. in [PR96]) with a high precision, we can effectively solve the following problem. Suppose that we are given:

1. a bulk file size in terms of h and W
2. a maximum allowable BFLU υ_{max}, and
3. an estimate $\hat{q}$ of the packet survival probability.

We need to find a minimum number of FEC parities $c \geq 0$ to be added to each data block so that at the end of sender's transmission we have that $\upsilon(\ell = (h+c)W - 1) \leq \upsilon_{max}$.

The Gaussian approximation approach is based on the independent packet survival assumption. We need to test the robustness of the approach when it is subject to the real time series $\{I(i)\}$. To do so, we setup a set of trace-driven simulation experiments where in most cases we select experiment trace portions that appear reasonably stationary in the mean. In some cases, we also deliberately choose the portions that are not stationary in the mean in order to compare. We further subdivide each specific trace portion into three periods. We initially find the estimate $\hat{q}$ of the packet survival probability from an *estimation* period. Then we use the Gaussian approximation to compute the number of parity packets c needed to compensate packet losses, given $\hat{q}, \upsilon_{max}, h$ and W. We reflect the time required for the Gaussian approximation computation a short *computation* period. Typically, we set the computation period to be much larger than the actually required computation time. Next, we have a *simulated file transmission* period where we assume the arrangement of the interleaved transmission of Figure 4.1.

The three periods of the simulation process is illustrated in Figure 4.2 below. In most cases, the estimation period lasts 120 seconds except for the DSL and campus network configurations that we set the estimation period to 500 seconds. The computation period lasts 60 seconds. The number c of RSE-based FEC parity packets is computed using the Gaussian approximation to satisfy the maximum BFLU $\upsilon_{max} = 0.01$ (the times required to compute c are in fact much less than 10 seconds).

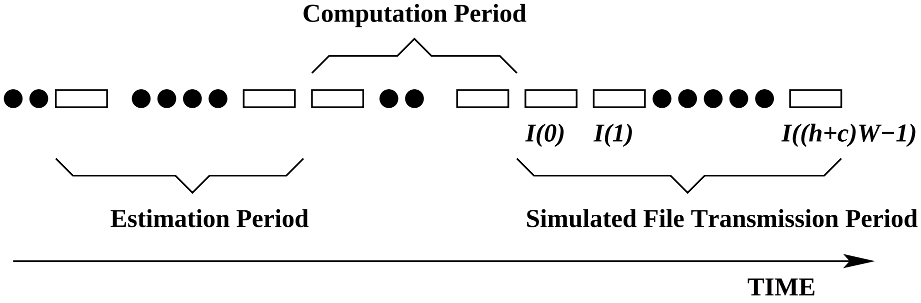


Figure 4.2: The three periods of the trace-driven simulations

We then look at the realization of the reception status $X_w(h+c)$ at all W data blocks. We further analyze the aspects of the realization of $X_w(h+c)$. Table 4.1 summarizes the simulations that we conducted. Table 4.2 summarizes the presentation format of the results we have from the simulations. We present one example of the Gaussian approximation simulation results in Figure 4.3 and all of the Gaussian approximation simulation results in Appendix B.

Date and Network Config.	TX Rate (Kbits/sec)	File Size (MB)	W (Blocks)	h (Packets)	c (Packets)	Appendix Figure
Aug 02, 2001	512	50	256	200	3	B.1
(HST)	256	30	154	200	1	B.2
	128	25	128	200	0	B.3
	64	12	62	200	16	B.4
	400	40	410	100	36	B.5
	100	20	205	100	2	B.6
Aug 06, 2001	512	80	410	200	2	B.7
(HST)	256	50	256	200	3	B.8
	128	25	128	200	2	B.9
	64	12	62	200	3	B.10
Aug 08, 2001	512	150	768	200	14	B.11
(HST)	256	100	512	200	14	B.12
	128	50	256	200	13	B.13
	64	25	128	200	12	B.14

Table 4.1: Trace-driven simulations to verify the Gaussian approximation approach

(Continued on next page)

Table 4.1 continued

Date and Network Config.	TX Rate (Kbits/sec)	File Size (MB)	W (Blocks)	h (Packets)	c (Packets)	Appendix Figure
Aug 09, 2001	512	150	768	200	13	B.15
(HST)	256	100	512	200	13	B.16
	128	50	256	200	13	B.17
	64	25	128	200	12	B.18
Aug 10, 2001	512	50	256	200	13	B.19
(HST)	256	50	256	200	13	B.20
	128	20	103	200	12	B.21
Aug 17, 2001	512	200	1024	200	14	B.22
(HST)	256	50	400	128	109	B.23
	128	20	103	200	3	B.24

(Continued on next page)

Table 4.1 continued

Date and Network Config.	TX Rate (Kbits/sec)	File Size (MB)	W (Blocks)	h (Packets)	c (Packets)	Appendix Figure
Aug 22, 2001 (HST)	512	50	256	200	13	B.25
	256	50	256	200	14	B.26
	128	25	128	200	13	B.27
	64	12	62	200	13	B.28
Aug 27, 2001 (HST)	512	200	1024	200	14	B.29
Aug 19, 2001 (DSL)	512	200	1024	200	10	B.30
Aug 23, 2001 (CAN)	512	200	1024	200	1	B.31

(Network Configurations: HST=Hybrid Satellite-Terrestrial Internet Network, DSL=Digital Subscriber Line, and CAN=Campus Area Network)

(A) The packet survival trace (i.e. the realization of $\{I(i)\}$). Also the visual marks that indicate the durations and the estimates of the packet survival probabilities ($\hat{q}$) corresponding to the *estimation* and *simulated file transmission* periods respectively.	(B) The sample autocorrelation function $\hat{\gamma}_I(k)$ of the packet survival process $\{I(i)\}$ during the *simulated file transmission* period.
(C) The realization of the block reception status $X_w(h+c)$ at each block $w = 1, 2, \ldots, W$ after the sender has transmitted h data packets and c parity packets.	(D) The Gaussian quantile-quantile plot of the realization of the block reception status $X_w(h+c)$ from all W blocks.
(E) The sample autocorrelation function of the block reception status referred to as the **Inter-block reception correlation**: $\hat{\gamma}_{X_w}(k)$.	(F) The sample autocorrelation function of the packet survivals that belong to each specific block w. We refer to this as the **Intra-block reception correlation**.

Table 4.2: Presentation format of the results from the Gaussian approximation simulations

Note: See Figure 4.3 for an example. See Appendix B for all the Gaussian approximation simulation results.

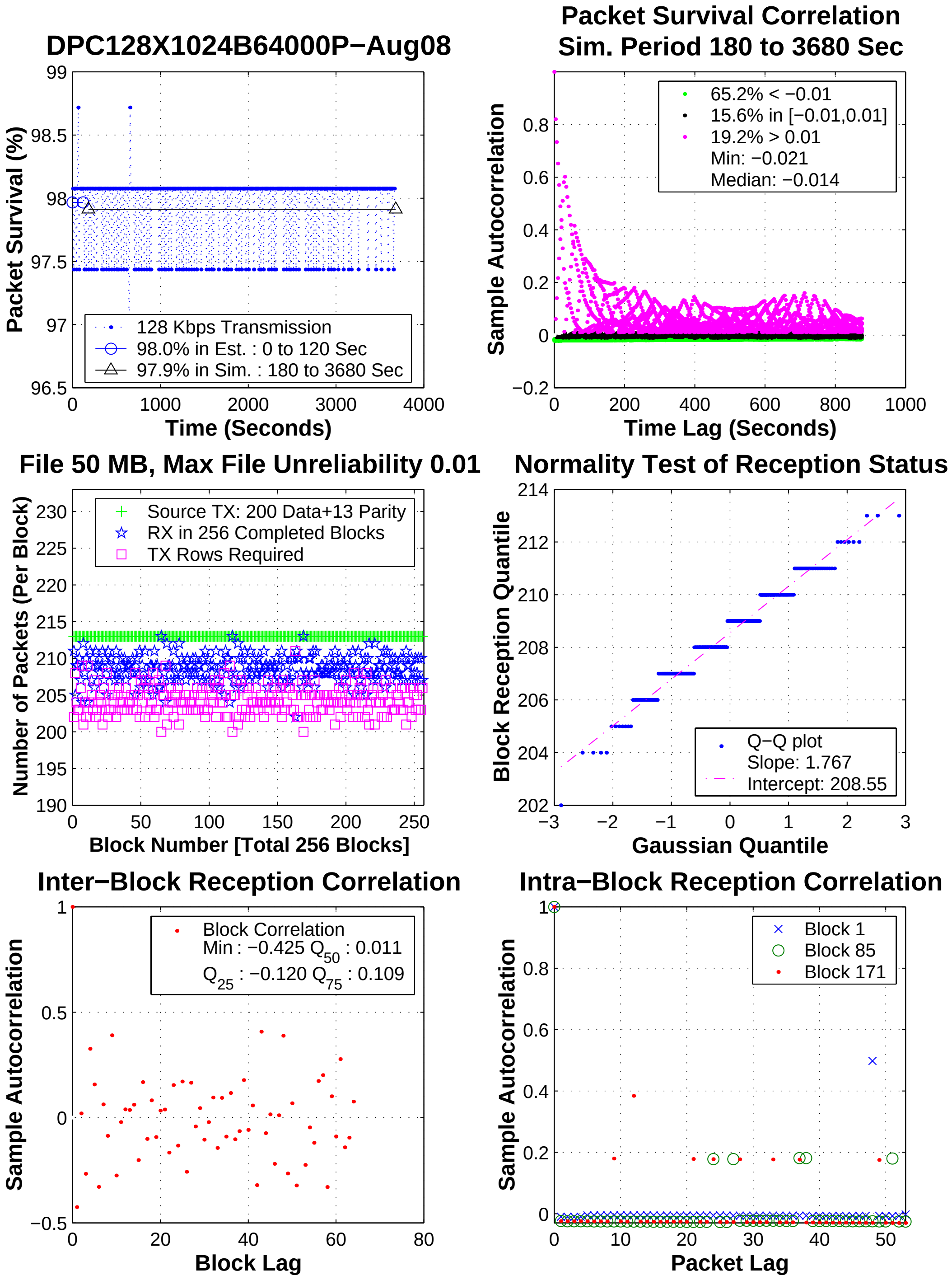


Figure 4.3: Trace-driven simulation : August 08, 2001 : HST 128 Kbits/sec transmission rate

(Note: Also listed in the Appendix as Figure B.13)

From Table 4.2, we describe the interpretation of the trace-driven simulation results as follows:

- The packet survival trace (A) generally reveals how stationary each trace is during the estimation, computation and simulated file transmission periods.

- The sample autocorrelation of the packet survival process (B) shows how strongly the elements $I(i)$ of the time series $\{I(i)\}$ depend on each other during the trace alloted for the simulated file transmission period. We show the sample autocorrelation function for time lags upto 1/4 of the duration of the simulated file transmission period.

- The realizations of the block reception status $X_w(h+c)$ in (C) shows the combined effect of both the interleaved transmission (Figure 4.1) and the packet survivals in each specific network channel. In the plots, a plus ('+') sign indicates how many data and parity packets the source transmits. For each block that receives at least h distinct packets (and thus considered completed), a star mark indicates the actual number of distinct packets received in that particular block and a square mark indicates the minimum number $j \geq h$ of transmission rows (i.e. Figure 4.1) that are required to allow the block to receive exactly distinct h packets. For each block that receives fewer than h distinct packets (and thus considered not completed), a cross ('×') mark indicates the number of distinct packets received in that particular block.

- The Gaussian quantile-quantile plot (D) reveals how close the one-dimensional distribution of $X_w(h+c)$ is to the Gaussian distribution. In particular, let $\Phi(z) = P[Z \leq z]$ be the cumulative distribution function (cdf) of a standard Gaussian $\mathcal{N}(0,1)$ random variable Z. On the horizontal and vertical axes respectively, the value of z_ι such that $\Phi(z_\iota) = p_\iota$ is plotted against the value of $P[X_w(h{+}c) \leq x_{(\iota)}] = p_\iota$, where $x_{(\iota)}$ are the ascendingly-ordered realizations of $X_w(h{+}c)$, and $\iota = 1, 2, \ldots, W$. We also show a least-square linear regression between z_ι and $x_{(\iota)}$ on each Gaussian quantile-quantile plot. The closer the Gaussian quantile-quantile plot to the linear regression, the closer the one-dimensional distribution of $X_w(h + c)$ to the Gaussian distribution ([Jai91]).

- The **Inter-block reception correlation** (E) reveals how strongly each block $w = 1, 2, \ldots, W$ is dependent on each other as the result of the interleaved transmission (Figure 4.1). We show the Inter-block reception correlation up to lag $W/4$.

- The **Intra-block reception correlation** (F) looks inside the packets that belong to each particular block w. From Figure 4.1, these packets of our interest would be $J_w(0) = I(w - 1)$, $J_w(1) = I(W + w - 1)$, $J_w(2) = I(2W + w{-}1), \ldots, J_w(j{-}1) = I((j{-}1)W{+}w{-}1)$ and finds the sample autocorrelation function $\hat{\gamma}_{J_w}(k)$ of lag k upto $(h + c)/4$. We choose to display the Intra-block reception correlation of blocks $w = 1$, $\lfloor \frac{W}{3} \rfloor$, and $\lceil \frac{2W}{3} \rceil$.

Table 4.3 summarizes how the block reception status across each of the W blocks behaves like a Gaussian random variable. Using the Gaussian quantile-quantile plots, we rate the empirical distribution of the block reception status into three (3) categories according to their similarity to the Gaussian distribution. *HIGH* means that the Gaussian distribution can very effectively approximate the block reception status. *MEDIUM* means that the Gaussian distribution may still effectively approximate the block reception status. *LOW* means that the Gaussian distribution may not be a good model.

Date and Network Config.	TX Rate (Kbits/sec)	Figure	Packet Survival (Sim. Period)	Reception Status : Similarity to Gaussian Distribution
Aug 02, 2001 (HST)	512	B.1	99.9%	*LOW*
	256	B.2	97.0%	*HIGH*
	128	B.3	99.7%	*MEDIUM*
	64	B.4	99.9%	*LOW*
	400	B.5	91.2%	*HIGH*
	100	B.6	99.9%	*LOW*
Aug 06, 2001 (HST)	512	B.7	99.4%	*MEDIUM*
	256	B.8	99.9%	*LOW*
	128	B.9	99.9%	*LOW*
	64	B.10	99.9%	*LOW*
Aug 08, 2001 (HST)	512	B.11	97.8%	*HIGH*
	256	B.12	97.9%	*HIGH*
	128	B.13	97.9%	*HIGH*
	64	B.14	97.5%	*HIGH*
Aug 09, 2001 (HST)	512	B.15	98.0%	*HIGH*
	256	B.16	97.9%	*HIGH*
	128	B.17	97.9%	*MEDIUM-HIGH*
	64	B.18	97.9%	*MEDIUM*

Table 4.3: Block reception status : convergence to Gaussian distribution

(Continued on next page)

Table 4.3 continued

Date and Network Config.	TX Rate (Kbits/sec)	Appendix Figure	Packet Survival (Sim. Period)	Reception Status : Similarity to Gaussian Distribution
Aug 10, 2001 (HST)	512	B.19	94.9%	*HIGH*
	256	B.20	98.0%	*HIGH*
	128	B.21	98.1%	*HIGH*
Aug 17, 2001 (HST)	512	B.22	97.9%	*HIGH*
	256	B.23	66.6%	*HIGH*
	128	B.24	99.9%	*LOW*
Aug 22, 2001 (HST)	512	B.25	97.8%	*HIGH*
	256	B.26	97.8%	*HIGH*
	128	B.27	97.6%	*HIGH*
	64	B.28	98.0%	*HIGH*
Aug 27, 2001 (HST)	512	B.29	97.8%	*HIGH*
Aug 19, 2001 (DSL)	512	B.30	98.6%	*MEDIUM-HIGH*
Aug 23, 2001 (CAN)	512	B.31	99.$\bar{9}$%	*LOW*

(Network Configurations: HST=Hybrid Satellite-Terrestrial Internet Network, DSL=Digital Subscriber Line, and CAN=Campus Area Network)

We observe that the *LOW* category generally results from a situation where most of the packets do survive (i.e. when the packet survival is 99.9% or greater). This makes sense because such a situation yields lower variance on the reception status outcomes, causing a significant degeneration in the convergence to the Gaussian distribution. We may expect an obvious opposite situation where almost all of the packets do not survive (i.e. when the packet survival is very close to 0%), the Gaussian distribution would be inaccurate to describe the block reception status with a similar reason. **In non-extreme cases, we have found that we may reasonably and accurately approximate the block reception status by the Gaussian distribution.** A closer inspection at the *Intra-block reception correlation* (i.e. item (F) of Table 4.2) reveals that the interleaved transmission technique significantly cuts the correlation of packets belonging to each particular w block down from the correlation of the original packet survival $\{I(i)\}$ (i.e. item (B) of Table 4.2). As expected, such correlation appears instead in the plots of the *Inter-block reception correlation* (i.e. item (E) of Table 4.2).

4.2 Related Theoretical Perspectives

We present three related theoretical perspectives here.

4.2.1 Markov Process Model

Yajnik et al ([YMKT99]) suggested that a finite-state Markov process model can be used to describe temporally correlated packet losses that occur in the real (terrestrial) Internet. Such a model relies on a finite history of packet losses to predict the next outcome. However, when we are involved with a long packet history such a model will quickly become too complicated for our purpose to implement a rate control system.

4.2.2 Multivariate Gaussian Model

Our study in Section 4.1 suggested that the one-dimensional view of the block reception status $X_w(j)$ evolves to be similar to a Gaussian random variable. It is natural to approximately model the joint block reception status across all blocks $w = 1, 2, \ldots, W$ as a **multivariate Gaussian random vector**. This can simplify how we represent the state of the block reception status. In the remaining of this section, we shortly write $X_w(j)$ as X_w, dropping the transmission row index (j) for readability in our notations. Let $\mathbf{X} = [X_1, X_2, \ldots, X_W]^T$ be a Gaussian vector of W elements, then it can be completely specified by its mean vector $\bar{\mu}_{\mathbf{X}} = [\mu_{X_1}, \mu_{X_2}, \ldots, \mu_{X_W}]^T$ where

$\mu_{X_w} = E[X_w]$ and its $W \times W$ positive definite **covariance matrix**:

$$\mathbf{\Sigma_X} = \begin{bmatrix} \sigma_{X_1,X_1} & \sigma_{X_1,X_2} & \sigma_{X_1,X_3} & \cdots & \sigma_{X_1,X_W} \\ \sigma_{X_2,X_1} & \sigma_{X_2,X_2} & \sigma_{X_2,X_3} & \cdots & \sigma_{X_2,X_W} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \sigma_{X_W,X_1} & \sigma_{X_W,X_2} & \sigma_{X_W,X_3} & \cdots & \sigma_{X_W,X_W} \end{bmatrix} \qquad (4.10)$$

where $\sigma_{X_a,X_b} = E[(X_a - \mu_{X_a})(X_b - \mu_{X_b})];\ a, b \in \{1, 2, \ldots, W\}$. We denote this concisely as

$$\mathbf{X} \sim \mathcal{N}_W(\bar{\mu}_{\mathbf{X}}, \mathbf{\Sigma_X}). \qquad (4.11)$$

In our case where the time series $\{I(i)\}$ is assumed to be stationary, and with the interleaved transmission, we can further simplifies the multivariate Gaussian approximation model in that:

- $\mu_{X_1} = \mu_{X_2} = \cdots = \mu_{X_W} = \mu_X$ (i.e. identical mean),
- $\sigma_{X_1,X_1} = \sigma_{X_2,X_2} = \cdots = \sigma_{X_W,X_W} = Var[X] = \sigma_X^2$ (i.e. identical variance)

where X denotes the (imaginary) one-dimensional state of the master block's reception status. Because of the stationarity of $\{I(i)\}$, we can write the covariance matrix in the following form:

$$\mathbf{\Sigma_X} = \begin{bmatrix} \sigma_{X_{|0|}} & \sigma_{X_{|1|}} & \sigma_{X_{|2|}} & \cdots & \sigma_{X_{|W-1|}} \\ \sigma_{X_{|1|}} & \sigma_{X_{|0|}} & \sigma_{X_{|1|}} & \cdots & \sigma_{X_{|W-2|}} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \sigma_{X_{|W-1|}} & \sigma_{X_{|W-2|}} & \sigma_{X_{|W-3|}} & \cdots & \sigma_{X_{|0|}} \end{bmatrix} \quad \{I(i)\} \text{ stationary} \qquad (4.12)$$

where $\sigma_{X_{|k|}} = E[(X_w - \mu_X)(X_{w+k} - \mu_X)]$ are the covariance between two blocks at lag $k = 0, 1, \ldots, W-1$. Note that (4.12) has a **Toeplitz structure**.

At the present, the general problem of evaluating the probabilities of the multivariate Gaussian random vector still proves to be computationally difficult (e.g. [Gen93]). Unless there is a breakthrough, we still may not be able to benefit much from any presently known multivariate Gaussian algorithm. However, there is a related interesting theoretical result known as the **Slepian inequality** (i.e. Theorem 5.1.7 of [Ton90]) which can be useful in some cases. Let $\mathbf{X} \sim \mathcal{N}_W(\bar{\mu}_{\mathbf{X}}, \mathbf{\Sigma}_{\mathbf{X}})$ and $\mathbf{Z} \sim \mathcal{N}_W(\bar{\mu}_{\mathbf{Z}}, \mathbf{\Sigma}_{\mathbf{Z}})$ be two Gaussian random vectors of the same dimension. $\mathbf{\Sigma}_{\mathbf{X}}$ and $\mathbf{\Sigma}_{\mathbf{Z}}$ are positive-definite covariance matrices. If the following conditions are true:

$$\begin{aligned} \bar{\mu}_{\mathbf{X}} &= \bar{\mu}_{\mathbf{Z}} \\ \sigma_{X_w,X_w} &= \sigma_{Z_w,Z_w} \text{ for } w = 1, 2, \ldots, W \text{ and,} \\ \sigma_{X_a,X_b} &\geq \sigma_{Z_a,Z_b} \text{ for } a \neq b; a, b \in \{1, 2, \ldots, W\} \end{aligned} \tag{4.13}$$

then it follows that

$$P[\bigcap_{w=1}^{W} X_w > h] \;\; \geq \;\; P[\bigcap_{w=1}^{W} Z_w > h]. \tag{4.14}$$

regardless of the signs of σ_{X_a,X_b} and σ_{Z_a,Z_b}.

A direct consequence of the Slepian inequality is that if all the joint reception status of W blocks are positively dependent. That is from (4.12), (4.13), and (4.14) when $\sigma_{X_{|k|}} \geq 0$ for $k = 1, 2, \ldots, W-1$, we may use a Gaussian random vector :

$$\mathbf{Z} \sim \mathcal{N}_W(\bar{\mu}_{\mathbf{X}}, \sigma_X^2 \mathbf{I}); \quad \text{where } \mathbf{I} \text{ is the identity matrix} \tag{4.15}$$

to approximate the upper probabilistic bound of the BFLU (4.3). Note that in fact $\mathbf{Z}$ in (4.15) is simply a vector of i.i.d. Gaussian random variables. Our results (i.e. the inter-block reception correlation) indicate that the block reception status may not exhibit positive dependence in general.

4.2.3 Alternative Bound on Reception Reliability

One way to find the probabilistic bound of the bulk file reception reliability is to look in terms of the zero-one (0-1) completion indicator of the block reception status. For each block $w = 1, 2, \ldots, W$ of the bulk file, the completion indicator of block w has a value of one if the block has received at least h distinct packets (i.e. the block is completed), otherwise the completion indicator has a zero value (i.e. the block is not completed). Therefore if there are W blocks, there are 2^W possible distinct joint outcomes. In Figure 4.4, if we have a control policy which can uniformly guarantee that the probability of each individual block w being not completed does not exceed ψ_{max} (shown on the left), then certain non-overlapping subsets of all 2^W possible outcomes are also guaranteed to have their probabilities not to exceed ψ_{max} (shown at the center). As a result, we can find the final probabilistic bound on a desirable case where all W blocks are completed (shown on the lower right).

By using the argument shown in Figure 4.4 and by construction, we have that if there are W blocks in a bulk file, and if we uniformly enforce one-dimensional unreliability across all W blocks to be at most ψ_{max}, then the **alternative bulk file reliability bound** will be:

$$P[\text{Bulk file is Completed}] \geq 1 - W \cdot \psi_{max} \tag{4.16}$$

The bound (4.16) would be meaningful only when $W \cdot \psi_{max} \ll 1$. In practice, it is more convenient to specify the maximum BFLU υ_{max}, by assuming independent packet survivals (which implies independent blocks). Then derive ψ_{max} using (4.8) and create a control policy based on such value. After that, we then can use (4.16) to find out what would happen if we are to implement such a

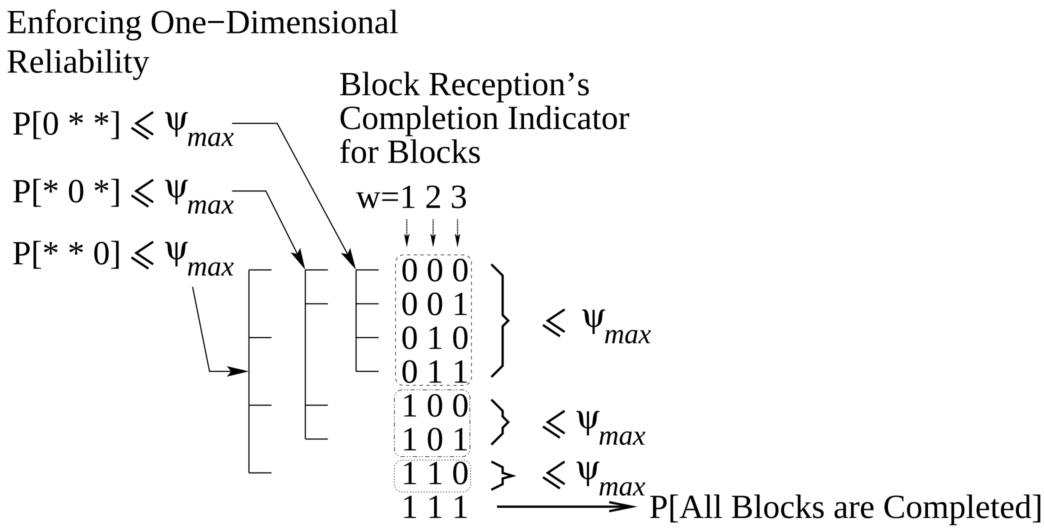


Figure 4.4: Example of enforcing one dimensional block reception reliability when $W = 3$

control policy in a general case where packet survivals may not be really independent. Figure 4.5 through Figure 4.8 illustrate the bounds resulting from this process when we specify the maximum BFLU υ_{max} to be $0.01, 0.05, 0.1$, and 0.5 respectively. We can observe the departures of the lower bounds on the bulk file reception reliability from the ideal value of $(1 - \upsilon_{max})$ when the independent packet survival assumption may not hold. We note however, that in applying (4.16), the stationarity assumption must still hold. In the trace-driven simulations of Section 4.1, we tried to guarantee the maximum BFLU value $\upsilon_{max} = 0.01$ and the corresponding alternative reception reliability bounds are shown in Figure 4.5. Therefore, **the violation of the independent packet assumption** was not the major cause of the observed incomplete receptions. **The deviation from the mean stationary assumption** was in fact responsible for the observed incomplete receptions.

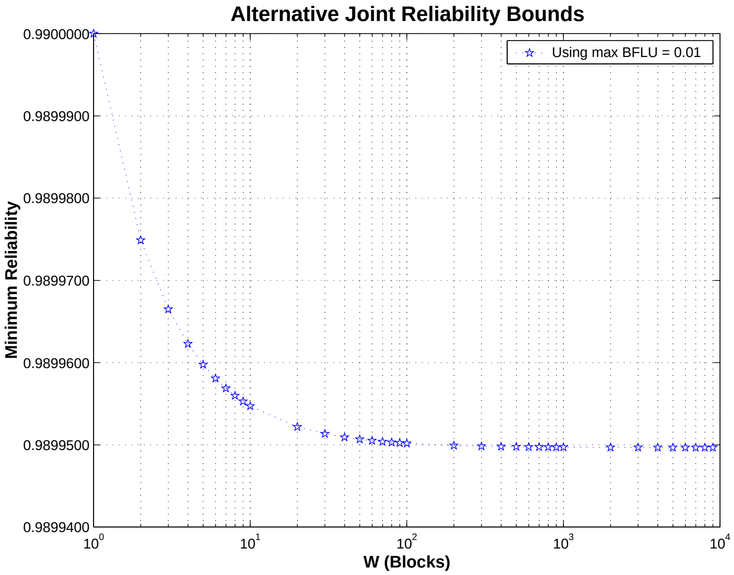


Figure 4.5: Alternative reception reliability bounds when setting max BFLU $v_{max} = 0.01$

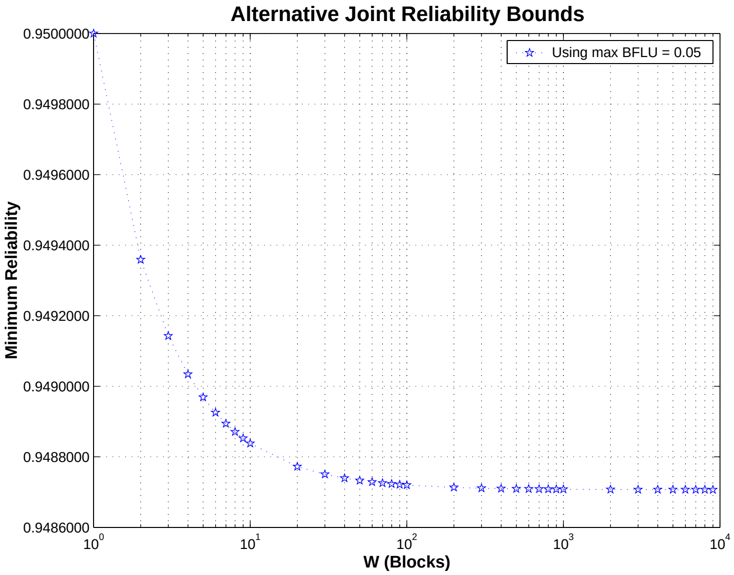


Figure 4.6: Alternative reception reliability bounds when setting max BFLU $v_{max} = 0.05$

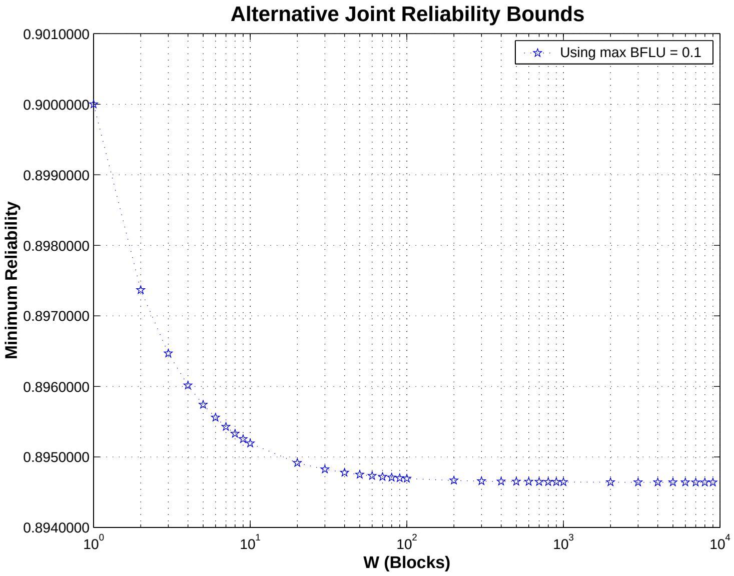


Figure 4.7: Alternative reception reliability bounds when setting max BFLU $v_{max} = 0.10$

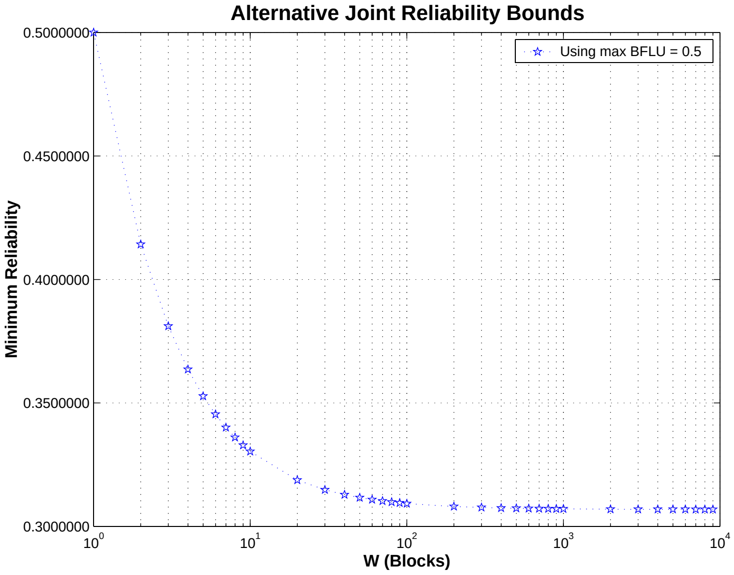


Figure 4.8: Alternative reception reliability bounds when setting max BFLU $v_{max} = 0.50$

4.3 Chapter Summary

We have introduced the Gaussian approximation approach in its simplest form. We can use it to estimate the status of the bulk file reception and to find the number of proactive RSE-based FEC parity packets needed for each bulk file transmission. We validated the approach using real satellite packet survival traces that we collected.

We have found the Gaussian approximation approach to be reasonably effective because it guarantees a certain degree of reliability for bulk file reception. In few cases where the Gaussian approximation fails to produce a reliable reception, it is generally because the time series exhibits some non-stationary behavior (a short-term black out, or a mean shift). In these cases however, the Gaussian quantile-quantile plots reveal that the reception status may still be approximated by the Gaussian distribution, but with a different mean and variance.

Chapter 5

End-to-End Multicast Rate Control

Evidences were found in the previous chapter that we can effectively use the Gaussian approximation approach to estimate the status of the bulk file reception. The Gaussian approximation approach appears to work in a real-world situation where the actual packet survivals are not independent.

In this chapter, we extend the Gaussian approximation approach towards the problem of end-to-end multicast rate control in hybrid satellite-terrestrial networks.

5.1 End-to-End Scenario

A general end-to-end scenario can be depicted in Figure 5.1. The sender transmits a bulk file via a satellite to a number of earth stations. Upon arrival at the earth station, each data packet is then multicasted through an Internet-like network to end receivers. Let the maximum capacity of the satellite channel be C bits per second. The sender can transmit at a rate r (measured in bits per second), selected from a finite set $\mathcal{R}$ of $|\mathcal{R}| = R$ distinct discrete rates, such that $0 < r \leq C$ for every $r \in \mathcal{R}$. Data packets, upon arriving at each earth station, are then multicasted through Internet-like networks to end receivers. The system works in an *end-to-end* fashion. All repairs come from the sender, and no intermediate nodes have any caching or local repair mechanism. The system can be easily extended to support the notion of **receiver representatives** making it scalable to a certain degree.

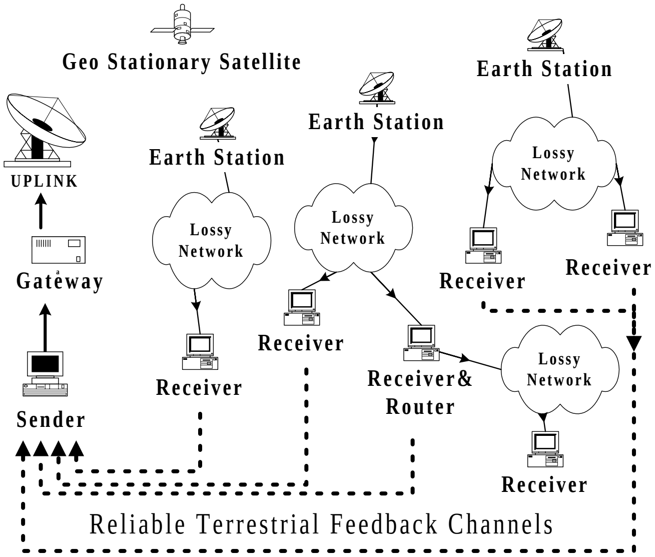


Figure 5.1: A sample end-to-end system.

We assume that there are M receivers, not necessarily homogeneous, each distinctly labeled by a number $m \in \mathcal{M} = \{1, 2, \ldots, M\}$. For simplicity, we assume, from the perspective of each particular receiver $m \in \mathcal{M}$ that when a data or parity packet is transmitted by the sender at a *specific rate* $r \in \mathcal{R}$, the packet survives and reaches receiver m as an independent Bernoulli trial with parameter $q_r^{(m)}$, which we denote as the **end-to-end packet survival probability**. We also assume that these probabilities are relatively time-invariant over the scheduling period, and are in the range:

$$0 < q_r^{(m)} \leq 1, \quad m \in \mathcal{M}, \, r \in \mathcal{R}. \tag{5.1}$$

Figure 5.2 depicts an example probabilistic multicast tree model that we assume. Two end-to-end paths are shown as an example. It is clear that at any two or more receivers, the Bernoulli outcomes of the same specific packet can be spatially (jointly) dependent. But when we restrict ourselves to a *one-dimensional* view at a specific receiver, the packet arrivals appear to be independent at that receiver, and we assume that its packet arrivals are *independent* Bernoulli trials. Accordingly, when the sender transmits n packets at one particular rate $r \in \mathcal{R}$ for a particular data block, the total number of packets received by a specific receiver m is $A^{(m)} \sim Binomial(n, q_r^{(m)})$.

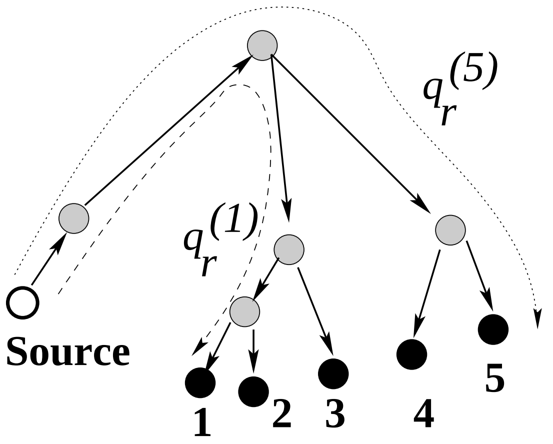


Figure 5.2: A sample probabilistic multicast tree model (Black, numbered nodes are receivers).

We assume the use of RSE-based FEC repairs, described earlier in Section 1.3. Each data block consists of h data and c parity packets, all of the same length. A receiver will need any combination of h distinct packets, which can be data or parity, from the total of $h + c$ packets to reconstruct the original data block. The bulk file consists of W data blocks, each distinctly labeled by a number

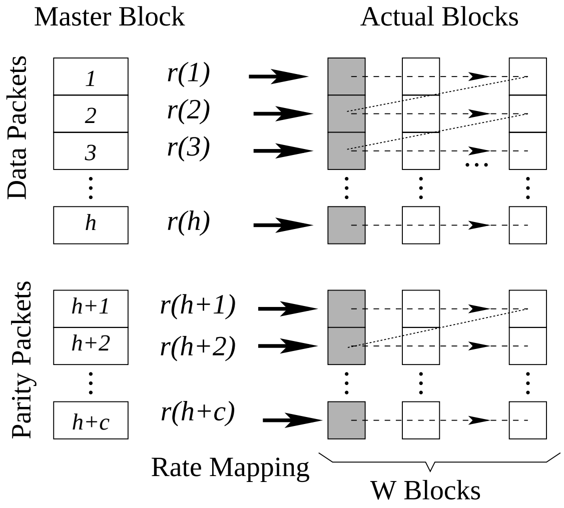


Figure 5.3: Mapping the master block to the actual data blocks

$w \in \mathcal{W} = \{1, 2, \ldots, W\}$. Each receiver $m \in \mathcal{M}$ monitors the history of multicast packet arrivals and periodically provides the sender with unbiased estimates of $q_r^{(m)}$ at each and every rate $r \in \mathcal{R}$.

5.2 The End-to-End Multicast Rate Scheduling Problem

The notions of **master block** and **interleaved transmission** described in Section 4.1 allow us not only to formally define the end-to-end multicast rate scheduling problem, but to systematically solve it.

Using the interleaved transmission technique and the **rate repetition mapping** of Figure 5.3, the **end-to-end multicast rate scheduling problem** to a population of heterogeneous receivers is equivalent to finding the rate assignments of the master block's data packets:

$$r(j_1), j_1 = 1, 2, \ldots, h. \tag{5.2}$$

and determining the amount of proactive parity packets $c \geq 0$, and their associating rates:

$$r(j_2), j_2 = h + 1, \ldots, h + c. \tag{5.3}$$

to be transmitted from the sender so that all the M receivers are subject to a certain master block's reliability criteria and that the rate assignments minimize a specific cost function.

Utilizing the rate repetition mapping of Figure 5.3, if the master block contains $h + c$ packets, the actual schedule will have $(h + c)W$ packets. We sequentially label each packet in the actual schedule by $\ell = 0, 1, \ldots, (h + c)W - 1$. Specifically, the rate $r_a(\ell)$, and the corresponding block label $w(\ell)$ of the ℓ-th actual packet, are respectively:

$$\begin{aligned} r_a(\ell) &= r(\left\lfloor \frac{\ell}{W} \right\rfloor + 1), & (5.4) \\ w(\ell) &= 1 + (\ell \; mod \; W). & (5.5) \end{aligned}$$

Besides the aforementioned ability to spread burst losses (if any) more evenly across all the W actual blocks, the rate repetition mapping of Figure 5.3 also allows faster receivers to complete their bulk file reception earlier because faster receivers can receive a combination of h distinct data or parity packets of all the W actual data blocks earlier. Note, however, that the rate repetition mapping of Figure 5.3 requires that the receivers have temporary space to hold all the incomplete data blocks while receiving each particular bulk file. This should not be a problem on modern computers because the size of the storage required is on the order of the size of the bulk file itself.

5.3 The State Space

It suffices to use the master block notion primarily to determine the state of any particular data block while the rate schedule is in progress. Let $Y^{(m)}(j)$ be a random variable denoting the number of the (virtual) distinct master block's packets received at each particular receiver $m \in \mathcal{M}$, after the sender has (virtually) transmitted j distinct packets for the master block, $j = 1, \ldots, (h + c)$. Initially $Y^{(m)}(0) = 0$. The evolution of the master block's state can be written as:

$$Y^{(m)}(j) = Y^{(m)}(j-1) + I^{(m)}(j), \tag{5.6}$$

where $I^{(m)}(j)$ is an independent Bernoulli (0,1) random variable with success probability $q^{(m)}_{r(j)}$. With FEC, a receiver needs at least h distinct data or parity packets to reconstruct a particular data block. From the *local* perspective of each receiver m, the **master block's local unreliability, MBLU**, after the j-th virtual packet is transmitted is defined as:

$$\psi^{(m)}(j) = P[Y^{(m)}(j) < h], \tag{5.7}$$

with $\psi^{(m)}(0) = 1$.

With the rate repetition mapping of Figure 5.3, the **bulk file's local unreliability, BFLU** at any multicast receiver can be established. At each receiver $m \in \mathcal{M}$, and at every block index $w \in \mathcal{W}$, let $X^{(m)}_w(\ell)$ be a random variable denoting the number of block w's distinct packets received by receiver m after the sender has transmitted the ℓ-th packet, $\ell = 0, 1, \ldots, (h + c)W - 1$. With an initial condition $X^{(m)}_w(-1) = 0, \forall w \in \mathcal{W}$, the state evolution is straightforwardly:

$$X^{(m)}_w(\ell) = \begin{cases} X^{(m)}_w(\ell-1) + I^{(m)}_{r(\ell)}, \text{ if } w = w(\ell) \\ X^{(m)}_w(\ell-1), \quad \text{otherwise.} \end{cases} \tag{5.8}$$

Here $I_{r(\ell)}^{(m)}$ is an independent Bernoulli (0,1) random variable with success probability $q_{r(\ell)}^{(m)}$, and the relationships $r(\ell)$, $w(\ell)$ were defined earlier in (5.4), (5.5) respectively.

A receiver cannot reconstruct the bulk file if the receiver has received, for some of the actual W data blocks, fewer than h distinct packets. By the independent packet, and independent data block assumptions, the BFLU at each receiver m after the sender has transmitted the ℓ-th packet is:

$$\begin{aligned} \upsilon^{(m)}(\ell) &= P[\ \exists w \in \mathcal{W} \text{ such that } X_w^{(m)}(\ell) < h] && (5.9)\\ &= 1 - P[\ \forall w \in \mathcal{W}, X_w^{(m)}(\ell) \geq h] && (5.10)\\ &= 1 - \prod_{w=1}^{W} \Big(1 - P[X_w^{(m)}(\ell) < h]\ \Big). && (5.11) \end{aligned}$$

We obviously have $\upsilon^{(m)}(\hat{\ell}) = 1$, for $\hat{\ell} = -1, 0, 1, \ldots, (h+c)W - 2$. In fact, we can establish a more rigorous relationship between BFLU and MBLU. For each master block packet $j = 1, \ldots, (h+c)$ and for each actual block packet $\ell = 0, 1, \ldots, (h+c)W - 1$, and using the notation:

$$\ell_j = (j-1)W \tag{5.12}$$

we may rewrite $\ell = \ell_j + w - 1$ where $w = 1, 2, \ldots, W$. Note that ℓ_j are those packets marked as gray in Figure 5.3. From (5.7) and (5.11) we then have:

$$\upsilon^{(m)}(\ell = \ell_j + w - 1) = 1 - [1 - \psi^{(m)}(j-1)]^{(W-w)}[1 - \psi^{(m)}(j)]^{w} \tag{5.13}$$

We showed in the previous chapter that we can establish a formal relationship between the maximum bound of the MBLU and that of the BFLU. It suffices to guarantee BFLU through uniformly enforcing the condition of MBLU to all W actual blocks. We require that the BFLU of any valid multicast rate schedule must not exceed some constant $0 < \upsilon_{max} \ll 1$. And this can be accomplished by requiring MBLU not to exceed another constant ψ_{max} through the relationships in (4.7) and (4.8).

5.4 The End-to-End Gaussian Approximation Approach

A crucial observation is that, for sufficiently large master block, $h \gg 0$, (5.6) is the sum of a large number of independent Bernoulli random variables. However they are not identically distributed. From (5.1), we additionally assume that the majority of receivers have $0 < q_r^{(m)} < 1$ (i.e. **they do not experience perfect receptions**). If $q_r^{(m)} = 1$, there is no packet loss and therefore no uncertainty. In such cases, it is trivial to determine the progress of the rate schedule. In (5.6), the mean and variance of each $I^{(m)}(j)$ are respectively $\mu_{I^{(m)}(j)} = q_{r(j)}^{(m)}$ and $\sigma^2_{I^{(m)}(j)} = q_{r(j)}^{(m)}(1 - q_{r(j)}^{(m)})$. Since $|I^{(m)}(j)| \leq 1$ and when $0 < q_{r(j)}^{(m)} < 1$, we have from (5.6) that the variance of $Y^{(m)}(j)$, as $j \to \infty$:

$$\sigma^2_{Y^{(m)}(j)} = \sigma^2_{Y^{(m)}(j-1)} + \sigma^2_{I^{(m)}(j)} \to \infty, \tag{5.14}$$

The above conditions satisfy the **"uniformly bounded case"** of the **Central Limit Theorem** as stated in [Ash72] and are sufficient to conclude that, as $j \to \infty$,

$$\frac{Y^{(m)}(j) - \mu_{Y^{(m)}(j)}}{\sqrt{\sigma^2_{Y^{(m)}(j)}}} \xrightarrow{\mathcal{D}} \mathcal{N}(0, 1), \tag{5.15}$$

where $\mu_{Y^{(m)}(j)}$ is the mean of $Y^{(m)}(j)$ and $\mathcal{N}(0, 1)$ is the standard Gaussian distribution. In practice, we can also use the Gaussian approximation approach to estimate the MBLU (5.7) through the Gaussian cdf. There is a special case, however, where $\sigma^2_{Y^{(m)}(j)} = 0$. This happens if either receiver m has experienced a perfect reception, or it is at the start of a rate schedule where $\mu_{Y^{(m)}(j)} = 0$. In this special case, we define

$$P[Y^{(m)}(j) < h | \sigma^2_{Y^{(m)}(j)} = 0] = \begin{cases} 1 & \text{if } \mu_{Y^{(m)}(j)} < h, \\ 0 & \text{otherwise.} \end{cases}$$

When using the Gaussian approximation for a rate schedule computation, we need to keep track of only the mean and variance of $Y^{(m)}(j)$ **for each receiver** $m \in \mathcal{M}$. The Gaussian approximation requires significantly smaller space and less computation time than keeping track of the actual probability mass function (pmf) of $Y^{(m)}(j)$. The schedule computation can now be geometrically interpreted on a mean-variance, (μ, σ^2), plane. First, we define the ratio

$$\Delta_{I^{(m)}(j)} = \frac{\sigma^2_{I^{(m)}(j)}}{\mu_{I^{(m)}(j)}} = 1 - q^{(m)}_{r(j)} \tag{5.16}$$

as the **slope** of the line segment representing a Bernoulli random variable with success probability $q^{(m)}_{r(j)}$ on the (μ, σ^2) plane. From (5.1), we correspondingly have

$$0 \leq \Delta_{I^{(m)}(j)} < 1. \tag{5.17}$$

Next, we consider the case where a rate schedule consists of K rate steps, at each step $k = 1, 2, \ldots, K$ the schedule 'virtually' transmits n_k master block packets at a particular rate $r_k \in \mathcal{R}$. In setting $(h + c) = \sum_{k=1}^{K} n_k$, we recall that the number of the master block's packets received by each receiver m during step k will be $A^{(m)}(k) \sim Binomial(n_k, q^{(m)}_{r_k})$. Note that the **slope** $\Delta_{A^{(m)}(k)}$ of the line segment representing $A^{(m)}(k)$ on the (μ, σ^2) plane:

$$\Delta_{A^{(m)}(k)} = \frac{\sigma^2_{A^{(m)}(k)}}{\mu_{A^{(m)}(k)}} = 1 - q^{(m)}_{r_k}, \tag{5.18}$$

depends solely on the success probability of the underlying Bernoulli trials. Similarly, we have:

$$0 \leq \Delta_{A^{(m)}(k)} < 1. \tag{5.19}$$

Figure 5.4 illustrates the conceptual view of scheduling computation using the Gaussian approximation. The MBLU of (5.7) can then be approximated at any intermediate packet $j = 1, 2, \ldots, (h + c)$. Since we know that the MBLU $\psi^{(m)}(j) = 1$ for $j = 1, 2, \ldots, h - 1$, we may start the approximation at indices $j = h, h + 1, \ldots, h + c$. In Figure 5.4, we also show the shaded region where $\psi^{(m)}(\jmath) \leq \psi_{max}$ (i.e. where each receiver m would become $(1 - \psi_{max})$ reliable if at some intermediate master block packet $\jmath \in \{h, h + 1, \ldots, h + c\}$, the 'end point' of the line segment representing $Y^{(m)}(\jmath)$ enters the shaded region). We call such region the **reliable subregion** of the mean-variance plane. The region is solely defined by max MBLU ψ_{max} which by itself is a function of max BFLU υ_{max}, W, and h. Therefore, **if we can ensure that the 'end point' of $Y^{(m)}(\jmath)$ enters the reliable subregion for each and every $m \in \mathcal{M}$, at some intermediate master block's packet $\jmath \in \{h, h + 1, \ldots, h + c\}$, the actual schedule based on the rate repetition policy of Figure 5.3 would be $(1 - \upsilon_{max})$ reliable.**

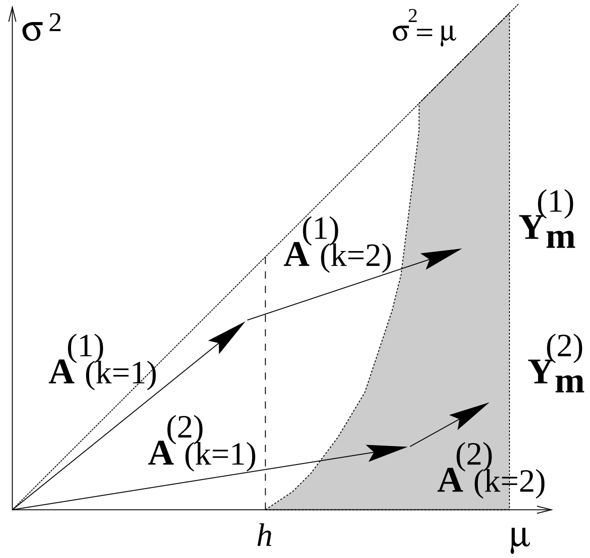


Figure 5.4: Schedule on (μ, σ^2) plane, M=2, K=2

5.5 Rate Schedule Computation

From the possible rate assignment space of (5.2) and (5.3), it is not practical to search for every possible solution. We make a few restrictions to the set of solutions in which we are interested. We restrict the solutions to those with either $K = 1$ (i.e. single-rate solutions), or $K = R$ (i.e. solutions utilizing every rate); from the latter case we shall seek performance improvements over the single-rate case. In the $K = R$ case, we allow only strictly decreasing rate steps (i.e. $r_1 > r_2 > \cdots > r_K$) and the corresponding $n_k > 0, k = 1, \ldots, K$. In our experiments, we have $R = 4$ with $K = 1$, and $K = 4$. Lastly, when $K > 1$, we restrict the choices of n_k, $k = 1, \ldots, K - 1$ to be $n_k = d_k f h$, where d_k is a non-zero, positive integer, and the **assignment quanta** is fh, where f is user-defined, $f \in (0, 1)$. Note that n_K depends on $\{n_1, \ldots, n_{K-1}\}$ since the last-step assignment needs to fulfill the schedule's $(1 - \upsilon_{max})$ reliability constraint. In our system, **the rate scheduling algorithm is based on a simple depth-first search with branch-and-bound technique** using the cost function to be described in the upcoming subsection. Intuitively, a smaller value of f generally leads to a rate schedule solution which has a better cost but it will take longer time to compute due to a larger search space. Figure 5.6 and Figure 5.7 illustrate how single-rate and four-rate schedules, respectively, can be computed on the (μ, σ^2) plane.

5.6 The Cost Function

The composite cost function which we try to minimize has two components: the bandwidth requirement and the aggregate delivery delay. Let $\phi \in [0, 1]$ be a

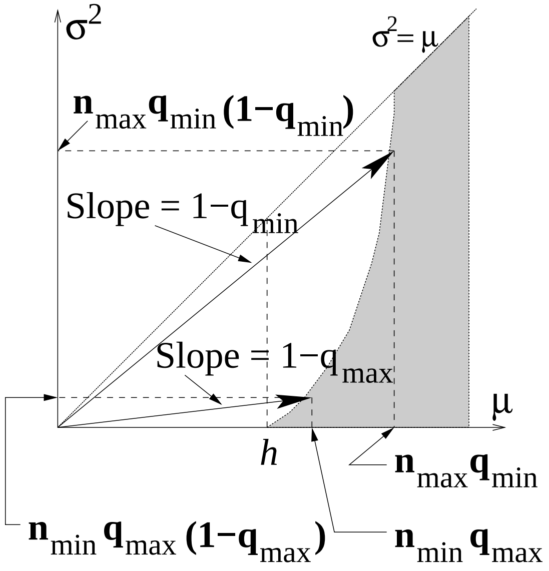


Figure 5.5: Finding n_{min}, n_{max}

system-wide tuning parameter called the bandwidth/delay trade off **knob** which adjusts the relative importance of the two cost components based on external system considerations. For any K-step, $(1-\upsilon_{max})$ reliable schedule, the **unnormalized bandwidth** (UBW) is the total number of data and parity packets in the master block:

$$UBW = h + c = \sum_{k=1}^{K} n_k. \tag{5.20}$$

We can find the (integer) lower and upper bounds of the UBW: n_{min}, and n_{max}. Define the *worst* and the *best* packet survival probabilities, respectively :

$$q_{min} = \min_{m\in\mathcal{M}, r\in\mathcal{R}} q_r^{(m)},$$

and

$$q_{max} = \max_{m\in\mathcal{M}, r\in\mathcal{R}} q_r^{(m)}.$$

With the Gaussian approximation, we can determine n_{min}, n_{max} by the process illustrated in Figure 5.5. For any $(1-\upsilon_{max})$ reliable schedule, the master block

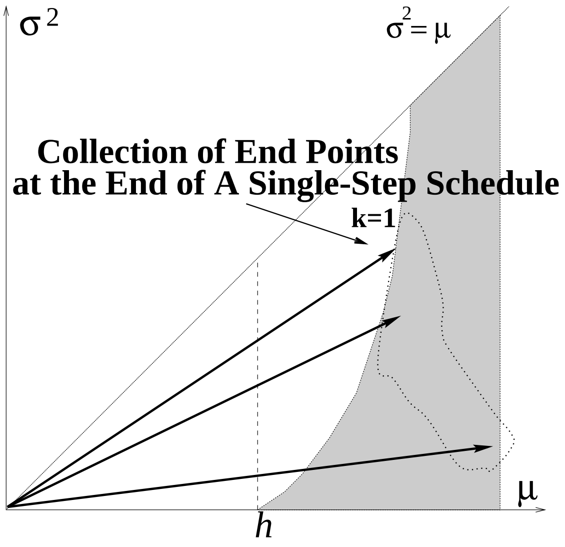


Figure 5.6: Single-step rate schedule, $M = 3$ receivers

requires at least n_{min} but at most n_{max} data plus parity packets. Because of receiver heterogeneity, we may safely assume that $q_{min} < q_{max}$, and therefore, $h \leq n_{min} < n_{max}$. The **normalized bandwidth requirement** is

$$BW = \frac{UBW - n_{min}}{n_{max} - n_{min}} \tag{5.21}$$

where $0 \leq BW \leq 1$. If a rate schedule has $BW = 0$ ($BW = 1$), the rate schedule can be interpreted as the *best* (*worst*) in terms of the bandwidth requirement.

The second cost component is the delivery delay. Since the proactive parity repair mechanism attempts to fit the bulk file transmission within a *single* transmission round (i.e. the best possible case when all receivers do not need to send NACK), we focus only on the time period between when each receiver sees the start of a new file and when the receiver receives sufficient data and parity packets to reconstruct the whole bulk file *completely.* We do not include, in our cost notion, the propagation delay because we cannot reduce it. Let L be the fixed packet length (in bits), the time used to transmit the ℓ-th actual packet,

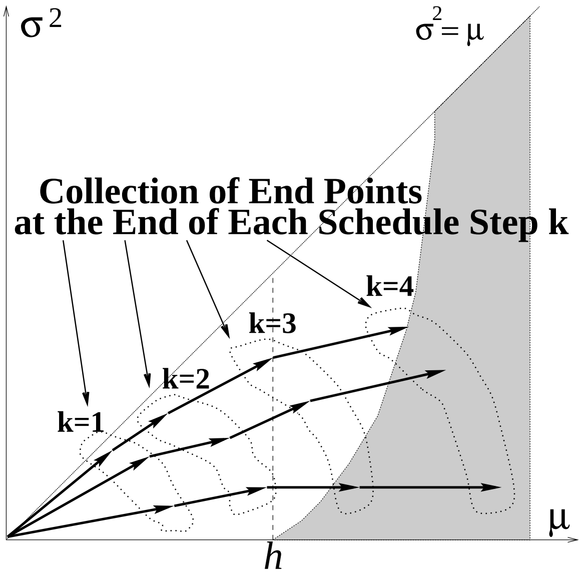


Figure 5.7: Four-step rate schedule, $M = 3$

$\ell = 0, 1, \ldots, (h+c)W - 1$ is $\tau(\ell) = \frac{L}{r(\ell)}$. At each receiver $m \in \mathcal{M}$, $\tau(\ell)$ can then be viewed as the time between two consecutive BFLUs, $\upsilon^{(m)}(\ell - 1)$ and $\upsilon^{(m)}(\ell)$. We define the delivery delay accumulated during the delivery of the ℓ-th packet at a specific receiver m to be the product $\upsilon^{(m)}(\ell - 1) \cdot \tau(\ell)$. The **unnormalized aggregate delivery delay** (UAD) is defined as the sum of these rectangular areas, across all the receivers, and across all the packets up to and including the last actual W blocks whose the associated master block first becomes $(1 - \psi_{max})$ reliable (i.e. after the master block of a particular receiver m has achieved $(1 - \psi_{max})$ reliability for the first time, we will no longer consider the delay from that receiver). Using the notation from (5.12), with the decomposition $\ell = \ell_j + w - 1$, we formally write:

$$UAD = \sum_{i=1}^{M} \left\{ \varphi(m) \sum_{j=1}^{h+c} \left[H^{(m)}(j) \sum_{w=1}^{W} \upsilon^{(m)}(\ell_j - w - 2)\tau(\ell_j + w - 1) \right] \right\}, \qquad (5.22)$$

where $H^{(m)}(j)$ is a 0,1 indicator function. $H^{(m)}(j) = 1$ from the beginning and only until the receiver m's master block has achieved $(1 - \psi_{max})$ reliability for the first time. $\varphi(m)$ is the weight of each receiver m. For now, $\varphi(m) = 1$ (i.e. $\sum_{i=1}^{M} \varphi(m) = M$).

Define $\tau_{min} = \min_{r \in \mathcal{R}} \frac{L}{r}$. We can then determine the lower bound of the UAD:

$$UAD \geq uad_{min} = W n_{min} \tau_{min} \sum_{i=1}^{M} \varphi(m). \tag{5.23}$$

Since we seek improvements in the multi-rate schedule ($K = R$ case), the upper bound uad_{max} of the UAD shall be derived from the *worst* UAD of the single-rate schedule. We will not consider any multi-rate solutions which may have their UADs greater than uad_{max} (i.e. their delays are worse than the single-rate schedule). The **normalized aggregate delivery delay** for the rate schedules under our consideration is:

$$AD = \frac{UAD - uad_{min}}{uad_{max} - uad_{min}}. \tag{5.24}$$

A $(1 - \upsilon_{max})$ reliable rate schedule that has $AD = 0$ ($AD = 1$) is the *best* (*worst*) schedule in terms of the aggregate delivery delay.

Finally, our **composite cost function** for a $(1 - \upsilon_{max})$ reliable rate schedule is:

$$COST = \phi BW + (1 - \phi) AD. \tag{5.25}$$

At this point, it should be clear that for the given **scheduling parameters** $\mathcal{R}$, W, h, and K, we have the knob ϕ and the maximum BFLU υ_{max} as our two **control parameters**. Additionally when $K > 1$, we also need to specify the assignment quanta f (described in Section 5.5), which partially controls the schedule computation time.

So far, each receiver $m \in \mathcal{M}$ has been assumed to represent itself. In some situations, for example, for a number of receivers residing within a single subnet or LAN and sharing a single incoming multicast traffic router, the packet survival statistics can be very similar and only one set of representative feedbacks is needed for every receiver in the subnet. Now we can think of each $m \in \mathcal{M}$ as one of the M receiver representatives. Each representative m is associated with a 'weight' $\varphi(m) \geq 1$ as the number of receivers it represents including itself. Since the definitions of reliability guarantee and the bandwidth consumption cost (i.e. UBW, BW) are unaffected by this modification, we will see the effects only in the delay cost component, i.e. UAD (5.22), and AD (5.24). This modification is also applicable to the case of having 'priority' receivers (where some receivers are more important than the others).

5.7 End-to-End Experiments and Results

We use the topology of Figure 5.8 to conduct our end-to-end multicast rate control experiments. Earth stations, main routers, and subnet routers are denoted by (E), (R), (S) respectively. A subnet contains a number of receivers (black or patterned nodes). All the receivers within a specific subnet are represented by a single representative (a patterned node). That is, we have 16 representatives representing 165 receivers in our simulations. Each representative is responsible for reporting the packet survival statistics and the number of receivers inside a subnet. Background traffic flows continuously in both directions between each receiver and its corresponding Internet (I) node, competing with our multicast traffic for the congested (R)-(S) links. The parameters are shown in Table 5.1. Before the first bulk file starts, we transmit 2000 probe packets at each and every $r \in \mathcal{R}$ to initialize the estimators of $q_r^{(m)}$. Note that the probe traffic is needed

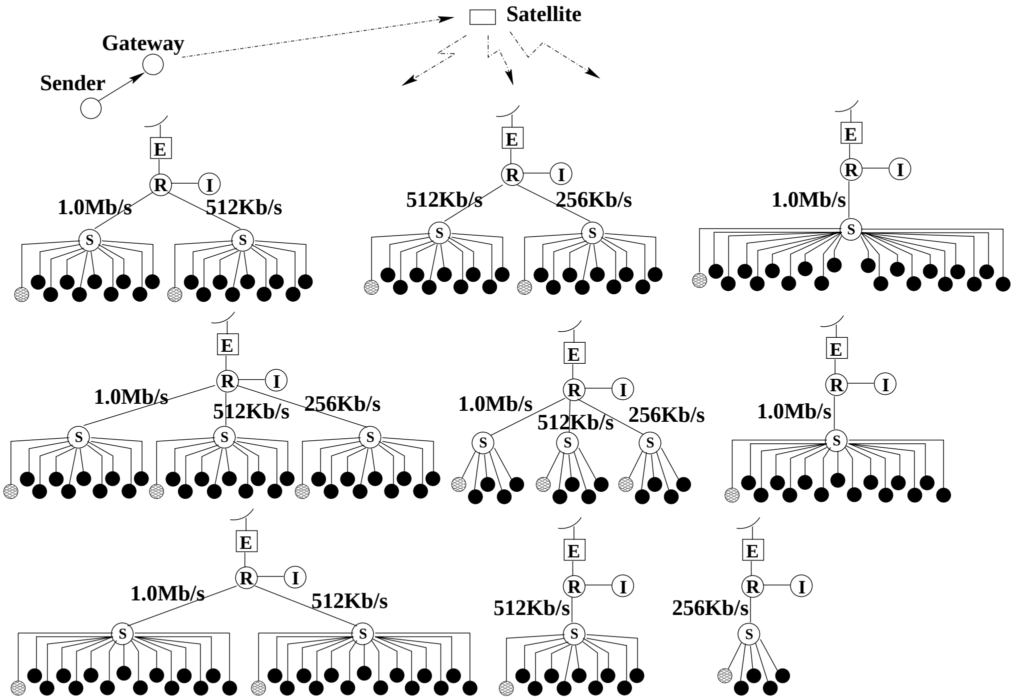


Figure 5.8: Network topology for end-to-end multicast rate schedule experiments

only once at startup, and in practice it can be part of the bulk file. Each new file transmission starts 5 minutes after the previous one has ended. To aid the rate schedule computation, each receiver representative reports packet survival statistics back to the sender after a file (or a probe) has ended, but before the start of the next file.

We have three major sets of the simulations based on the choices of the background traffic, which is either FTP-based using DropTail or Random Early Detection (RED) queueing, or Exponential (EXP) in nature with only DropTail queueing in place. The three sets are tabulated in Table 5.2 The EXP traffic rate is 45 Kbits/s, with 20ms and 50ms as mean burst time and mean idle time, respectively. The FTP traffic has the maximum TCP window set to 128. Within each major experiment set, there are six scenarios (A-F) designed to see the various effects of the control and scheduling parameters.

Parameters	Values, Notes
Satellite Channel	$C = 512$ Kbits/s, delay=270ms
Transmission Rates	$\mathcal{R} = \{512, 256, 128, 64\}$ Kbits/s
(E)-(R) links	512 Kbits/s, 5ms
(R)-(S) links	See Figure5.8, 5ms
(R)-(I) links	40Mbps, 20ms
(S)-(Receiver) links	10Mbps, 5ms
Bulk Files	Ten 5-MB files
Packets	1080 bytes (1024-byte data)

Table 5.1: Experiment parameters for end-to-end multicast rate scheduling

Set	Background Traffic	Queue
I	EXP No Flow Control	DropTail
II	FTP with renoTCP	DropTail
III	FTP with renoTCP	RED

Table 5.2: Experiment sets for end-to-end multicast rate scheduling

We conducted the simulations using the *ns-2* network simulator [NS2]. Each scenario A through F is conducted at knob settings $\phi = 0.0$ through 1.0 with an increment of 0.1. In each scenario, we first show the **global reliability**[1] which is the number of the files (out of the ten transmitted files) that were completed at

[1] Not the same as BFLU which is local to each receiver.

Scenario	Steps (K)	Total Blocks (W)	Data Packets Per Block (h)	f	Max BFLU υ_{max}
A	1	40	128	N/A	0.05
B	4	40	128	$\frac{1}{4}$	0.05
C	4	40	128	$\frac{1}{8}$	0.05
D	1	80	64	N/A	0.05
E	4	80	64	$\frac{1}{4}$	0.05
F	4	80	64	$\frac{1}{4}$	0.005

Table 5.3: Scenarios in each end-to-end experiment set

all the receivers without any receiver needing to request for repairs (NACK). Note that we did not incorporate any NACK repair mechanism in our simulations in order to isolate the pure effects of the rate schedules we computed. It is to understand that a deferred version of the NACK-based repair technique, where the NACK-based repair cycle starts only after the whole proactive transmission schedule has finished, may still be required to supplement our rate scheduling approach for 100% global reliability.

Secondly we show the average **redundancy ratios**, defined as $(h+c)/h$, as the results of the schedule computation, along with the sample standard deviations. Thirdly, only for those files which were completed without any need of NACK, we show the **average file reception latency** of the receivers for a globally completed file. This average reception latency was first computed at each receiver across all the globally completed files, then the average and the sample standard deviation were taken from these receiver average waiting times across all receivers.

The results from our experiment sets I, II and III are shown from Figure 5.9 through Figure 5.26 respectively.

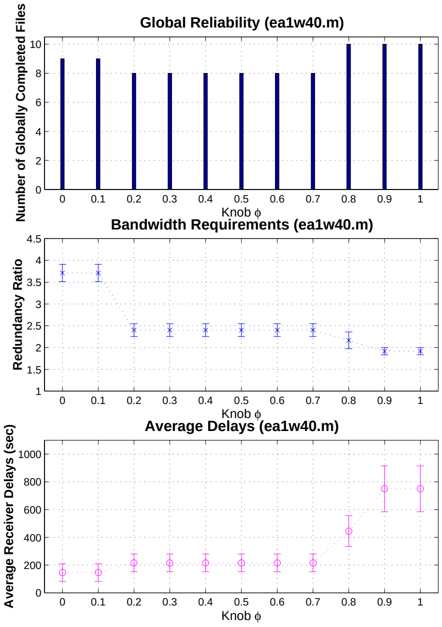


Figure 5.9: Results from end-to-end experiment I.A : exponential background traffic, DropTail queues : K=1, W=40, h=128, υ_{max}=0.05

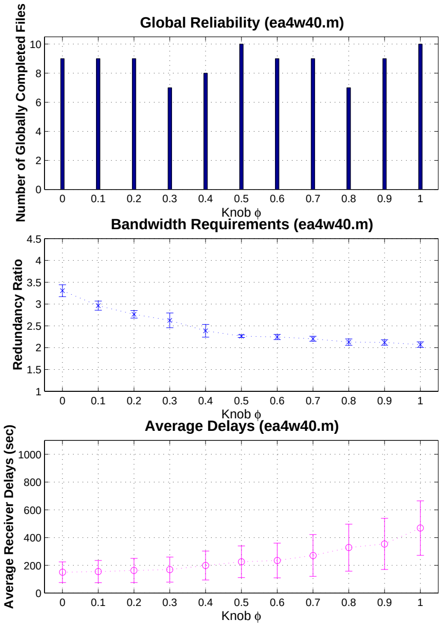


Figure 5.10: Results from end-to-end experiment set I.B : exponential background traffic, DropTail queues : K=4, W=40, h=128, f=0.25, υ_{max}=0.05

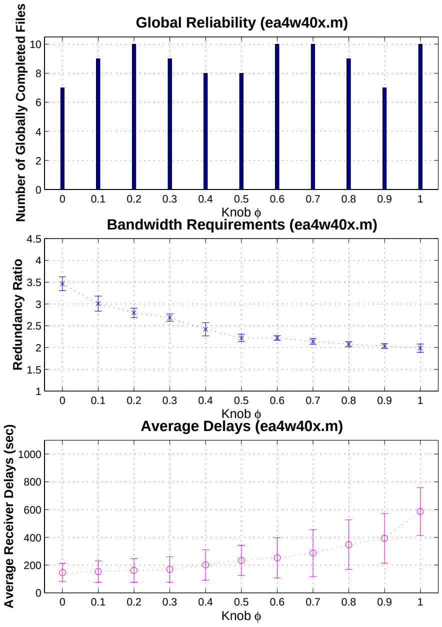


Figure 5.11: Results from end-to-end experiment set I.C : exponential background traffic, DropTail queues : K=4, W=40, h=128, f=0.125, υ_{max}=0.05

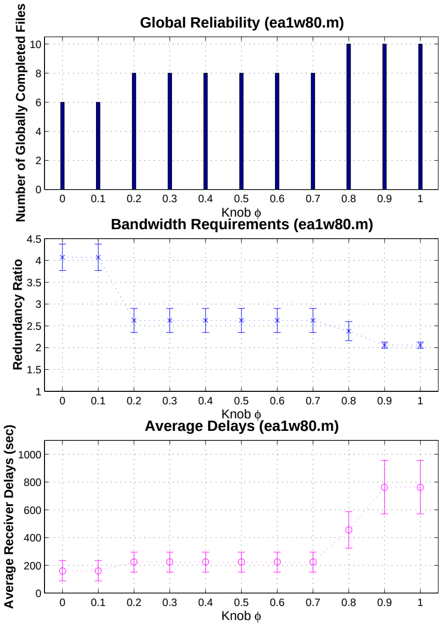


Figure 5.12: Results from end-to-end experiment set I.D : exponential background traffic, DropTail queues : K=1, W=80, h=64, υ_{max}=0.05

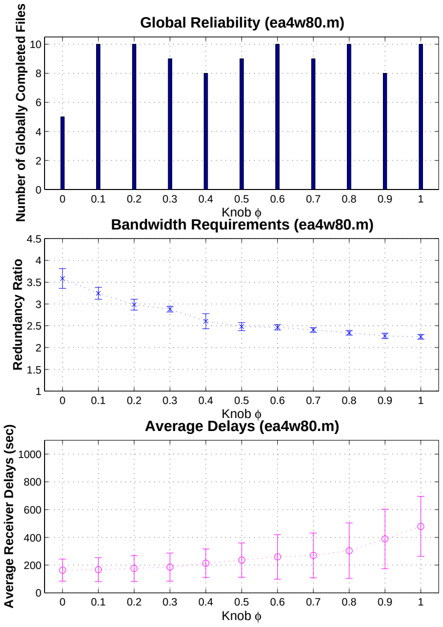


Figure 5.13: Results from end-to-end experiment set I.E : exponential background traffic, DropTail queues : K=4, W=80, h=64, f=0.25, υ_{max}=0.05

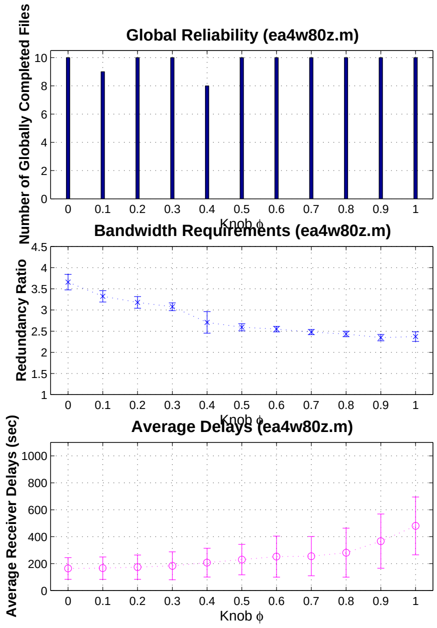


Figure 5.14: Results from end-to-end experiment set I.F : exponential background traffic, DropTail queues : K=4, W=80, h=64, f=0.25, υ_{max}=0.005

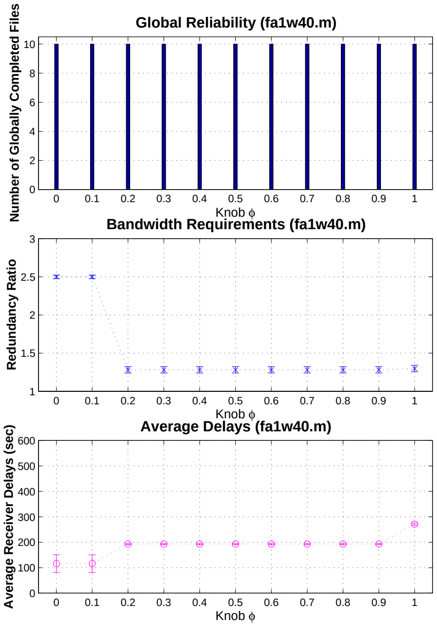


Figure 5.15: Results from end-to-end experiment set II.A : FTP background traffic (renoTCP), DropTail queues : K=1, W=40, h=128, υ_{max}=0.05

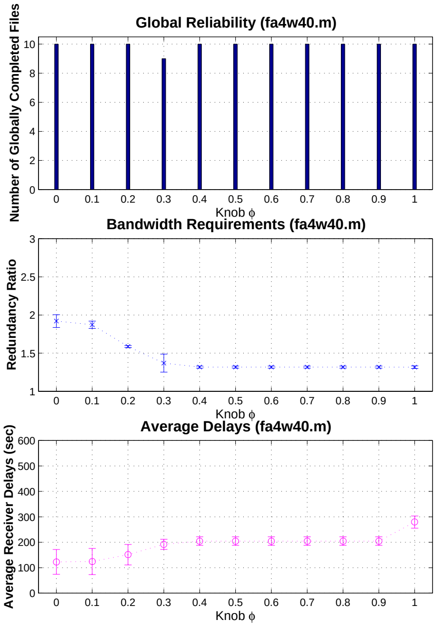


Figure 5.16: Results from end-to-end experiment set II.B : FTP background traffic (renoTCP), DropTail queues : K=4, W=40, h=128, f=0.25, υ_{max}=0.05

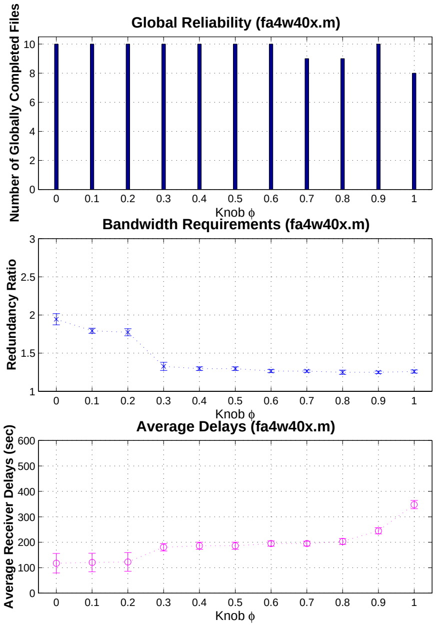


Figure 5.17: Results from end-to-end experiment set II.C : FTP background traffic (renoTCP), DropTail queues : K=4, W=40, h=128, f=0.125, υ_{max}=0.05

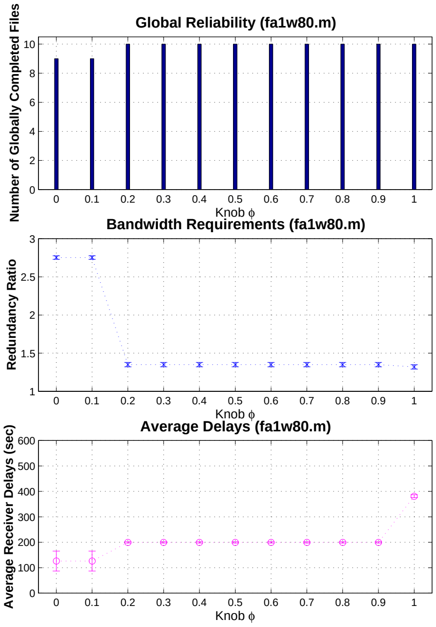


Figure 5.18: Results from end-to-end experiment set II.D : FTP background traffic (renoTCP), DropTail queues : K=1, W=80, h=64, υ_{max}=0.05

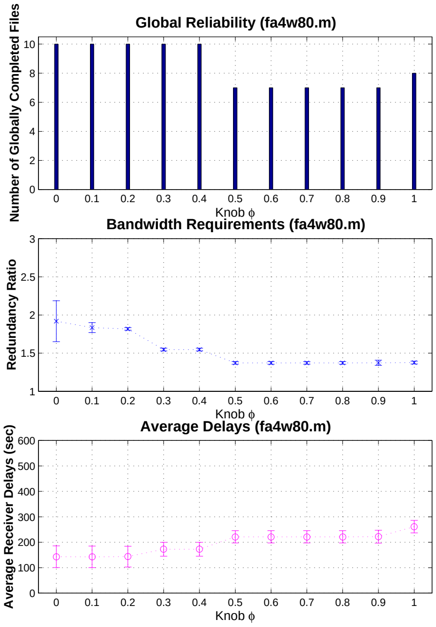


Figure 5.19: Results from end-to-end experiment set II.E : FTP background traffic (renoTCP), DropTail queues : K=4, W=80, h=64, f=0.25, υ_{max}=0.05

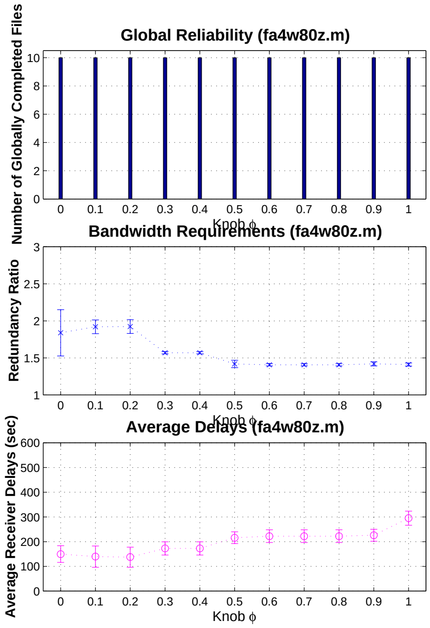


Figure 5.20: Results from end-to-end experiment set II.F : FTP background traffic (renoTCP), DropTail queues : K=4, W=80, h=64, f=0.25, υ_{max}=0.005

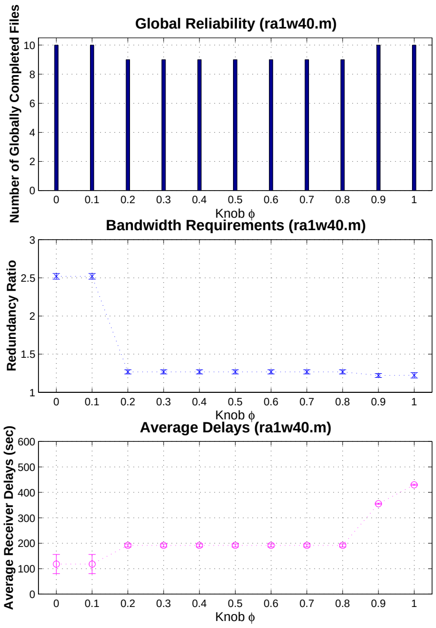


Figure 5.21: Results from end-to-end experiment set III.A : FTP background traffic (renoTCP), RED queues : K=1, W=40, h=128, υ_{max}=0.05

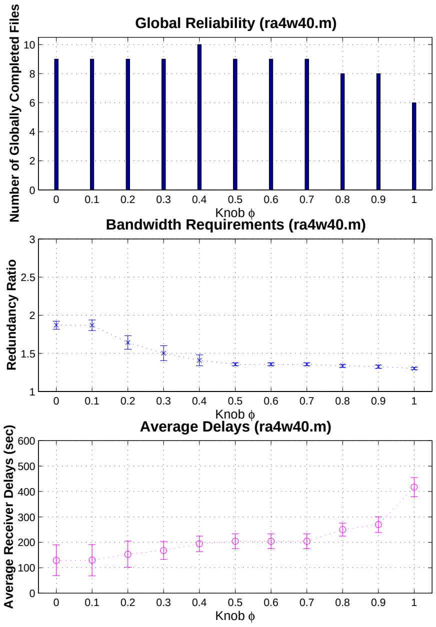


Figure 5.22: Results from end-to-end experiment set III.B : FTP background traffic (renoTCP), RED queues : K=4, W=40, h=128, f=0.25, υ_{max}=0.05

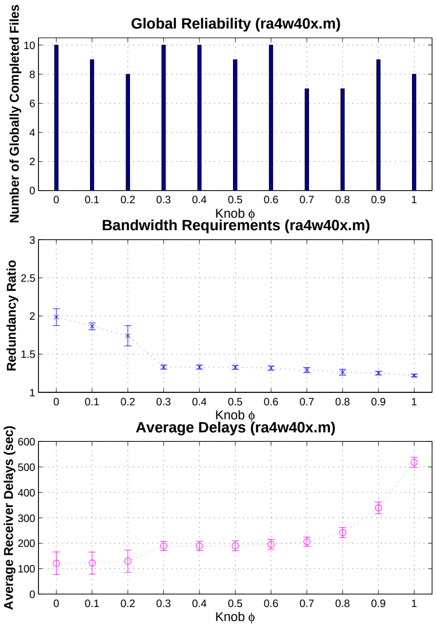


Figure 5.23: Results from end-to-end experiment set III.C : FTP background traffic (renoTCP), RED queues : K=4, W=40, h=128, f=0.125, υ_{max}=0.05

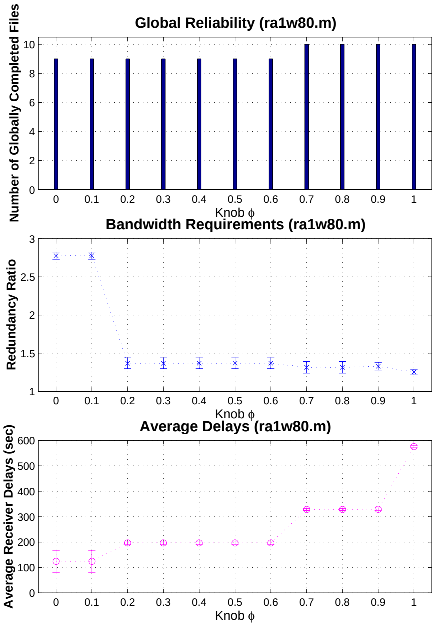


Figure 5.24: Results from end-to-end experiment set III.D : FTP background traffic (renoTCP), RED queues : K=1, W=80, h=64, υ_{max}=0.05

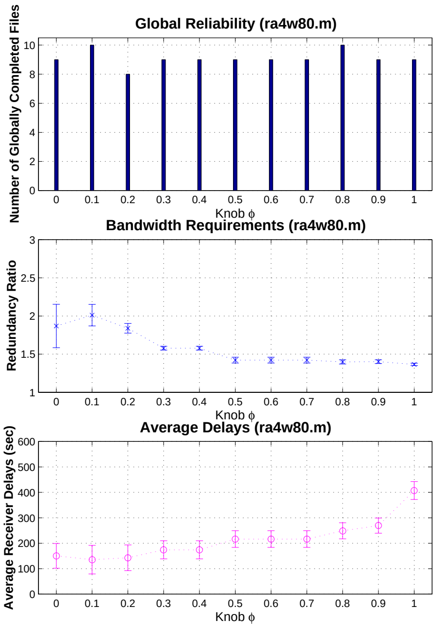


Figure 5.25: Results from end-to-end experiment set III.E : FTP background traffic (renoTCP), RED queues : K=4, W=80, h=64, f=0.25, υ_{max}=0.05

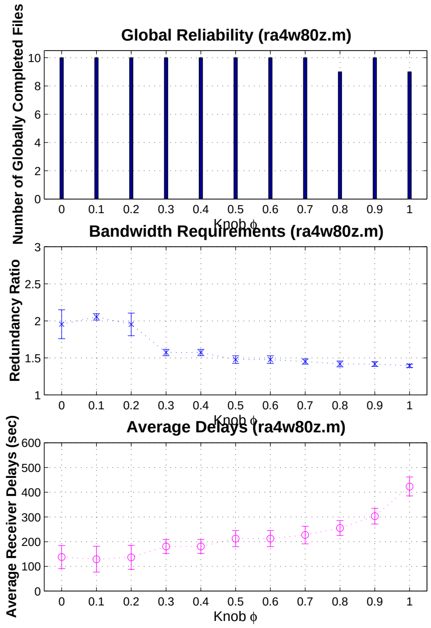


Figure 5.26: Results from end-to-end experiment set III.F : FTP background traffic (renoTCP), RED queues : K=4, W=80, h=64, f=0.25, υ_{max}=0.005

We will now describe key observations from our results. First, although we cannot directly compare between experiment set I and experiment sets II, III due to the differences in their background traffic nature, we can see that the rate scheduling method effectively trades the bandwidth requirement and the receiver delay in response to the knob setting ϕ. From (5.25), when $\phi \rightarrow 0$, the goal is to minimize delay, while when $\phi \rightarrow 1$, the goal is to minimize the bandwidth utilization.

Second, we can use the parameter υ_{max} to control reliability. The reader may compare the numbers of globally completed files in Scenario E (where $\upsilon_{max} = 0.05$) with those in Scenario F (where $\upsilon_{max} = 0.005$) within each specific experiment set. Recall that the maximum BFLU υ_{max} is associated with a one-dimensional, or local, view of each receiver's reliability. In general, the relationships between this local reliability view and the global reliability view (e.g. the number of globally completed files) cannot be explicitly drawn unless we know the underlying network topology. However, it appears that by carefully lowering the maximum BFLU υ_{max}, we can achieve high degree of global reliability. This is a desirable property when we consider an end-to-end approach where we do not have a complete knowledge of the network topology. Also note that we see a higher degree of global reliability with the FTP background traffic cases (Figures 5.15 through 5.26) because the underlying TCP flow control tries to adapt to accommodate our multicast traffic. The TCP flow control also contributes to a smaller 'swing' in the ranges of bandwidth requirement or the delivery delay. For example the average redundancy ratios range only from approximately 1.3 to 1.9 in experiment II.B, while they range from approximately 2.1 to 3.3 in experiment I.B.

Third, the four-step algorithm ($K = 4$) generally was able to find the more bandwidth-efficient solutions than the single-step algorithm's, while still maintaining relatively the same or better average user delays (for the globally completed files). Readers can see this effect when comparing Scenario A with Scenario B or when comparing Scenario D with Scenario E within each specific experiment set. However, when the block is small ($h = 64$), the four-step schedule suffers adverse effects on its global reliability during the FTP with Droptail cases, especially when the goal is to minimize bandwidth $\phi \to 1$ (i.e. compare the number of the globally completed files in II.D v.s. II.E). We believe that it results from the correlation among TCP flows which violates the independent packet assumption. That is, in a four-step schedule, we first transmit at the highest rate (512 Kbits/s) then drop down to the lower rates (256, 128 and 64 Kbits/s). This allows TCP to gradually regain the network bandwidth. But when many TCP flows restart simultaneously, there is an initial adverse effect to our multicast traffic. We can counteract such effects by decreasing the maximum BFLU υ_{max} from 0.05 to 0.005 (II.E v.s. II.F), effectively adjusting the lengths of our multicast transmission rate steps ($n_k, k = 1, \ldots, K$) or putting more proactivity (c) into the schedules. We did not see these effects in the case of RED queuing (III.D v.s. III.E). The RED queue is known to reduce start/restart correlations among TCP traffic. We believe that this also explains why the graphs of experiment set III (Figures 5.21 through 5.26) show more responsiveness to ϕ than those in experiment II (Figures 5.15 through 5.20). In runs II.F, III.E and III.F, we also see some slight anomalous 'rebounds' of the bandwidth requirements and the delivery delay around knob settings $\phi = 0.0$, 0.1 and 0.2. We also believe this results from the behavior of TCP, since there is no such anomaly in the EXP background traffic cases where there is no flow control. It may be advisable to not fully set the knob ϕ to 0.0 (i.e. minimizing delays through arbitrarily large bandwidth) when we are dealing with TCP background traffic.

Fourth, in the cases of the four-step schedules ($K = 4$), Scenarios B and C of each specific experiment set can be compared to see the effect of f. In Scenario B, the assignment quanta is $fh = 32$, while in Scenario C, the assignment quanta is $fh = 16$. Structurally, the scheduling algorithm in Scenario C considers all the solutions considered by Scenario B and additionally explores potential solutions with smaller assignment quanta (this increases the computation time, of course). In our experiment sets I-III, we see only slight differences in the schedule bandwidth requirements, but we see some noticeable differences in the receiver delays especially when $\phi \to 1$.

Fifth, we consider the complexity of the encoding and decoding process. In the $W = 40$ cases ($h = 128$), the rate schedule's redundancy ratios $\frac{(h+c)}{h} < 4$, implying that we can use a parallel FEC encoder/decoder based on GF(2^9) (i.e. symbol size $\beta = 9$ bits). However, this is not very common. In the $W = 80$ cases ($h = 64$), most of the rate schedule's redundancy ratios (especially those schedules utilizing multi-rate ($K = 4$), i.e. Scenarios E, F) have $\frac{(h+c)}{h} < 4$. This implies that we can use the more commonly available parallel FEC encoder/decoder based on GF(2^8), i.e. where symbol size is $\beta = 8$ bits. It is important to note that by using multiple-rate schedules, we can use a less complex and therefore faster FEC encoder/decoder which might not be feasible if we only employed the single-rate schedules. The ability to choose a less complex encoder/decoder is very desirable when dealing with near- realtime or delay-sensitive applications. However, when W is large, the bandwidth usage is increased due to the fact that a data or parity packet belonging to a particular data block is useless to the recovery of the other data blocks. A new class of erasure correcting codes, i.e. the Tornado codes [BLR98], has been shown to have significantly lower encoding/decoding complexity with a very small bandwidth increase penalty. However, the multicast rate control problem formulation based on such code would be slightly more complicated.

Finally, we consider the schedule computation time. The averages of the schedule computation times are presented in Table 5.4. Note that the times reported were measured on SUN Ultra 10 with 128MB memory and reflect the actual time spent in the rate computation module while we run the simulations. The space required for the scheduling algorithm is very much smaller than the main memory. For the single-step schedules ($K = 1$: Scenarios A and D), the computation times were very fast. For multi-step schedules (e.g. when $K = 4$ in Scenarios B, C, E, and F), the computation times vary greatly depending on the number of receiver representatives, the assignment quanta fh, the maximum BFLU υ_{max}, and the packet survival statistics. In our experiments, the schedule computations took more time in the EXP traffic cases because the multicast traffic experienced more losses. The algorithm must compensate for these losses while scheduling the transmission rates, and this scheduling processing is necessarily more lengthy. We also need to consider the trade off between the small assignment quanta fh (which might yield lower-cost solutions) and the small schedule computation time (the practical aspect). In our cases, the setting $f = 0.25$ seems more appropriate for the four-step schedules than the setting $f = 0.125$ as the latter setting gave little benefit at much greater expense. Adjusting the maximum BFLU υ_{max} from $\upsilon_{max} = 0.05$ to $\upsilon_{max} = 0.005$ (Scenarios

E and F respectively) has only small effects on the schedule computation times. Note that the receiver representatives transmit packet survival feedbacks only after the end of a file or the probe (i.e. one feedback from each representative per each file) the feedback processing overhead is thus significantly lower than that of the NACK-based approach.

Experiments	A	B	C	D	E	F
I (EXP)	1.2	27.8	164.7	1.0	19.5	23.4
II (FTP,DropTail)	0.8	3.9	18.7	0.9	4.2	4.7
III (FTP,RED)	0.9	4.3	25.2	1.2	5.9	4.7

Table 5.4: Average schedule computation times (seconds) in each end-to-end experiment scenario

5.8 Chapter Summary

We have presented an end-to-end rate scheduling approach for multicasting bulk data over a satellite-terrestrial network. Our end-to-end rate scheduling approach automates the selection of the proper transmission rate sequence and the amount of proactive parity packets for a given trade-off between delivery latency and bandwidth utilization. When the network is relatively stable, our approach needs only a small amount of control overhead to gather long-term packet survival statistics, and can reduce and oftentimes eliminate the use of the NACK-based repair mechanism. We have shown through simulations that our approach is effective both with and without the presence of background traffic flow control and in various network settings. The approach still remains reasonably effective even when the underlying independent packet assumption is violated.

Chapter 6

On Co-Existence with TCP Background Traffic

One of the unsolved problems that remains is that when the satellite multicast traffic from the sky merges with existing local terrestrial traffic, the two might not share the available terrestrial bandwidth fairly, especially when the terrestrial traffic uses a TCP congestion control mechanism. TCP's effective throughput can drop sharply in a very short period in response to network congestion that the long-term multicast rate scheduling mechanism might have caused. The large propagation delay would also make the multicast sender appear more sluggish in adapting to terrestrial network congestion if the multicast sender were to implement some sort of short-term rate control or congestion control. One solution to this problem is to slightly break the end-to-end model and let some terrestrial entities handle the problem.

We assume that each earth station receiving multicast data from the satellite is located very near (i.e. a few hops away) to the multicast receivers. This configuration is reasonable for many deployment scenarios as the satellite multicast traffic avoids the congested network routes in the terrestrial network. We also assume that there are some background TCP traffic streams originating from some nearby terrestrial sender, also being received by multicast receivers, and thus sharing a portion of the terrestrial network links with the multicast traffic. Recent work such as [GB00] has called for network support for multicast in heterogeneous environments. In a similar fashion this chapter focuses on a simple use of fair queueing algorithms at the routers where the multicast traffic meets terrestrial traffic near the receivers (i.e. at the very edge of the networks), as shown in Figure 6.1. The fair queueing algorithm blocks the excess part of the multicast traffic, protects the TCP traffic from starvation, and at the same time makes the multicast network appear heterogeneous (from the sender's perspective) due to both the presence of the terrestrial traffic and the varying link capacity. Because the abstract of the problem does not change, the multicast rate scheduling mechanism, introduced in the previous chapter, can handle this heterogeneity.

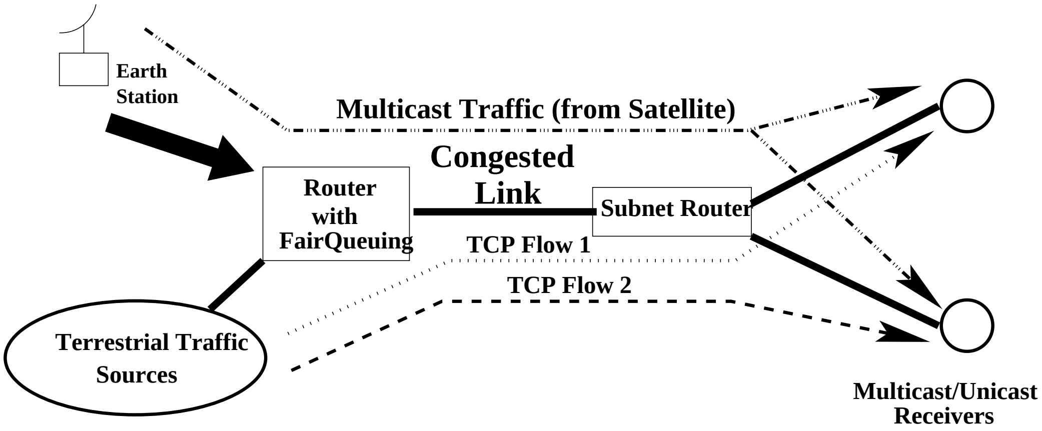


Figure 6.1: Utilizing fair queueing in satellite multicast

6.1 The Fair Queueing Algorithms

Two broad types of fair queueing algorithms have been proposed. The first type of fair queueing algorithms requires some per-flow state to be maintained at the routers (e.g. "Deficit Round Robin" or DRR [SV95]). The second type of fair queueing algorithms improve scalability by dividing routers into *edge* routers and *core* routers. The edge routers, which reside near the edge of the network, maintain per-flow states, and tag each packet by certain value that represents the flow properties (e.g. rate) before forwarding the packet into the network core – where the core routers compare only the information on the packet's tag to decide whether to drop or forward the packet (e.g. "Core-Stateless Fair Queueing" or CSFQ [SSZ98] and $\mathcal{T}$UF[CD01]).

Recall that the satellite multicast traffic merges with the terrestrial network traffic near the edges of the network. Using the fair queueing algorithms which maintain per-flow state is still reasonable because fewer active flows exist at the network edges than those inside the network core. We shall focus on two fair queueing algorithms: the well-known DRR [SV95] and its recently proposed extension DRR+ [HMMM00]. DRR has a theoretical fairness for flows with different packet lengths. DRR serves active flows in a round-robin manner. When it is the turn of a particular flow that has a packet waiting to transmit, the DRR algorithm adds a pre-defined quantum to the flow's *deficit counter*. The packet belong to that flow is allowed to be forwarded only if the value in the flow's deficit counter is not less than the length (e.g. in bytes) of the packet. If DRR forwards the flow's packet, the flow's deficit counter will be decremented by the length of the flow's packet. DRR+ extends DRR by replacing the DRR's per-flow first-in-first-out (FIFO) mechanism by a per-flow random early detection (RED)

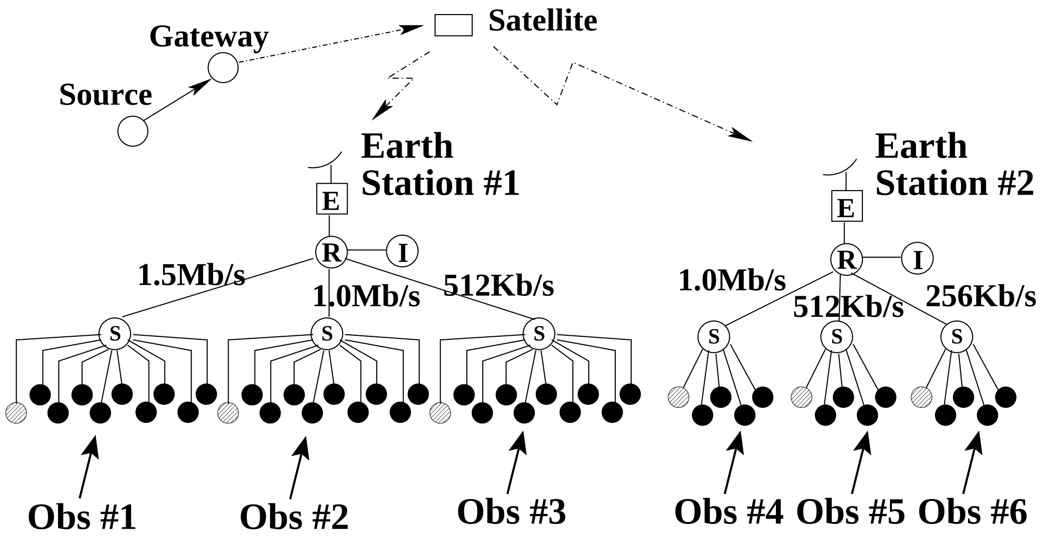


Figure 6.2: Topology used in the experiments

mechanism. DRR+ was shown in [HMMM00] to provide better fairness among reno TCP traffic flows than its DRR counterpart.

In this chapter, we utilize DRR and DRR+ mainly both to block excess part of the long-term multicast traffic from reducing background Reno TCP throughput and to create a favorable operating environment for our long-term rate scheduling mechanism.

6.2 Multicast with Fair Queueing Experiments and Results

We can use the end-to-end rate control in Chapter 5 We still use the network simulator (*ns-2*) [NS2] to study the performance. The topology used in our simulation is depicted in Figure 6.2; (E) denotes an earth station, (R) denotes a router, (S) denotes a sub-net router, and (I) denotes the source of the terrestrial background traffic. Each receiver, denoted by either black or shaded leaf nodes, receives both multicast traffic and unicast FTP traffic; the unicast FTP traffic

Parameters	Values and Notes
Satellite Channel	$C = 512$ Kbits/s, delay=270ms
(E)-(R) links	512 Kbits/s, 5ms
(R)-(S) links	bandwidths as in Figure 6.2, 5ms
(I)-(R) links	40Mbps, 20ms
(S)-(Receiver) links	10Mbps, 1ms
Bulk Files	Ten 5-MB files, each encoded in $W = 160$ blocks, $h = 32$ packets
Packet Size	1K bytes (both multicast and TCP)
DRR/DRR+	unicast quantum 1K bytes

Table 6.1: Parameters for the multicast with fair queueing experiments

transmits infinite-length data, utilizes reno TCP and originates from the corresponding (I) node. A shaded leaf node denotes a receiver which also functions as a representative who reports packet survival statistics for its subnet back to the multicast sender. Only at the links between (R) and (S) nodes where multicast traffic merges with local terrestrial traffic do we implement one of the following queueing policies: DropTail(FIFO), RED, DRR and DRR+. All other links use DropTail queues. For DRR and DRR+ queues, we can adjust the parameter w_m which is the multiple of 'quantum' the multicast flow receives per service round with respect to that of a unicast flow. In the results shown below, we set $w_m = 4$ which means that, when a service turn arrives, our active multicast flow receives four times of the service quantum of any other active unicast flow receives. Other simulation parameters are shown in Table 6.1.

The multicast sender can transmit at rate $r \in \mathcal{R} = \{512, 256, 128, 64\}$ Kbits/s. All the background TCP traffic starts at 0.1 seconds (virtual time). After 30 seconds, the multicast sender starts transmitting 2000 probe packets at each and every $r \in \mathcal{R}$ to initialize the packet survival probability estimators (the probe traffic is needed only once at startup, and in practice it can be part of the bulk file). After the probe transmission has ended (and subsequently after each bulk file transmission has ended) all the receiver representatives reliably report their packet survival statistics back to the multicast sender via the terrestrial feedback channels. Upon the reception of all the packet survival statistics, the multicast sender computes the rate schedule for the next file transmission which starts 5 minutes after the previous file transmission has ended – with an exception of the first bulk file which starts at 600 seconds into the simulation. The rate schedules are computed in such a way that theoretically (i.e. with both one-dimensional independent packet survival and time-invariant assumptions), the probability that each rate schedule provides unreliable delivery of any bulk file to a specific receiver would be at most 0.001 (in reality, these two assumptions are violated, for example by burst losses, thus we may see larger unreliability).

We consider the **bandwidth-delay trade off** in Figures 6.3 through 6.6. On the top of each of these plots and at each of the knob ϕ settings we used in the experiments, we present the **global reliability** which is the number of bulk files that were received completely at **all** of the receivers as a result of actuating the rate schedule alone (i.e. without utilizing any NACK mechanism). In the middle of these plots, and at each knob settings, we present the average **redundancy ratios**, which is defined as $(h + c)/h$ where c is the amount of master block's FEC parity packets, as the result of the rate schedule computation, along with the sample standard deviation from the ten bulk file transmissions. Since we use $h = 32$, if $(h + c)/h < 8$ then the multicast sender can use a RSE encoder based on the finite field GF(256), i.e. having 8-bit symbols, which is very common and fast. At the bottom of these plots, and at each knob setting, we show the average **file reception latency** of the files that are globally completed. The file reception latency at a specific receiver is defined as the elapsed time from the moment the receiver sees the start of a bulk file until the receiver receives sufficient packets (data or parity) to reconstruct the whole bulk file.

We also consider the **throughput dynamics** of the multicast and TCP traffic resulting from the queueing algorithms chosen for the (R)-(S) links (Figures 6.7 through 6.10). We choose to see the throughputs during the transmission of the 7th bulk file at knob setting $\phi = 0.1$ (i.e. emphasizing latency reduction rather than bandwidth reduction). In each of these figures, the 'raw' multicast and TCP throughputs as seen from observers Obs #1, #3, and #6 are shown on the top, middle and bottom plots respectively; each data point represents the average throughput in a 5-second interval immediately prior to and excluding the point.

We now describe our findings. First we notice an effective bandwidth-delay trade off in most of the results we show in Figures 6.3 through 6.6. However, in the use of DropTail queues at the traffic merging points, the global reliability is low (Figure 6.3) at knob settings around $\phi = 0.5 - 0.9$ (i.e. trying to minimize multicast bandwidth consumption rather than minimizing the reception latency). The main reason is that burst losses are more common in DropTail queues and thus the receivers connecting to the slowest (R)-(S) link (i.e. the 256 Kbits/s link) sometimes do not receive sufficient packets for some of the $W = 160$ data blocks. The use of RED, DRR, and DRR+ queues at the traffic merging points help in

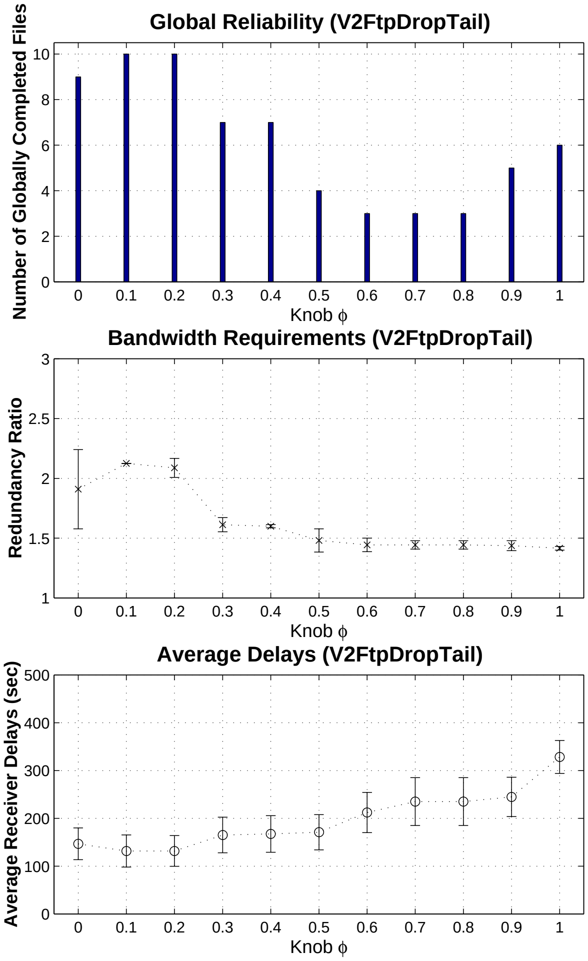


Figure 6.3: Multicast bandwidth v.s. delay when using DropTail queues

coping with this unreliability problem in different ways. **The RED queues helps reduce burst losses by probablistically dropping the packets. DRR has a mechanism to reserve a certain amount of bandwidth for a specific flow. The bursty effect of TCP flows is thus mostly blocked from interfering with the multicast flow and vise versa. DRR+ combines the benefits of both DRR and RED.**

Next, we look at the impact of different queueing algorithms on the traffic throughput dynamics. The use of the DropTail queues at the traffic merging

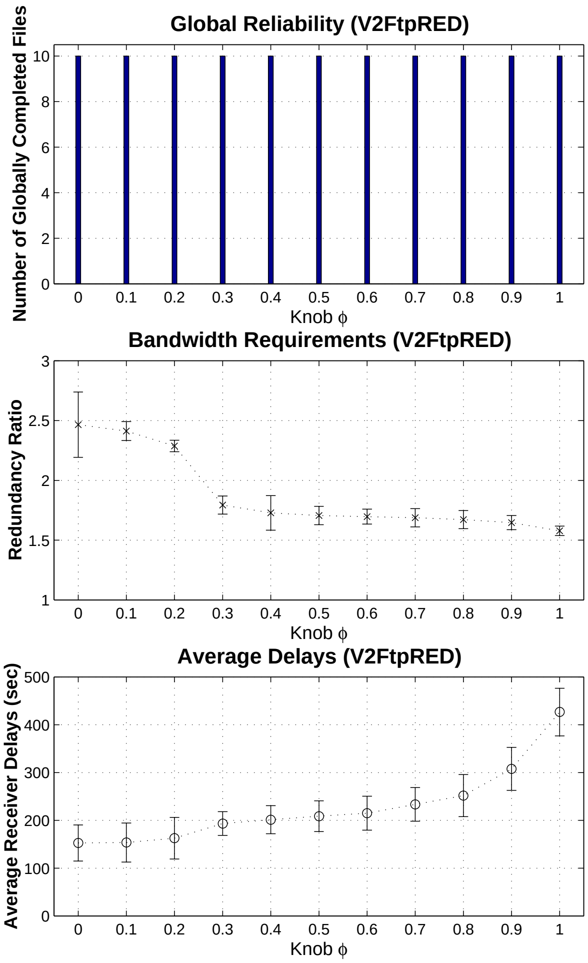


Figure 6.4: Multicast bandwidth v.s. delay when using RED queues

points cannot provide any protection for the TCP flow from being bandwidth-starved by the multicast flow, especially at observers (Obs) #3 and #6 (i.e. the middle and bottom plots of Figure 6.7 respectively). When compared to the DropTail queues, the RED queues allow a slightly smaller average amount of multicast bandwidth to be passed on to multicast receivers (Figure 6.8); this is due to the probabilistic nature of the packet drops. The use of RED queues at the traffic merging points also allows slightly quicker recovery of the TCP throughput at the slowest (R)-(S) links or Obs #6, i.e. comparing the bottom plots of

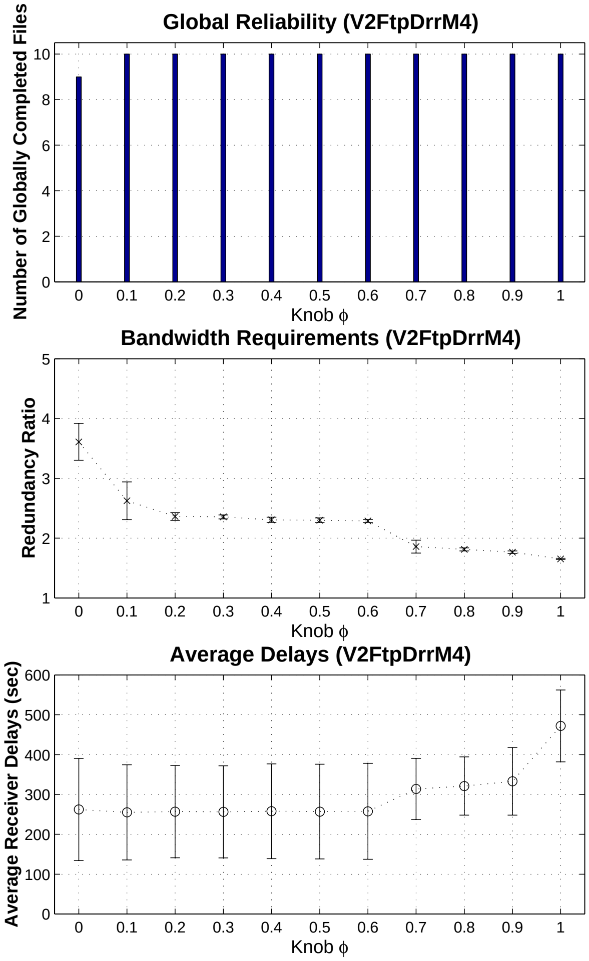


Figure 6.5: Multicast bandwidth v.s. delay when using DRR queues (w_m=4)

Figure 6.7 and Figure 6.8. The former shows that it is not until the multicast traffic is being transmitted at rate 128 Kbits/s by the multicast sender that the TCP traffic begins to recover, while the latter shows that TCP traffic starts to recover 'sooner' when the multicast traffic is being transmitted at rate 256 Kbits/s. However, the use of RED queues does not completely solve the TCP starvation problem. **The use of the DRR queues at the traffic merging point gives better protection for TCP** (Figure 6.9). For the 'fastest' (R)-(S) links (i.e. at observer Obs #1), DRR even gives a 'smoother' TCP throughput

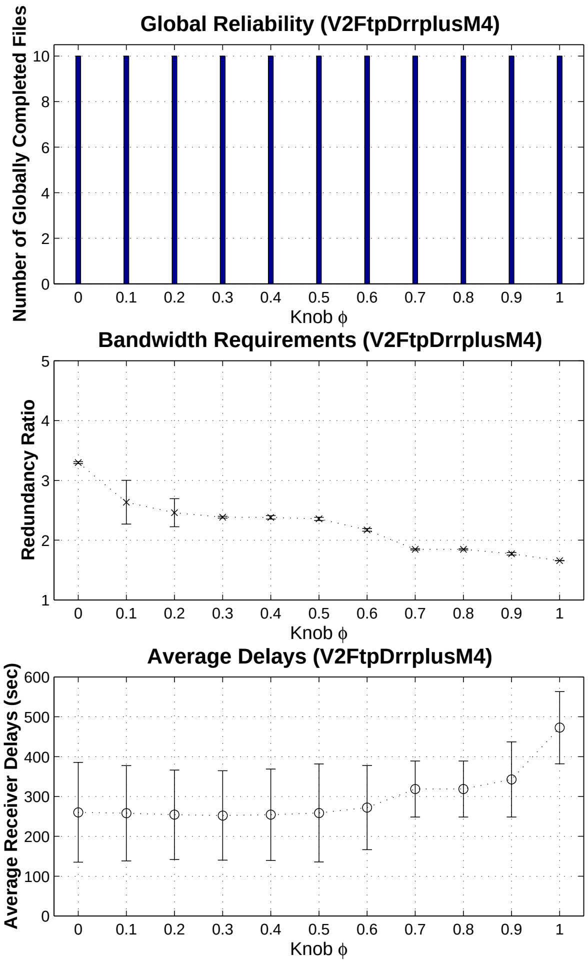


Figure 6.6: Multicast bandwidth v.s. delay when using DRR+ queues (w_m=4)

than the DropTail's counter part (i.e. the topmost plot of Figure 6.9 vs. the topmost plot of Figure 6.7). However at the slowest link (i.e. at observer Obs #6, or the bottom plot of Figure 6.9), we still notice a recurring pattern of short TCP starvations and restarts. **DRR+ almost completely eliminates this TCP starvation problem** (Figure 6.10). In fact, the DRR+ algorithm, like the RED algorithm, provides an operating environment which is closet to the independent packet survival assumption we used for the rate scheduling mechanism.

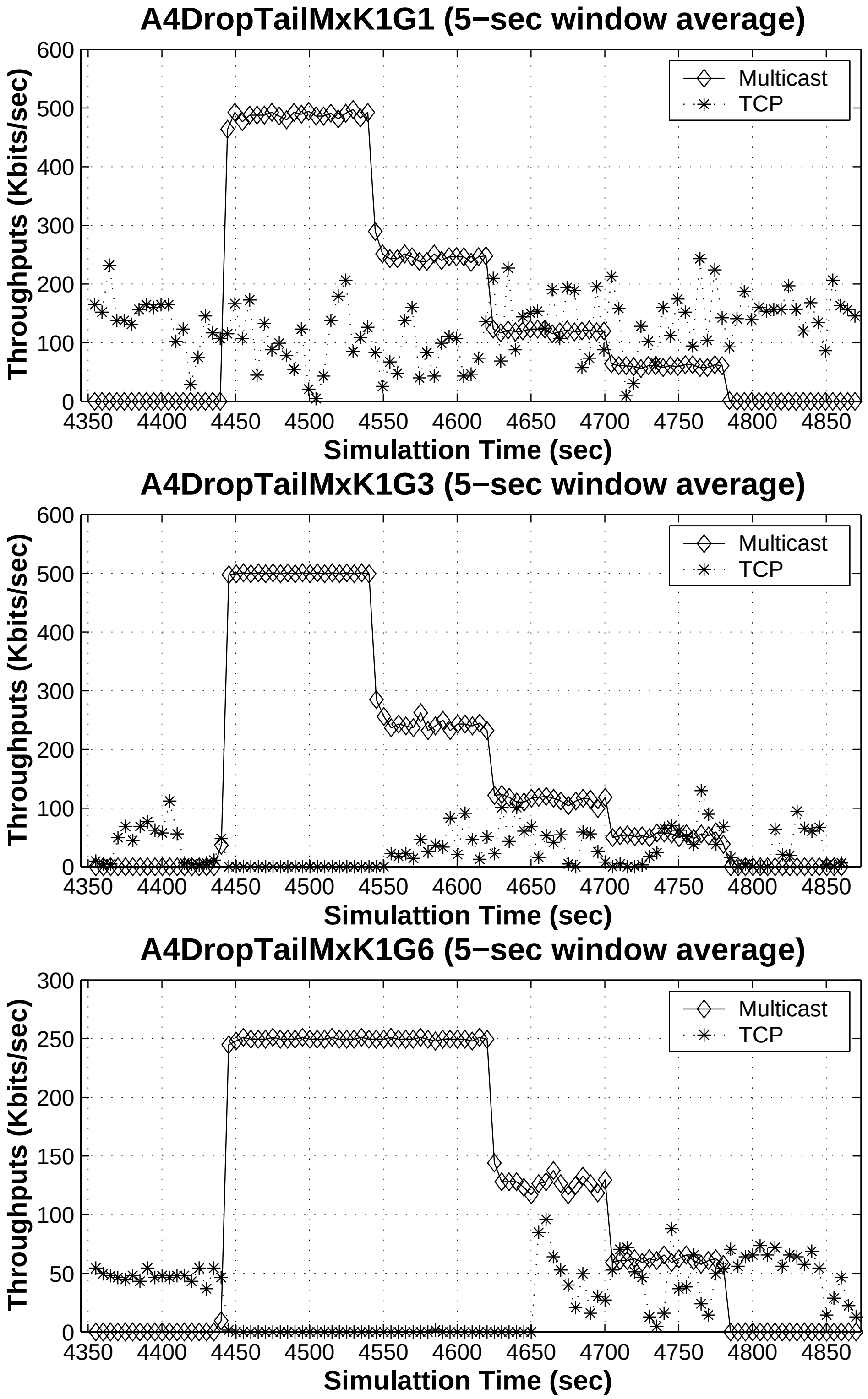
We also experimented with different w_m settings for DRR/DRR+ queueing algorithms. If $w_m = 1$, the multicast flow receives no more bandwidth share than any other unicast flow sharing the same (R)-(S) link. In the topology that we use, however, we found that the rate scheduling algorithm is less effective in providing bandwidth-delay trade off because most of the multicast receivers do not benefit from the multicast transmission rates 512 Kbits/s and 256 Kbits/s. This suggests that the choice of rates in the set $\mathcal{R}$ is also important. A solution to this problem can be realized by either using an entirely different set of multicast transmission rates $\mathcal{R}$ or by using some dynamic mechanisms to re-adjust the members of $\mathcal{R}$. The multicast sender can generally learn if the rate member in the set $\mathcal{R}$ is appropriate or not by looking at the probed packet survival probabilities $q_r^{(m)}$ at each rate $r \in \mathcal{R}$ and at every receiver $m \in \mathcal{M}$.

Lastly, for DRR/DRR+ queueing algorithms, it is also possible that one can administratively set different w_m values for different subnets according to some local fairness measures, but we do not cover this issue in this dissertation.

6.3 Chapter Summary

In this chapter, we have studied a simple way in allowing the multicast rate control that was proposed in the previous chapter to co-exist with terrestrial background TCP traffic. Through the use of fair queueing algorithms at the traffic merging points, we show through simulations that both the satellite-based multicast traffic and the terrestrial background traffic can co-exist. Between the two fair queueing algorithms that we tested, the DRR and the DRR+, our prefered choice is DRR+ ([HMMM00]). This is partly due to the random early drop (RED) mechanism incorporated in the DRR+ queueing that makes packet losses appear more randomized. The DRR+ is the most effective queue to help reduce TCP traffic's fluctuations. This helps maintain the packet survival stationarity assumption required by the multicast rate control mechanism.

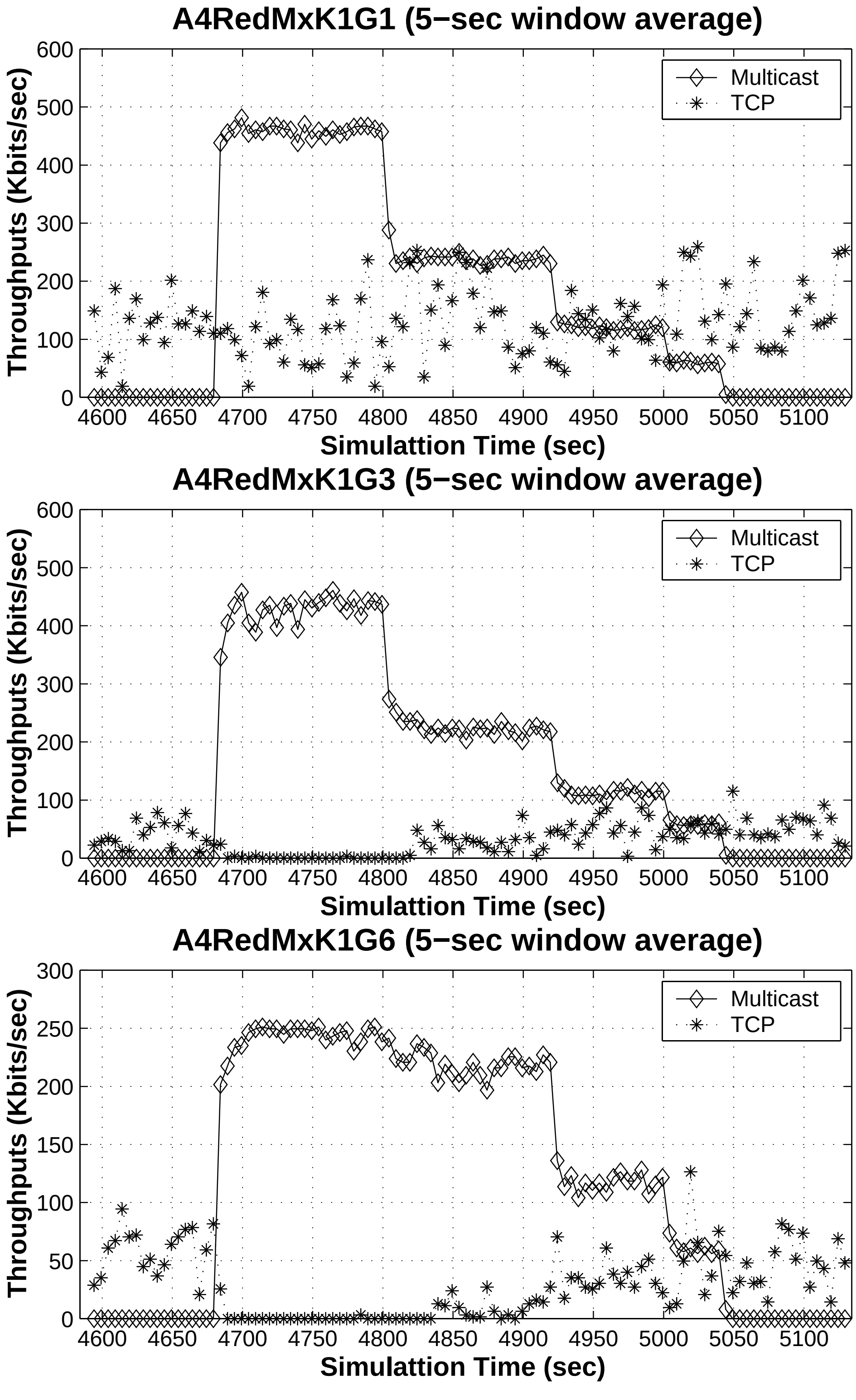
By putting a fair queueing algorithm at the traffic merging points in a hybrid satellite-terrestrial network, the abstraction of the multicast rate control problem does not change. Therefore we can effectively use the multicast rate control mechanism that we proposed in the previous chapter.


A4DropTailMxK1G1 (5−sec window average)

A4DropTailMxK1G3 (5−sec window average)

A4DropTailMxK1G6 (5−sec window average)

Figure 6.7: Throughput dynamics at observers 1, 3, and 6 when using DropTail queues (knob $\phi = 0.1$)


A4RedMxK1G1 (5−sec window average)

A4RedMxK1G3 (5−sec window average)

A4RedMxK1G6 (5−sec window average)

Figure 6.8: Throughput dynamics at observers 1, 3, and 6 when using RED queues (knob $\phi = 0.1$)

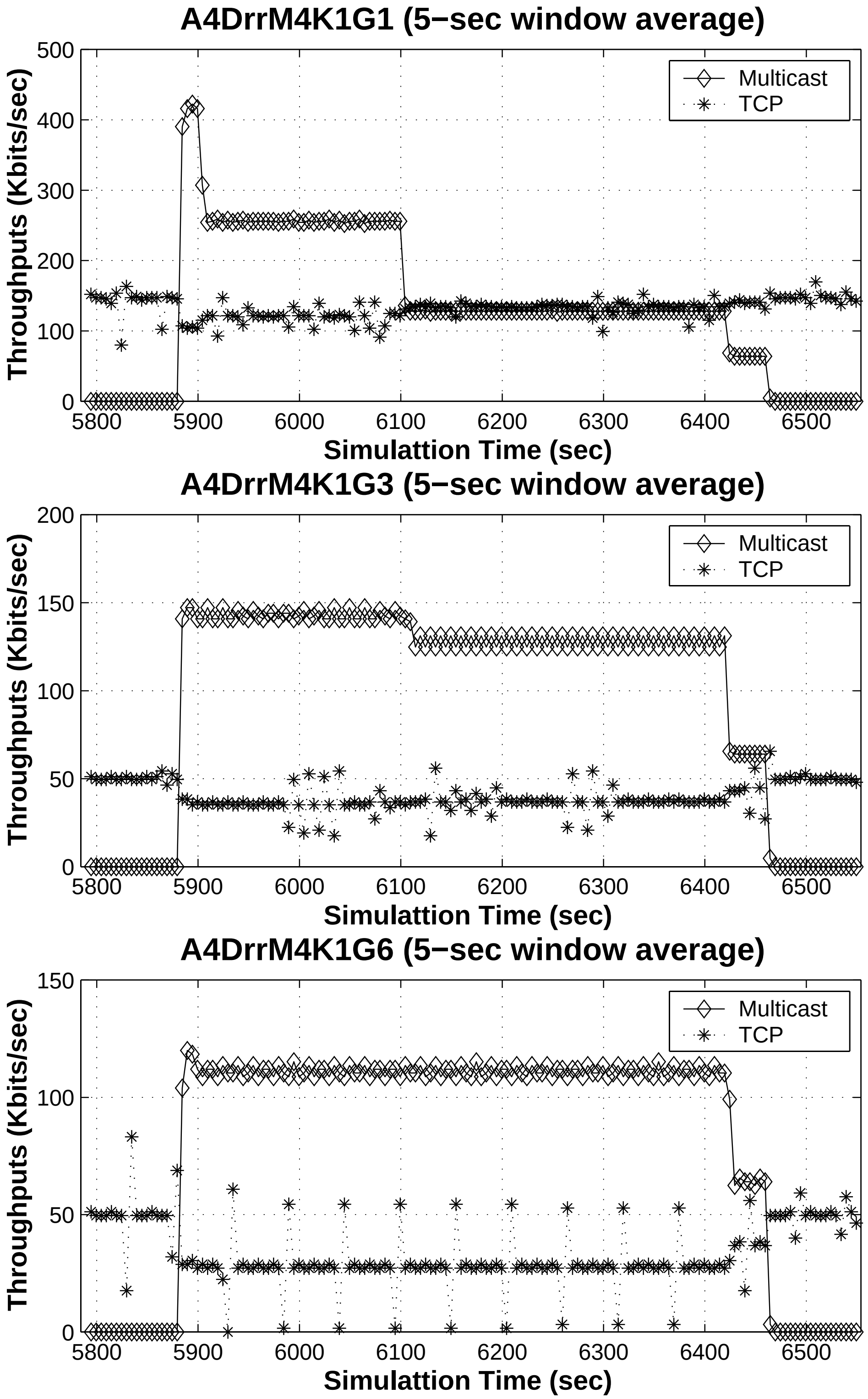


Figure 6.9: Throughput dynamics at observers 1, 3, and 6 when using DRR queues with w_m=4 (knob $\phi = 0.1$)

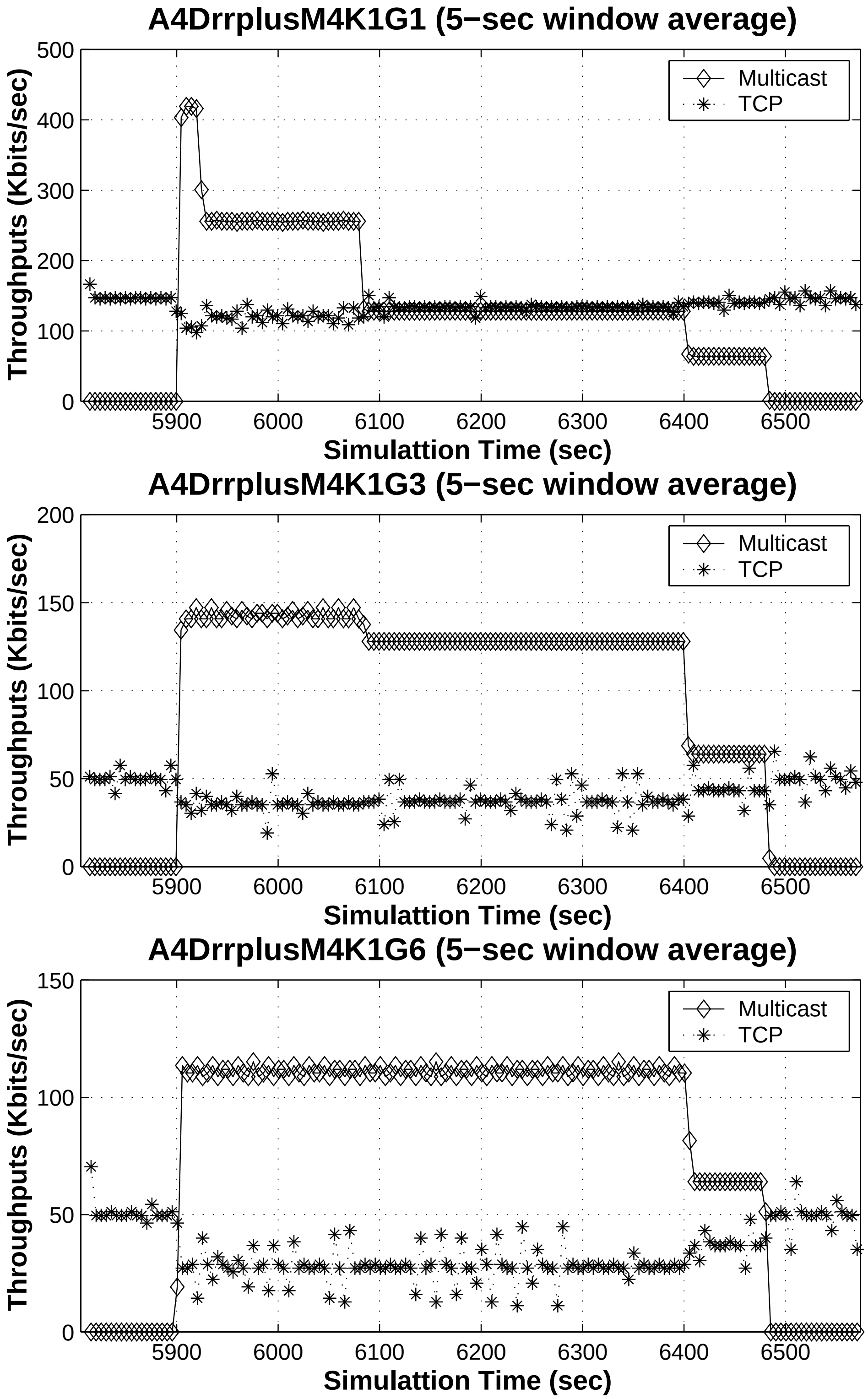


Figure 6.10: Throughput dynamics at observers 1, 3, and 6 when using DRR+ queues with w_m=4 (knob $\phi = 0.1$)

Chapter 7

Two-Staged Multicast Rate Control

In previous chapters, the multicast sender took all the responsibility in computing the multicast rate schedule. Scalability, partly achieved through the use of receiver representatives, would still be limited. In this chapter we propose a way to deploy the Gaussian approximation approach for the multicast rate control problem in a distributed manner. To achieve greater scalability, we will need supports and computational powers from elements in the terrestrial networks.

7.1 A Logical View of The System

From the experiments conducted in Chapter 3, we observeed that although the satellite channel causes packet losses, the loss rate is not significantly affected by the sender's transmission rate. This suggests that the satellite channel, when served as a network backbone, still has sufficient available capacity. To allow better scalability, our rate control system will work in a **two-staged** fashion. The logical view of our system is shown in Figure 7.1.

The sender will transmit the bulk file at some fixed constant rate $r_s = \max r \in \mathcal{R}$. In this chapter, we let $r_s = 512$ Kbits/s. Each earth station will

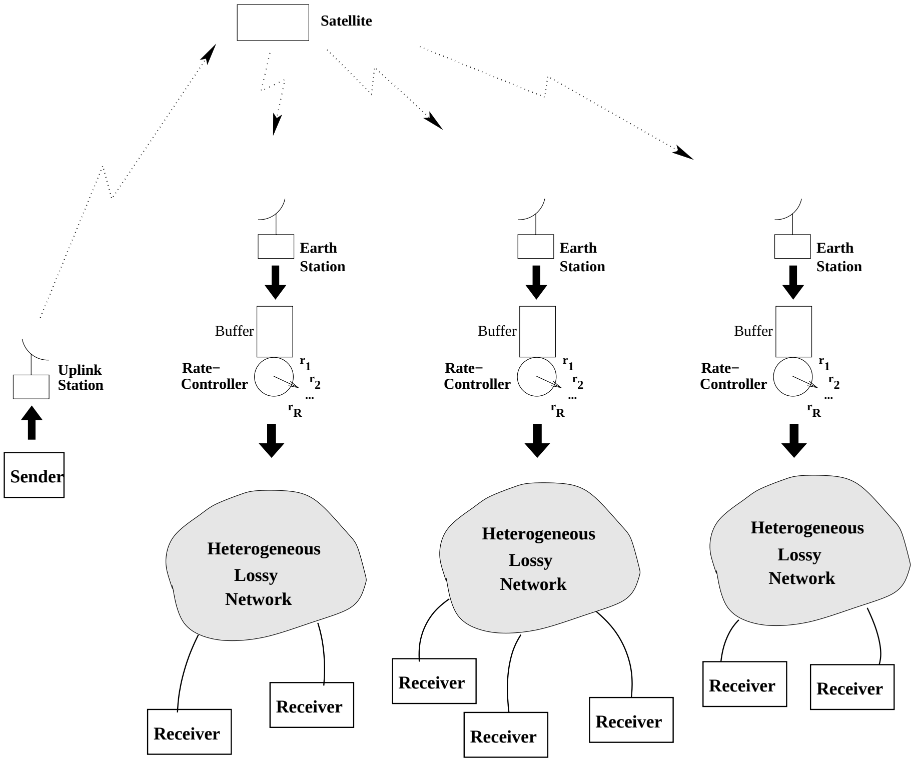


Figure 7.1: A logical view of a two-staged rate control system.

now have the ability to control the rate of the multicast traffic that passes through it. Each earth station **does not** have the FEC decoding and encoding capability, but it has a sufficiently large, finite buffer (i.e. to the order of the bulk file size) when rate adjustments are needed. This comes under the impression that the sender's fast transmission rate, that may satisfy many of the fast receivers, would require to be slowed down for other slower receivers. The earth station is now responsible for computing the rate schedule which is deemed suitable for its downstream receivers.

7.2 The Earth Stations and Receivers

We assume that there are G earth stations, each labeled distinctly by $g \in \mathcal{G} = \{1, 2, \ldots, G\}$, receiving multicast traffic directly from the satellite. There are a total of M distinct receivers, not necessarily homogeneous, each distinctly labeled by $m \in \mathcal{M} = \{1, 2, \ldots, M\}$ receiving multicast traffic from its uniquely associated parent earth station.

For simplicity, we assume that when a packet is transmitted by the sender at rate r_s, the packet survives and reaches an earth station g as a independent Bernoulli (0,1) random outcome[1] with survival probability $\zeta_{r_s}^{(g)}$. We assume that $0 < \zeta_{r_s}^{(g)} \leq 1$. The earth station g has two possible actions:

1. **Patches through** the sender's traffic. That is the earth station transmits at some rate $r_e = \zeta_{r_s}^{(g)} r_s$ where r_e needs not be in $\mathcal{R}$.
2. **Lowers the transmission rate** to some rate $r_e \in \mathcal{R} - \{r_s\}$ such that $r_e < \zeta_{r_s}^{(g)} r_s$ with the help of the buffering mechanism.

Also, we assume that packet survivals in the satellite portion of the network are independent from those in the terrestrial network. Let $\Omega(g) \subset \mathcal{M}$ be the set of downstream receivers associated with the earth station g. When the earth station g chooses to transmit the multicast packet at some rate r_e (i.e. either $r_e = \zeta_{r_s}^{(g)} r_s$ or $r_e < \zeta_{r_s}^{(g)} r_s$ and $r_e \in \mathcal{R}$), the packet survives and reach each receiver $m \in \Omega(g)$ as an independent Bernoulli (0,1) random outcome with survival probability $\rho_{r_e}^{(m)}$ such that $0 < \rho_{r_e}^{(m)} \leq 1$.

From the perspective of the sender, when the earth station g chooses to transmit at some rate $r_e \leq r_s$, each multicast packet independently survives and

[1] 'One' denotes a survival.

reaches a specific receiver $m \in \Omega(g)$ as an independent Bernoulli (0,1) random variable with **end-to-end packet survival probability**:

$$q^{(g,m)}_{r_s,r_e} = \zeta^{(g)}_{r_s}\ \rho^{(m)}_{r_e}. \tag{7.1}$$

From the bounds of $\zeta^{(g)}_{r_s}$ and $\rho^{(m)}_{r_e}$ above, we also have that

$$0 < q^{(g,m)}_{r_s,r_e} \leq 1. \tag{7.2}$$

We assume that these probabilities are relatively time-invariant over the scheduling period. This time-invariant condition is generally achieved when the packet survival in the satellite channel exhibits mean stationarity and when terrestrial background traffic remains unaltered.

Figure 7.2 illustrates an example probabilistic multicast tree model that we assume.

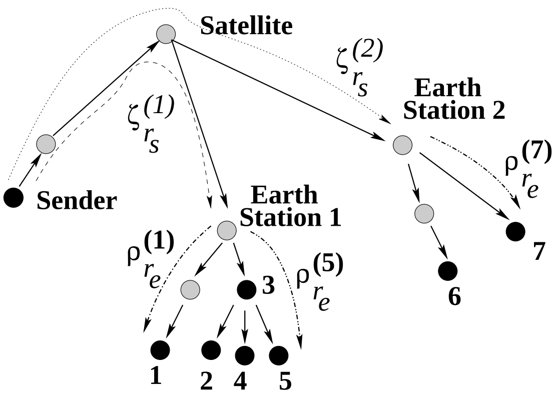


Figure 7.2: A two-staged probabilistic multicast tree model (with two earth stations and seven receivers).

In practice, we can estimate $\zeta^{(g)}_{r_s}$, $\rho^{(m)}_{r_e}$, and $q^{(g,m)}_{r_s,r_e}$ by using probe traffic and unbiased estimators. The earth station g can directly observe from the probe traffic to estimate $\zeta^{(g)}_{r_s}$. Also the earth station g may *patch through* the multicast traffic, so that each end receiver $m \in \Omega(g)$ can directly estimate the **patched-through packet survival probability** $q^{(g,m)}_{r_s,*}$ (i.e. there is no need to infer r_e in this case). On the other hand, the earth station may transmit the probe traffic at each and every rate $r_e \in \mathcal{R} - \{r_s\}$ so that each downstream receiver m can estimate $\rho^{(m)}_{r_e}$. When the downstream receiver reports this value to its parent earth station, the earth station can estimate the end-to-end packet survival probability $q^{(g,m)}_{r_s,r_e}$ by using (7.1).

7.3 The Gaussian Approximation Approach for the Two-Staged Scenario

We have introduced the Gaussian approximation approach in earlier chapters. We now show that there is a concise geometrical interpretation of the two-staged rate scheduling problem on the mean-variance (μ, σ^2) plane. There is also a distributed way to compute the rate schedule. Since the multicast sender is assumed to transmit at only one particular rate r_s, it is easy to track the reception status of the earth stations on the mean-variance (μ, σ^2) plane as illustrated in Figure 7.3.

Each earth station g is now responsible for ensuring the reliable receptions of its downstream receivers $m \in \Omega(g)$. Now the multicast rate control problem at the earth station reduces to a subproblem which is similar to the end-to-end rate control problem described in Chapter 5, with the following modifications:

1. The slope of the line segment representing the progress of any specific end receiver m which has parent earth station g will be bounded from below by $1 - \zeta^{(g)}_{r_s}$ which the slope of the parent earth station status shown in Figure 7.3.

Instead of (5.19), we now have:

$$1 - \zeta_{r_s}^{(g)} \leq \Delta_{A^{(m)}(k)} < 1. \tag{7.3}$$

We illustrate this in Figure 7.4.

2. The cost function to be minimized is now local to the scope of the earth station's downstream receivers only. This however, still has a global impact on both the satellite bandwidth requirement and the receiver aggregate delivery delay, depending on the knob ϕ setting.

3. Once the earth station finishes computing a rate schedule, the earth station reports the total number $h + c$ of data and parity packets required per block back to the sender. The sender must find the maximum of such numbers across all the earth stations, and transmit additional parity packets per block correspondingly.

We are interested in multi-step solutions, specifically the solutions that utilize all the rate $r \in \mathcal{R}$ in decreasing order. The computation of a multi-step schedule at each earth station can still use the approach based on **depth-first search with branch-and-bound technique** described in Section 5.5. The notion of **receiver representatives** is still valid. The new lower- and upper- bounds on the unnormalized bandwidth UBW (5.20) can be found from the end-to-end packet survival probabilities $q_{r_s,r_e}^{(g,m)}$. This information is readily available at each earth station. Similarly the new lower- and upper- bounds on the unnormalized aggregate delivery delay UAD can be obtained independently at each earth station. **Scalability is now significantly improved because all the earth stations can independently solve the multicast rate control subproblem in parallel.**

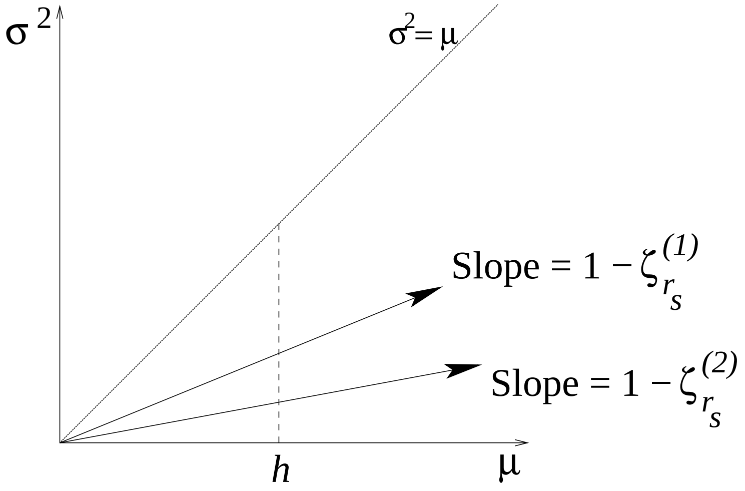


Figure 7.3: Tracking the reception status of the earth stations on the mean-variance (μ, σ^2) plane. An example case for two earth stations

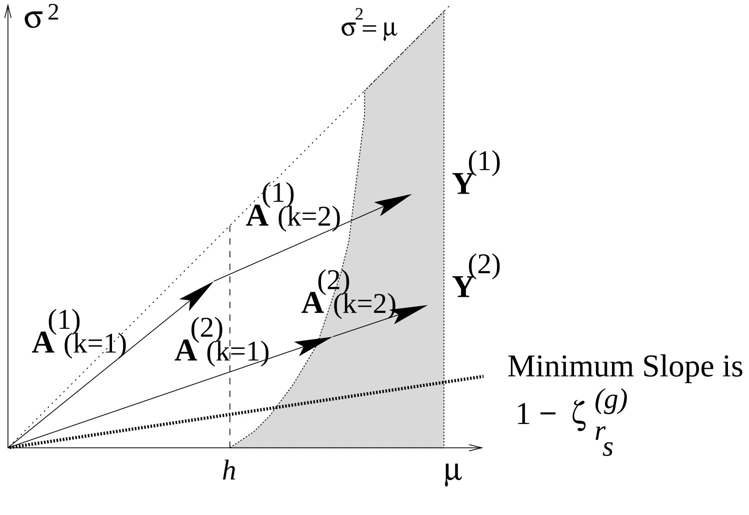


Figure 7.4: The slopes of the line segments representing receivers' reception status in the two-staged scenario

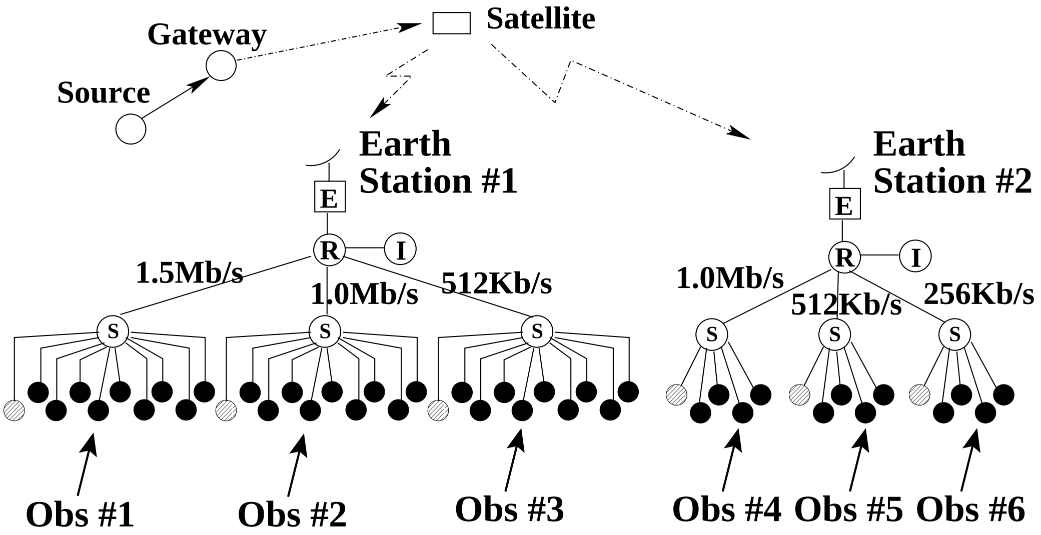


Figure 7.5: Topology used in the two-staged experiments

(E) = an earth station, (R) = a router, (I) = an Internet node (source of TCP),

(S) = a subnet router

7.4 Two-Staged Experiments and Results

In this section we validate our approach by simulations. To be more realistic, we extend the *ns-2* [NS2] simulator so that we can **drive the simulations using some of the real satellite traces** that we collected and analyzed in Chapter 3. We use the topology shown in Figure 7.5 which is the same as the one that we used in the previous chapter. Similarly, we only consider reno TCP background traffic (in the form of long running FTP applications). We use the DRR+ queues [HMMM00] at the traffic merging points because our experiments in the previous chapter indicate that DRR+ is a better choice. The experiment parameters are shown in Table 7.1. Table 7.2 shows the traces specifically used in our trace-driven simulations.

Parameters	Values and Notes
Satellite Channel	$C = 1024$ Kbits/s, delay=270ms
Multicast Rates	$\mathcal{R} = \{512, 256, 128, 64\}$ Kbits/s
(E)-(R) links	512 Kbits/s, 5ms
(R)-(S) links	bandwidths as in Figure 7.5, 5ms
(I)-(R) links	40Mbps, 20ms
(S)-(Receiver) links	10Mbps, 1ms
Bulk Files	Four 5-MB files, each encoded in $W = 160$ blocks, $h = 32$ packets
Packet Size	TCP 1024 bytes; Multicast 1052 bytes
DRR+ flow quantum	TCP 1024; Multicast 5× of TCP

Table 7.1: Parameters for the two-staged rate control experiments

Trace Date	Trace Figure	Results in Figures
August 08, 2001	A.27	7.6, 7.10, 7.14
August 09, 2001	A.31	7.7, 7.11, 7.15
August 17, 2001	A.38	7.8, 7.12, 7.16
August 27, 2001	A.45	7.9, 7.13, 7.17

Table 7.2: Trace-driven simulations and their results

Each simulation consists of the following steps:

1. The sender transmits probe packets at rate $r_s = 512$ Kbits/s for 180 seconds. Each earth station *puts through* all the surviving sender's probes to its downstream receivers.

2. Each earth station starts transmits its own probes at each rate $r_e \in \{256, 128, 64\}$ Kbits/s for 180 seconds.

3. Each earth station g can now determine the unbiased estimate of $\zeta_{r_s}^{(g)}$. Similarly each receiver can find the unbiased estimate of $\rho_{r_e}^{(m)}$ for every $r_e \in \mathcal{R}$. Each receiver m reports the estimate of $\rho_{r_e}^{(m)}$ to its parent earth station.

4. Each earth station infers the estimate of $q_{r_s,r_e}^{(g,m)}$ and then computes a multi-rate schedule for its downstream receivers. The solution of the rate schedule specifies how many (master block) packets should be put through and how many should be transmitted at the other reduced rates $r_1 > r_2 > \ldots > r_{|R-1|}$

5. Each earth station reports the number of the distinct packets required per data block to the sender.

6. The sender finds the maximum of such numbers, and transmits the bulk file accordingly using rate $r_s = 512$ Kbits/s.

7. Each earth station actuates its previously computed rate schedule and determines how long it should pass through and slow down the traffic at each rate. The earth station may in fact cut off the transmission before the sender finishes if all the downstream receivers receive the file completely.

8. The same rate schedule is re-used for the next bulk file (i.e. without a new computation). The next bulk file is assumed to be ready for transmission

after an amount of time has passed (i.e. there is only one active bulk file in the multicast system at a time).

We consider a number of performance-related aspects. First, we consider the **global reliability** which is the number of files that were completely received at *all* the receivers due to the multicast rate schedule alone. All the experiments here produce 100% global reliability as shown on the top of Figures 7.6 through 7.9.

Second, we consider the **satellite bandwidth**. This is related to the multicast sender's response to the maximum of the numbers of data plus parity packets requested by all the earth stations. We show the satellite bandwidth as the **redundancy ratios** of $(h + c)/h$ in the middle of Figures 7.6 through 7.9. The closer to the value of one (1) indicates more saving on the satellite bandwidth requirement.

Third, like the end-to-end scenario, we consider the **average file reception latency**. This is the global average of the elapsed times between the moment that each receiver sees the start of a new file until the moment that the receiver completes the reception of such file. We show the average file reception latency at the bottom of Figures 7.6 through 7.9.

Fourth, specifically for the two-staged scenario, we additionally consider the **multicast bandwidth at each earth station**. For all the bulk files transmitted by the multicast sender (not including the probe traffic), we count the number of satellite multicast packets that each earth station receives *and* forwards to its downstream receivers. Note that **the earth station may not receive every packet that the multicast sender transmits** because of losses indicated in the satellite traces, and that **not every multicast sender's packet may get forwarded** to downstream receivers if the rate schedule computed and actuated at the earth station does not require so. The results are shown in Figures 7.10 through 7.13. These plots clearly show the impact of the local multicast rate schedule at different knob ϕ settings. Earth station #1 has relatively 'faster' receivers in that their links have higher capacities. We see that when we set the knob $\phi \to 0$ where we wish to minimize reception delay, the multicast rate control at this earth station allows significantly more multicast packets to pass through at higher rates. Obviously this has a partial impact on reducing the overall average file reception latency as seen in Figures 7.6 through 7.9.

By having two control stages for satellite multicast, the earth station may block certain excess multicast traffic deemed not necessary to the earth station's downstream receivers thus allowing better use of terrestrial network bandwidth. Further refinements, such as having a different knob ϕ setting for each earth station, are also possible, but we have not studied them here.

Fifth, we consider the throughput dynamics of both the multicast flow and the TCP flow on the potentially congested (R)-(S) links which utilize DRR+ queues. The results shown in Figures 7.14 through 7.17 reveal that the DRR+ queues perform well to protect TCP flows.

7.5 Chapter Summary

In this chapter, we have proposed a scalable two-staged multicast rate control system that utilizes the Gaussian approximation approach. We verify our approach by trace-driven simulations, using real satellite traces. We observe that the two-staged multicast rate control can effectively control multicast bandwidth

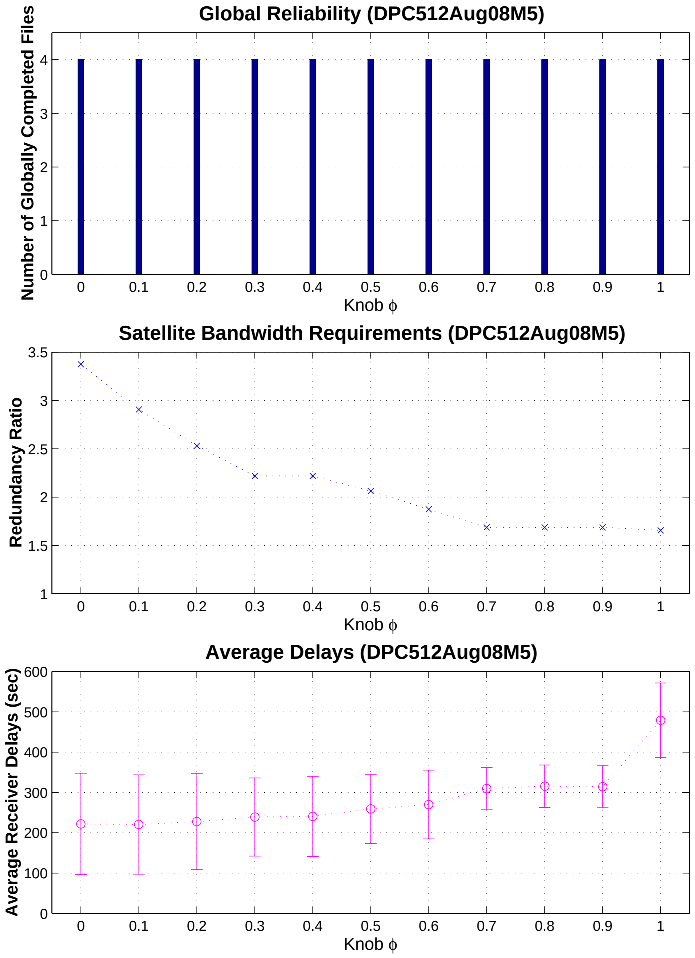


Figure 7.6: Satellite multicast bandwidth v.s. receiver delay results, based on the 512Kbits/s satellite trace of August 08, 2001

usages (i.e. both on the satellite link and in terrestrial networks), as well as the receiver reception delays. At the same time, the two-staged multicast rate control also provides very good global reliability. By distributing the schedule computation to the earth stations, the two-staged multicast rate control provides significantly improved scalability.

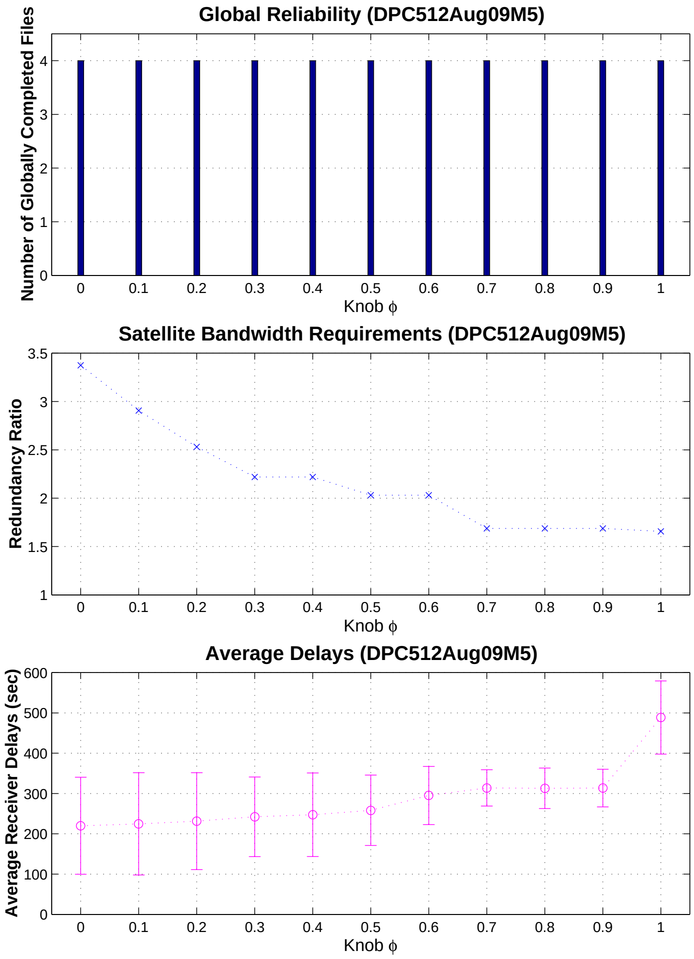


Figure 7.7: Satellite multicast bandwidth v.s. receiver delay results, based on the 512Kbits/s satellite trace of August 09, 2001

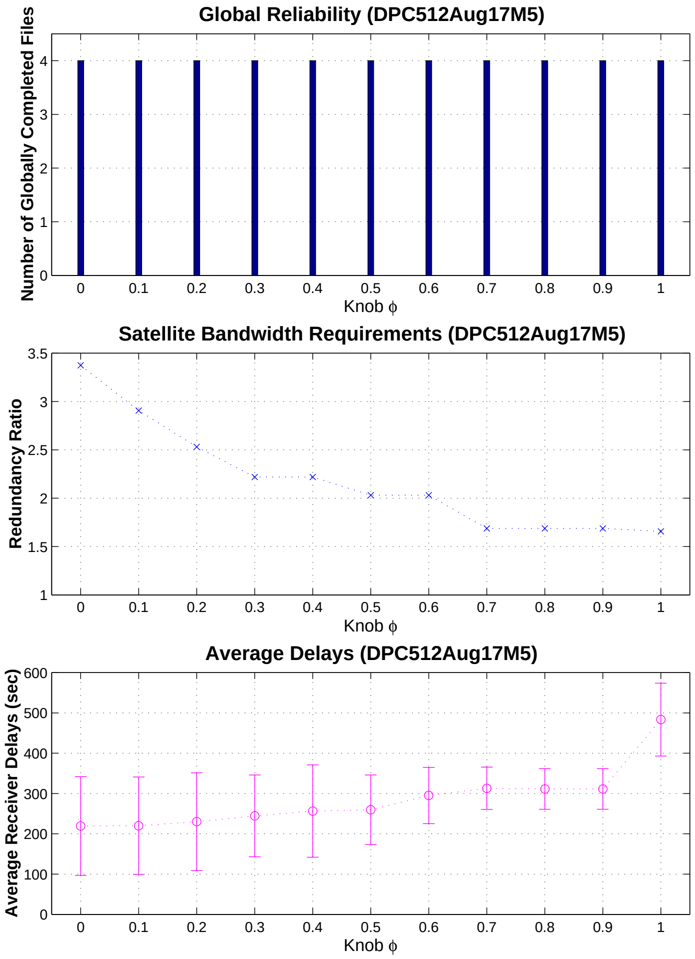


Figure 7.8: Satellite multicast bandwidth v.s. receiver delay results, based on the 512Kbits/s satellite trace of August 17, 2001

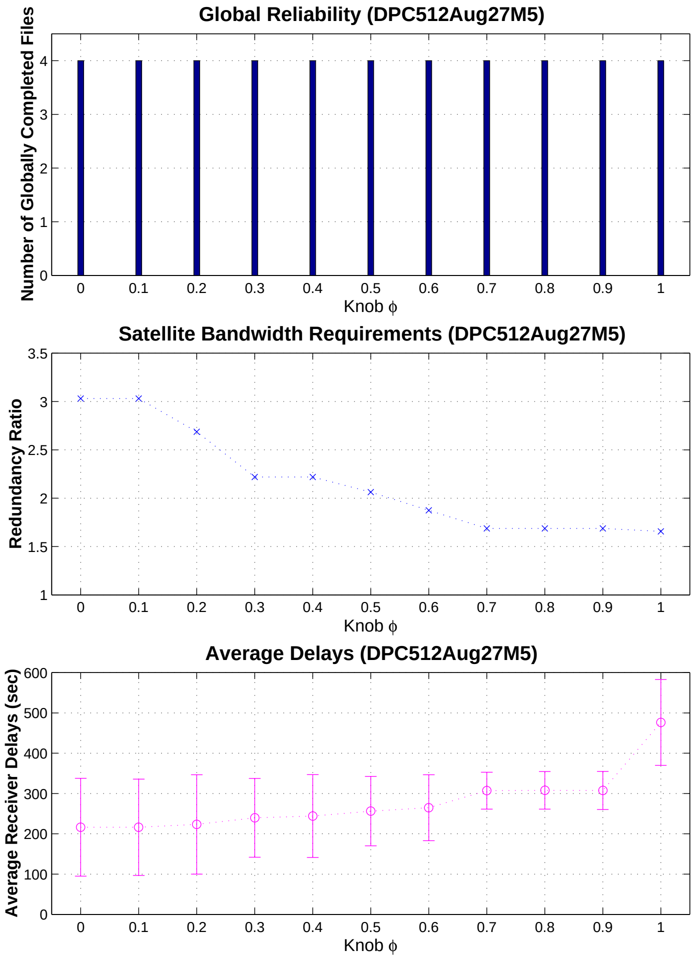


Figure 7.9: Satellite multicast bandwidth v.s. receiver delay results, based on the 512Kbits/s satellite trace of August 27, 2001

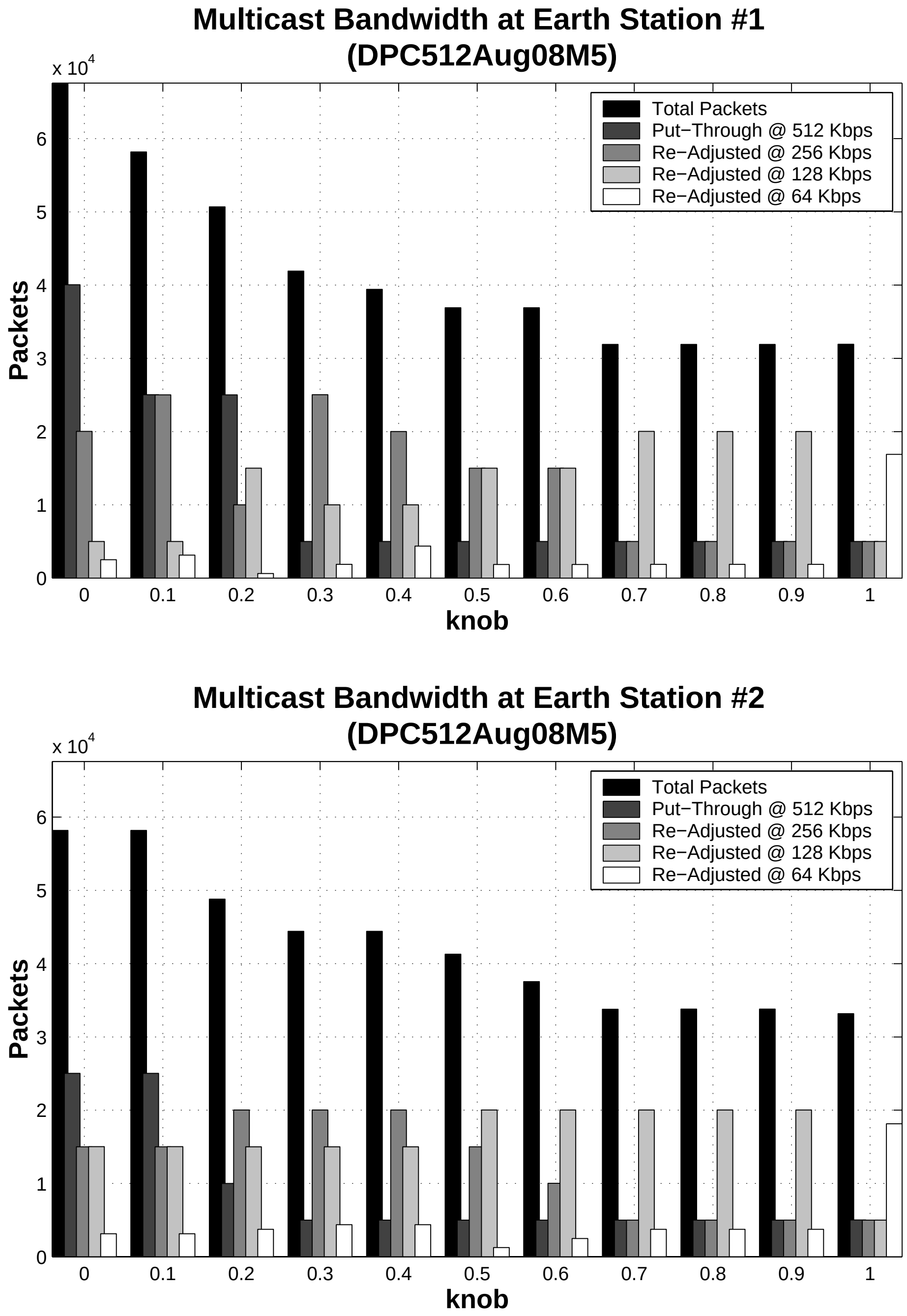


Figure 7.10: Earth station multicast bandwidth, based on the 512Kbits/s satellite trace of August 08, 2001

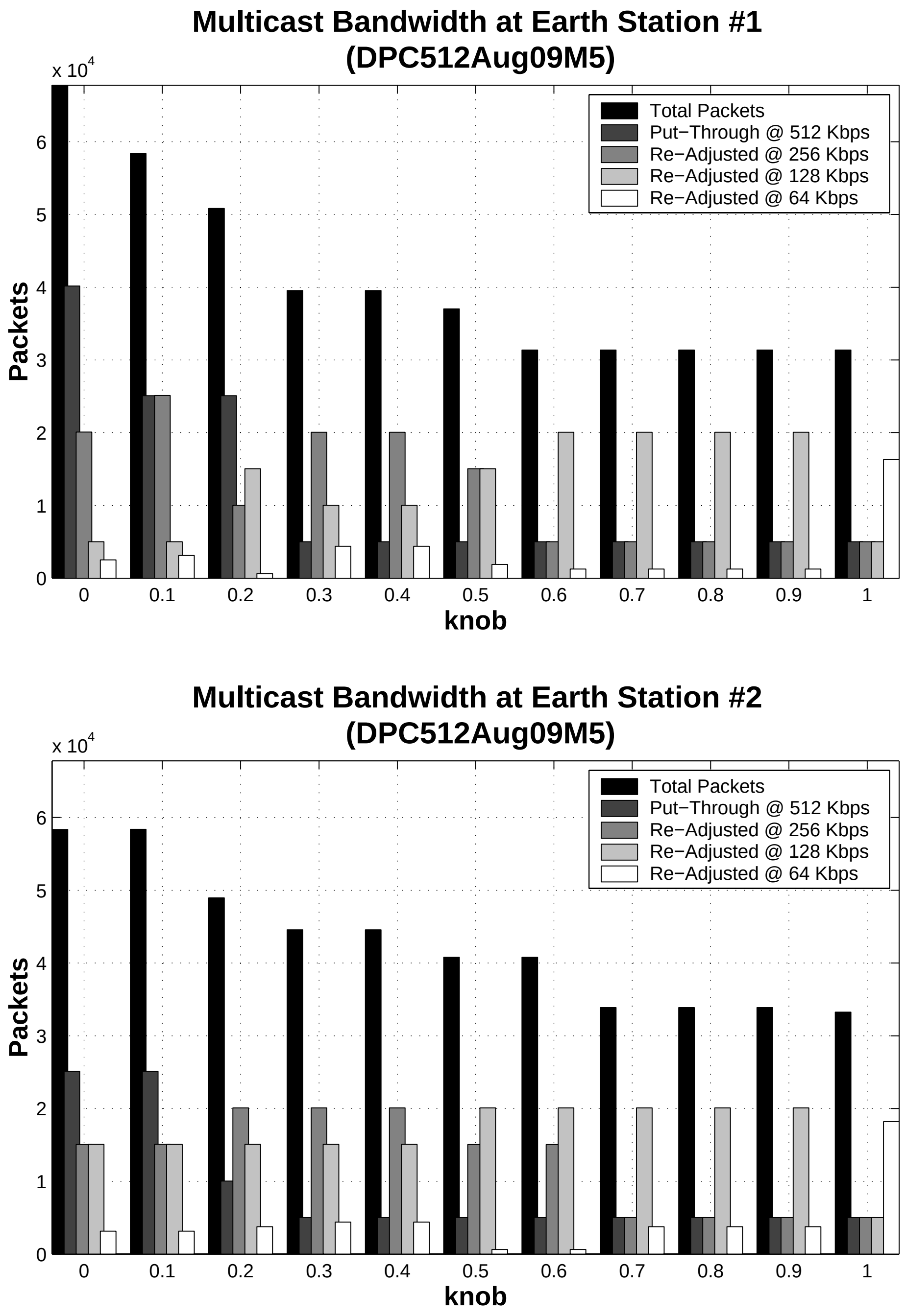


Figure 7.11: Earth station multicast bandwidth, based on the 512Kbits/s satellite trace of August 09, 2001

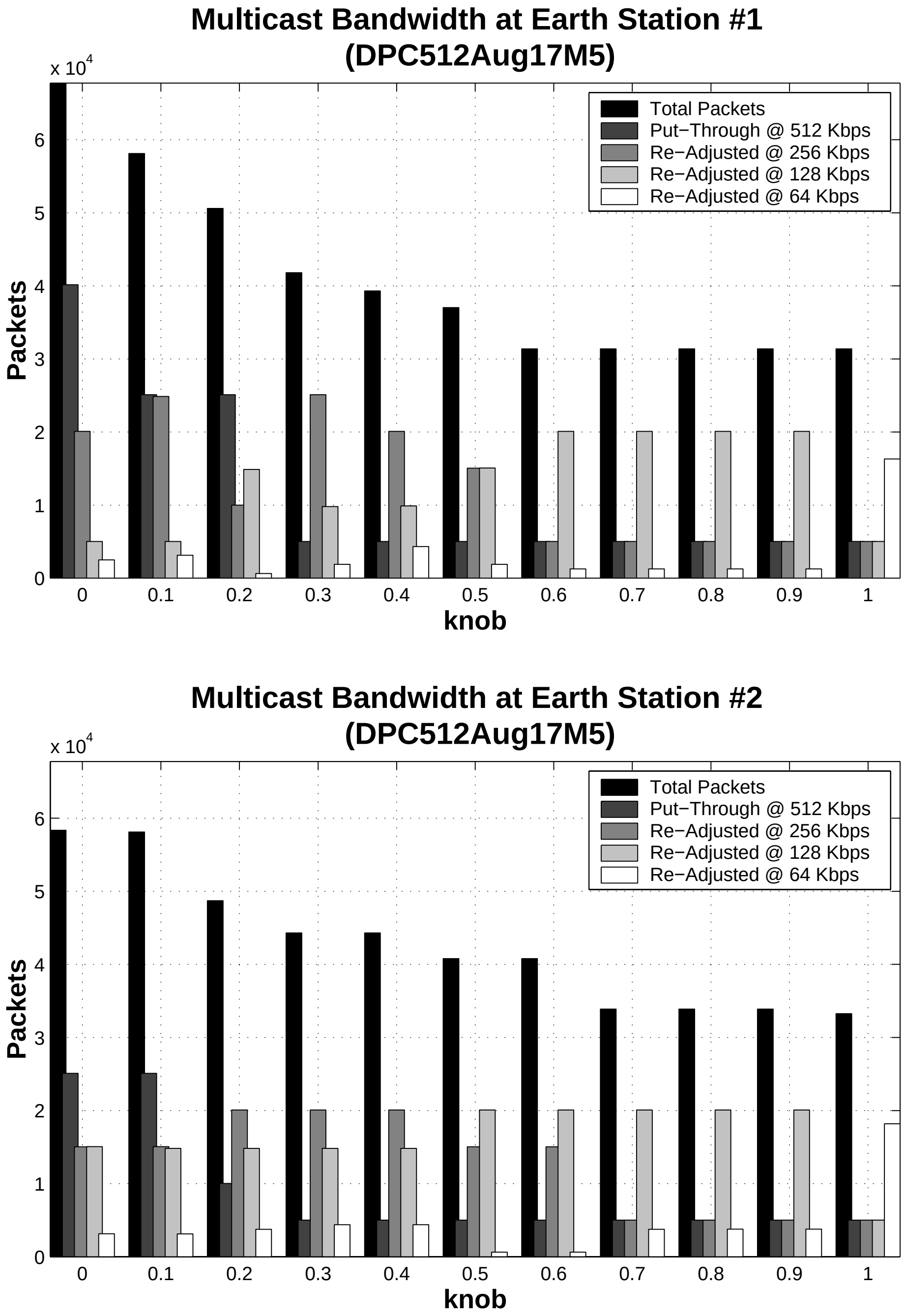


Figure 7.12: Earth station multicast bandwidth, based on the 512Kbits/s satellite trace of August 17, 2001

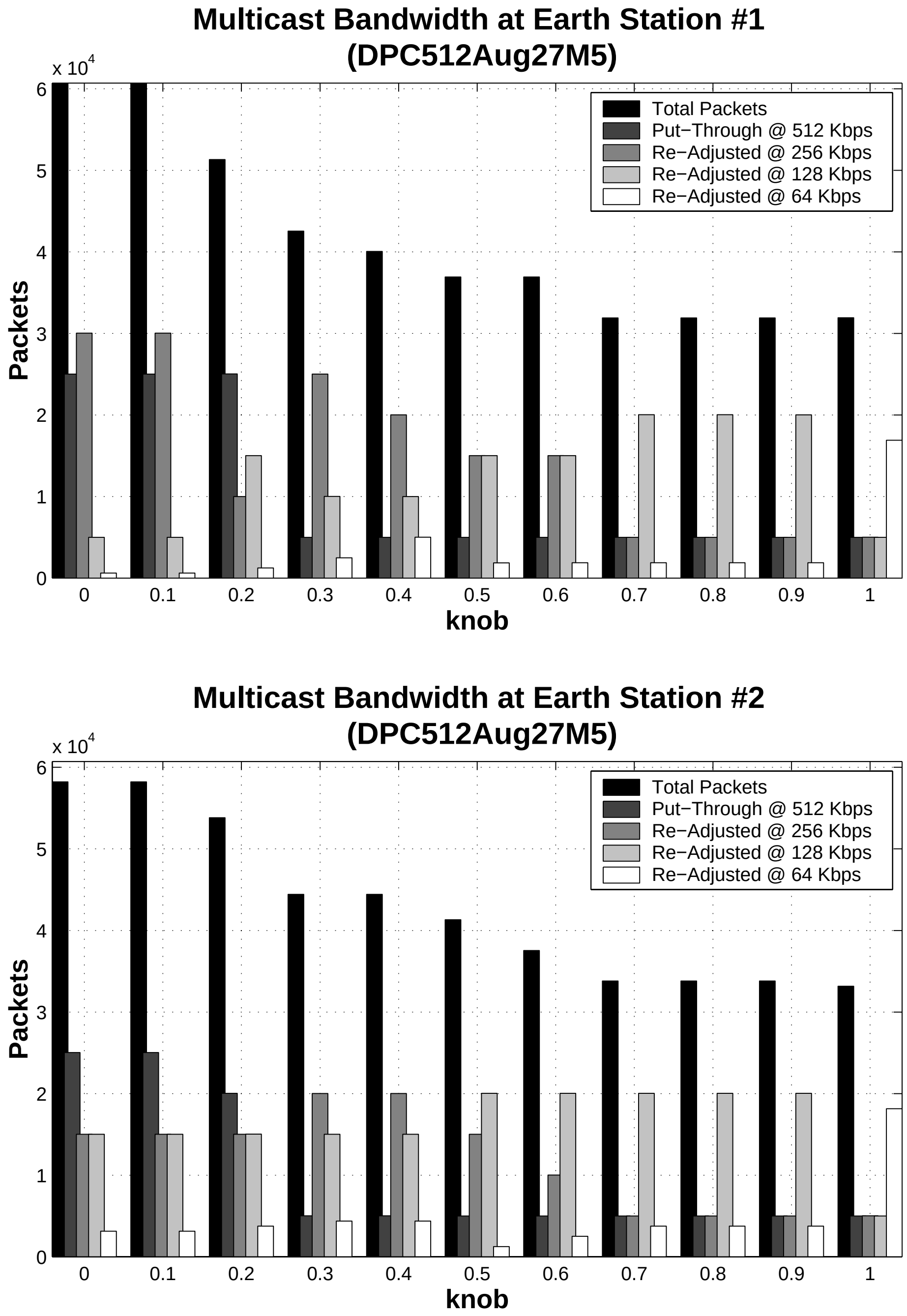


Figure 7.13: Earth station multicast bandwidth, based on the 512Kbits/s satellite trace of August 27, 2001

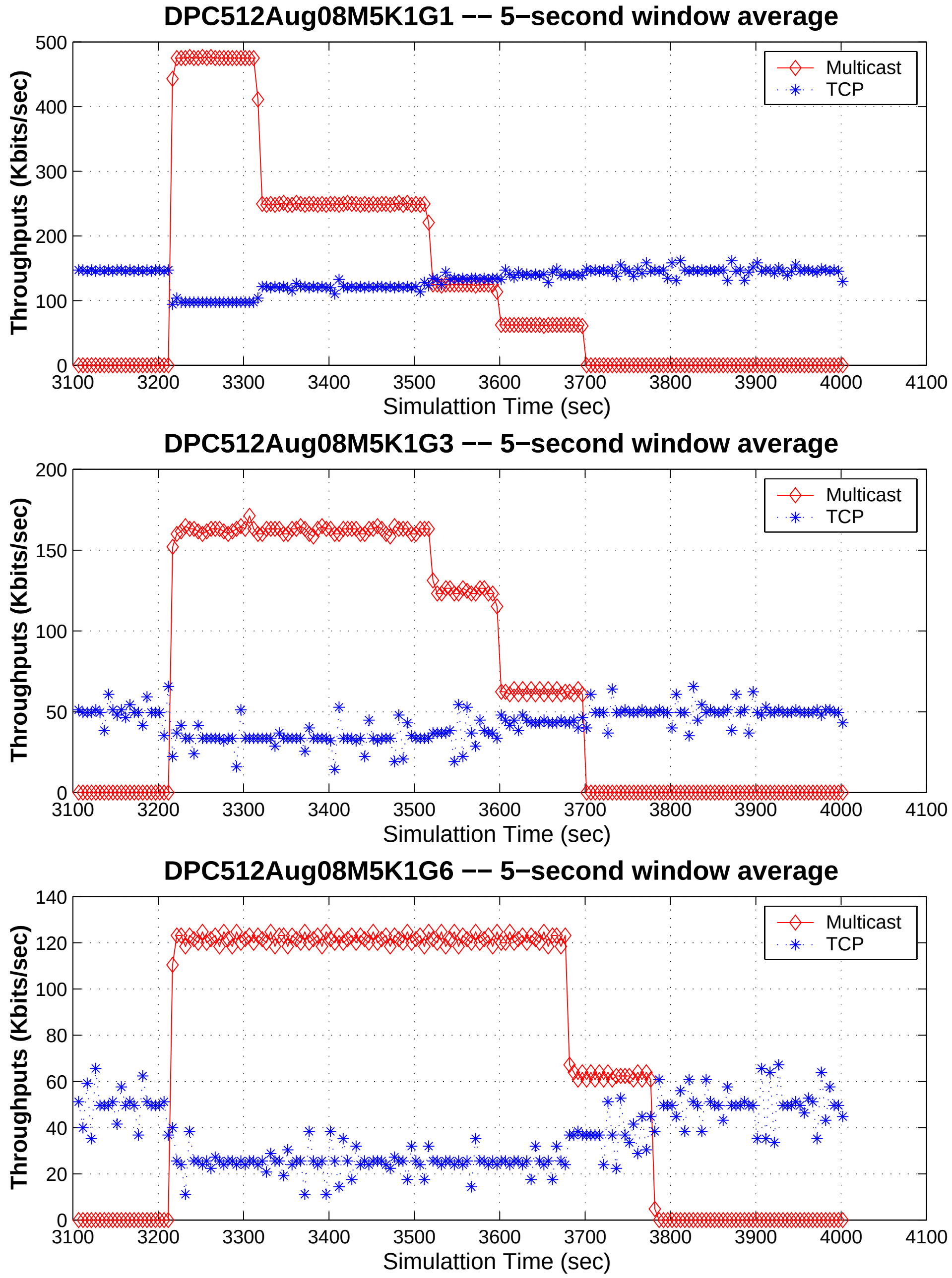


Figure 7.14: Throughput dynamics at observers 1, 3, and 6, based on the 512Kbits/s satellite trace of August 08, 2001

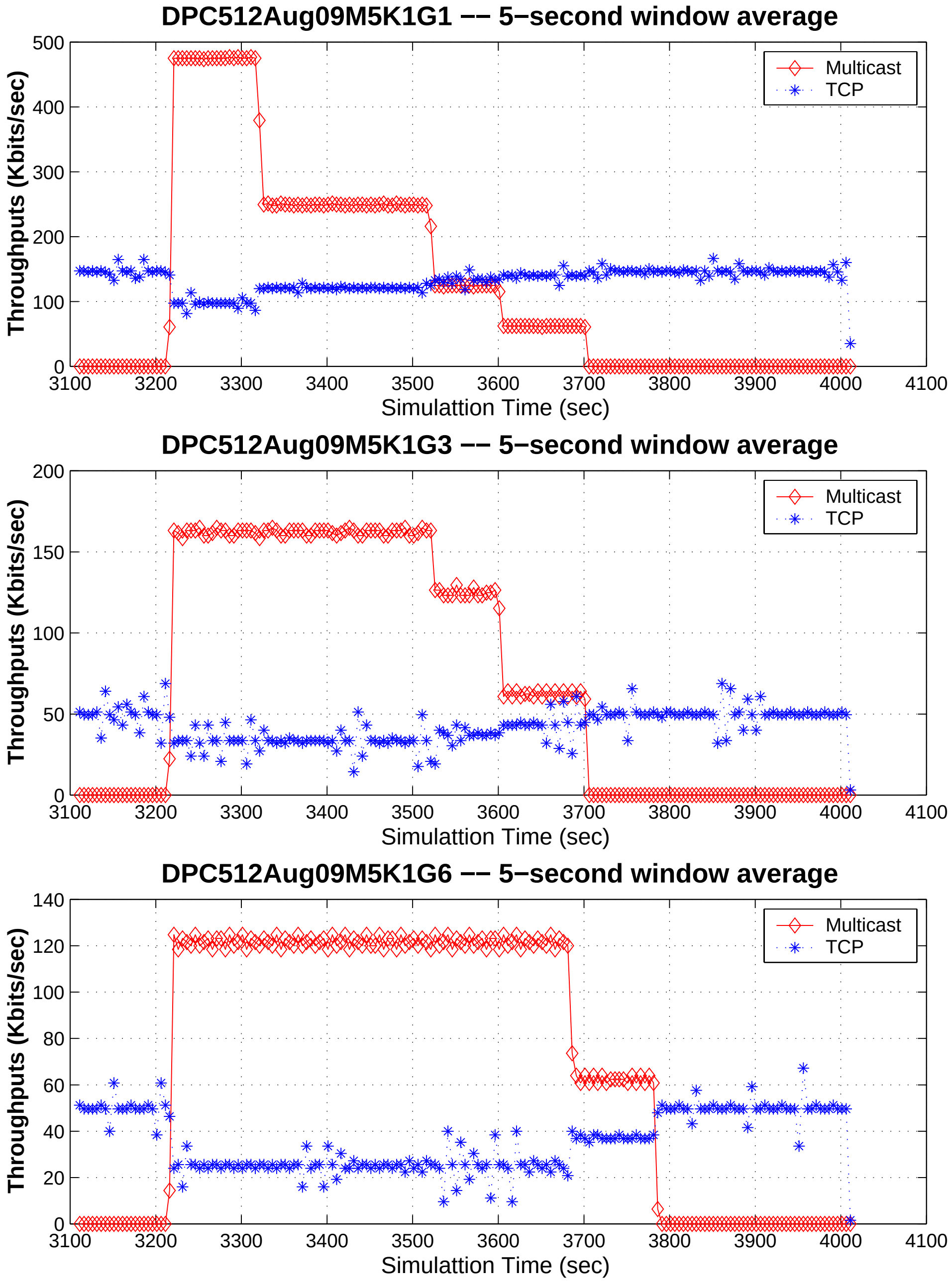
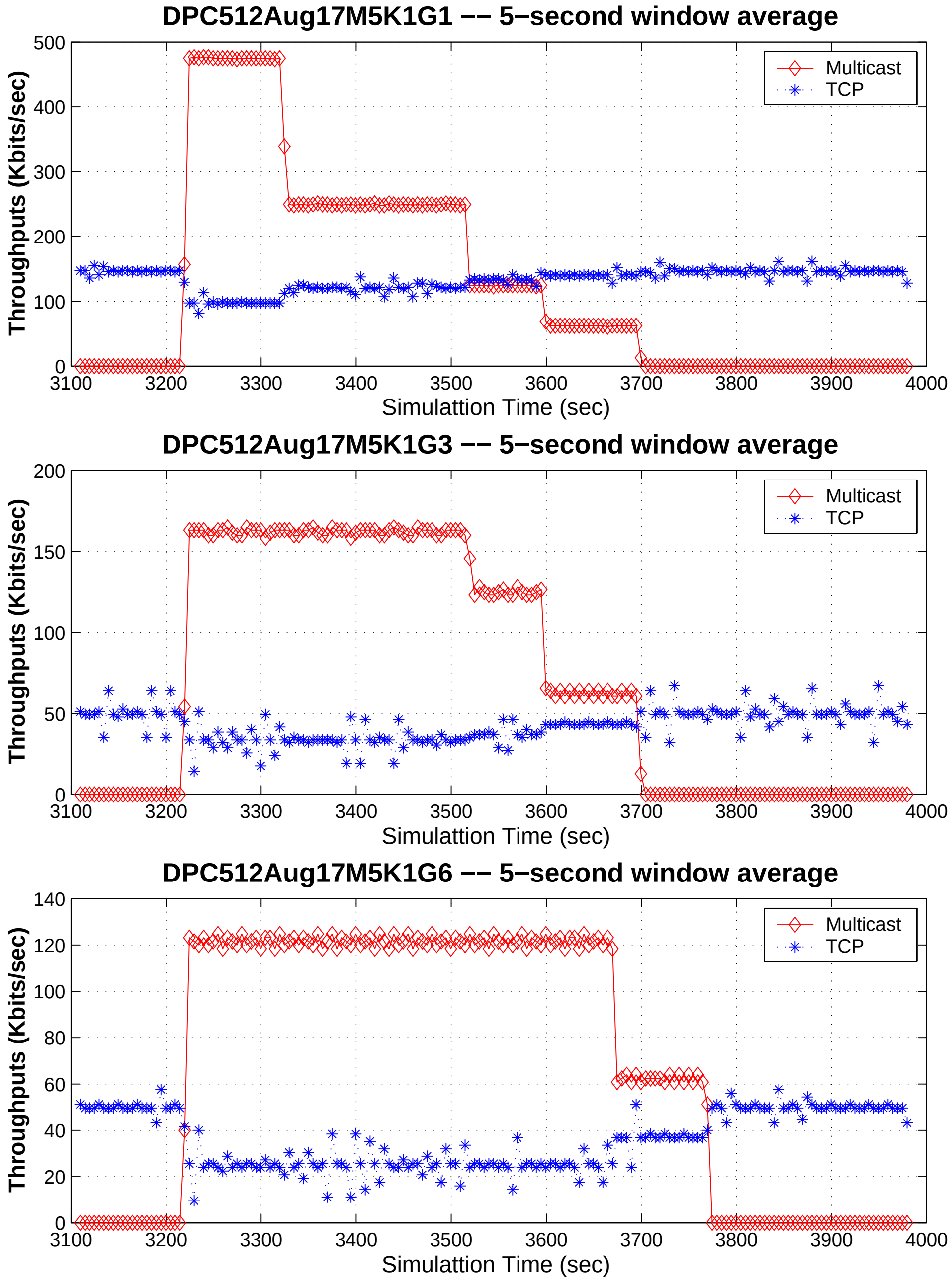


Figure 7.15: Throughput dynamics at observers 1, 3, and 6, based on the 512Kbits/s satellite trace of August 09, 2001

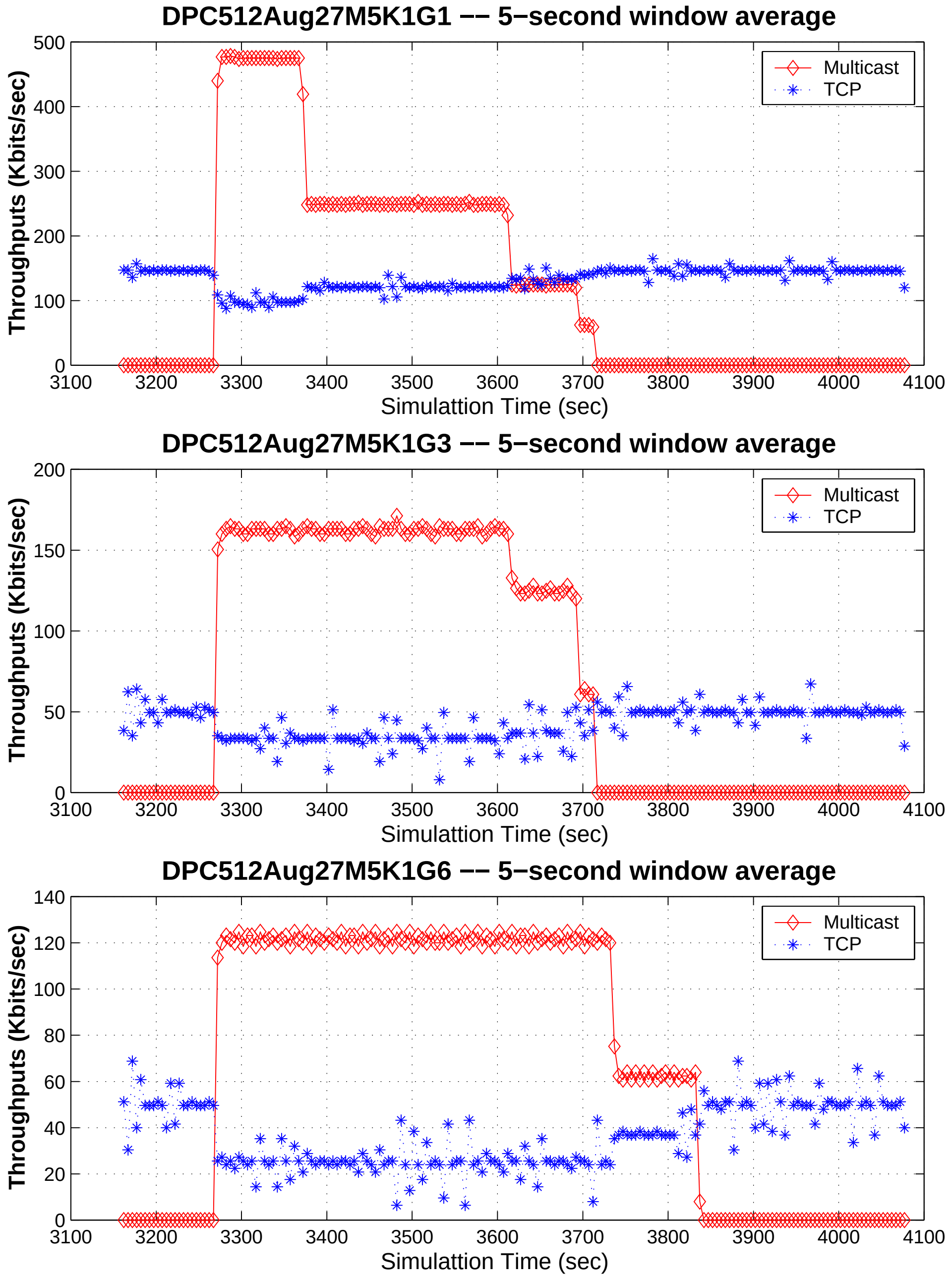


DPC512Aug17M5K1G1 –– 5−second window average

DPC512Aug17M5K1G3 –– 5−second window average

DPC512Aug17M5K1G6 –– 5−second window average

Figure 7.16: Throughput dynamics at observers 1, 3, and 6, based on the 512Kbits/s satellite trace of August 17, 2001


DPC512Aug27M5K1G1 –– 5–second window average

DPC512Aug27M5K1G3 –– 5–second window average

DPC512Aug27M5K1G6 –– 5–second window average

Figure 7.17: Throughput dynamics at observers 1, 3, and 6, based on the 512Kbits/s satellite trace of August 27, 2001

Chapter 8

Conclusions

In this dissertation, we have studied a long-term rate control problem for multicast dissemination of bulk data to heterogeneous receivers via hybrid satellite-terrestrial networks. Our proposed rate control mechanisms offer a controllable trade off between bandwidth consumption (i.e. the administrative issue) and reception latency (i.e. the users' satisfaction issue) through a given parameter called the **knob**. At the same time our rate control mechanisms can be set to provide a certain degree of probabilistical delivery reliability. We have made the following contributions:

- We have provided the results from real satellite experiments which we conducted to get a better understanding on the characteristics of the hybrid satellite-terrestrial internet channel. We have also introduced the Gaussian approximation approach and validated it through real satellite experiment traces.

- We have formulated and solved the end-to-end multicast rate control mechanism using the Gaussian approximation approach.

- We have studied how the long-term rate control system can co-exist with terrestrial TCP background traffic.

- Based on the Gaussian approximation approach, we have proposed the two-staged multicast rate control mechanism that will allow greater scalability.

Like most research work, this dissertation also generates a number of possible future work directions. There are interesting variants of the multicast rate scheduling problem that can be further studied. One example would be to study a multicast rate control system that would take into account the occasional non-stationary behaviors of the hybrid satellite-terrestrial internet networks. Another example would be a multicast rate control system that would allow multiple files to be scheduled to transmit overlappingly.

Future generations of satellites will allow more sophisticated on-board processing capabilities [Ale00]. One generic architectural question would be what type of processing should be put aboard the next generation satellite in order to better support multicast services. A constellation of low earth orbit (LEO) satellites may soon become more common and may prove useful as a communication backbone [Rod01]. The LEO satellite constellation can offer a shorter propagation delay and inter-satellite links. These will pose numerous scheduling and control challenges, because the LEO satellites would appear to be moving with respect to the earth surface. In order to set up LEO satellites to provide some form of guaranteed multicast service, key problems are required to be solved. Among these key problems would be the inter-satellite (multicast) routing, and the inter-satellite traffic control. We will look forward to solving some of these key challenges.

Appendix A

Results from Satellite Experiments

This appendix contains the experiment results that are referred to in Section 3.1 and Section 3.2.

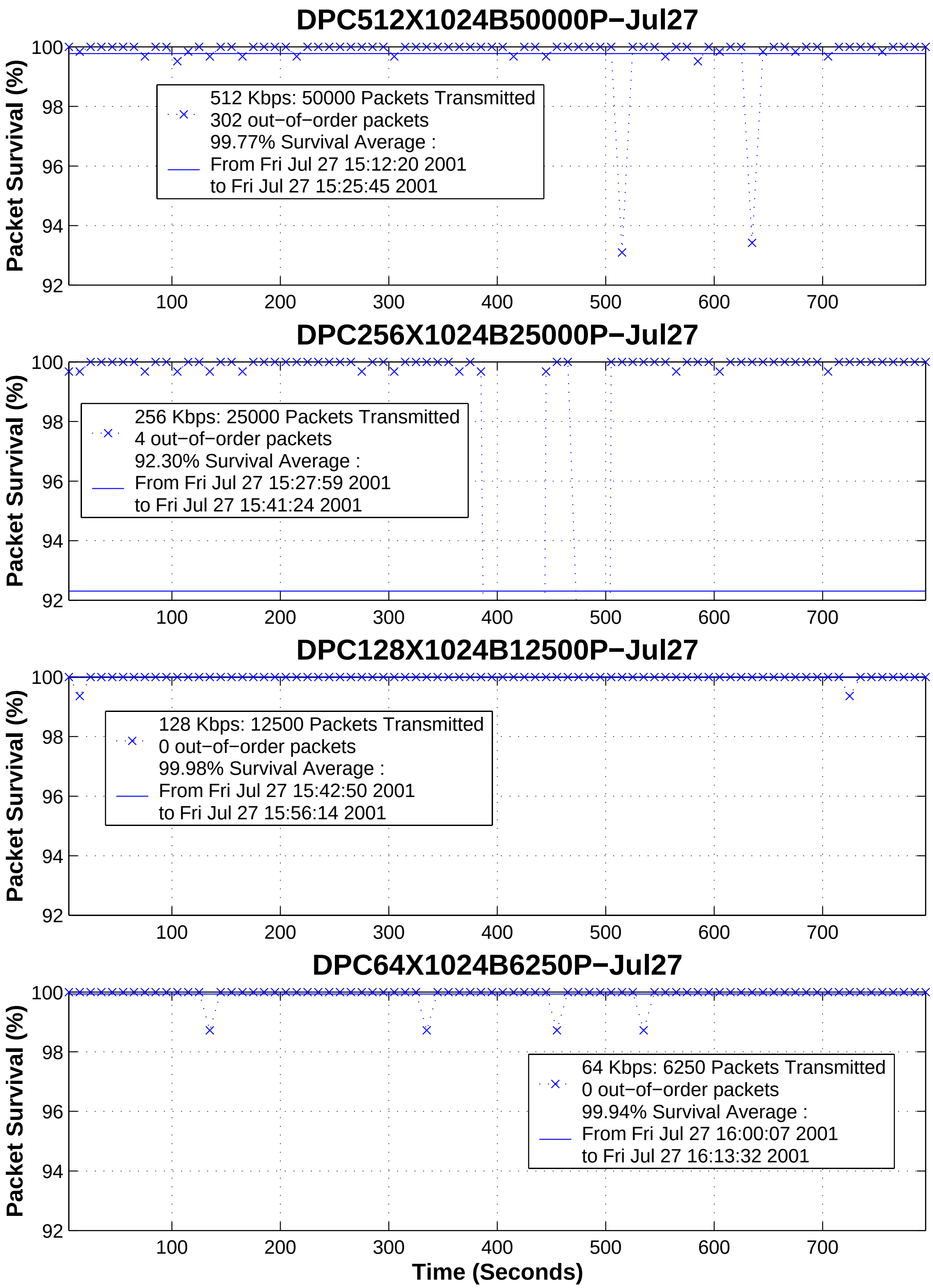


Figure A.1: Throughput dynamics of the hybrid satellite-terrestrial internet channel on July 27, 2001 (part 1)

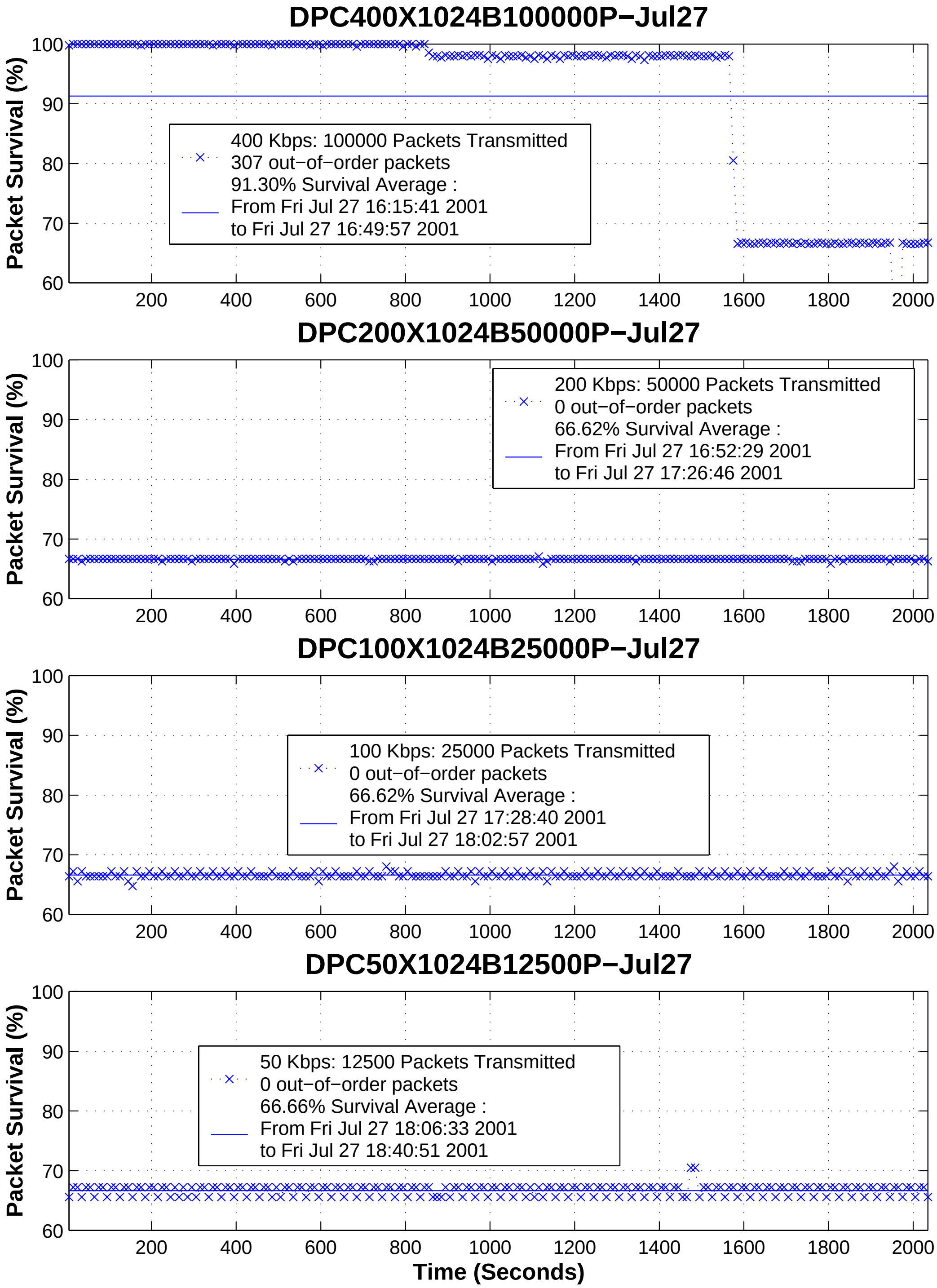


Figure A.2: Throughput dynamics of the hybrid satellite-terrestrial internet channel on July 27, 2001 (part 2)

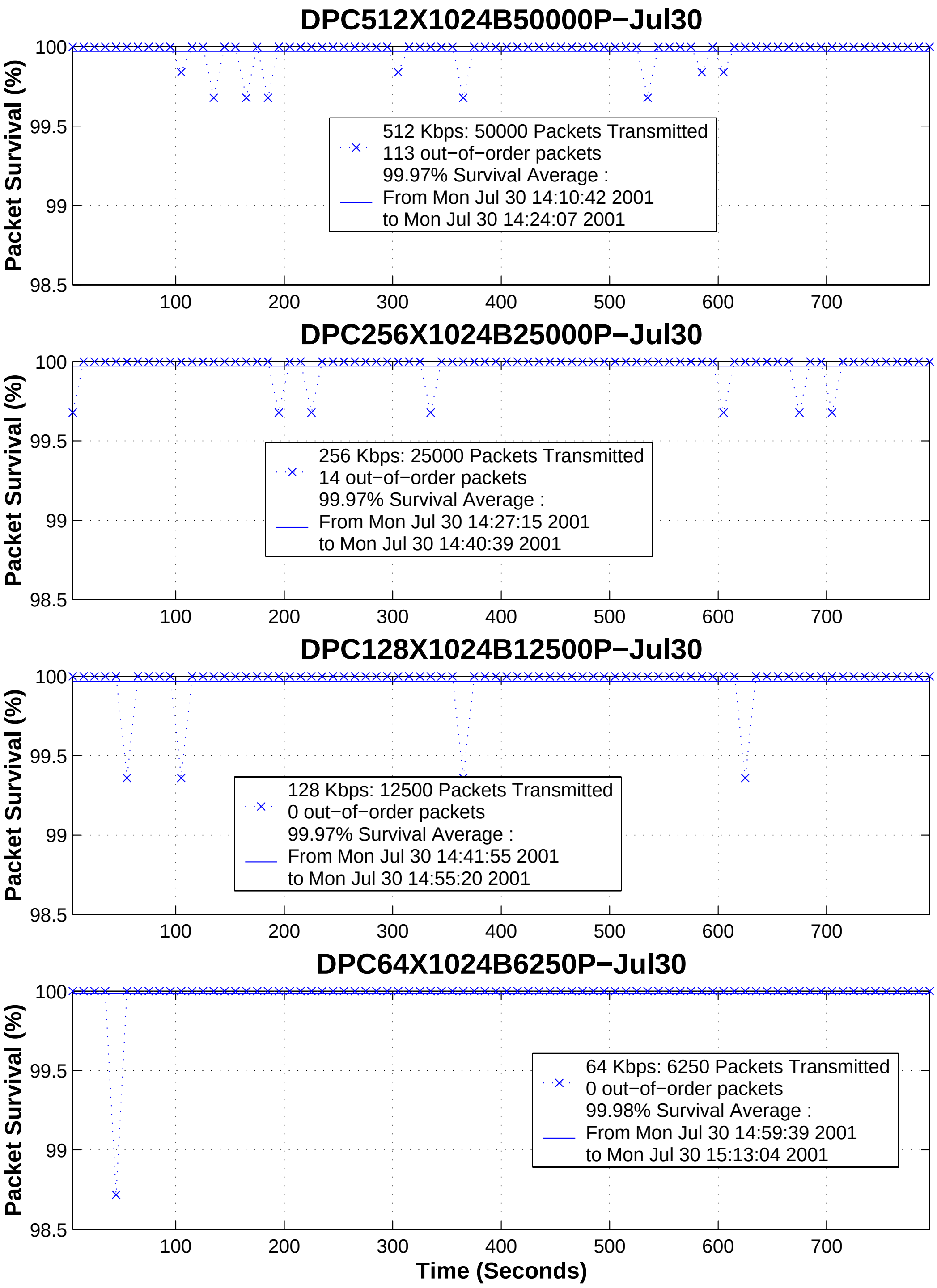


Figure A.3: Throughput dynamics of the hybrid satellite-terrestrial internet channel on July 30, 2001 (part 1)

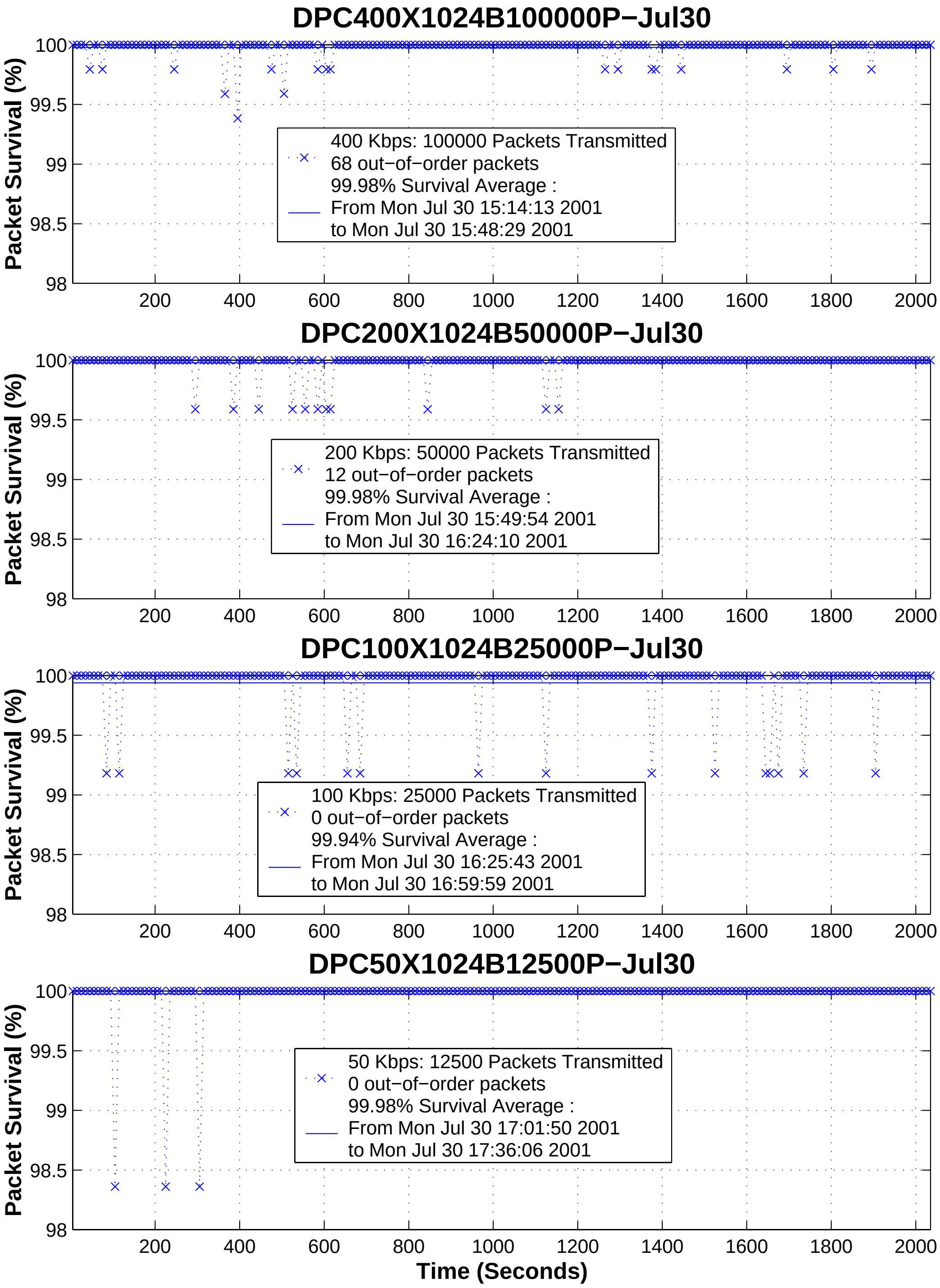


Figure A.4: Throughput dynamics of the hybrid satellite-terrestrial internet channel on July 30, 2001 (part 2)

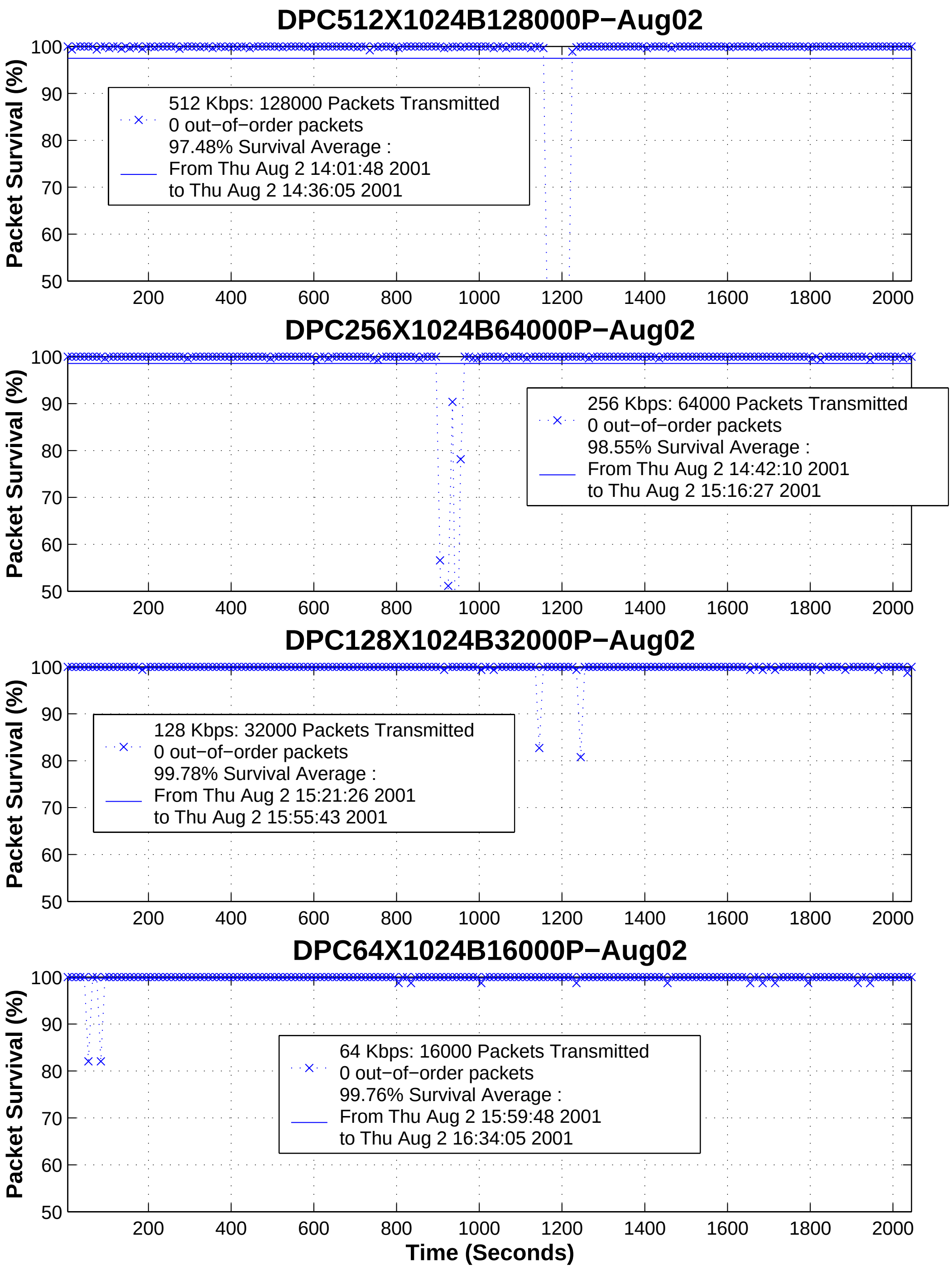


Figure A.5: Throughput dynamics of the hybrid satellite-terrestrial internet channel on August 2, 2001 (part 1)

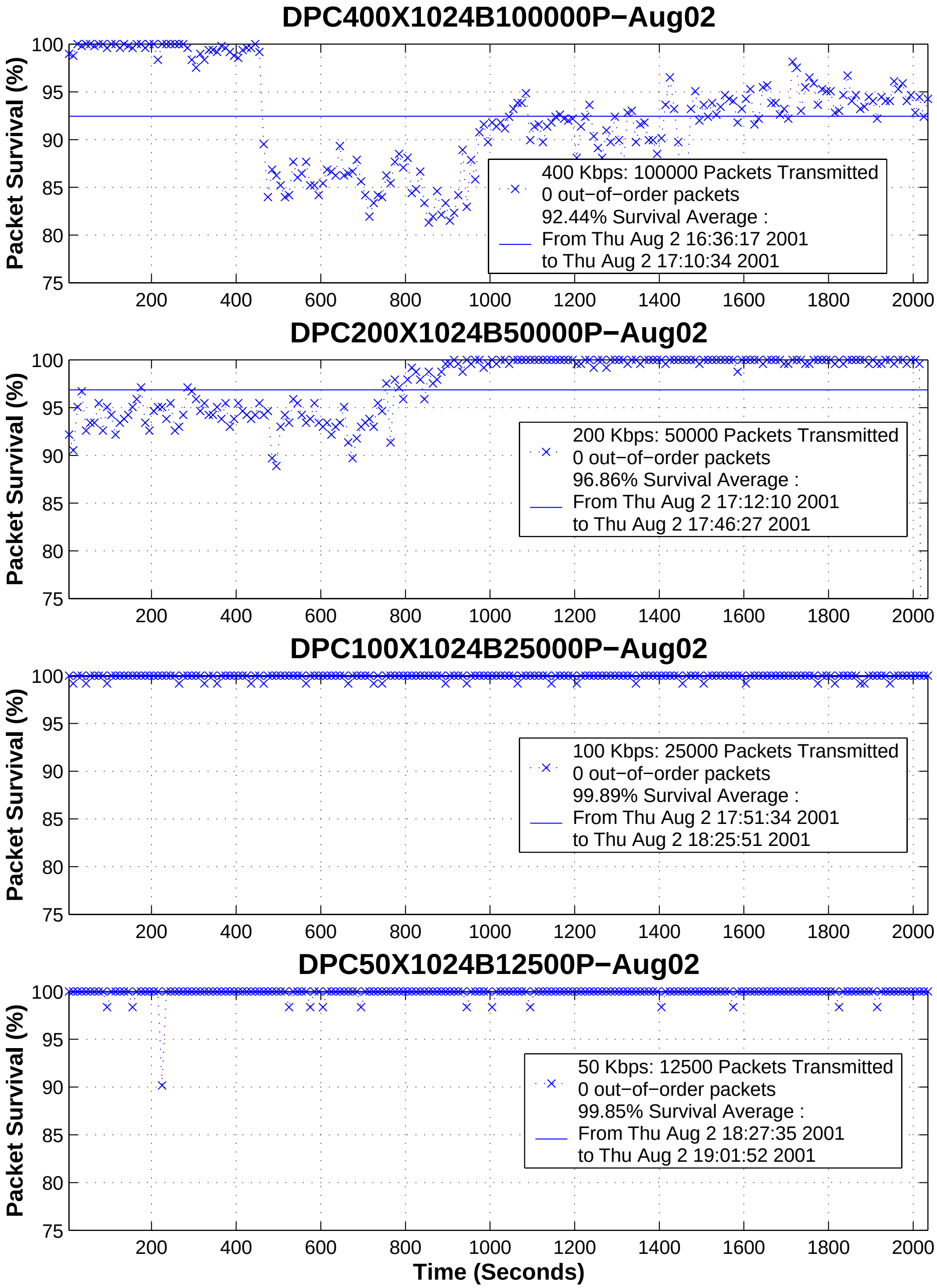


Figure A.6: Throughput dynamics of the hybrid satellite-terrestrial internet channel on August 2, 2001 (part 2)

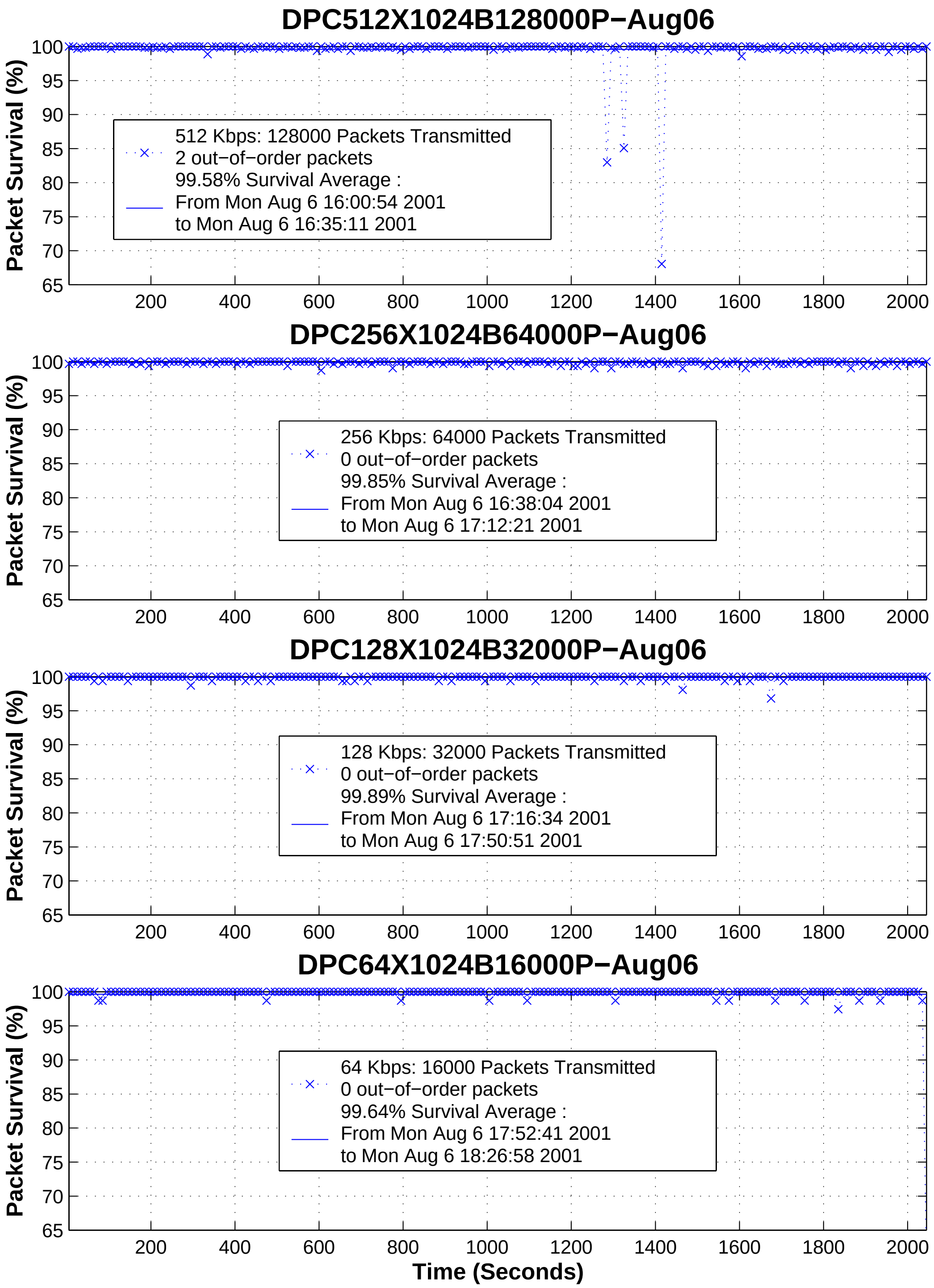


Figure A.7: Throughput dynamics of the hybrid satellite-terrestrial internet channel on August 6, 2001

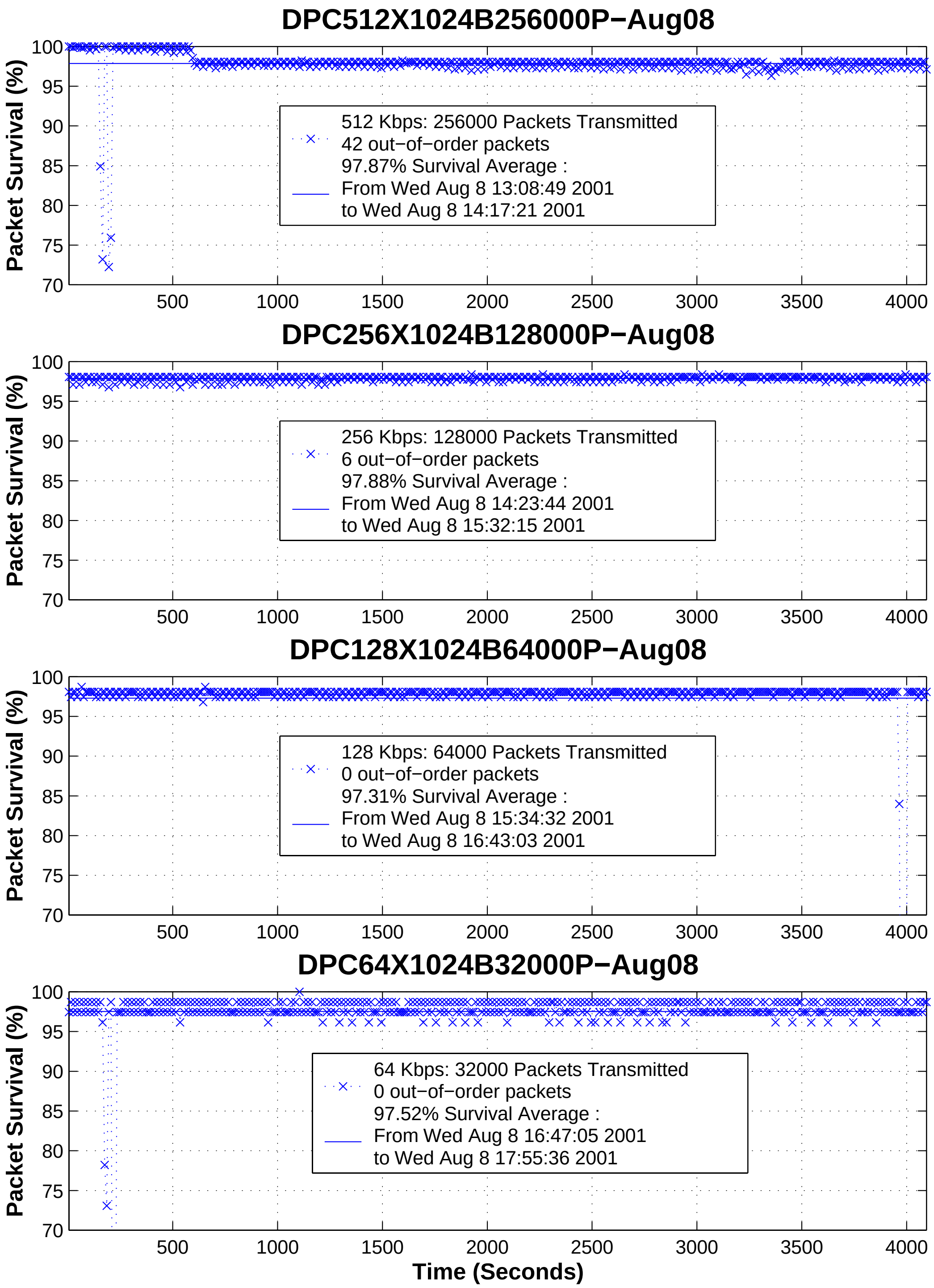


Figure A.8: Throughput dynamics of the hybrid satellite-terrestrial internet channel on August 8, 2001

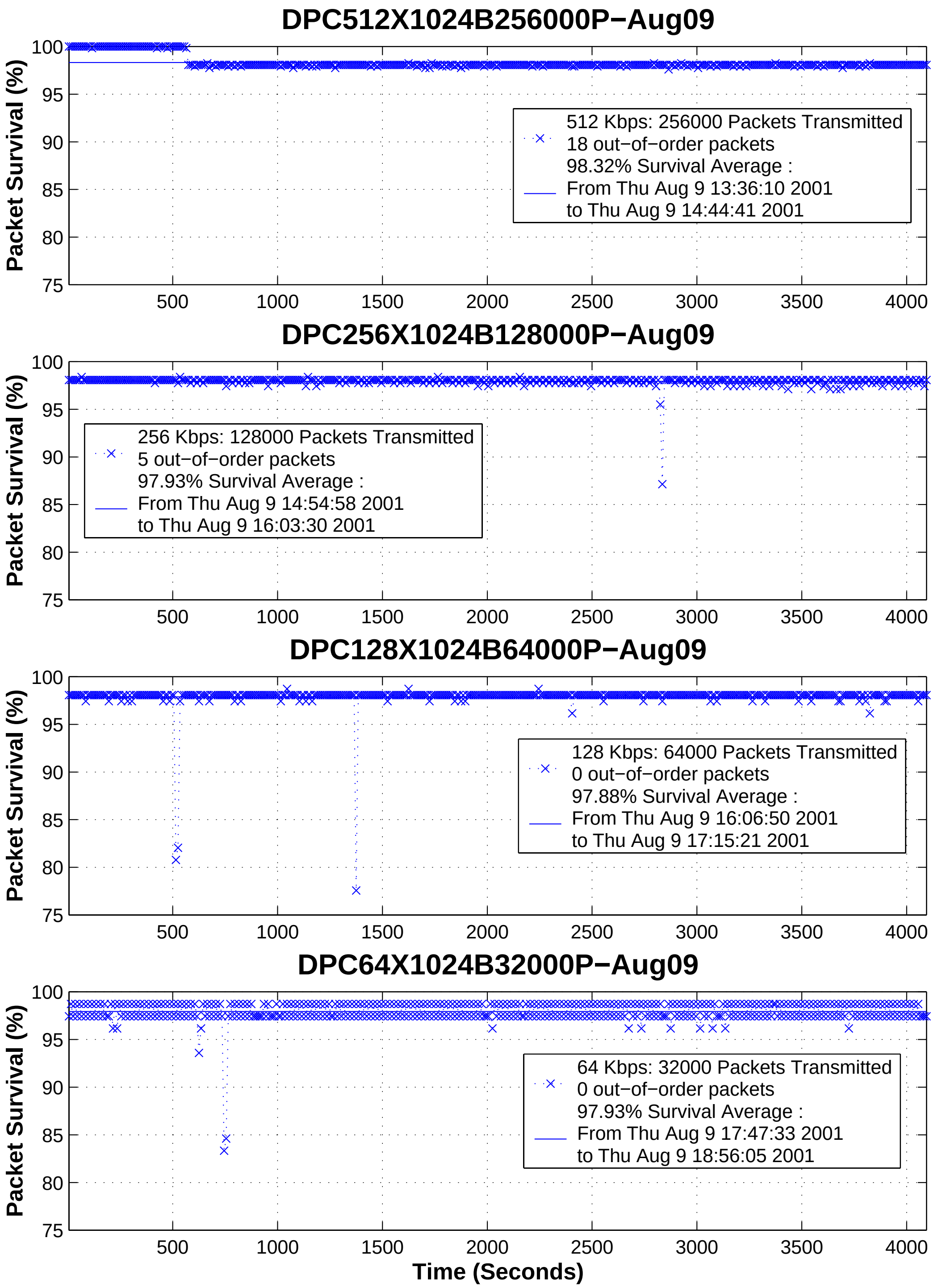


Figure A.9: Throughput dynamics of the hybrid satellite-terrestrial internet channel on August 9, 2001

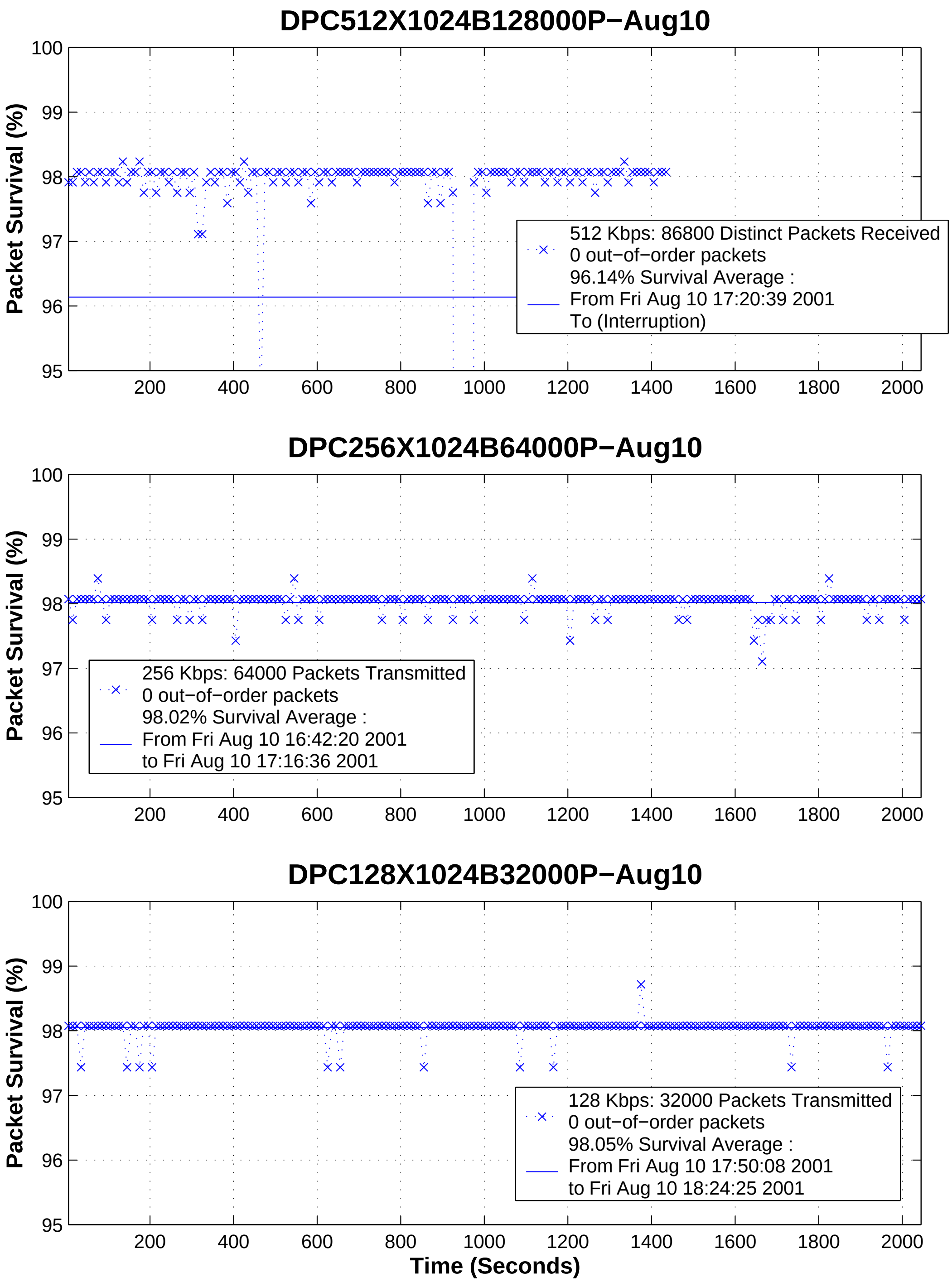


Figure A.10: Throughput dynamics of the hybrid satellite-terrestrial internet channel on August 10, 2001

The experiment in the top figure was interrupted by heavy rain.

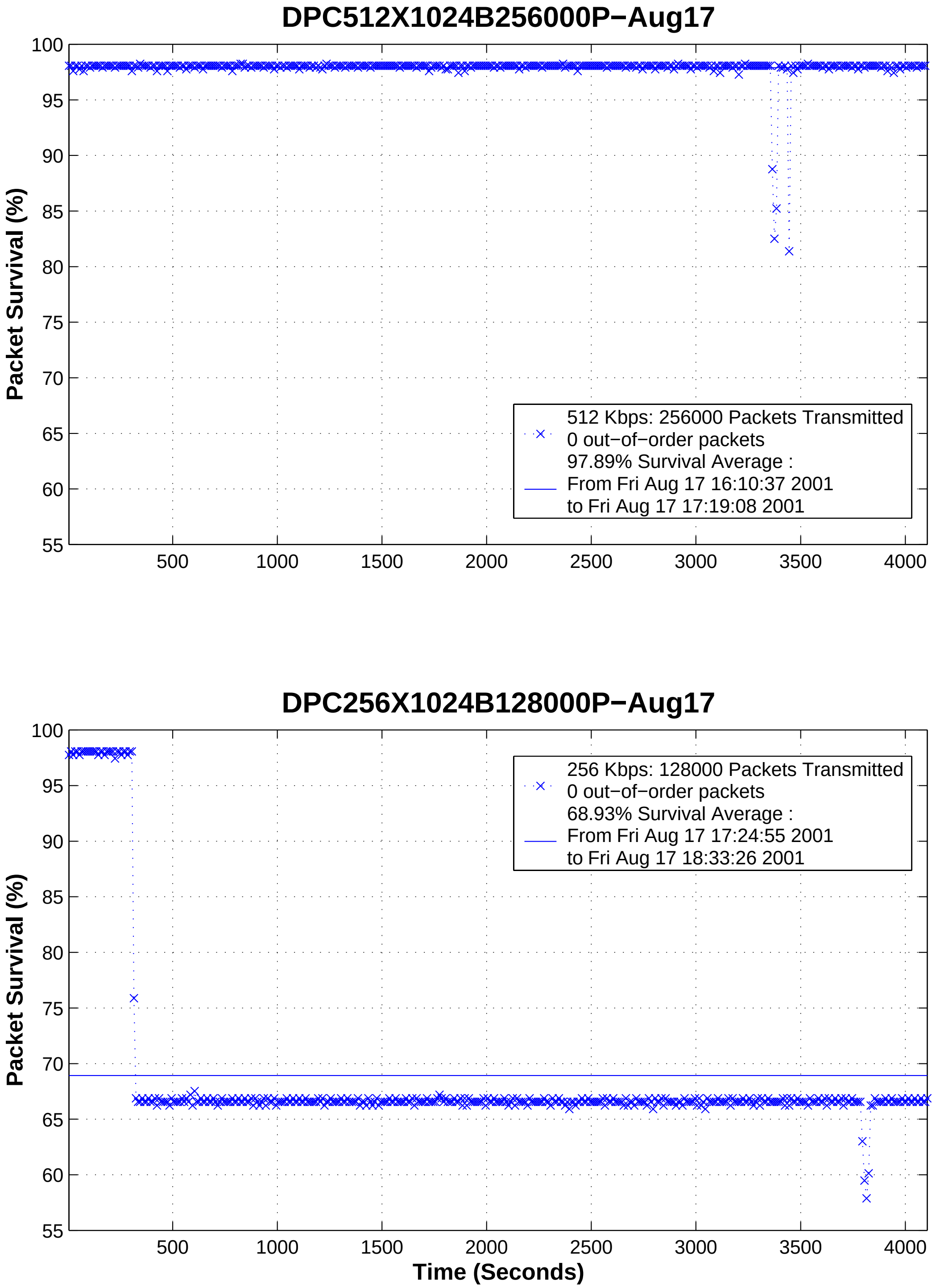


Figure A.11: Throughput dynamics of the hybrid satellite-terrestrial internet channel on August 17, 2001 (part 1)

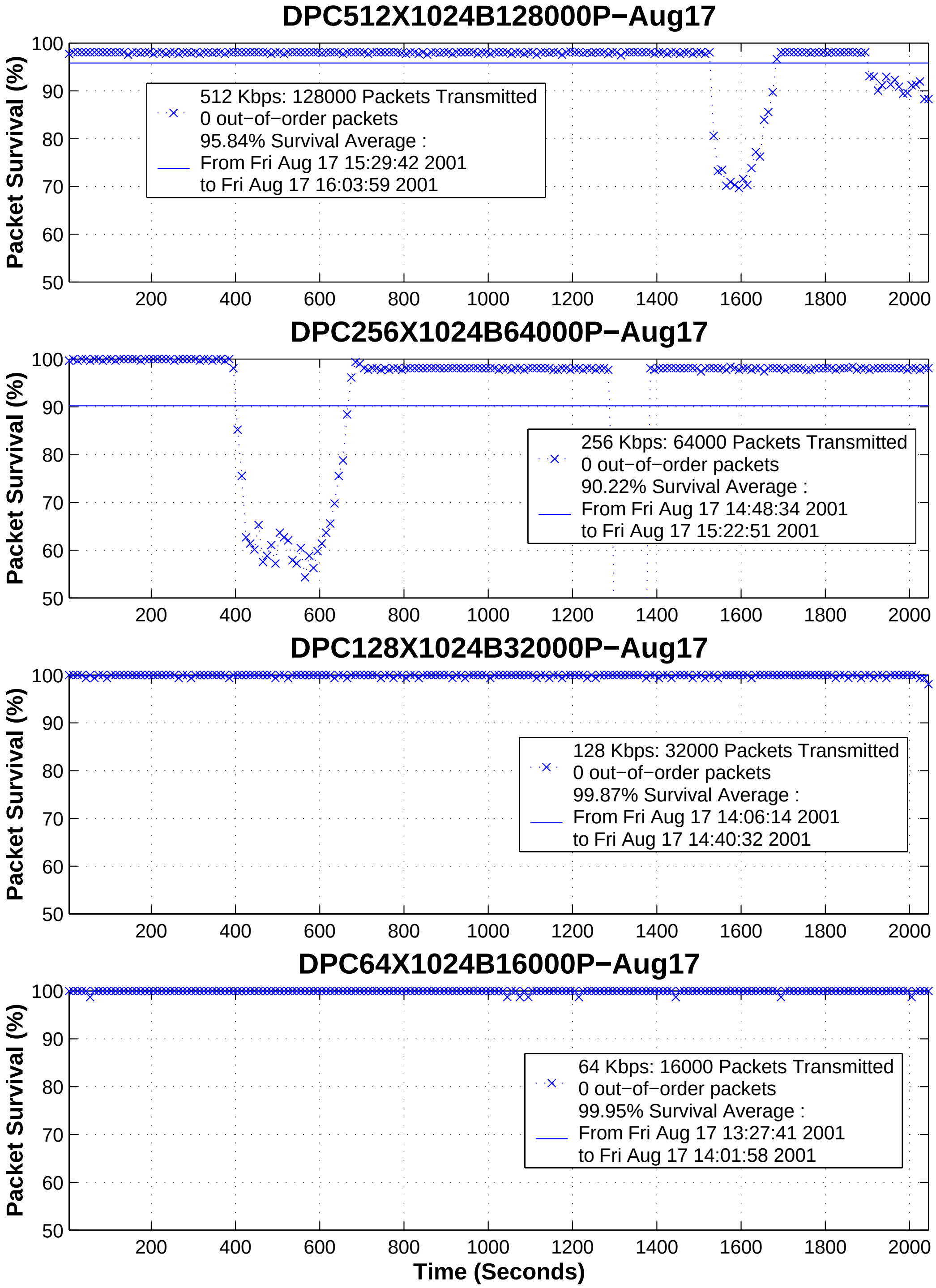


Figure A.12: Throughput dynamics of the hybrid satellite-terrestrial internet channel on August 17, 2001 (part 2)

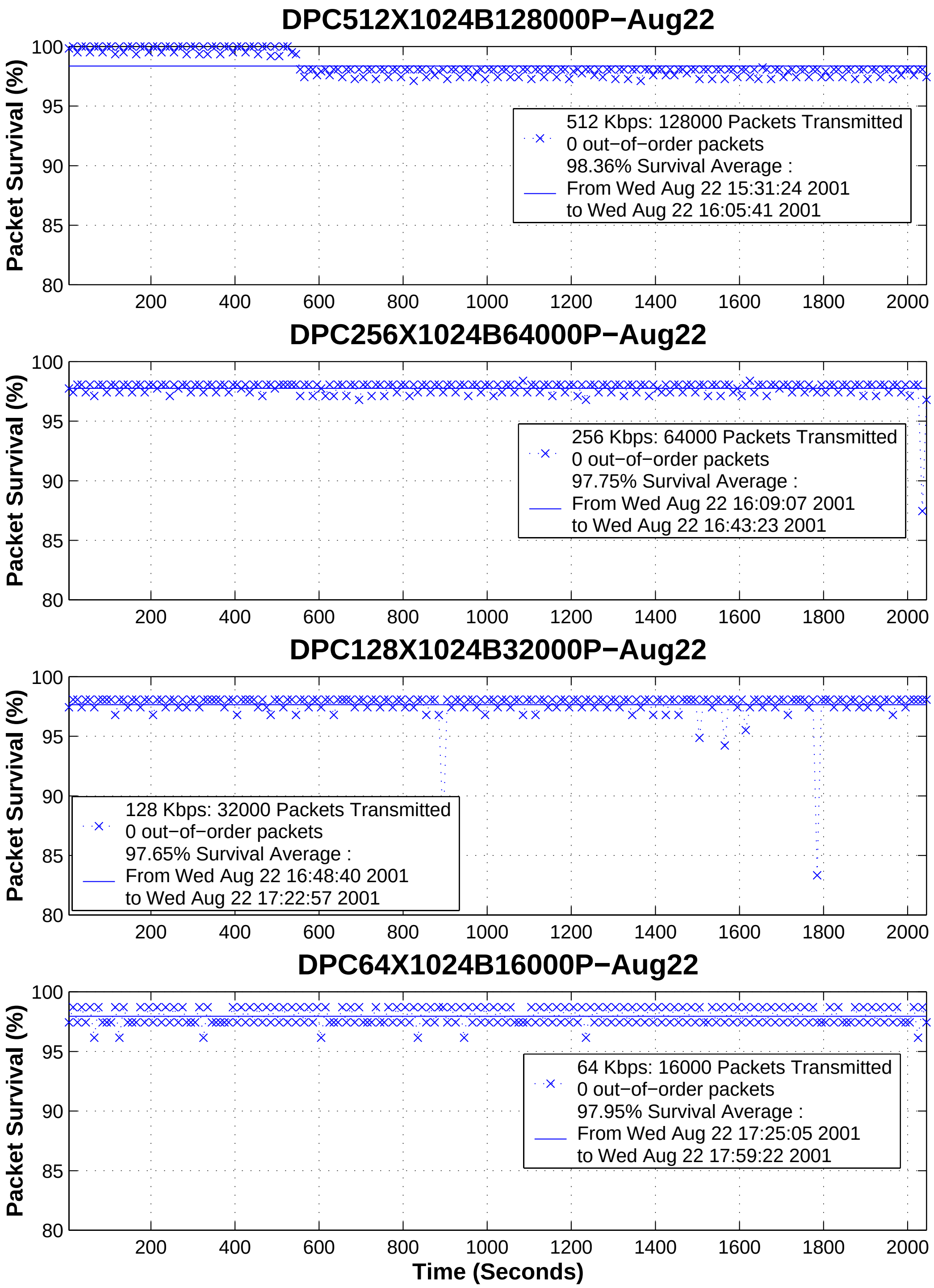


Figure A.13: Throughput dynamics of the hybrid satellite-terrestrial internet channel on August 22, 2001

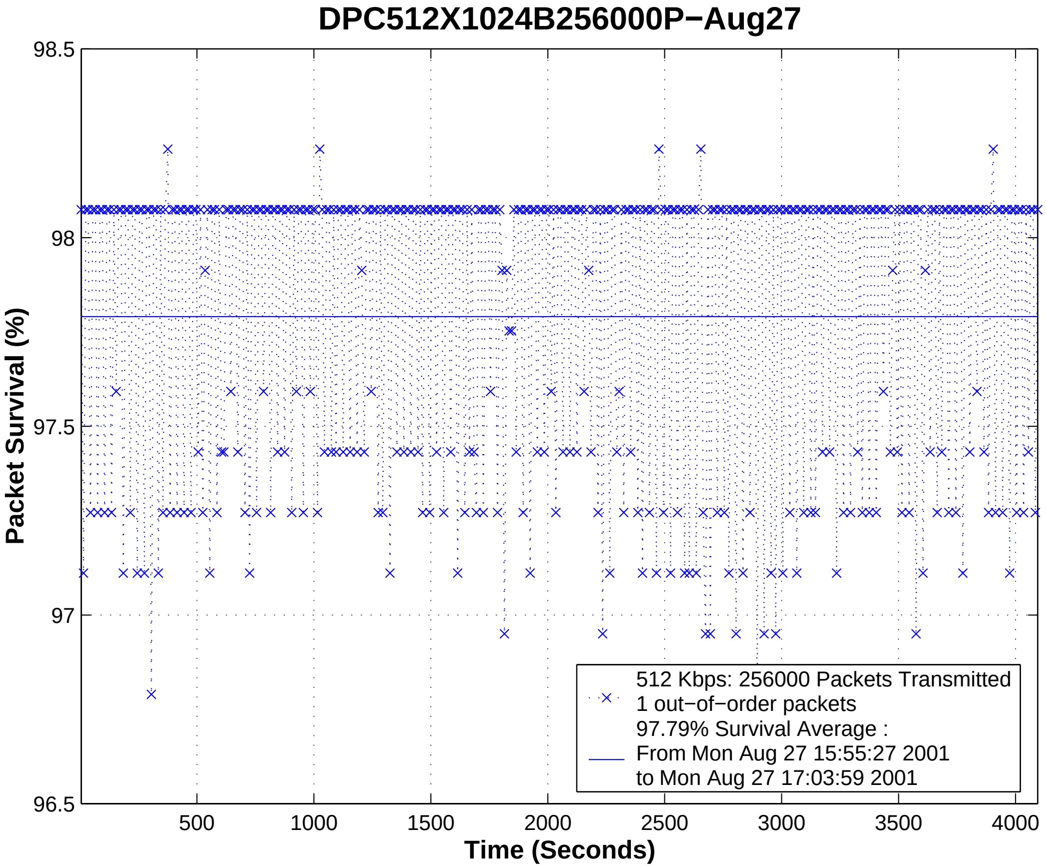


Figure A.14: Throughput dynamics of the hybrid satellite-terrestrial internet channel on August 27, 2001

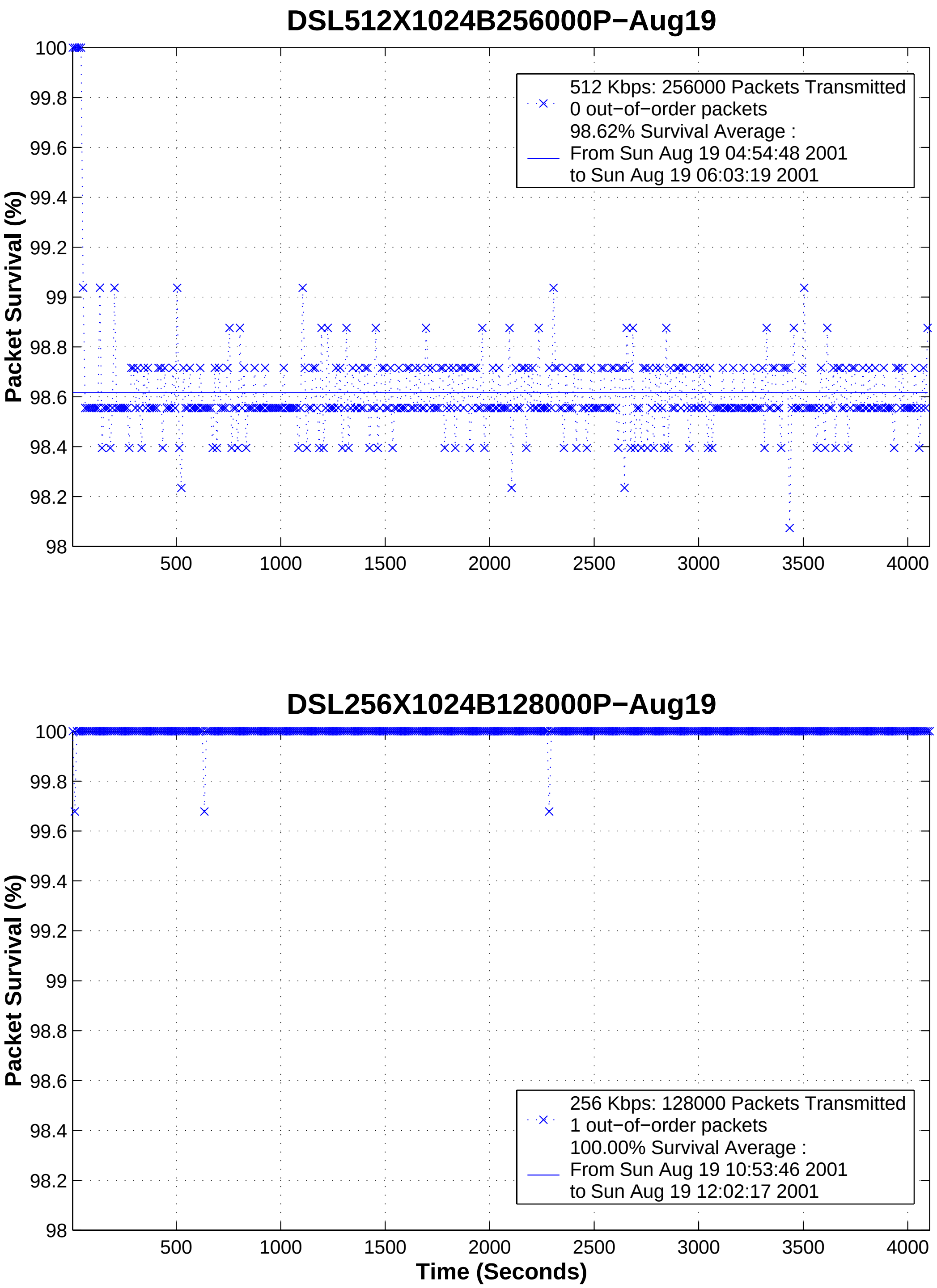


Figure A.15: Throughput dynamics of the all-terrestrial DSL internet channel on August 19, 2001

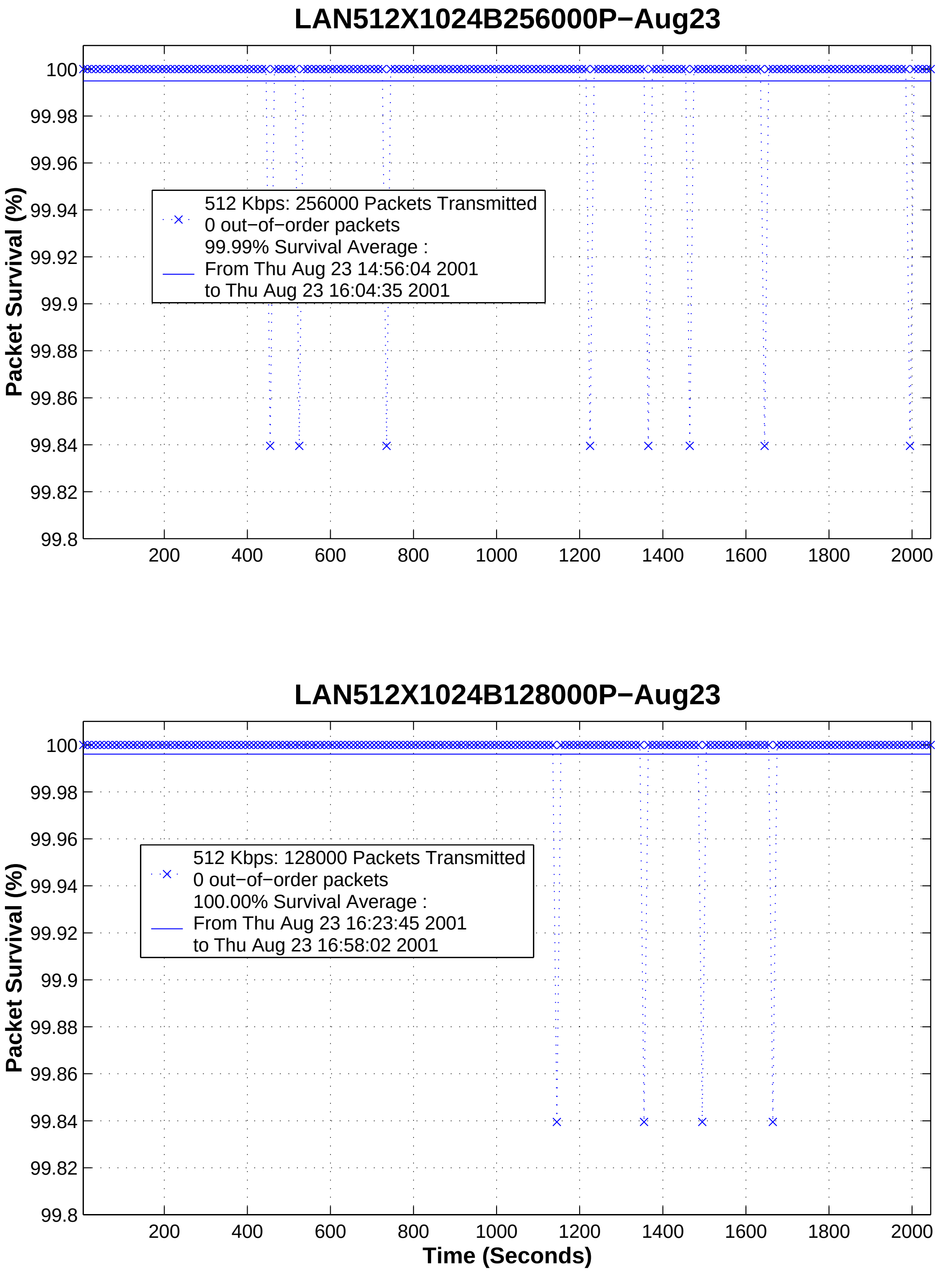


Figure A.16: Throughput dynamics of the campus network on August 23, 2001. Note that there were almost negligible packet losses in these two runs (8 packet losses in the top figure and 4 packet losses in the bottom figure).

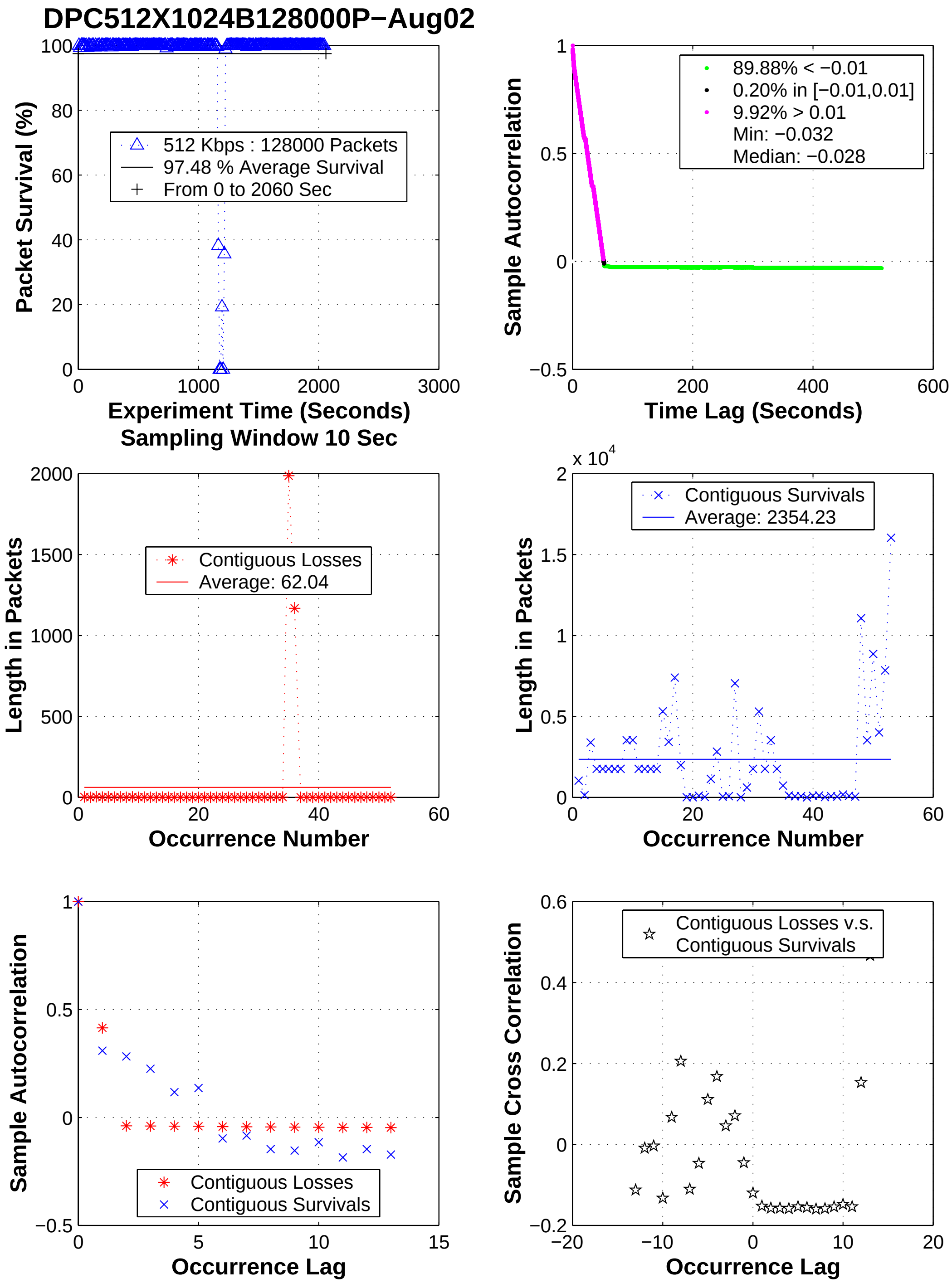


Figure A.17: Time series analysis: August 02, 2001 : HST 512 Kbits/sec transmission rate

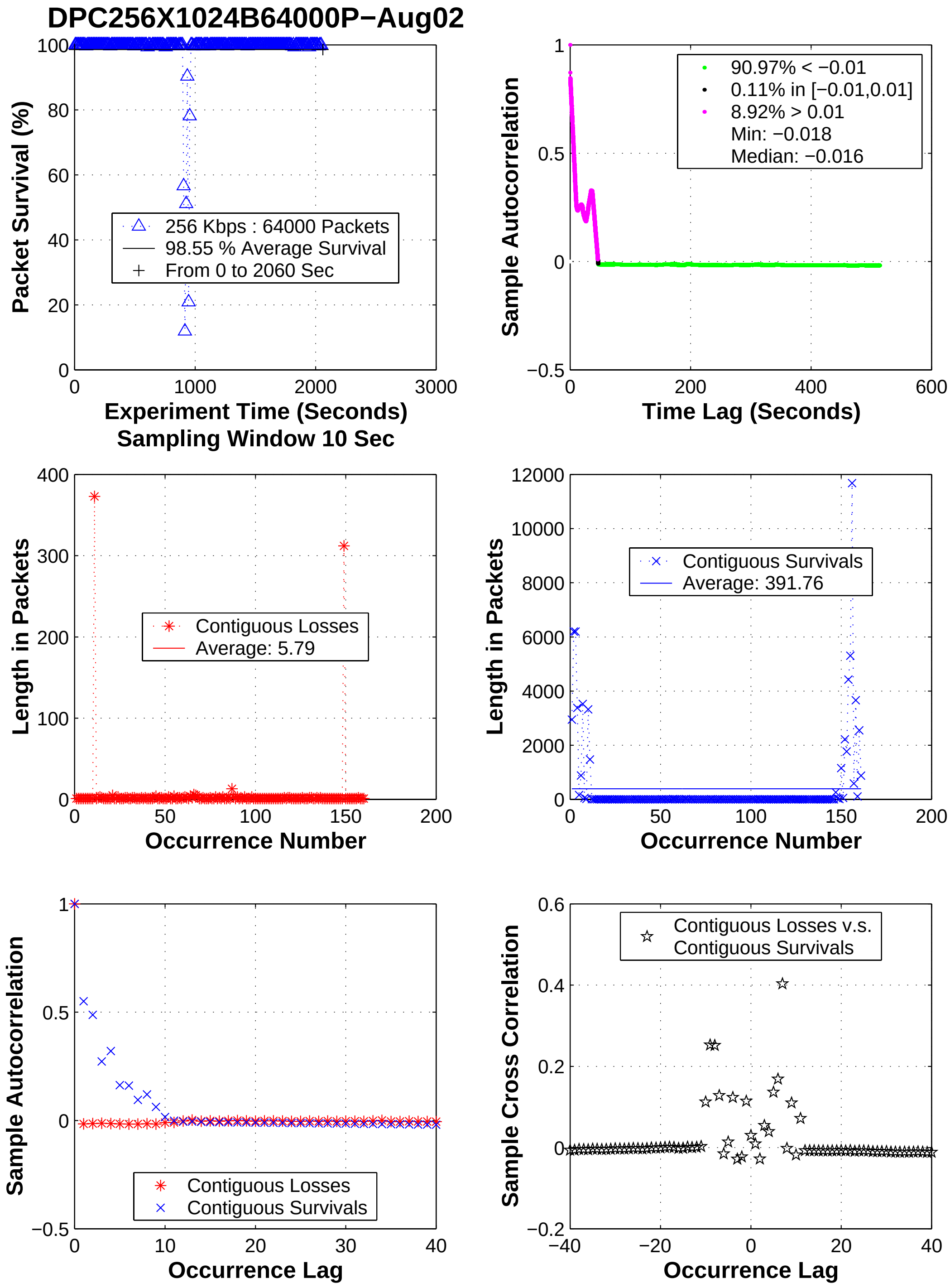


Figure A.18: Time series analysis: August 02, 2001 : HST 256 Kbits/sec transmission rate

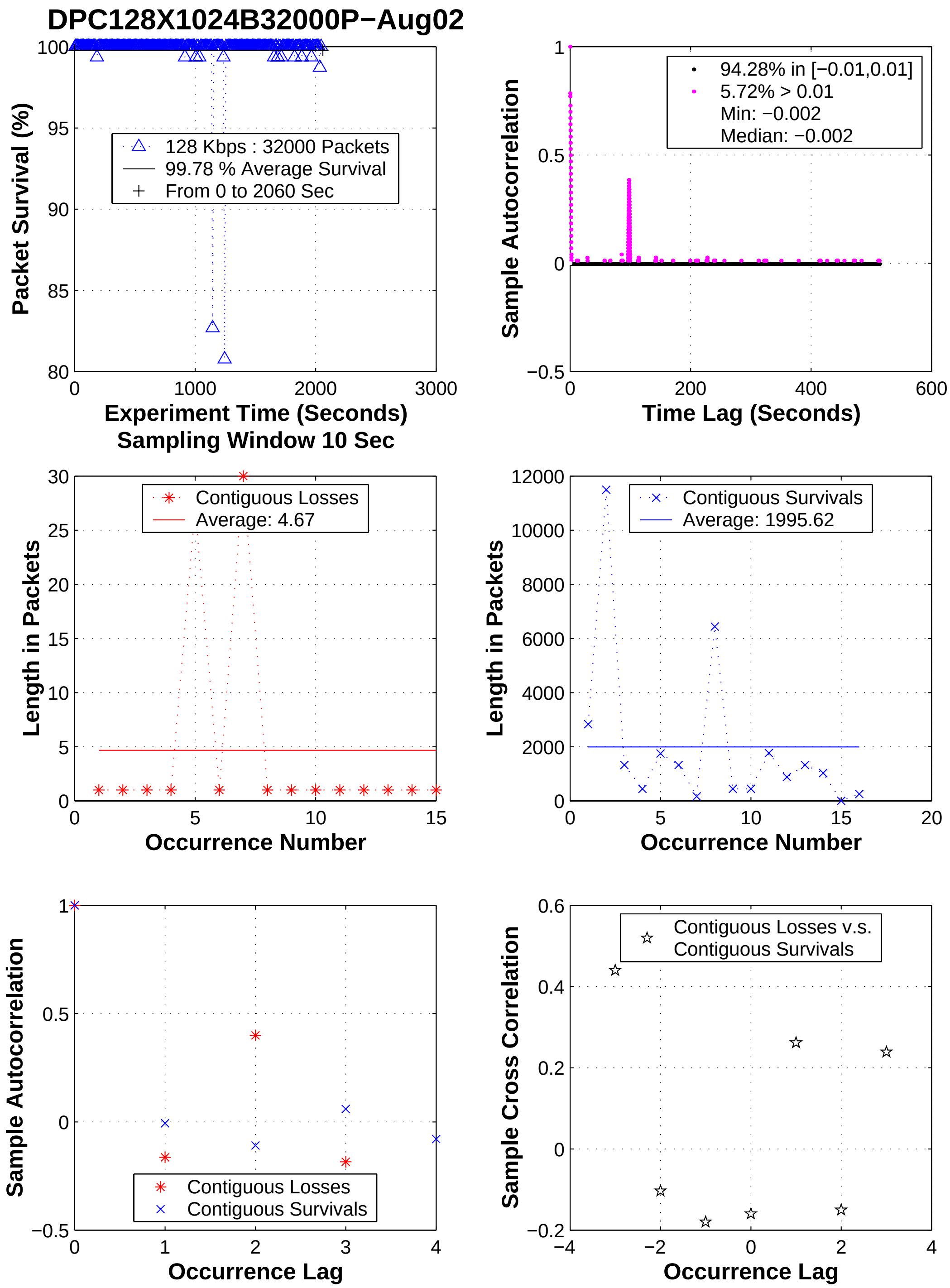


Figure A.19: Time series analysis: August 02, 2001 : HST 128 Kbits/sec transmission rate

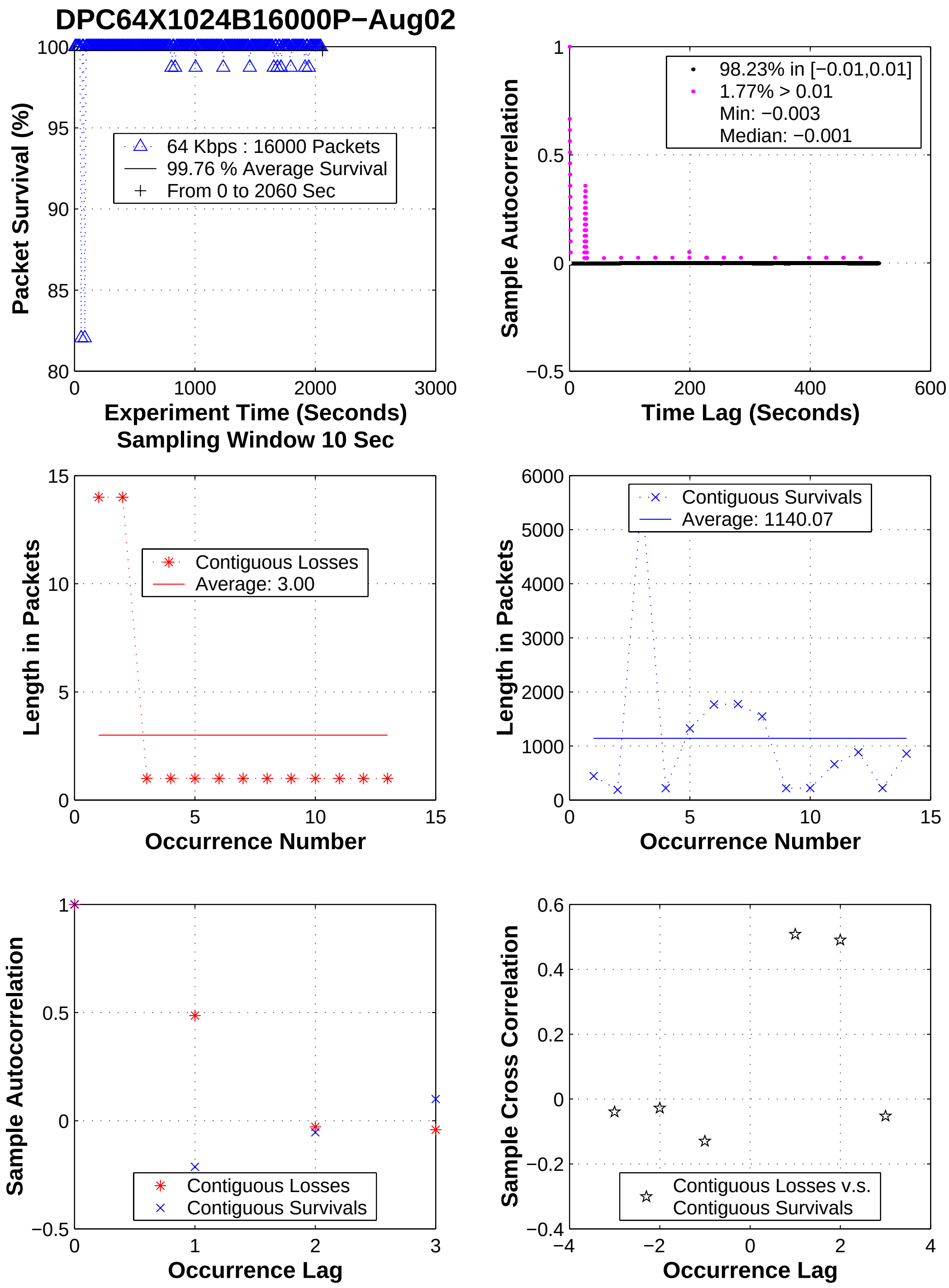


Figure A.20: Time series analysis: August 02, 2001 : HST 64 Kbits/sec transmission rate

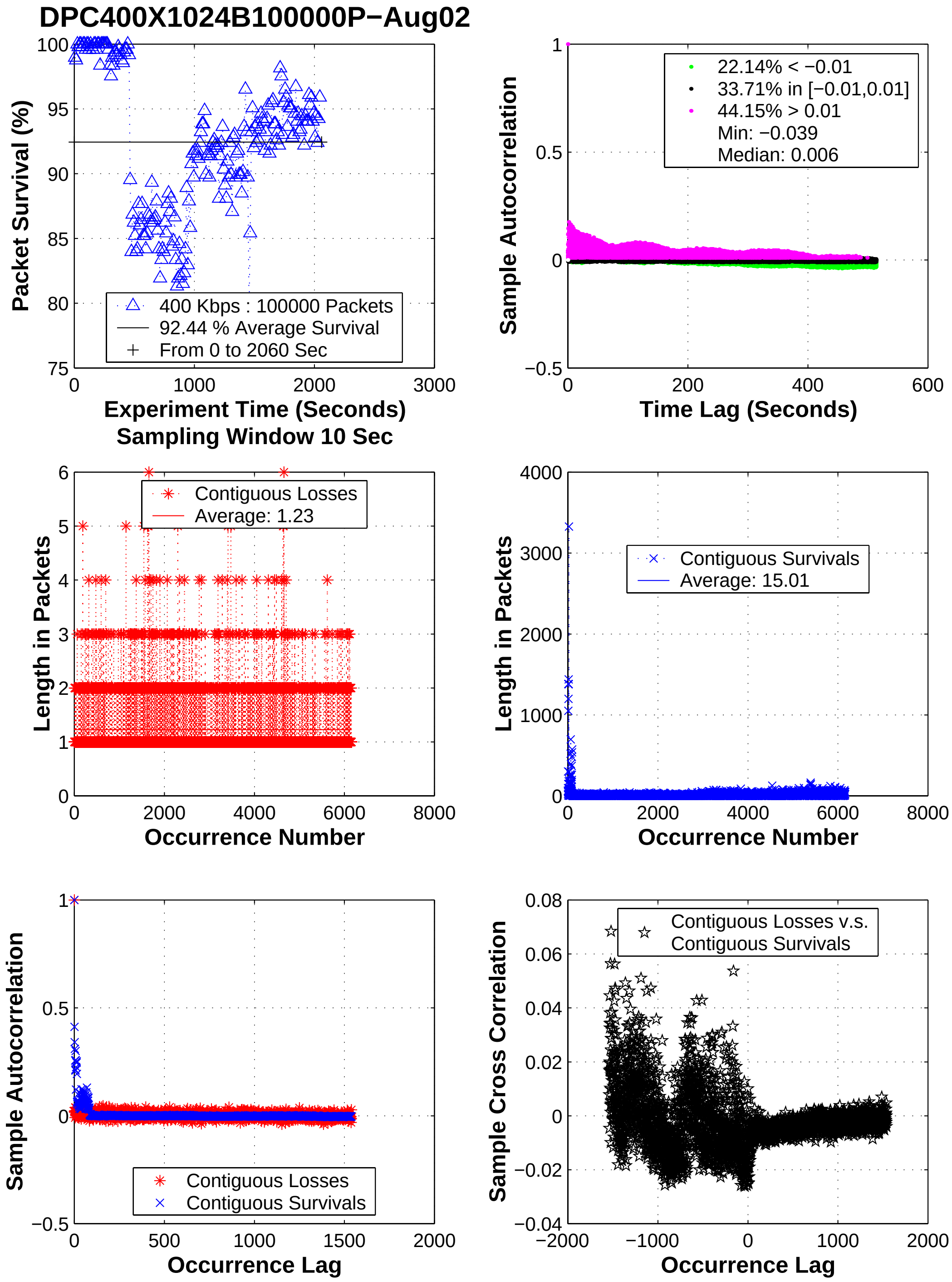


Figure A.21: Time series analysis: August 02, 2001 : HST 400 Kbits/sec transmission rate

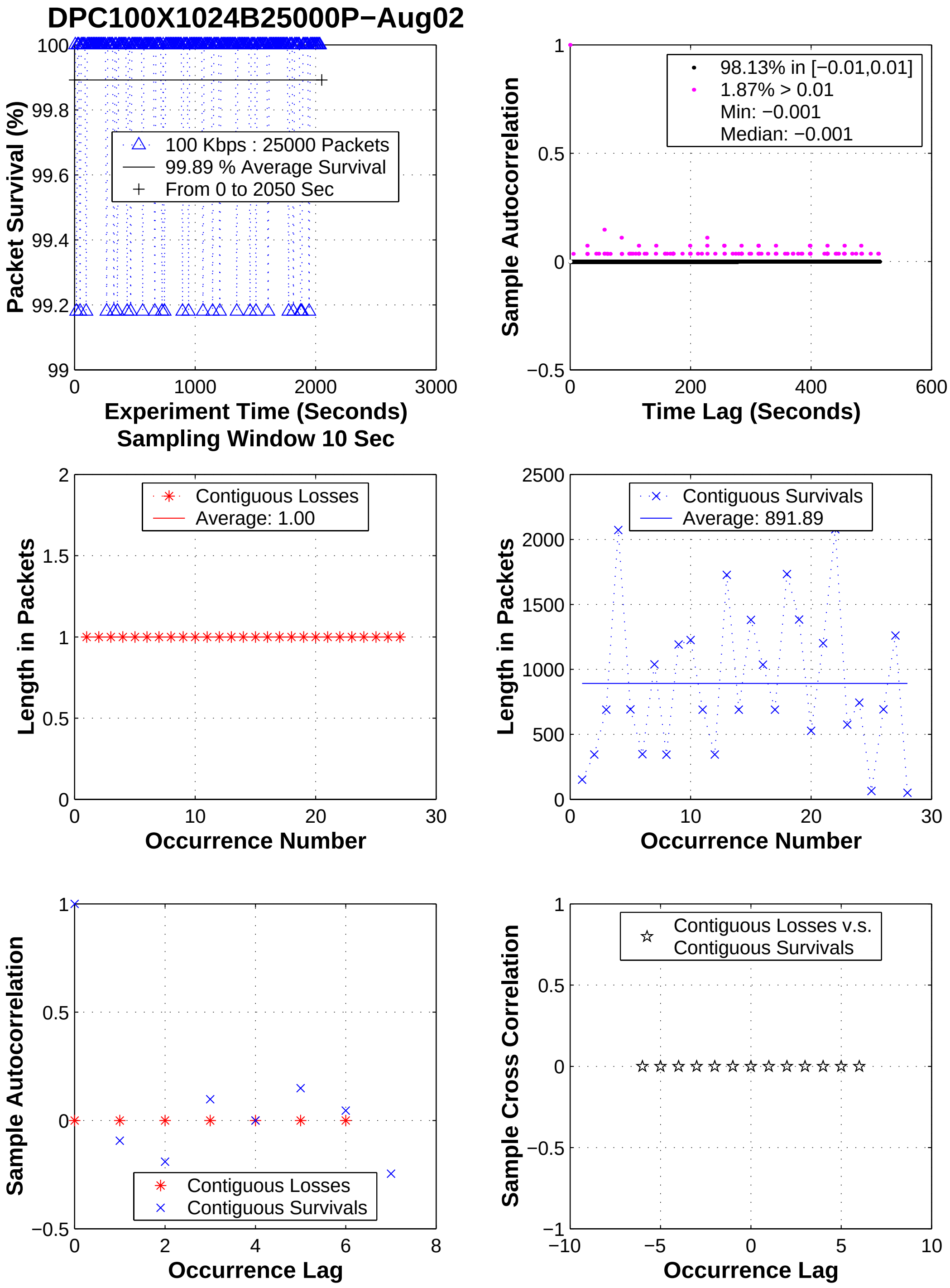


Figure A.22: Time series analysis: August 02, 2001 : HST 100 Kbits/sec transmission rate

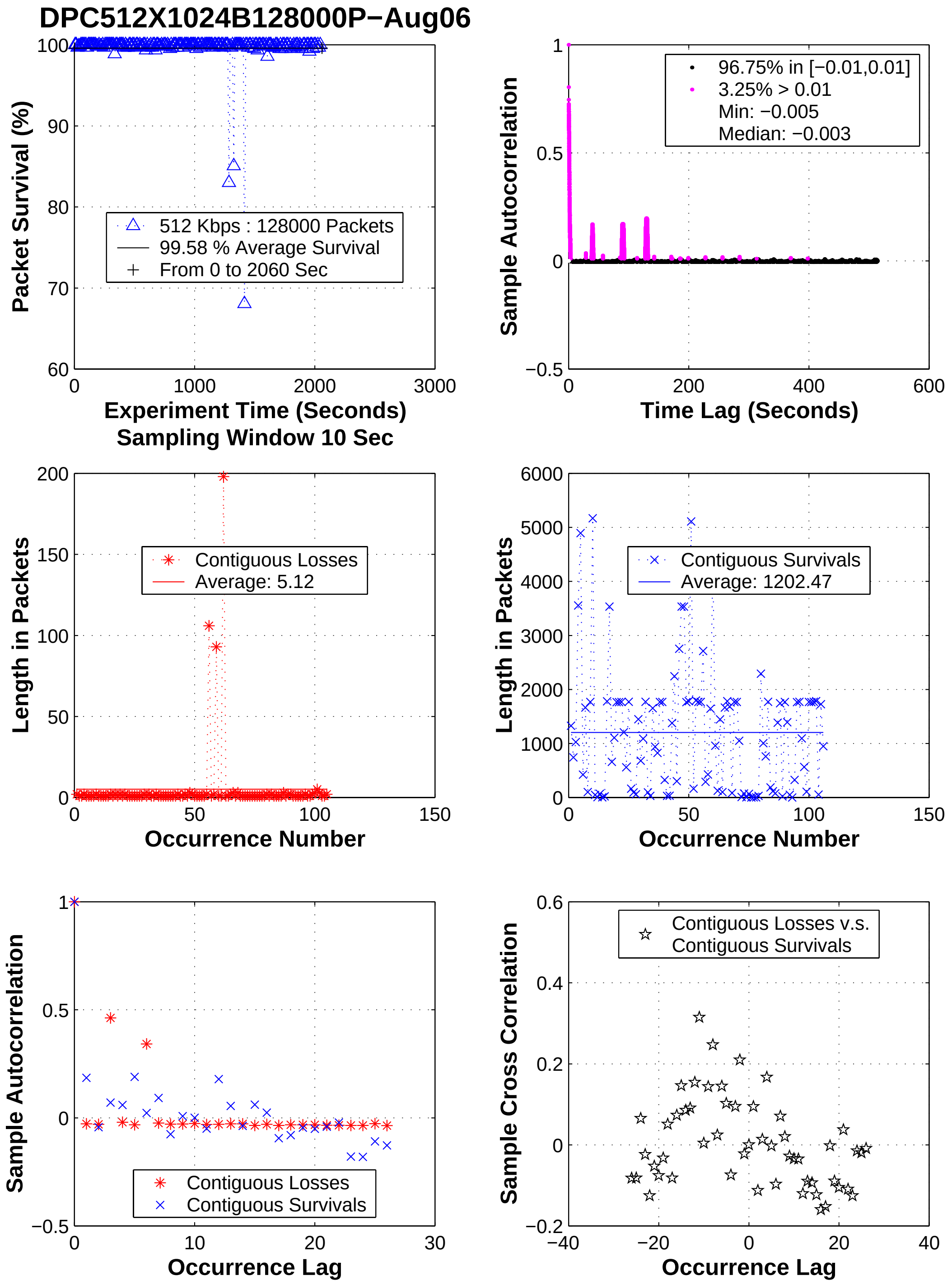


Figure A.23: Time series analysis: August 06, 2001: HST 512 Kbits/sec transmission rate

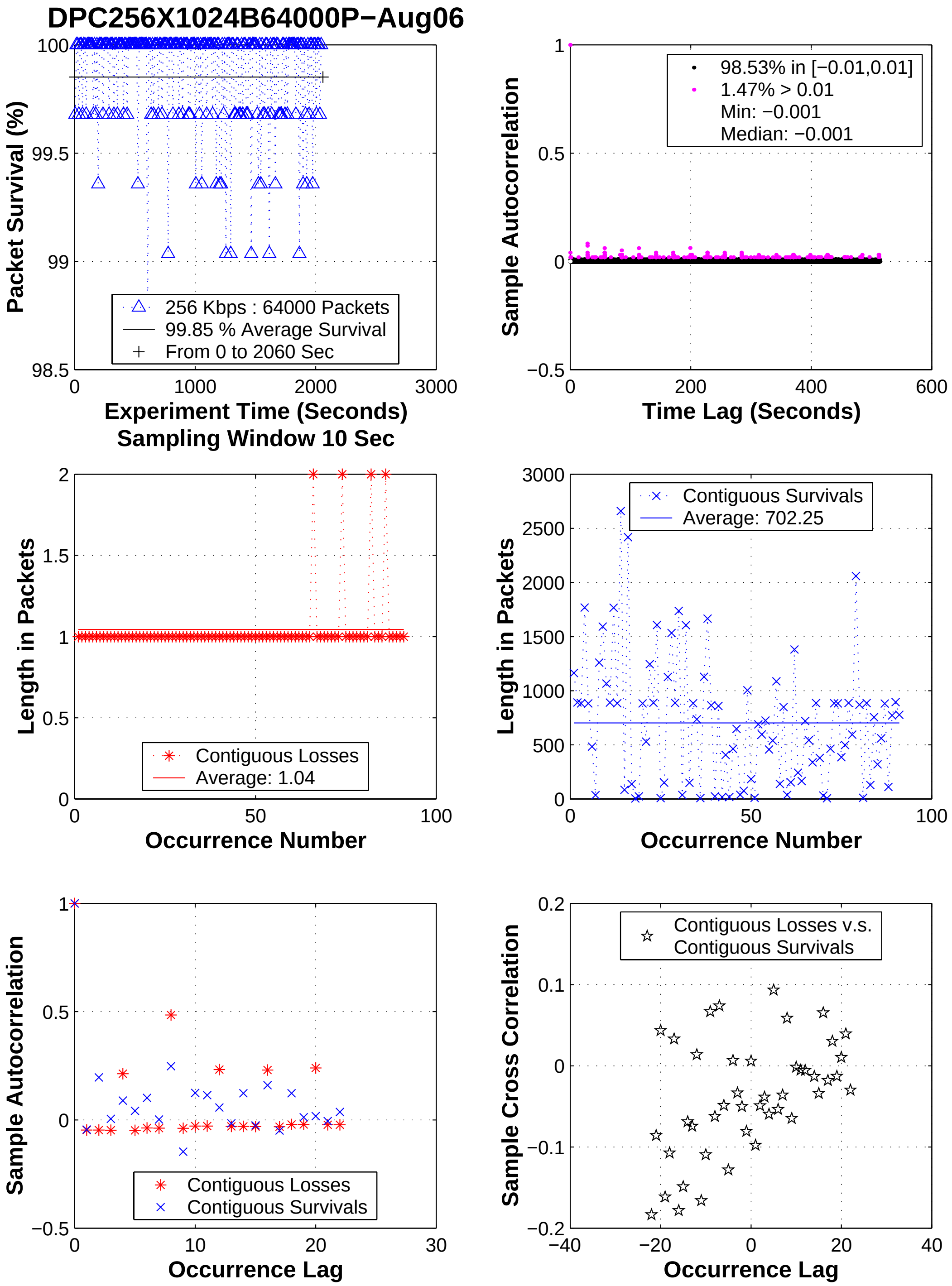


Figure A.24: Time series analysis: August 06, 2001 : HST 256 Kbits/sec transmission rate

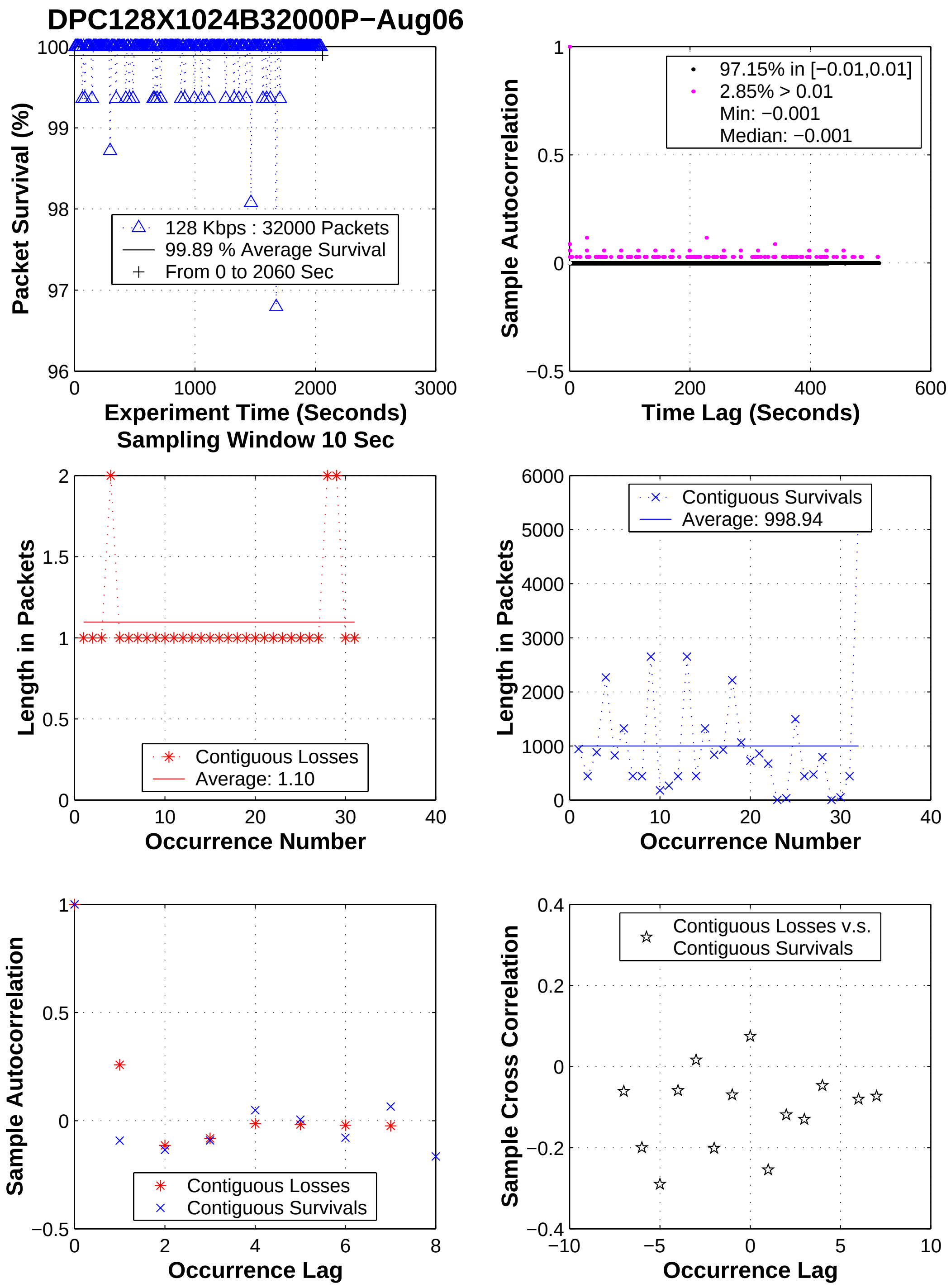


Figure A.25: Time series analysis: August 06, 2001 : HST 128 Kbits/sec transmission rate

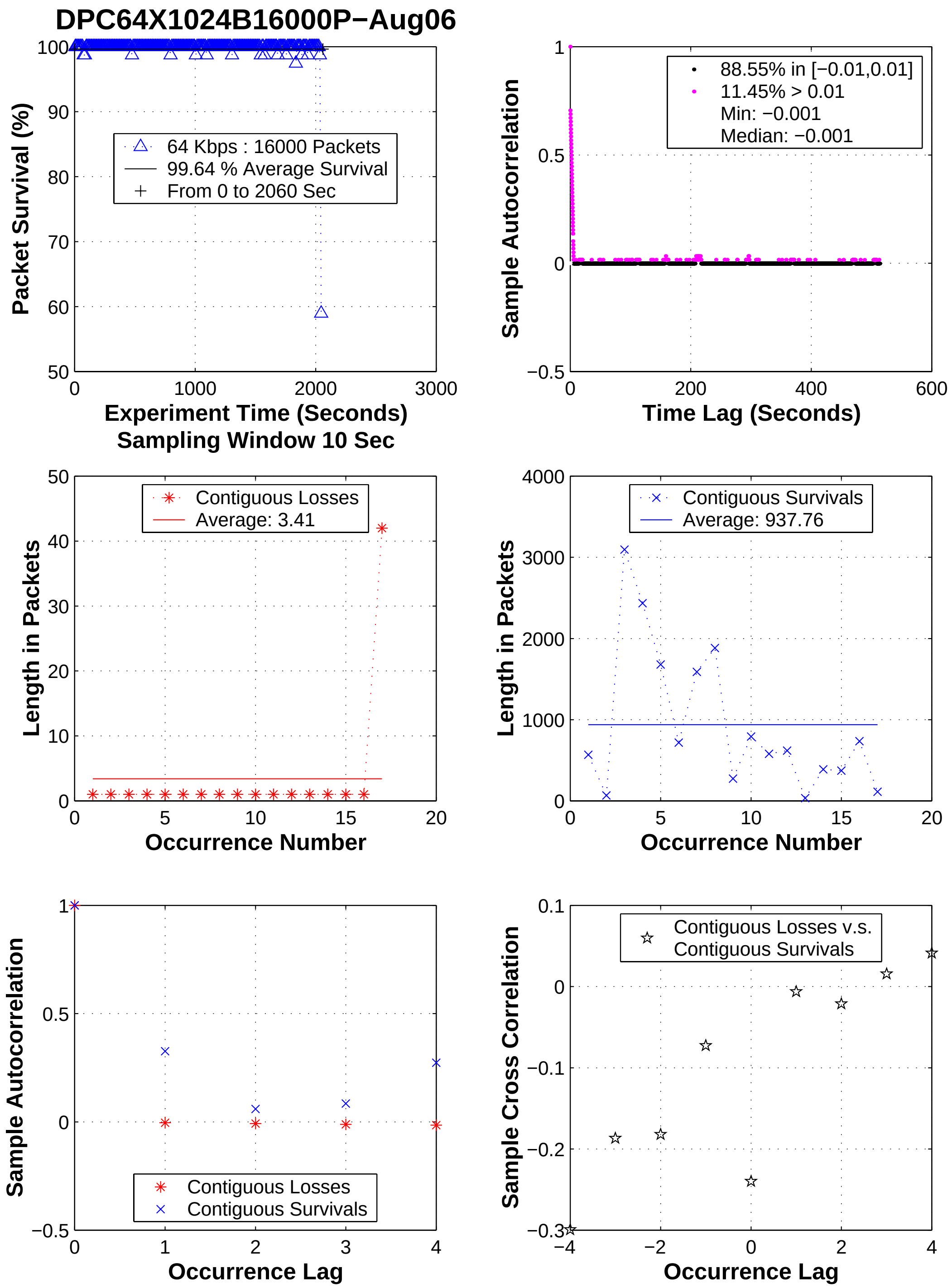


Figure A.26: Time series analysis: August 06, 2001 : HST 64 Kbits/sec transmission rate

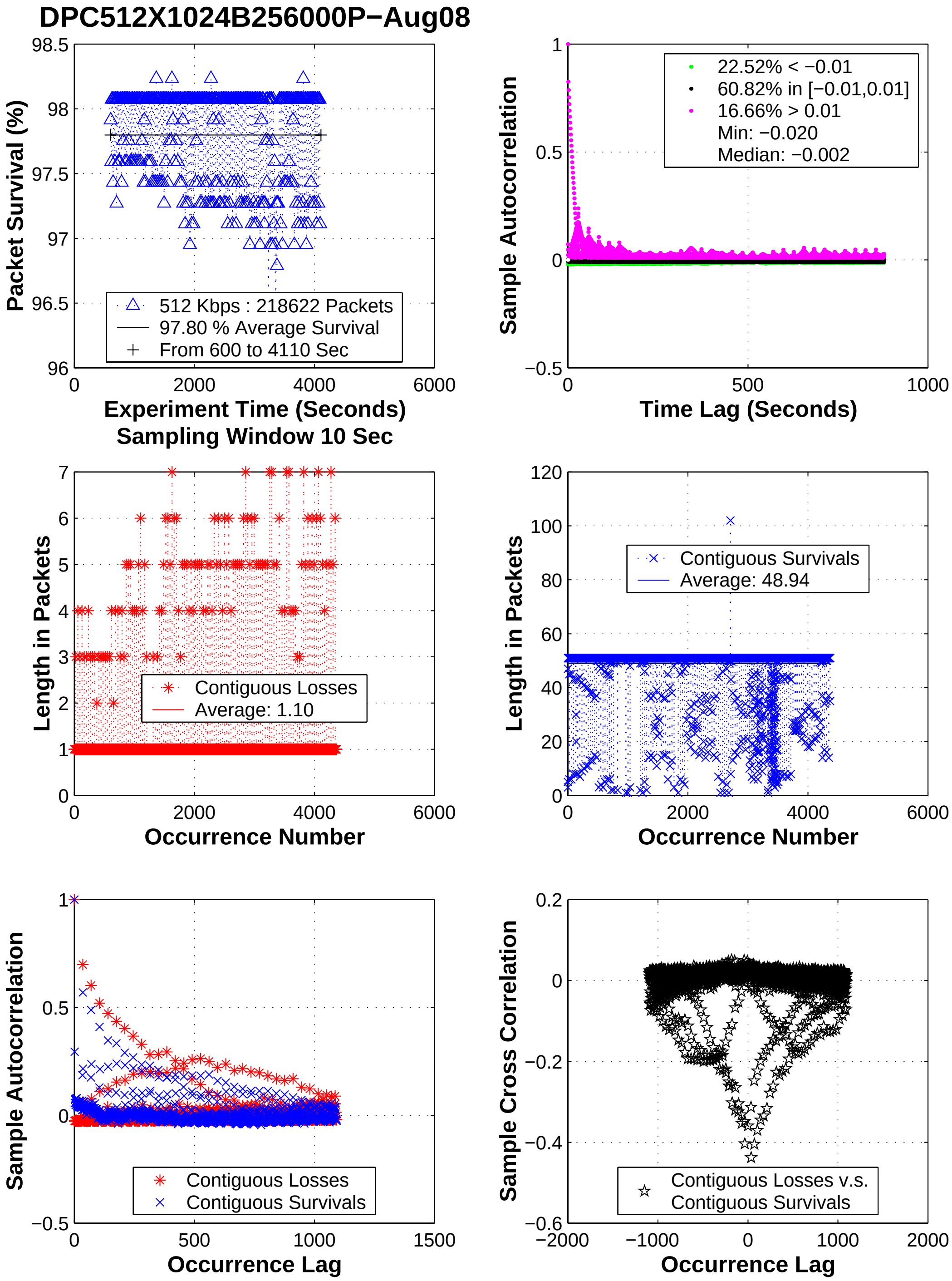


Figure A.27: Time series analysis: August 08, 2001: HST 512 Kbits/sec transmission rate

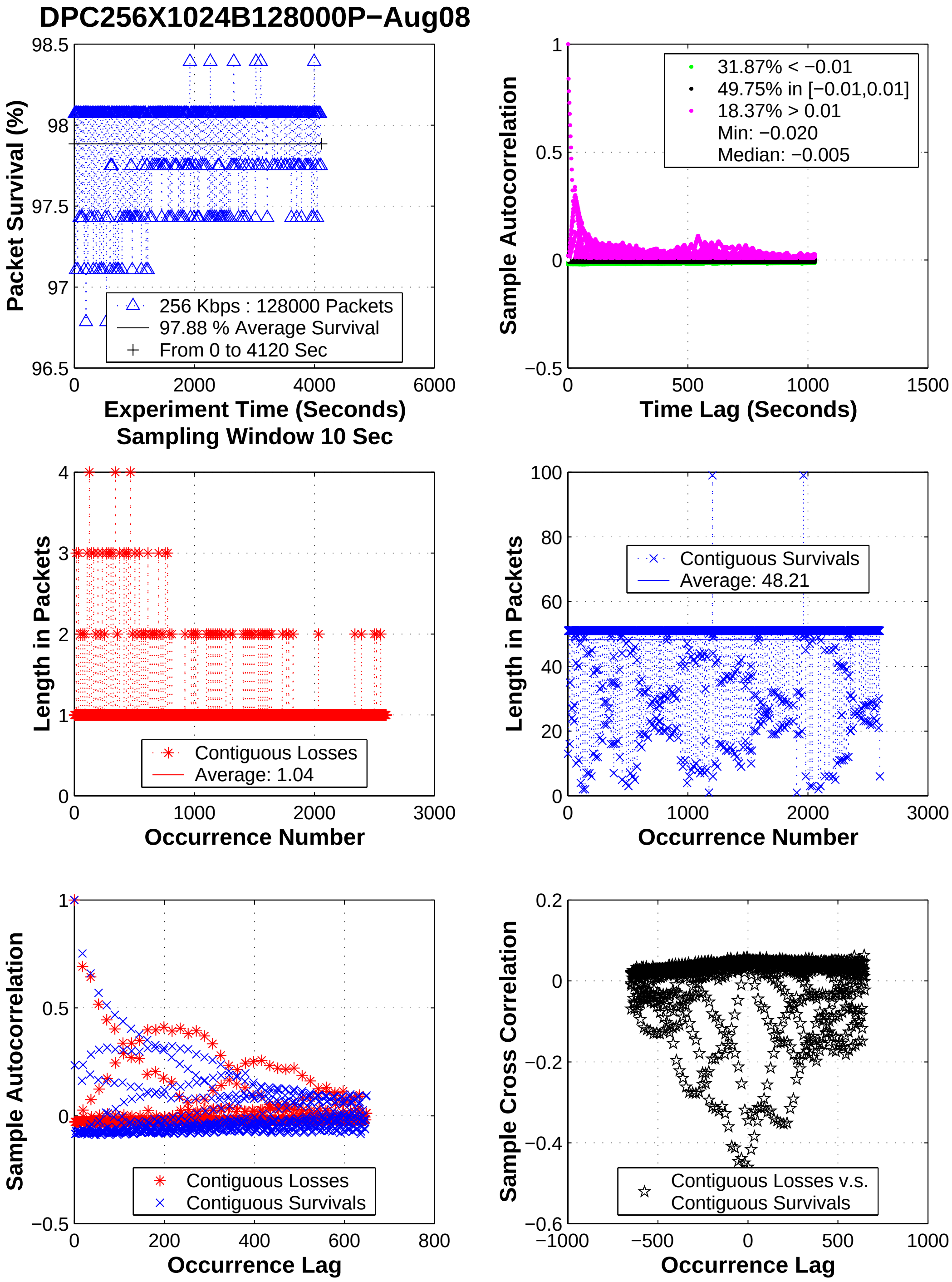


Figure A.28: Time series analysis: August 08, 2001: HST 256 Kbits/sec transmission rate

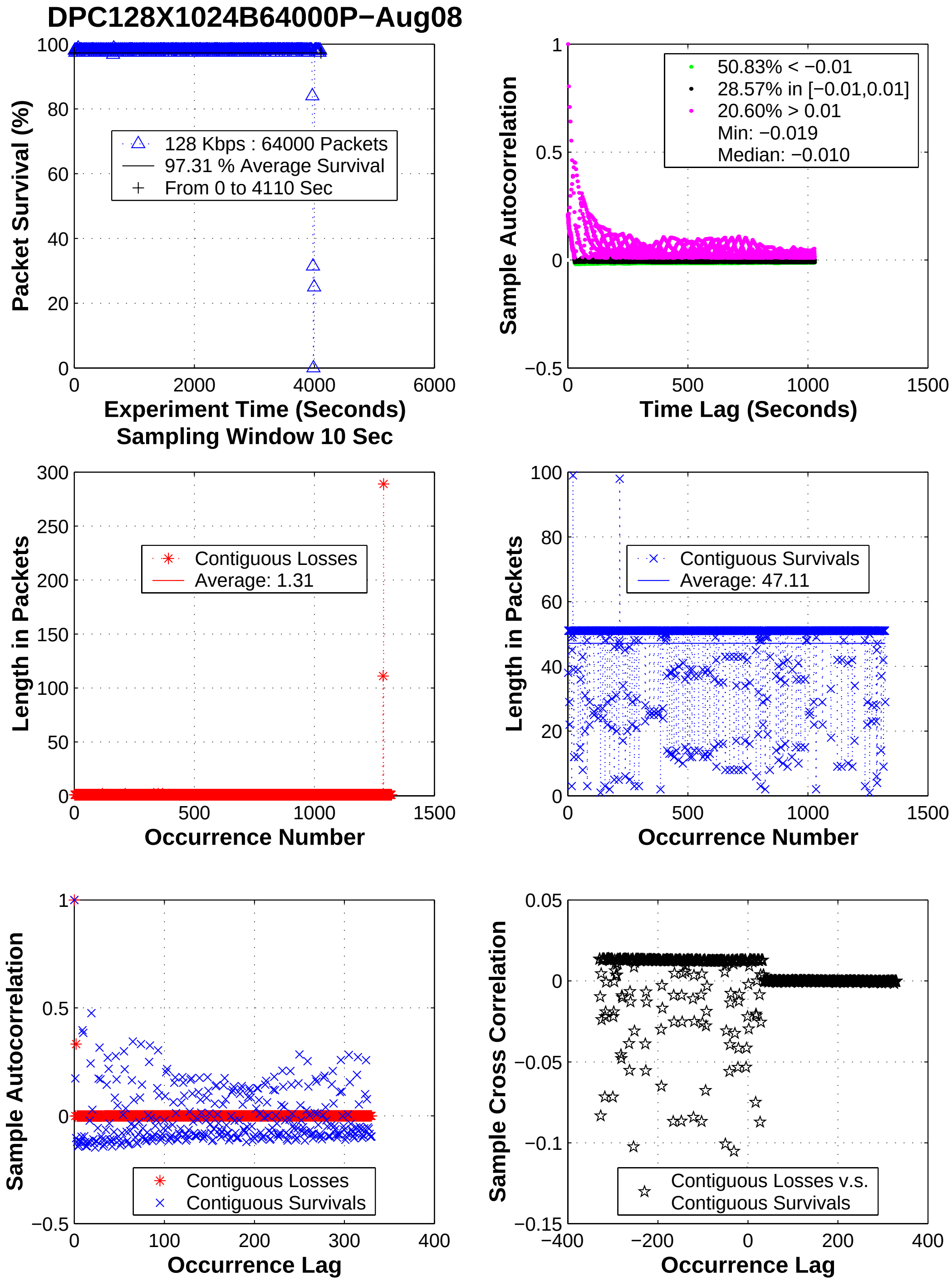


Figure A.29: Time series analysis: August 08, 2001: HST 128 Kbits/sec transmission rate

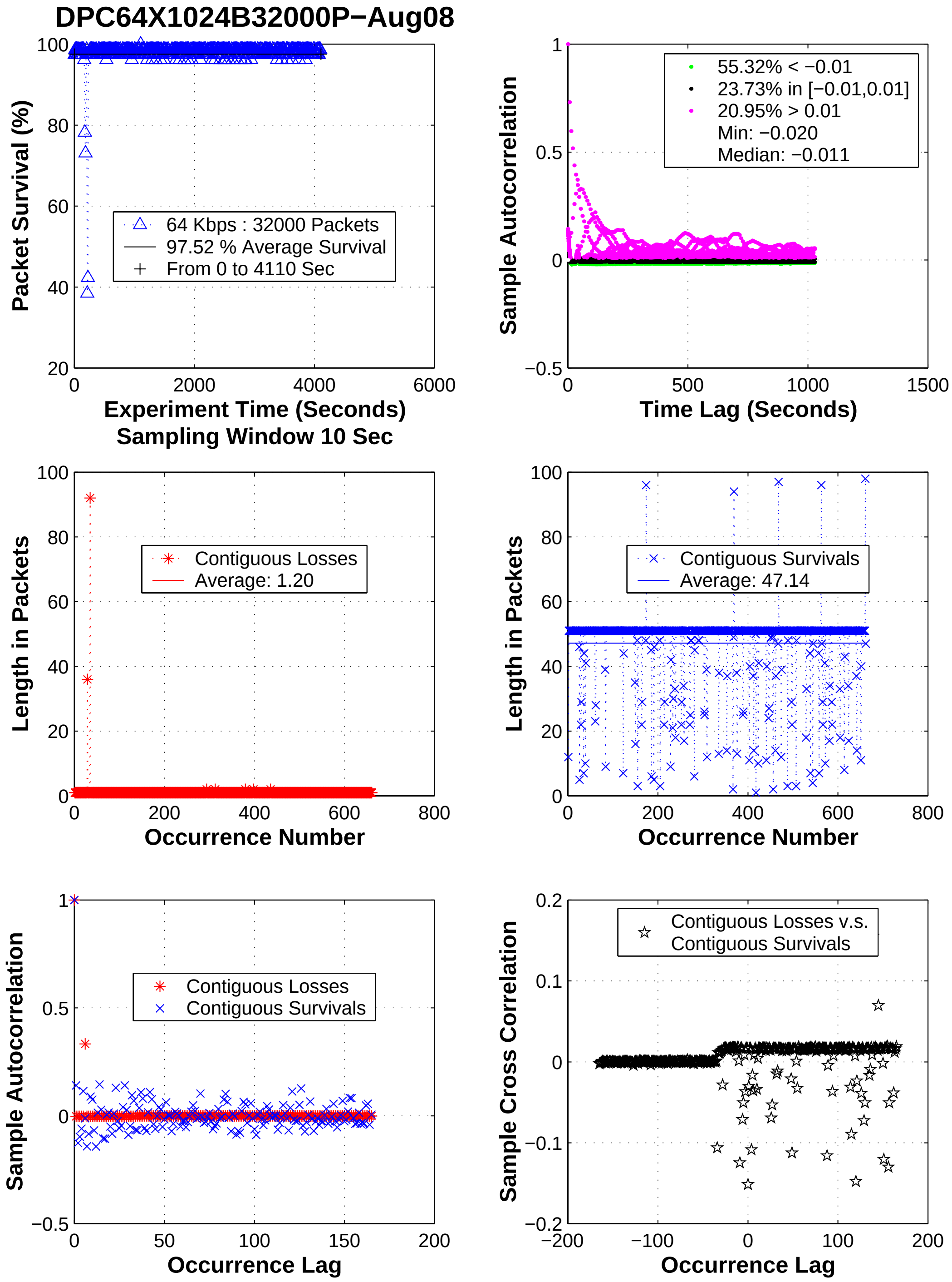


Figure A.30: Time series analysis: August 08, 2001: HST 64 Kbits/sec transmission rate

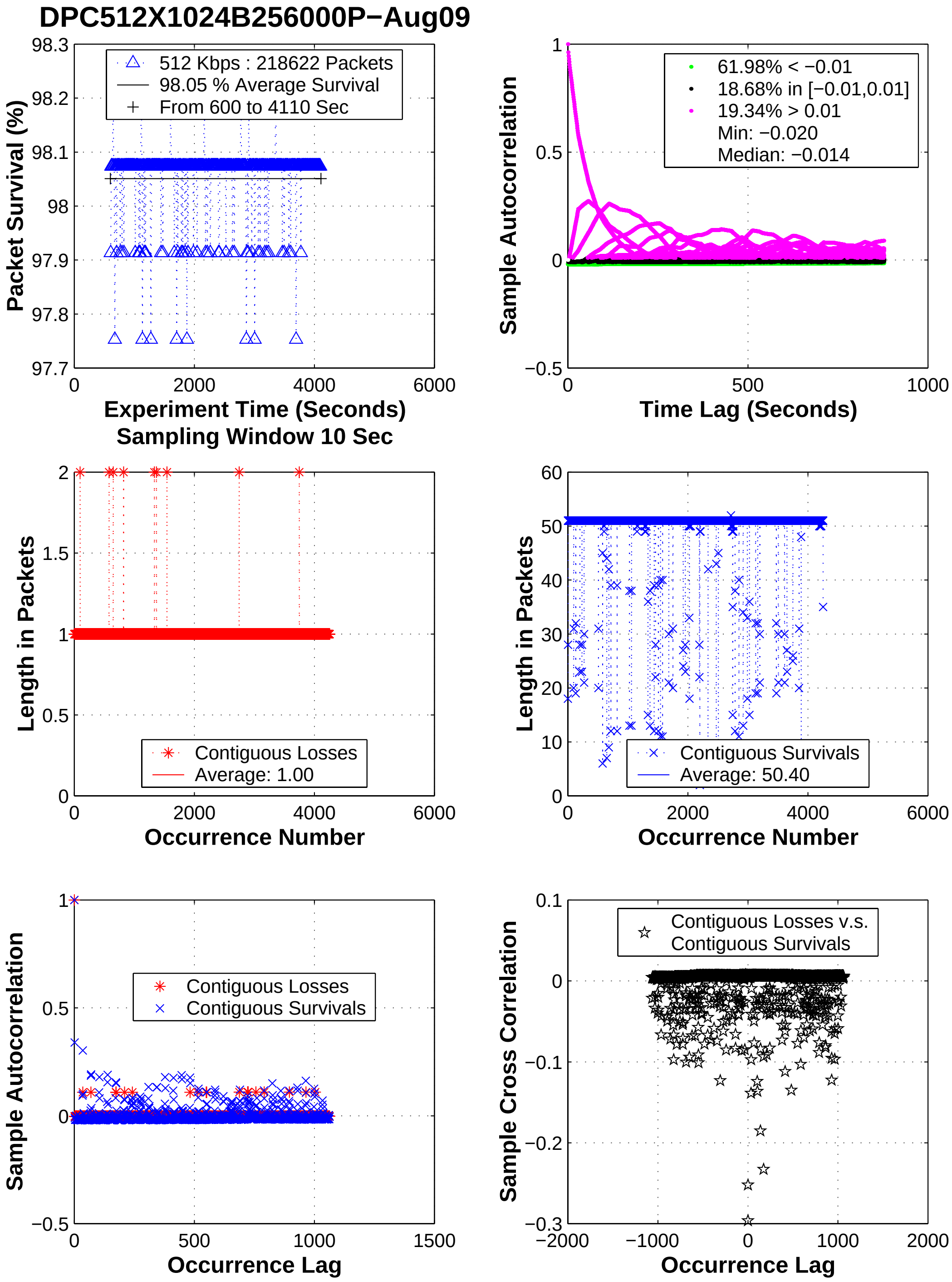


Figure A.31: Time series analysis: August 09, 2001: HST 512 Kbits/sec transmission rate

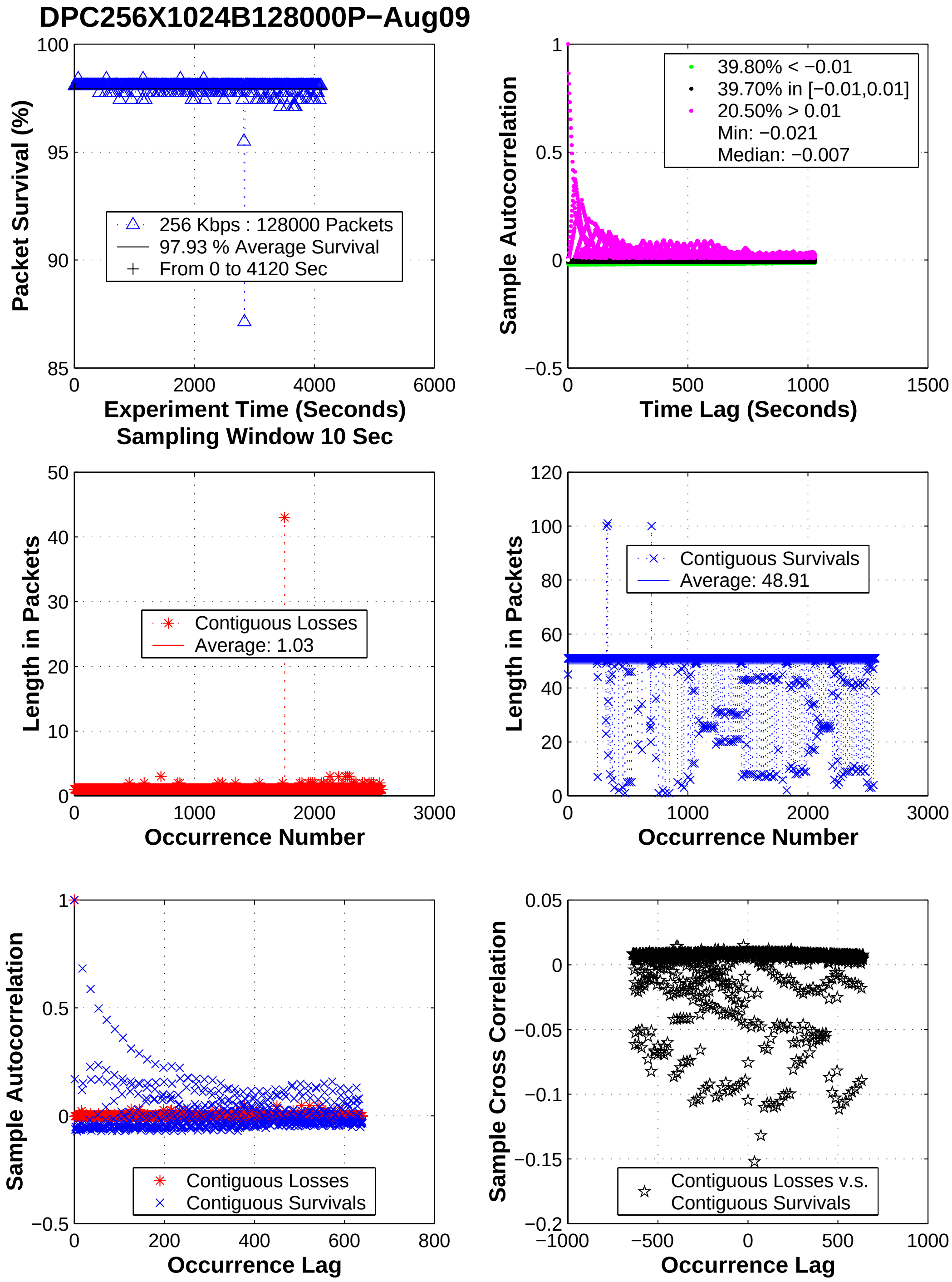


Figure A.32: Time series analysis: August 09, 2001: HST 256 Kbits/sec transmission rate

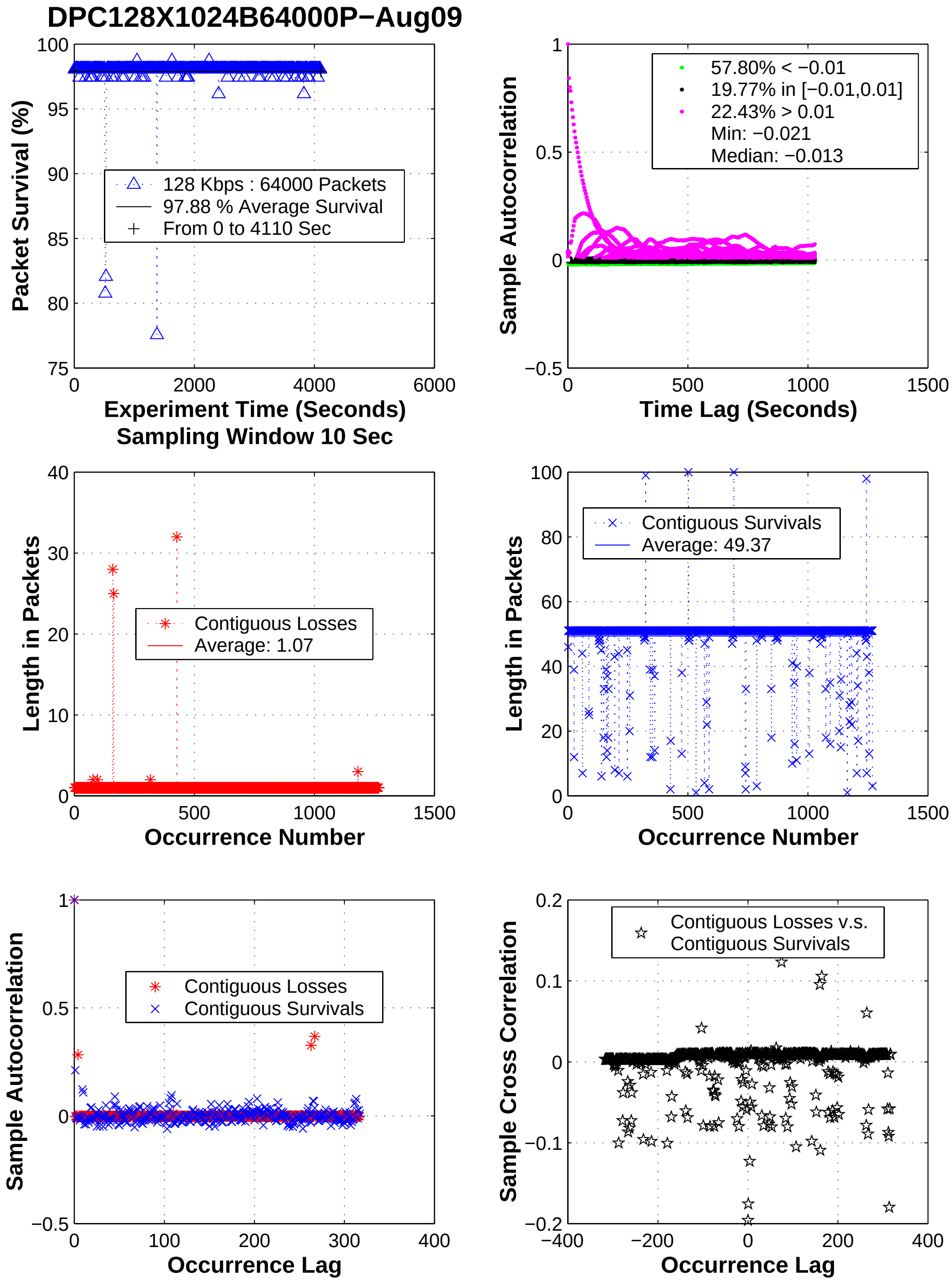


Figure A.33: Time series analysis: August 09, 2001: HST 128 Kbits/sec transmission rate

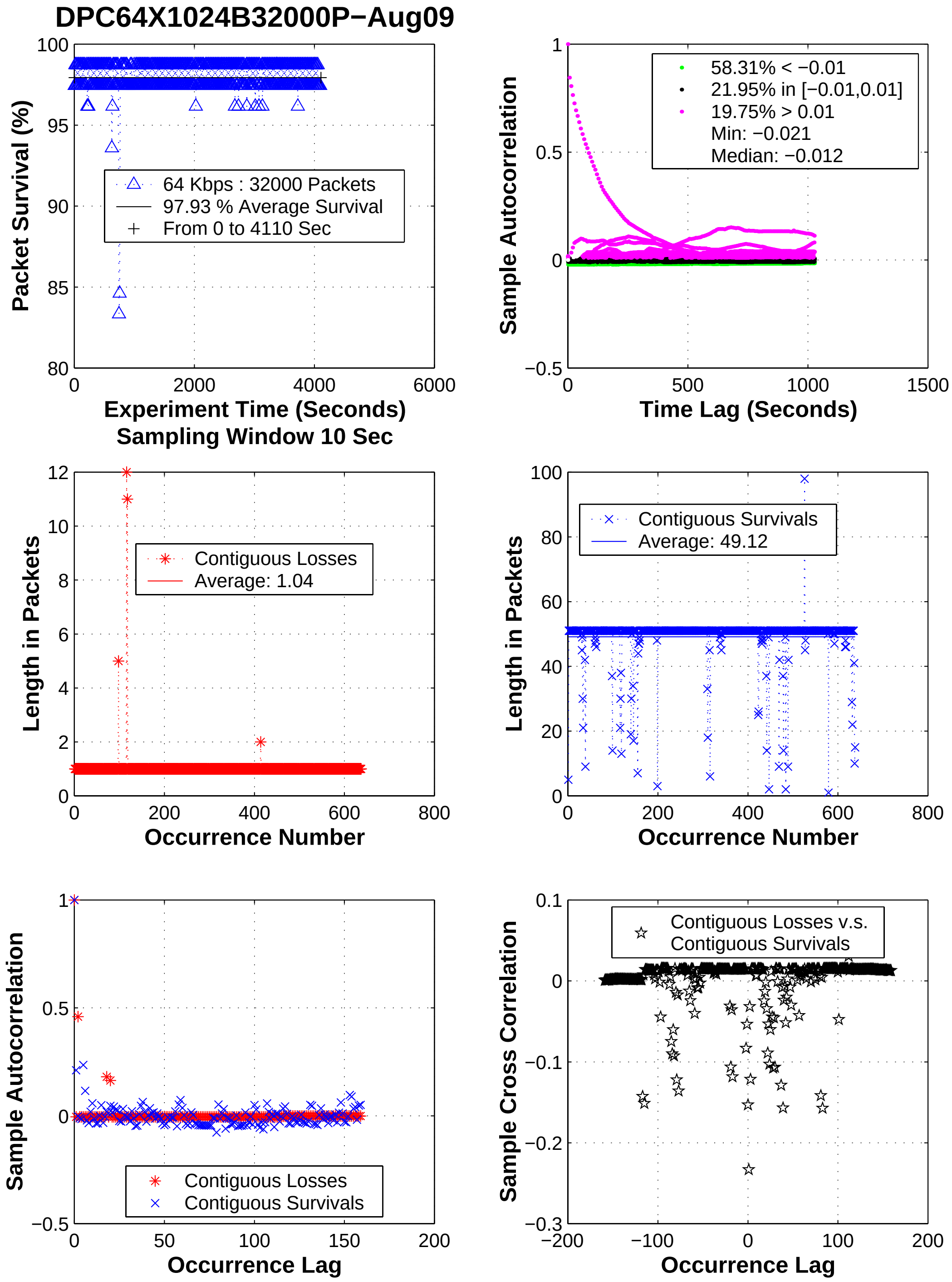


Figure A.34: Time series analysis: August 09, 2001: HST 64 Kbits/sec transmission rate

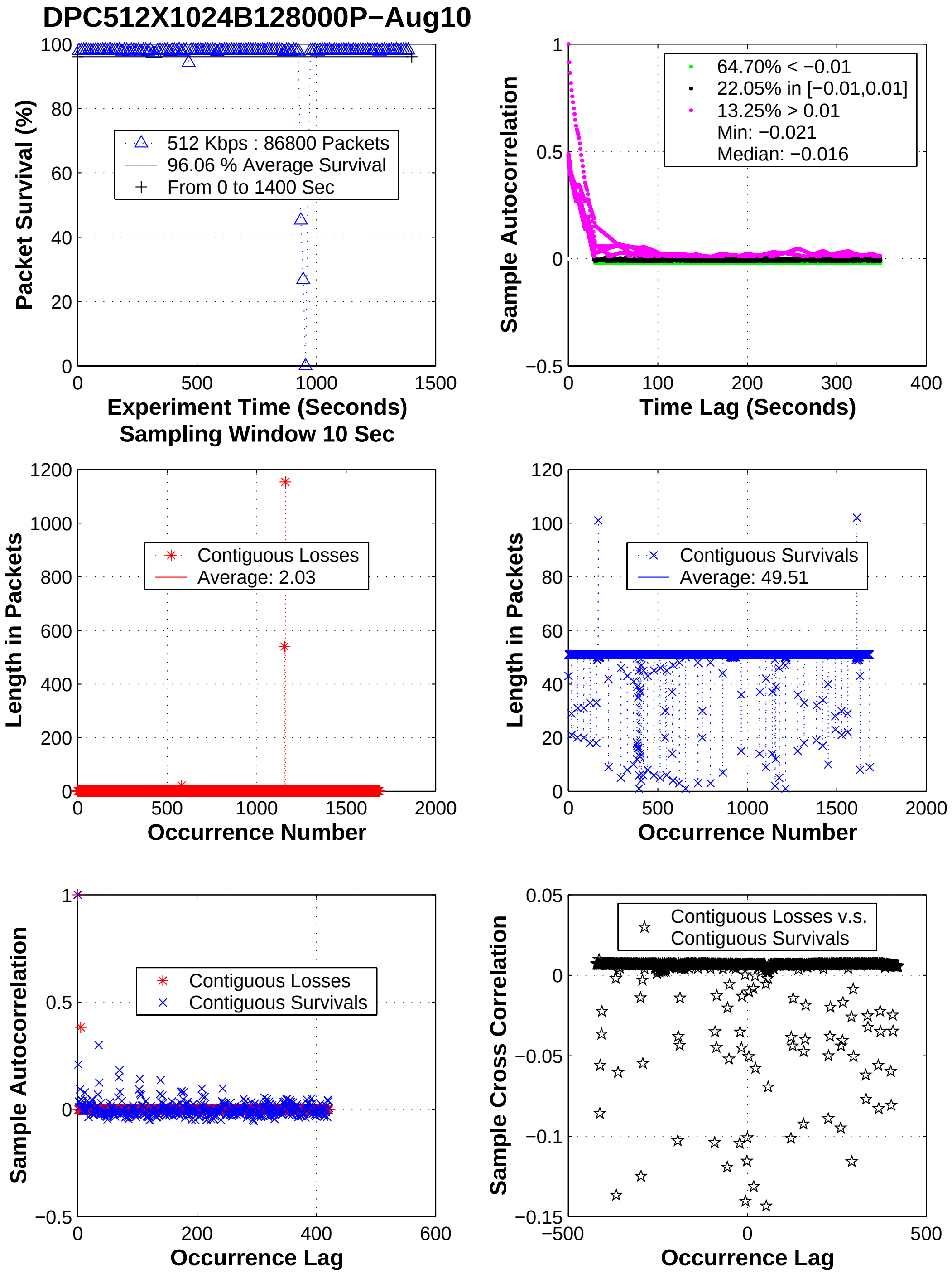


Figure A.35: Time series analysis: August 10, 2001 : HST 512 Kbits/sec transmission rate

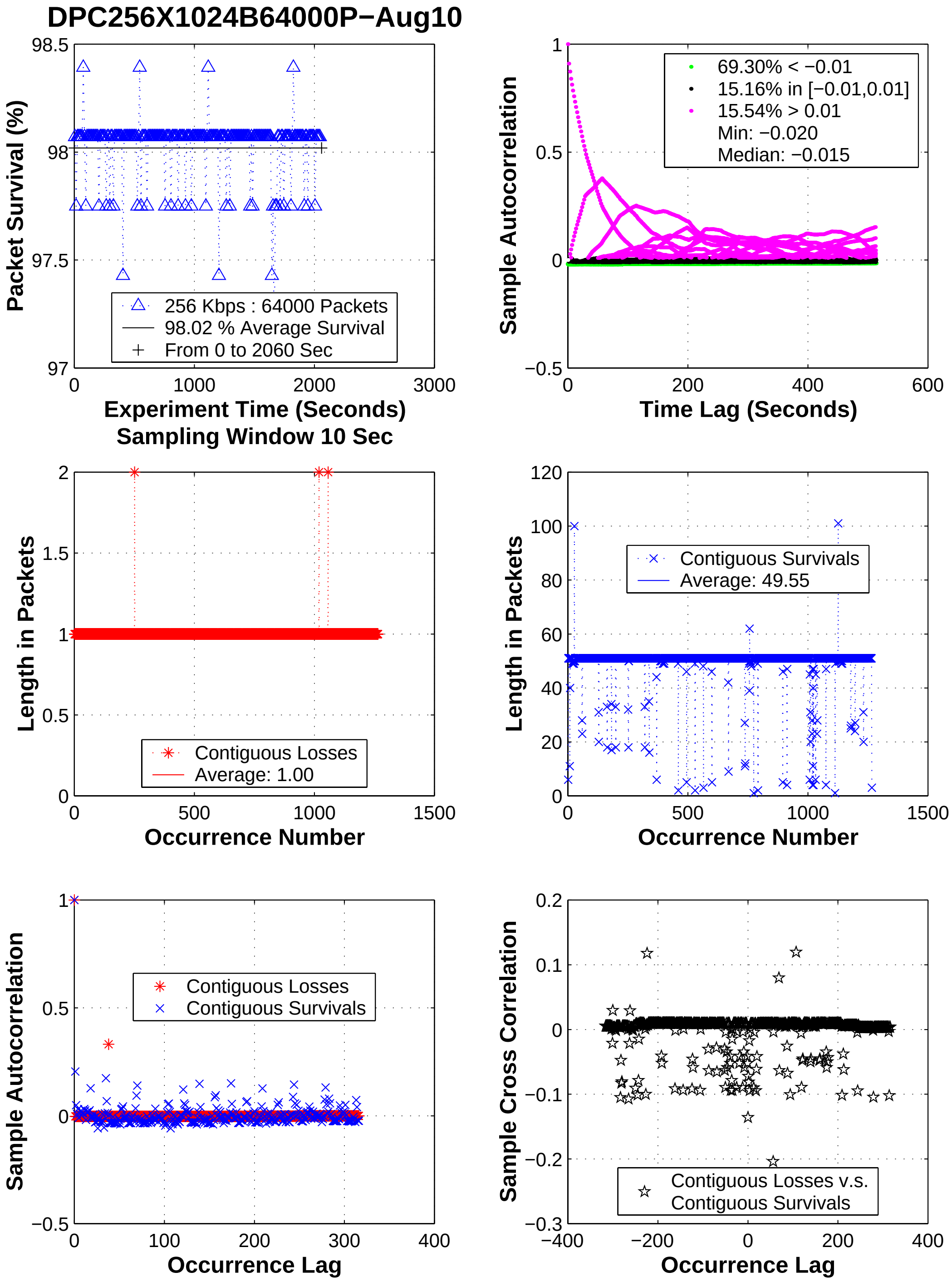


Figure A.36: Time series analysis: August 10, 2001 : HST 256 Kbits/sec transmission rate

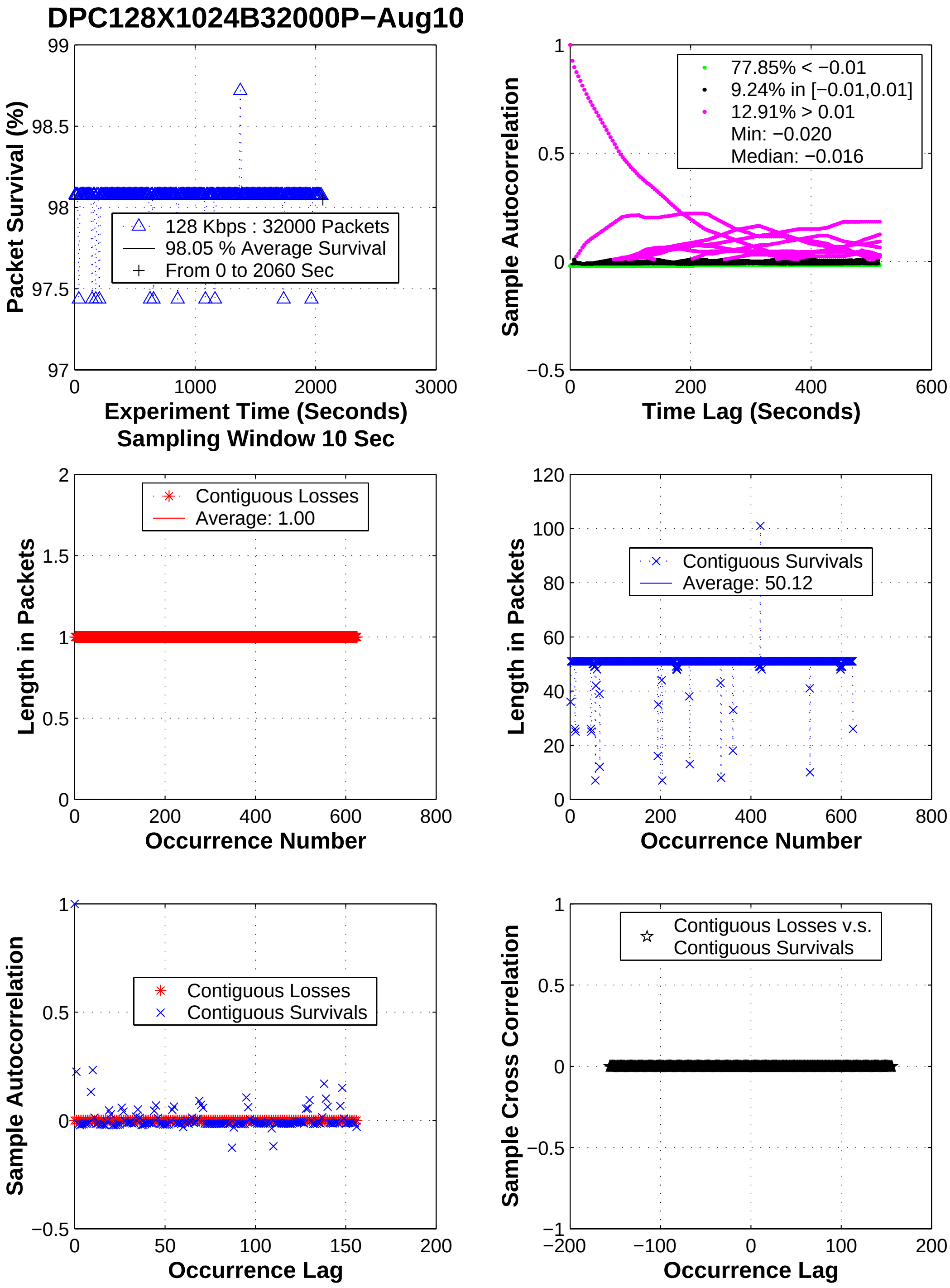


Figure A.37: Time series analysis: August 10, 2001 : HST 128 Kbits/sec transmission rate

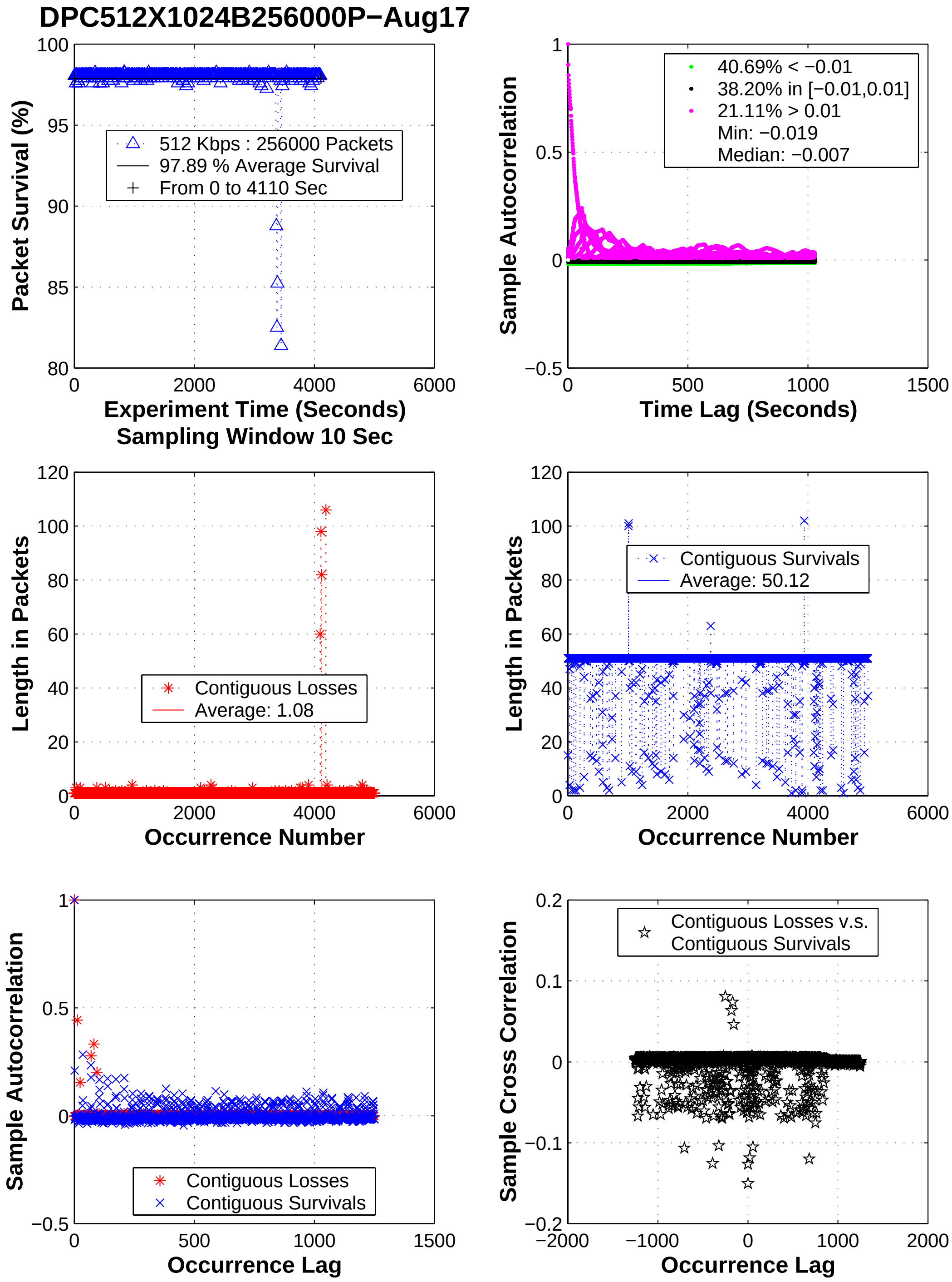


Figure A.38: Time series analysis: August 17, 2001 : HST 512 Kbits/sec transmission rate

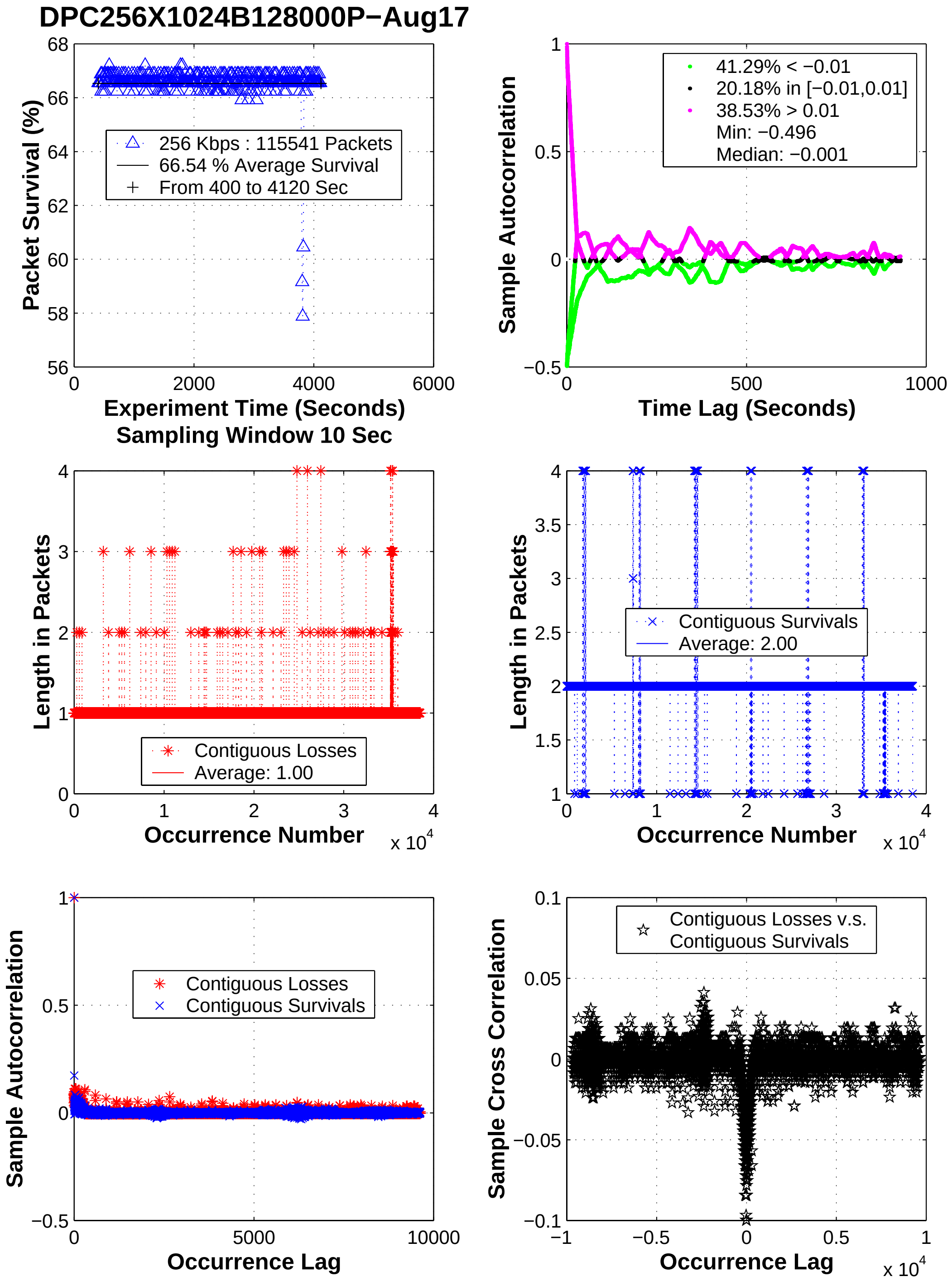


Figure A.39: Time series analysis: August 17, 2001 : HST 256 Kbits/sec transmission rate

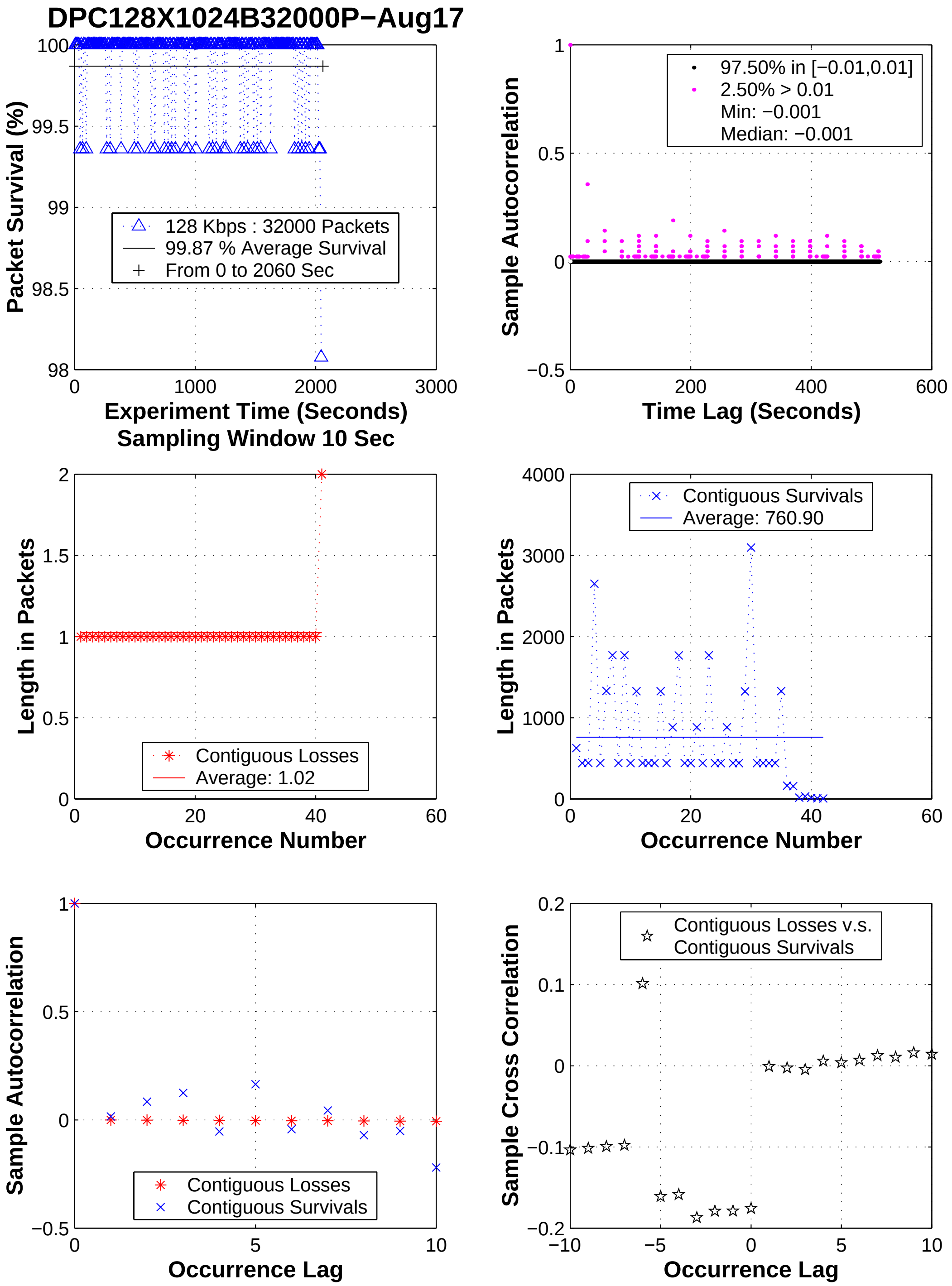


Figure A.40: Time series analysis: August 17, 2001 : HST 128 Kbits/sec transmission rate

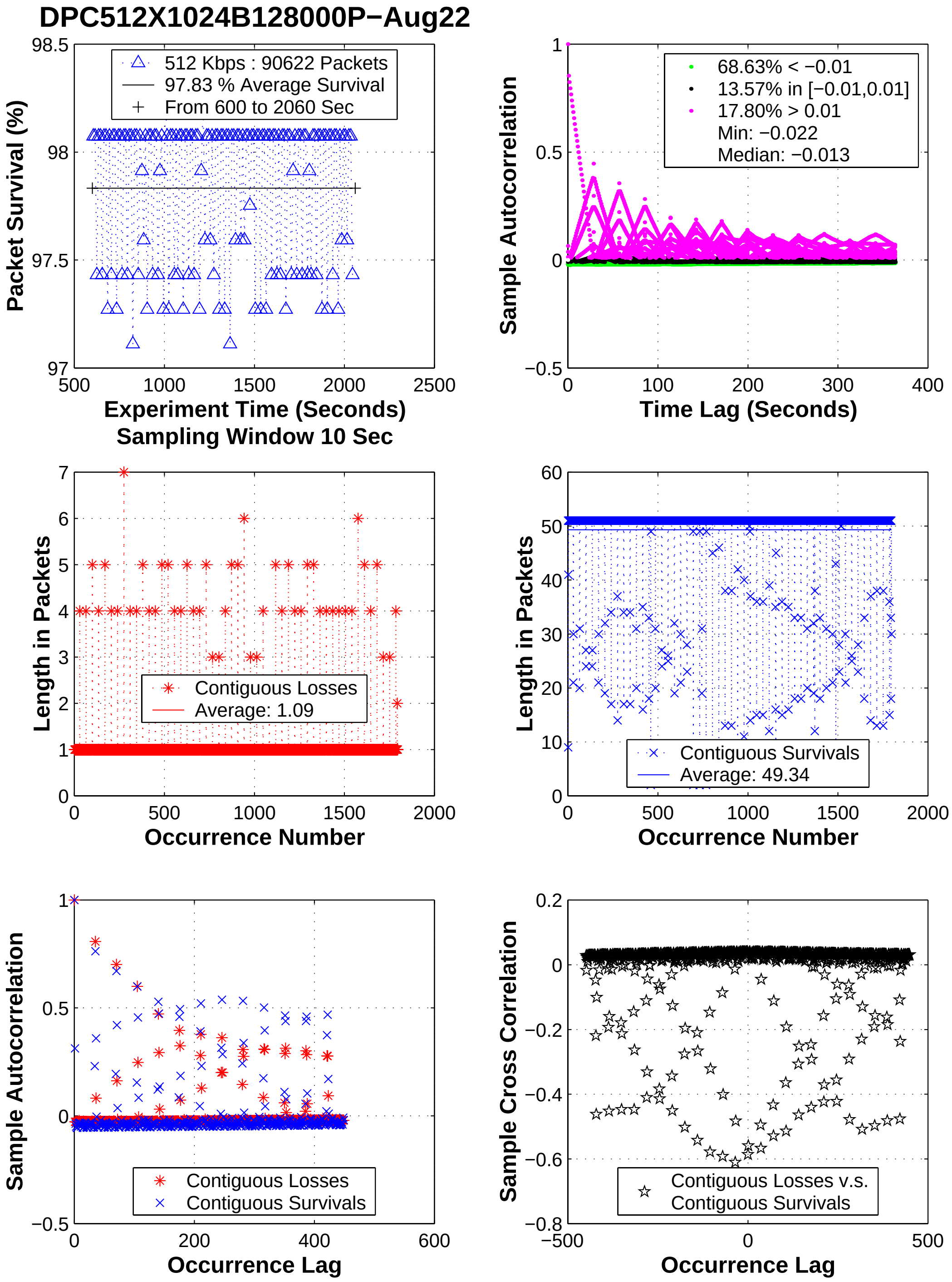


Figure A.41: Time series analysis: August 22, 2001 : HST 512 Kbits/sec transmission rate

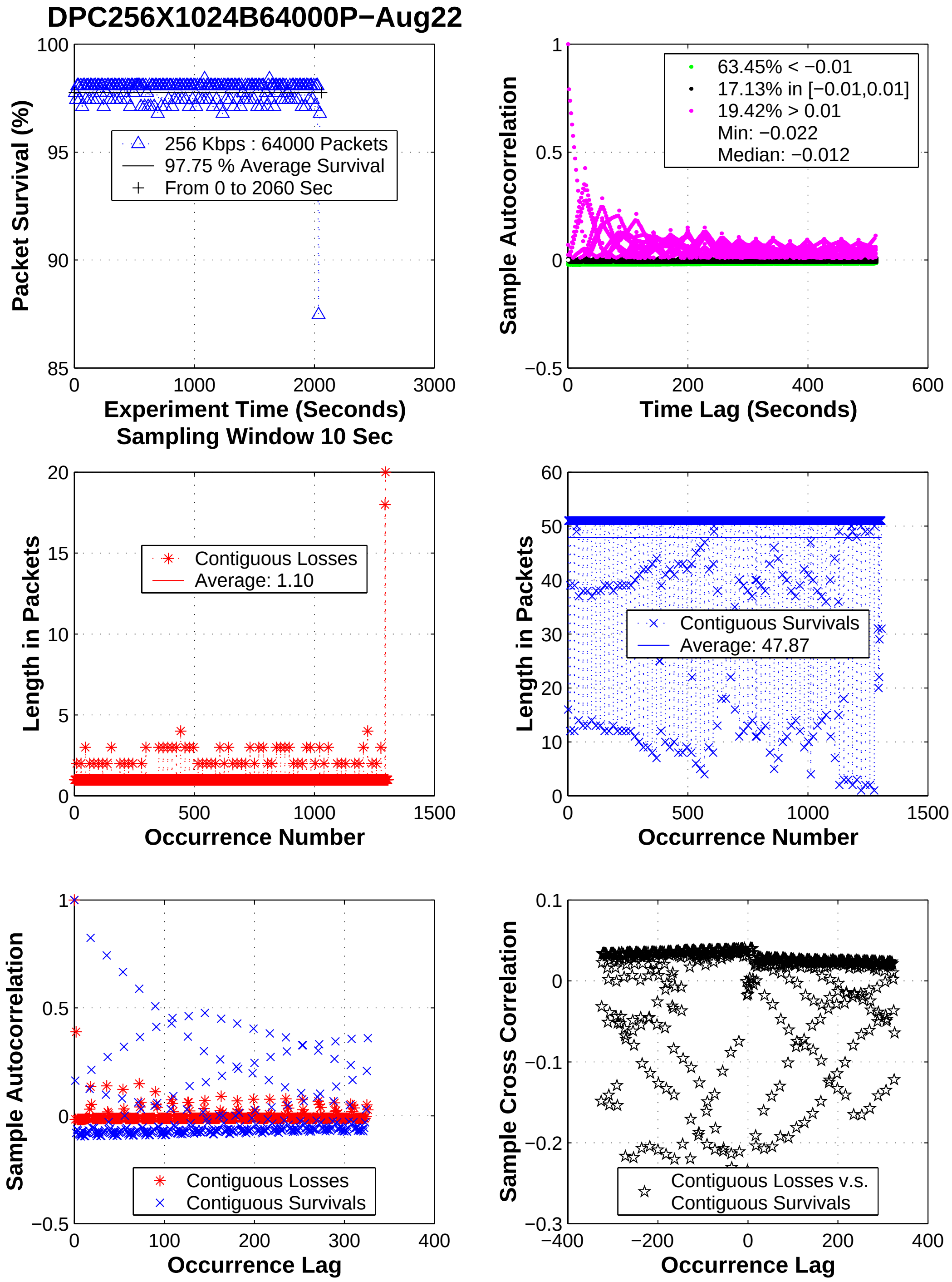


Figure A.42: Time series analysis: August 22, 2001 : HST 256 Kbits/sec transmission rate

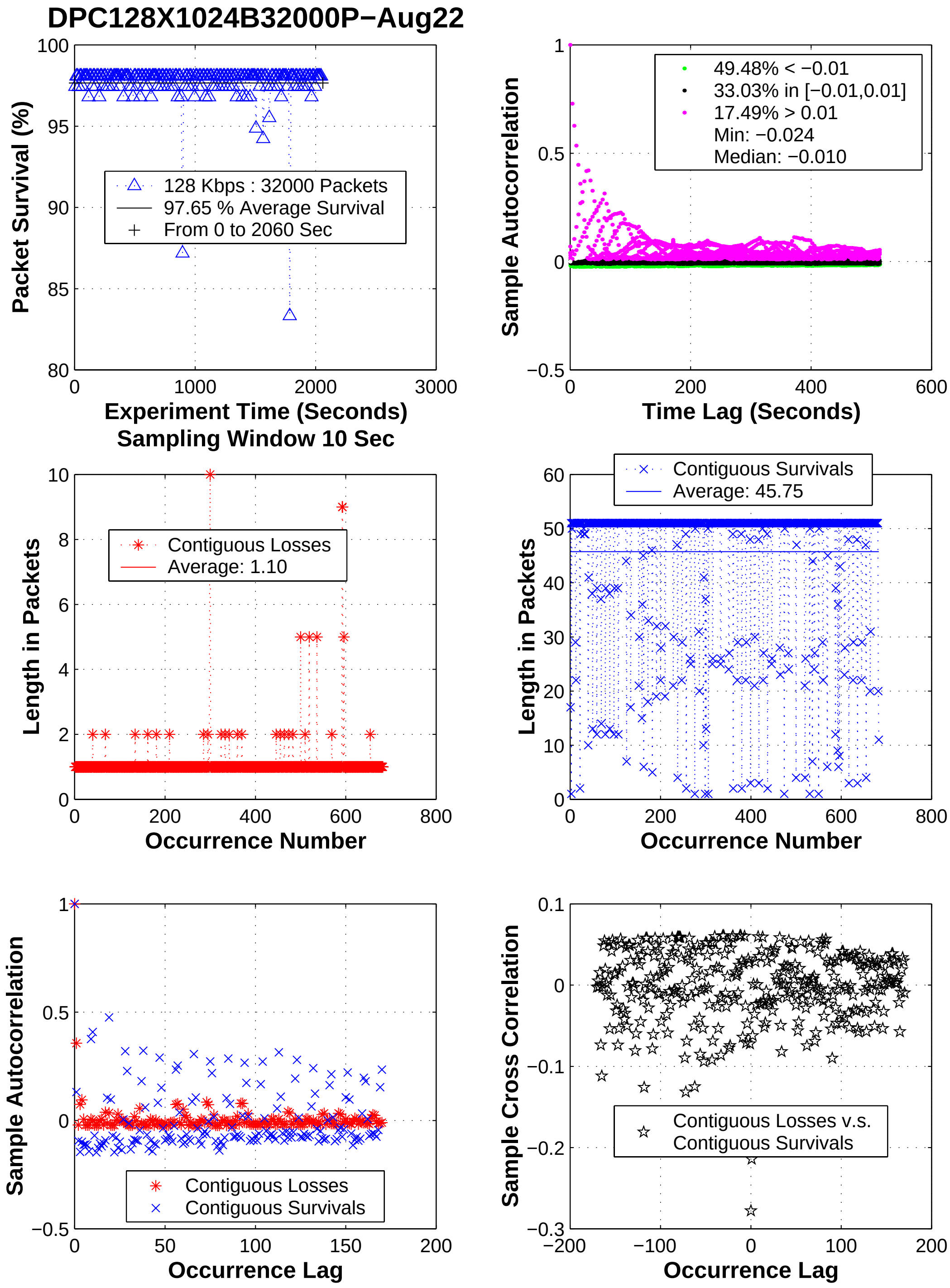


Figure A.43: Time series analysis: August 22, 2001 : HST 128 Kbits/sec transmission rate

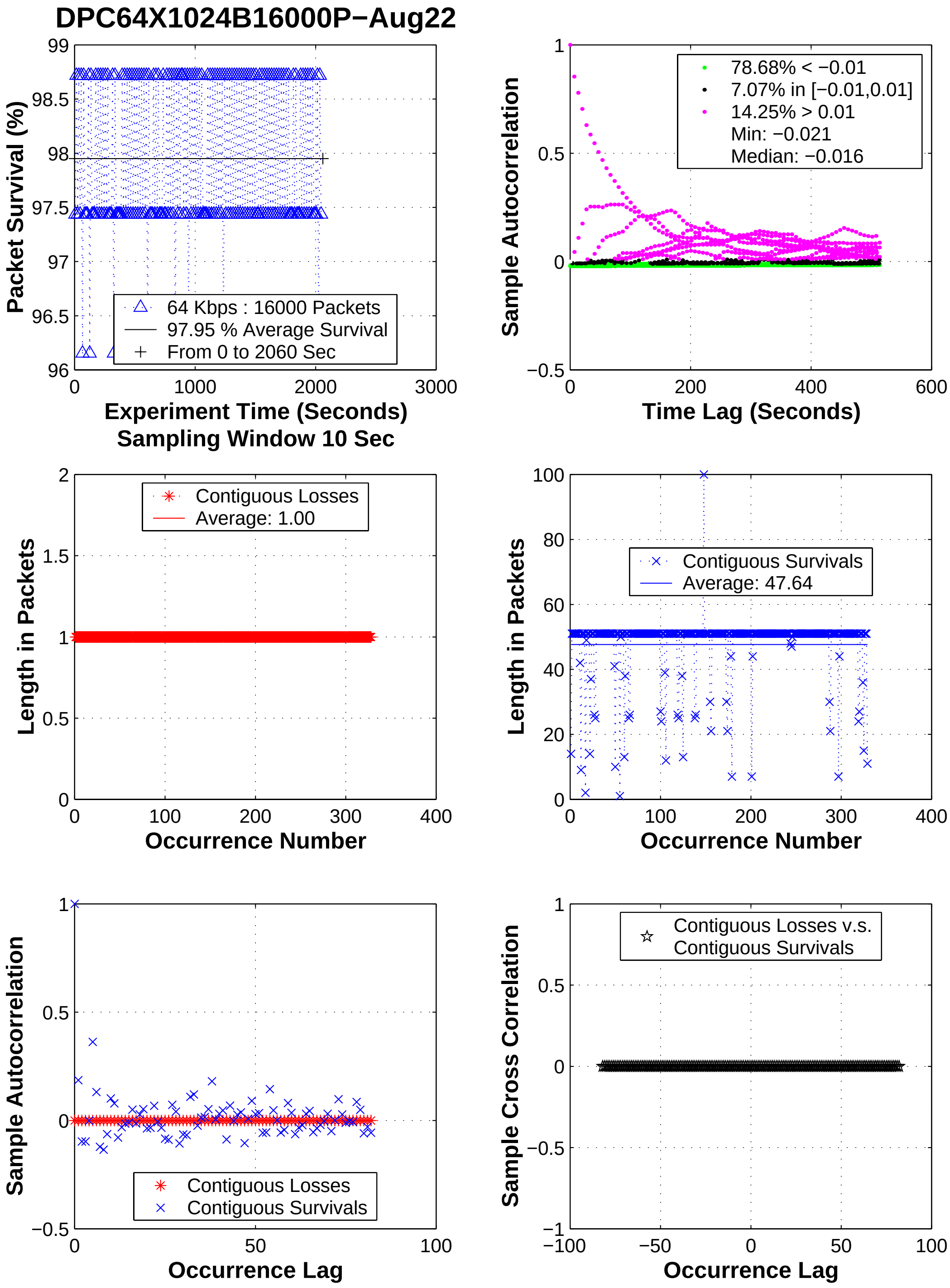


Figure A.44: Time series analysis: August 22, 2001 : HST 64 Kbits/sec transmission rate

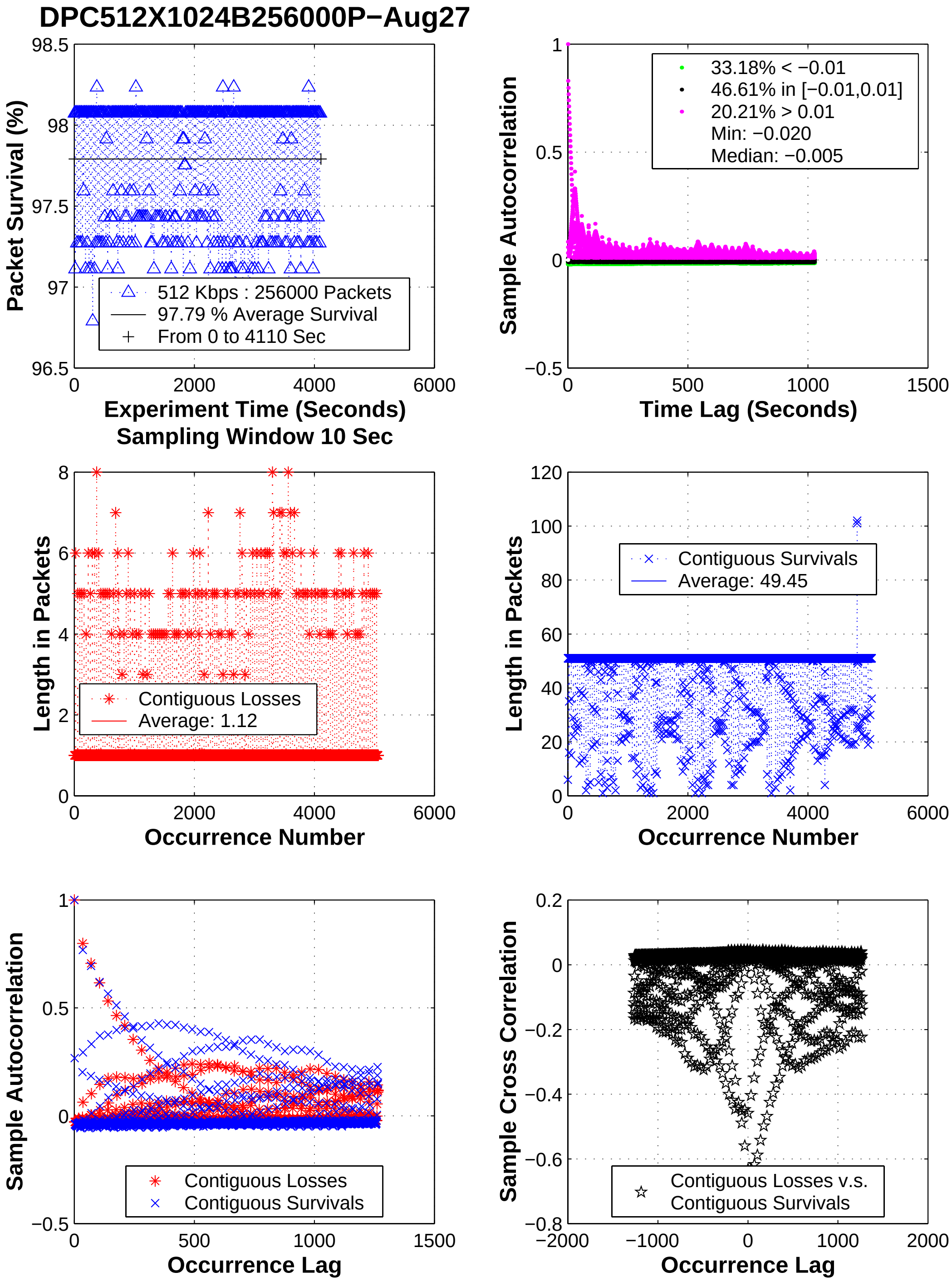


Figure A.45: Time series analysis: August 27, 2001 : HST 512 Kbits/sec transmission rate

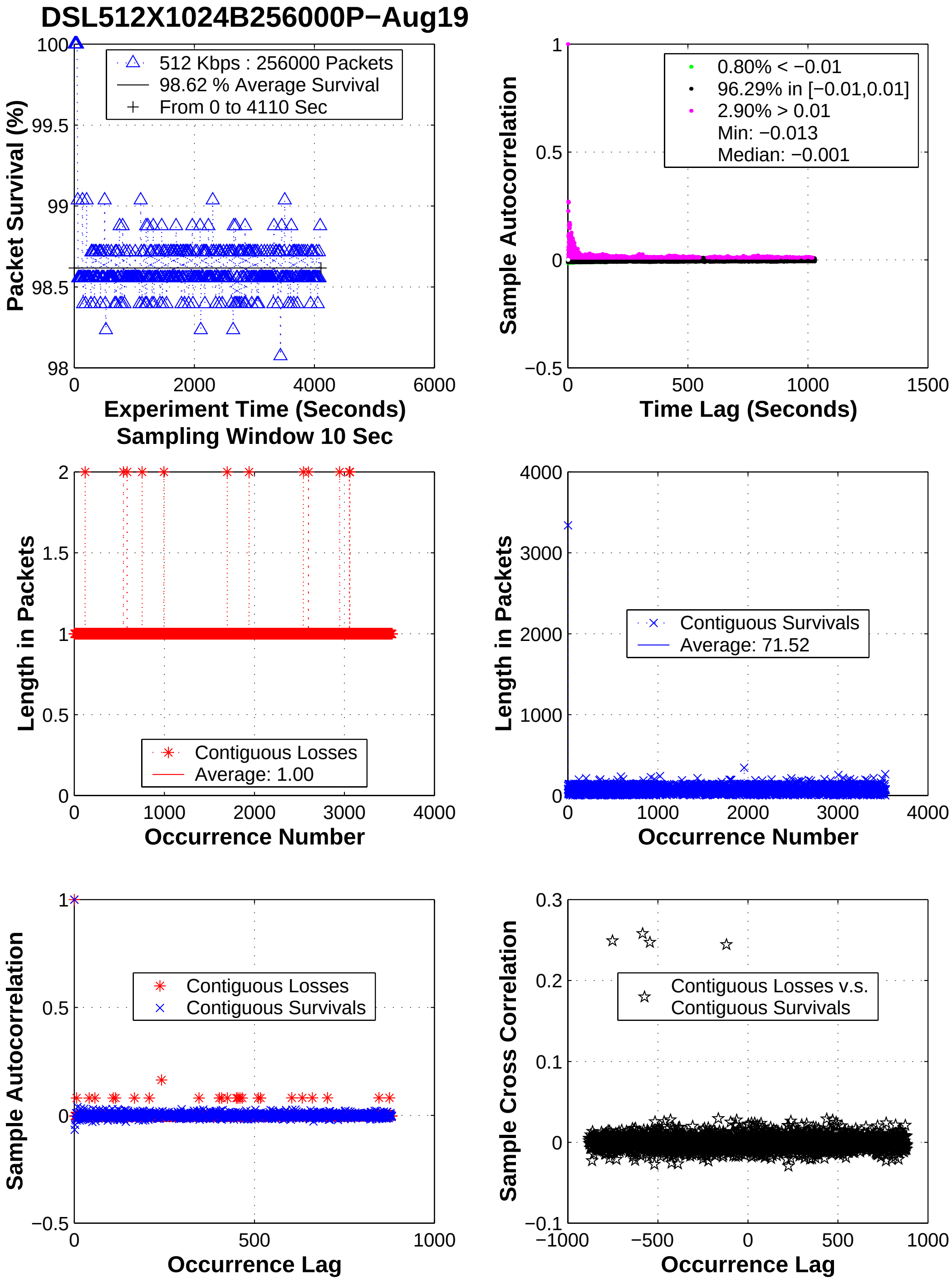


Figure A.46: Time series analysis: August 19, 2001 : DSL 512 Kbits/sec transmission rate

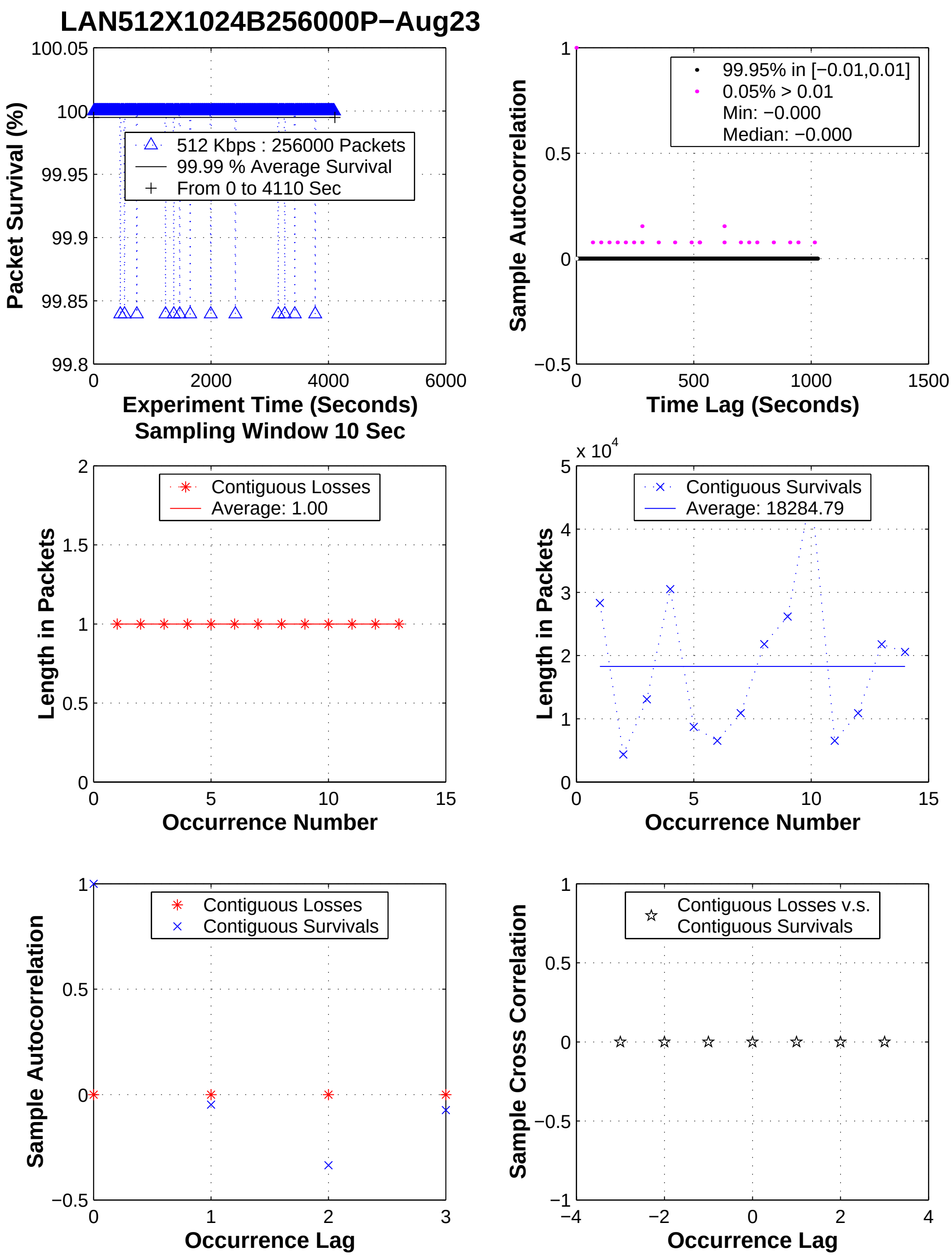


Figure A.47: Time series analysis: August 23, 2001 : campus network : 512 Kbits/sec transmission rate

Appendix B

Results from The Gaussian Approximation Approach

This appendix contains the results referred to in Chapter 4.

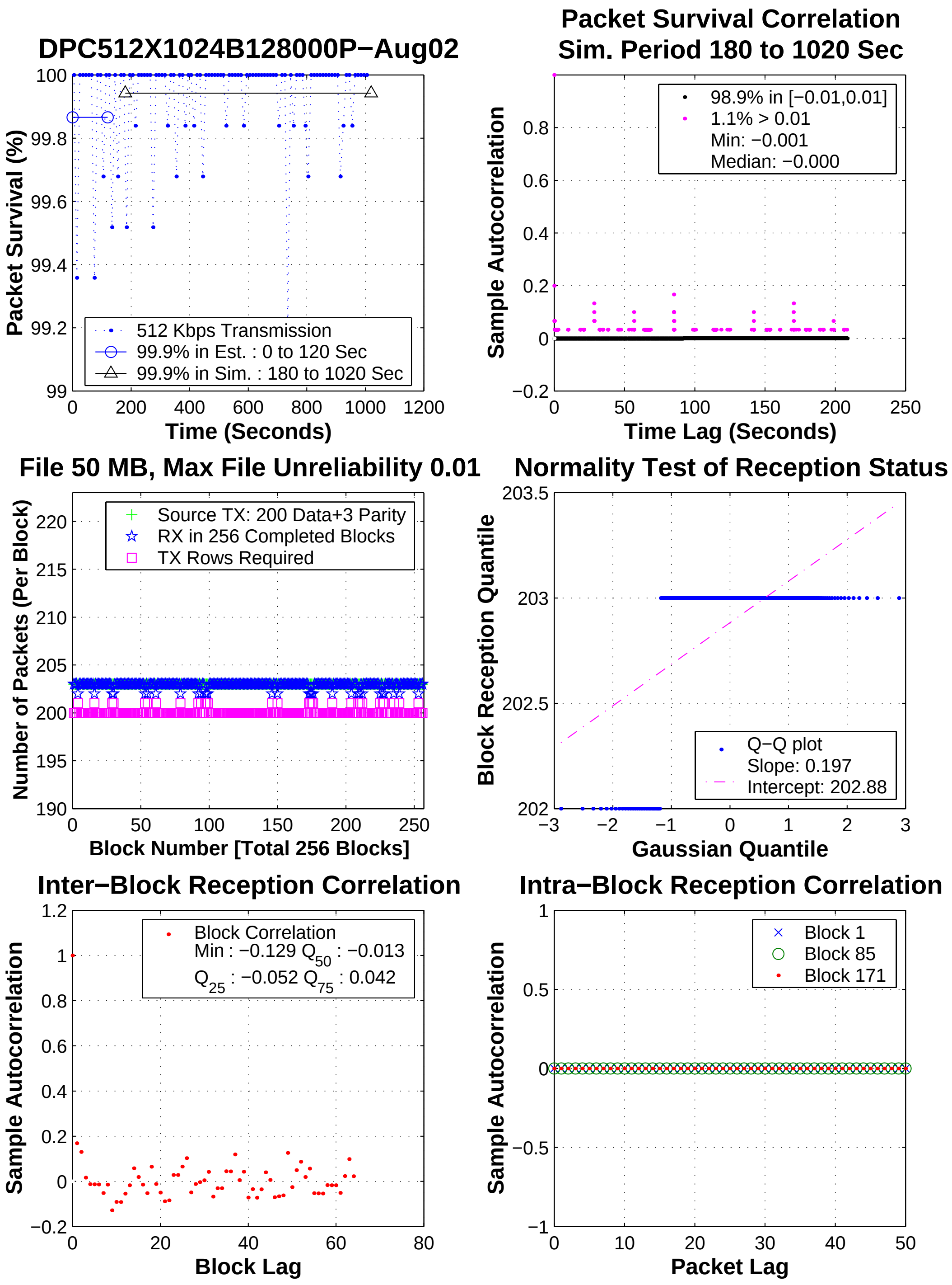


Figure B.1: Trace-driven simulation : August 02, 2001 : HST 512 Kbits/sec transmission rate

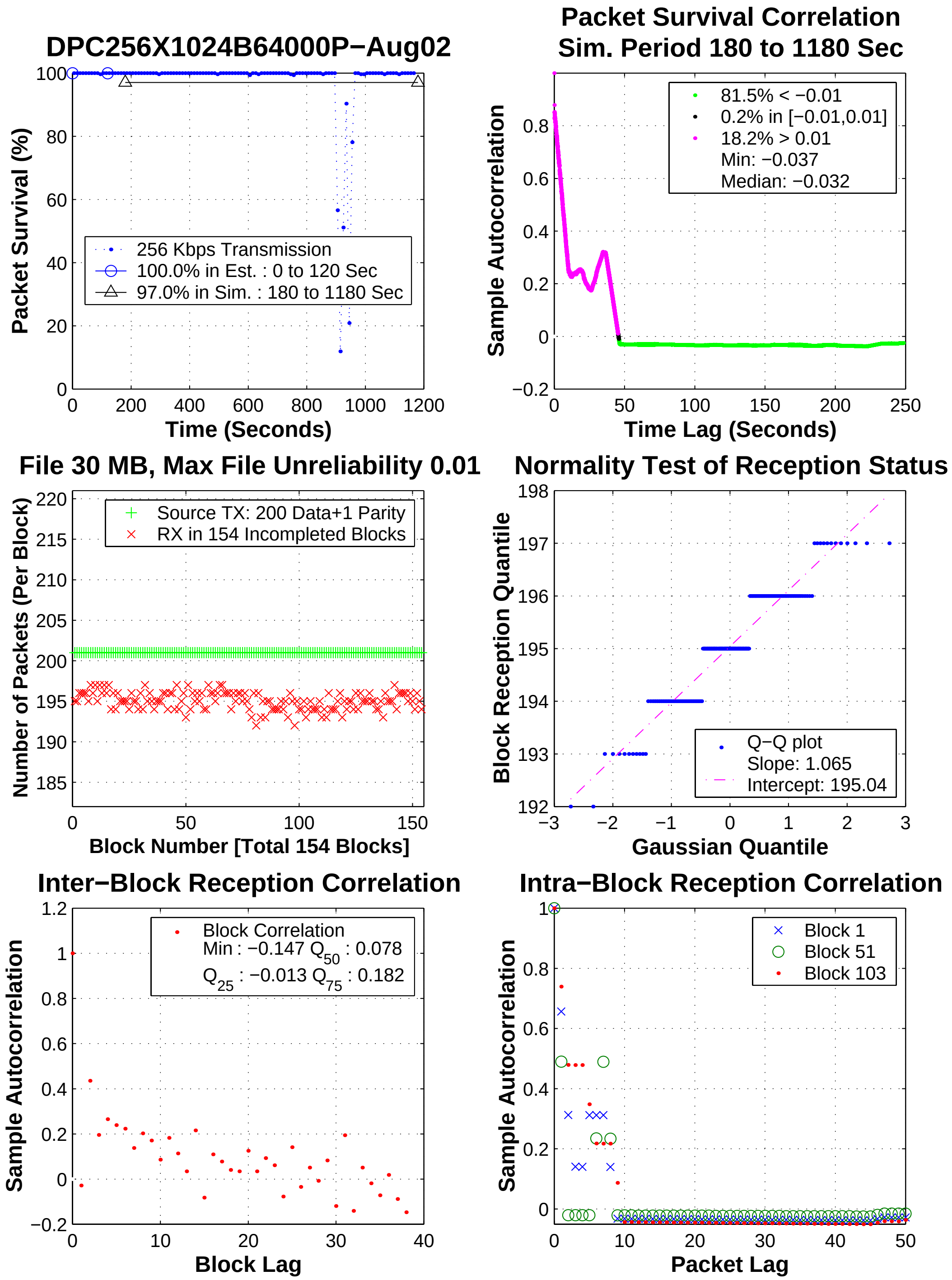


Figure B.2: Trace-driven simulation : August 02, 2001 : HST 256 Kbits/sec transmission rate

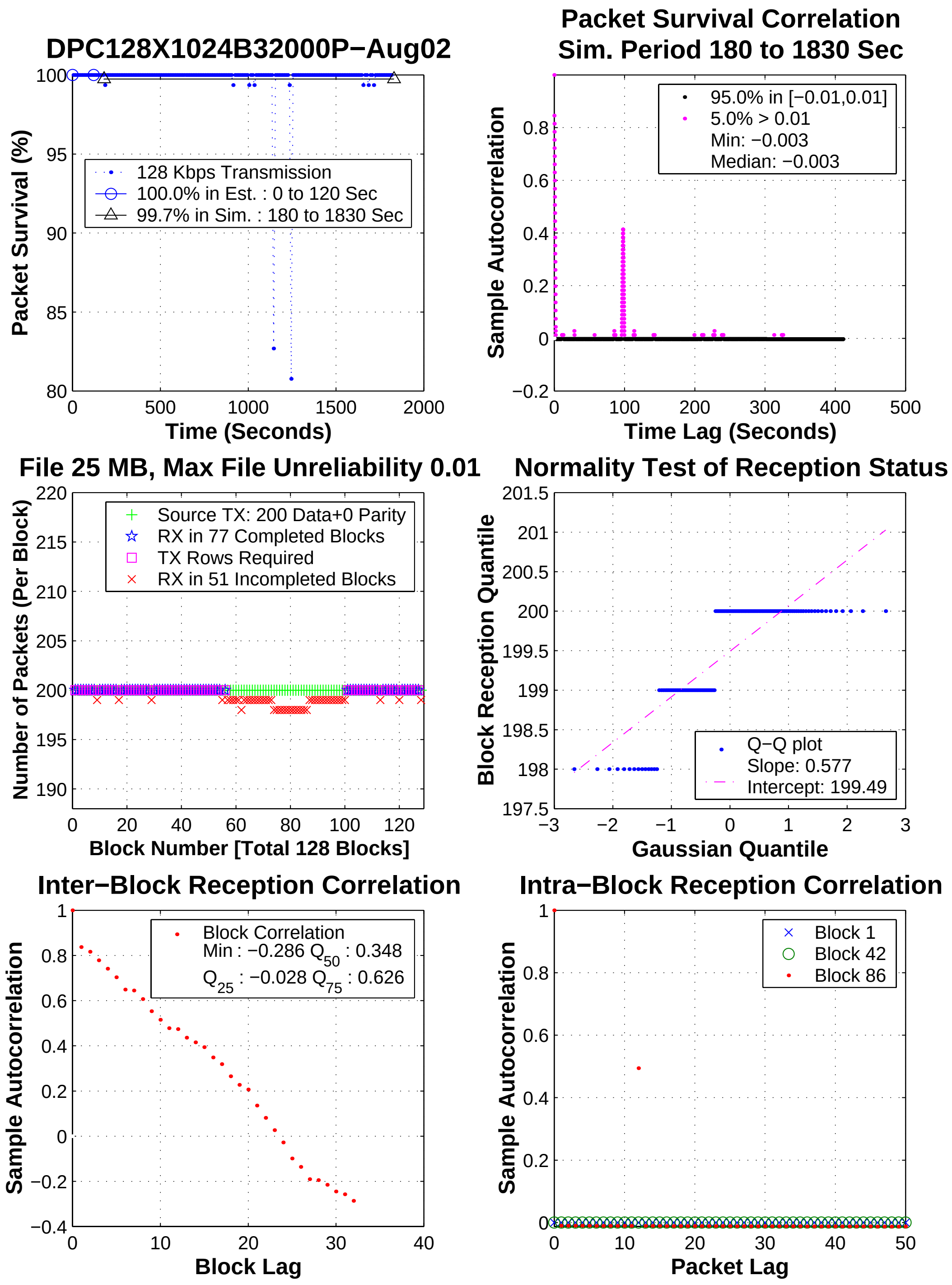


Figure B.3: Trace-driven simulation : August 02, 2001 : HST 128 Kbits/sec transmission rate

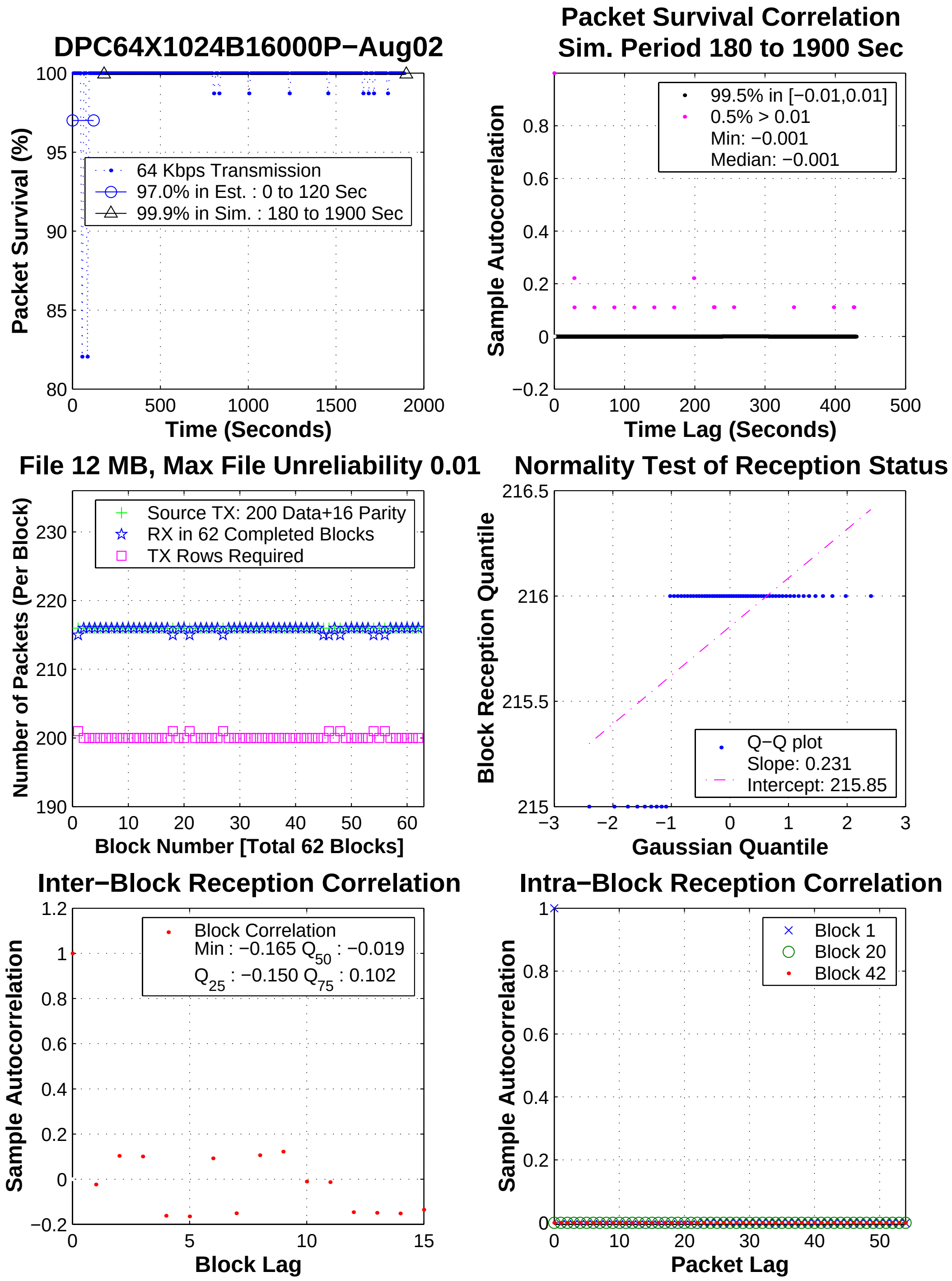


Figure B.4: Trace-driven simulation : August 02, 2001 : HST 64 Kbits/sec transmission rate

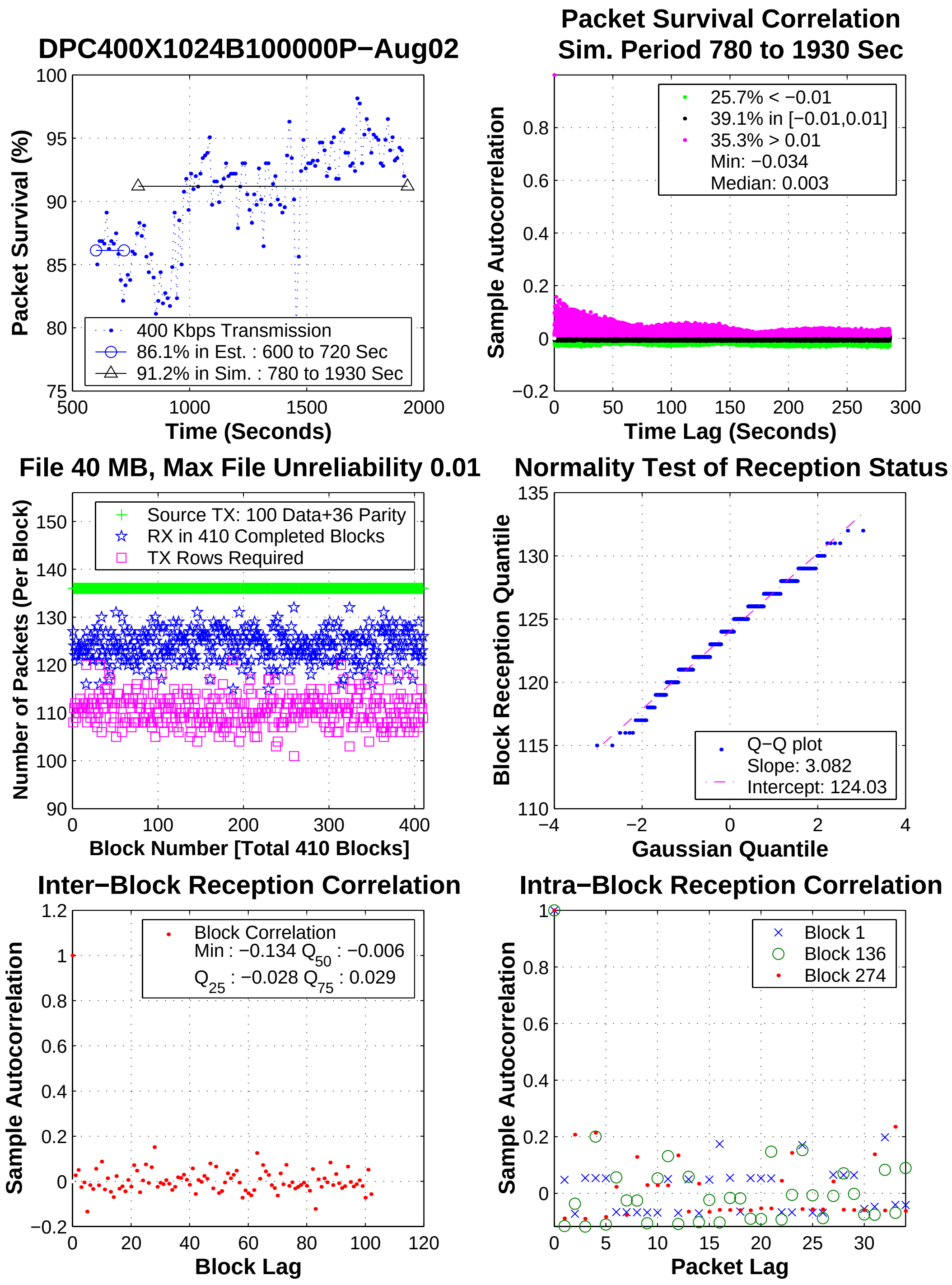


Figure B.5: Trace-driven simulation : August 02, 2001 : HST 400 Kbits/sec transmission rate

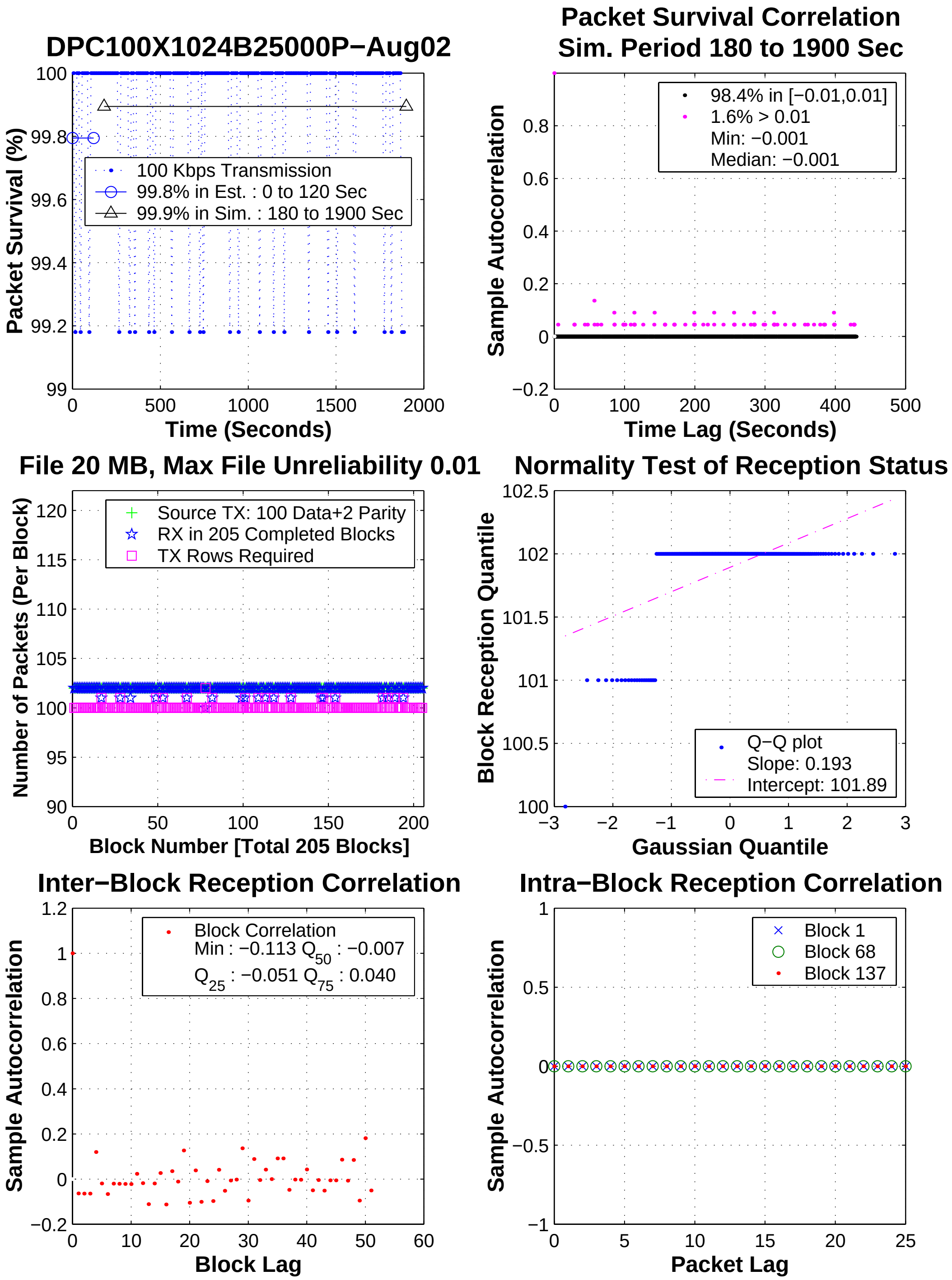


Figure B.6: Trace-driven simulation : August 02, 2001 : HST 100 Kbits/sec transmission rate

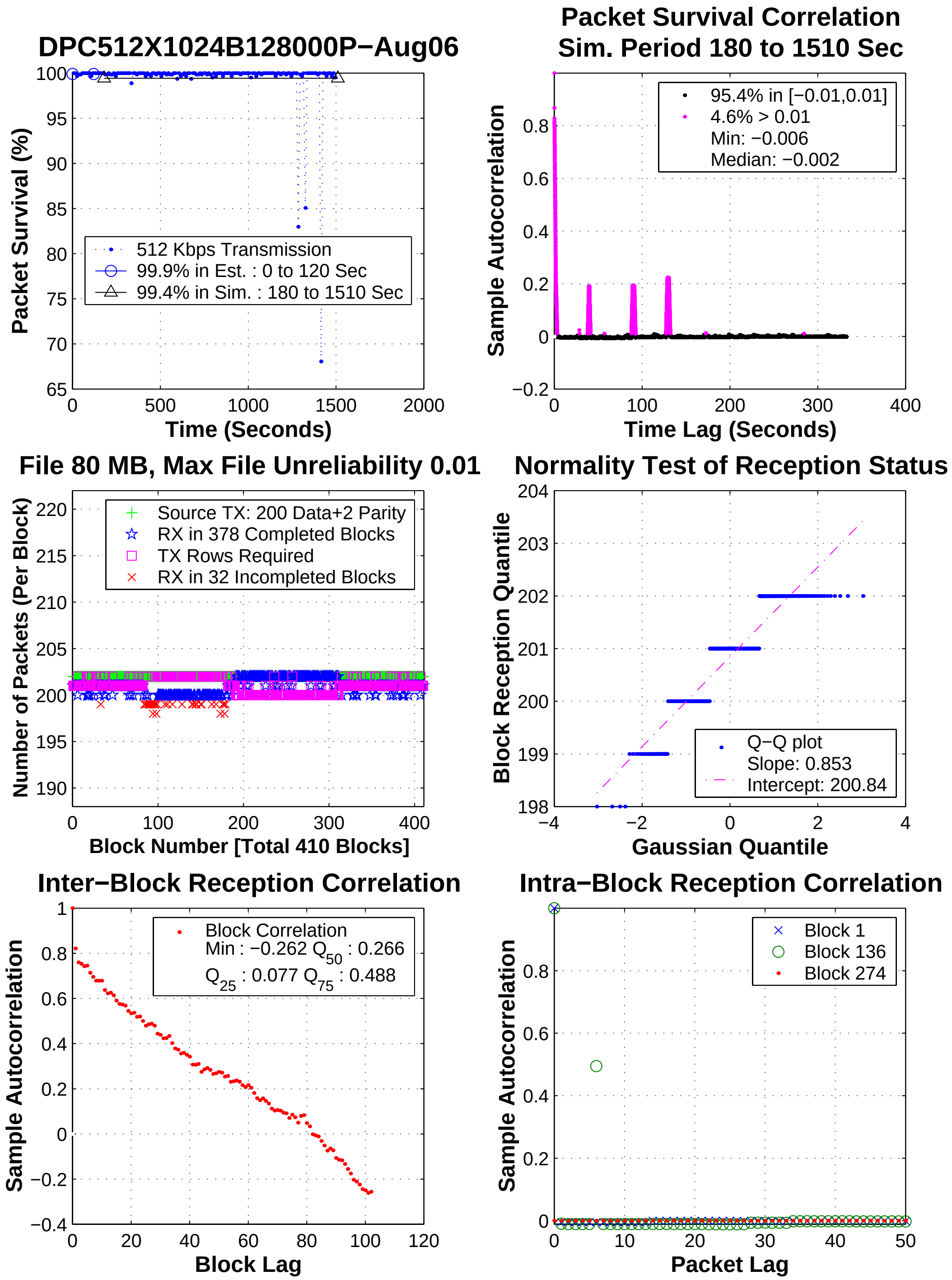


Figure B.7: Trace-driven simulation : August 06, 2001 : HST 512 Kbits/sec transmission rate

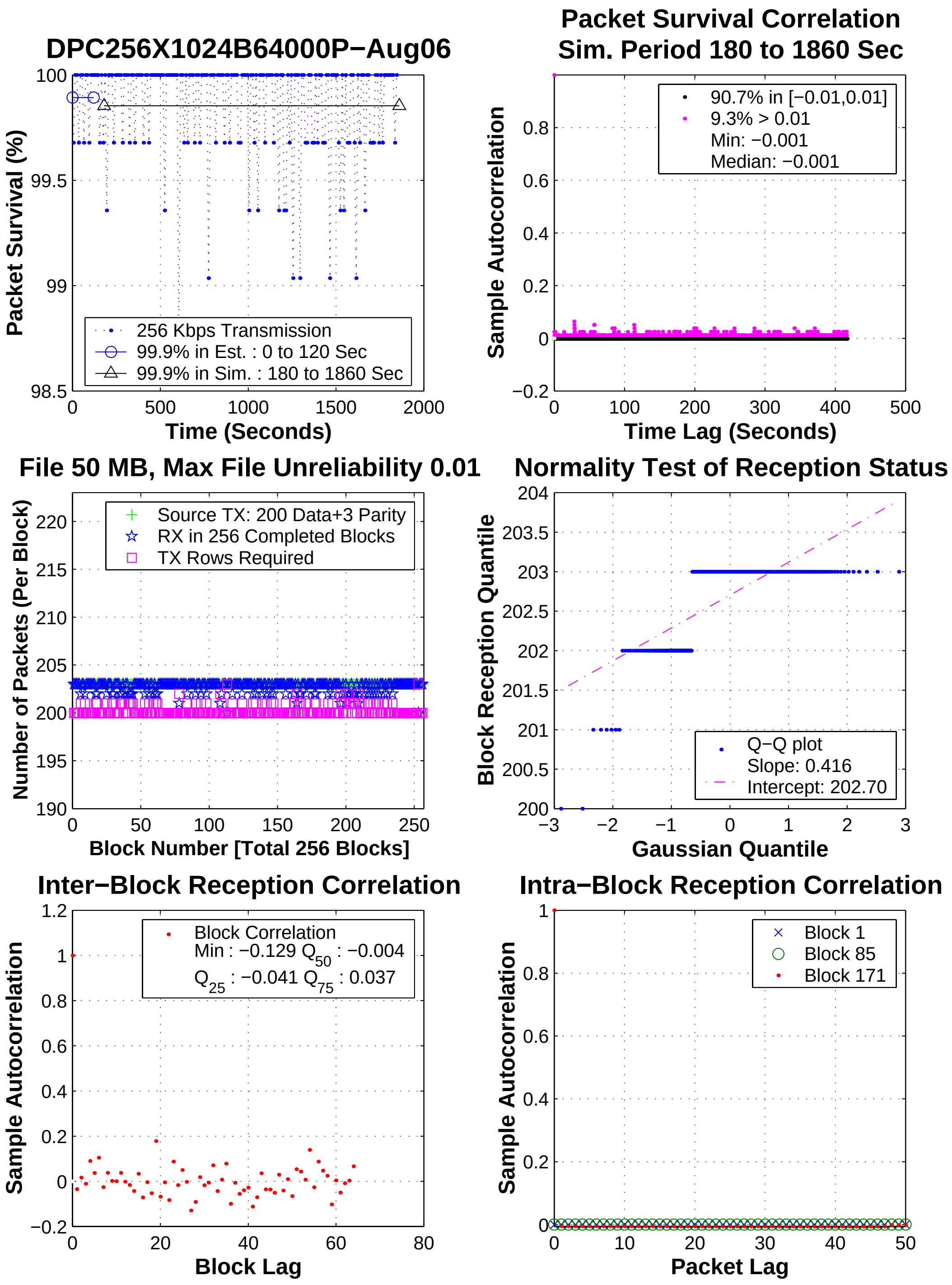


Figure B.8: Trace-driven simulation : August 06, 2001 : HST 256 Kbits/sec transmission rate

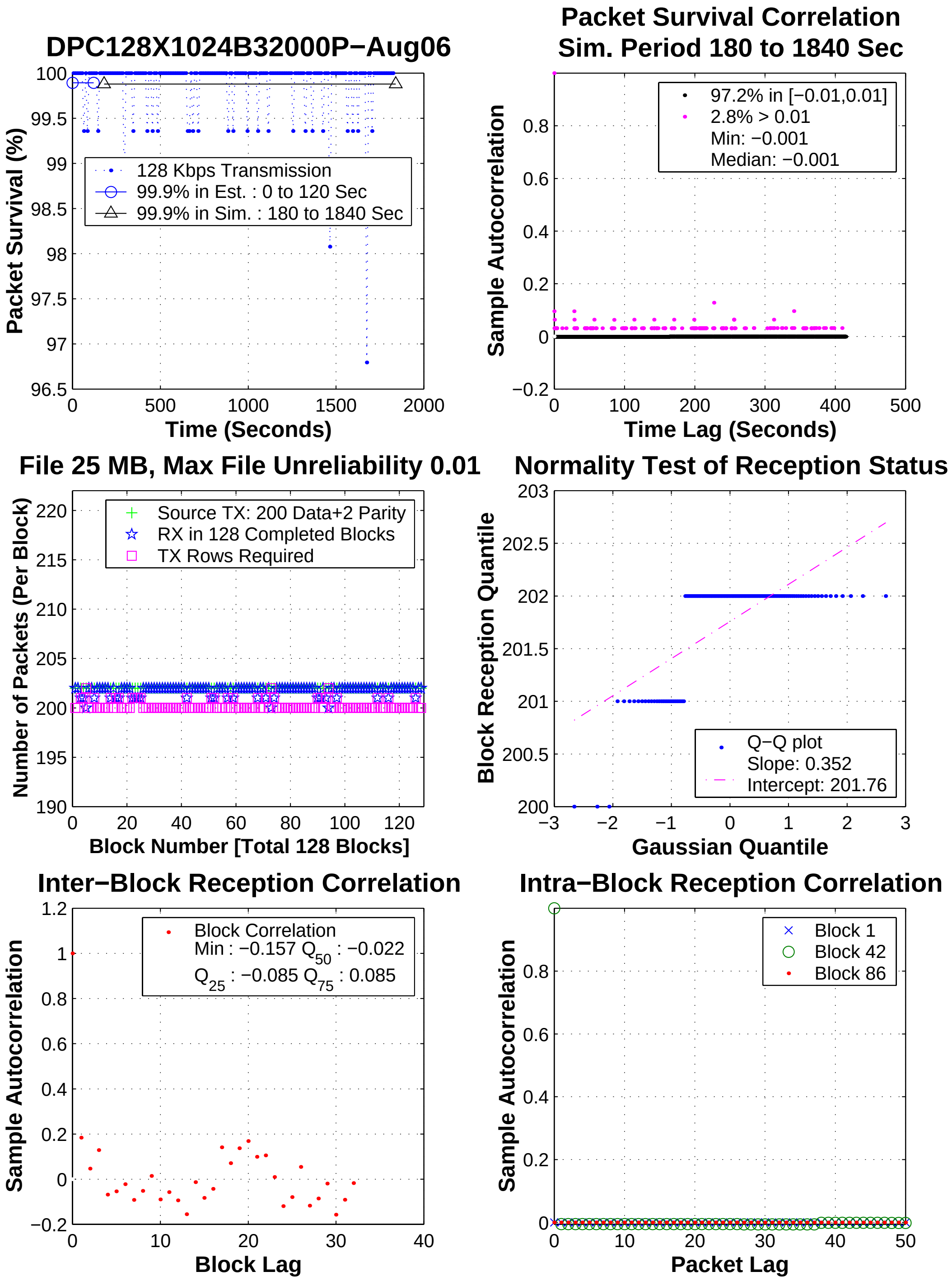


Figure B.9: Trace-driven simulation : August 06, 2001 : HST 128 Kbits/sec transmission rate

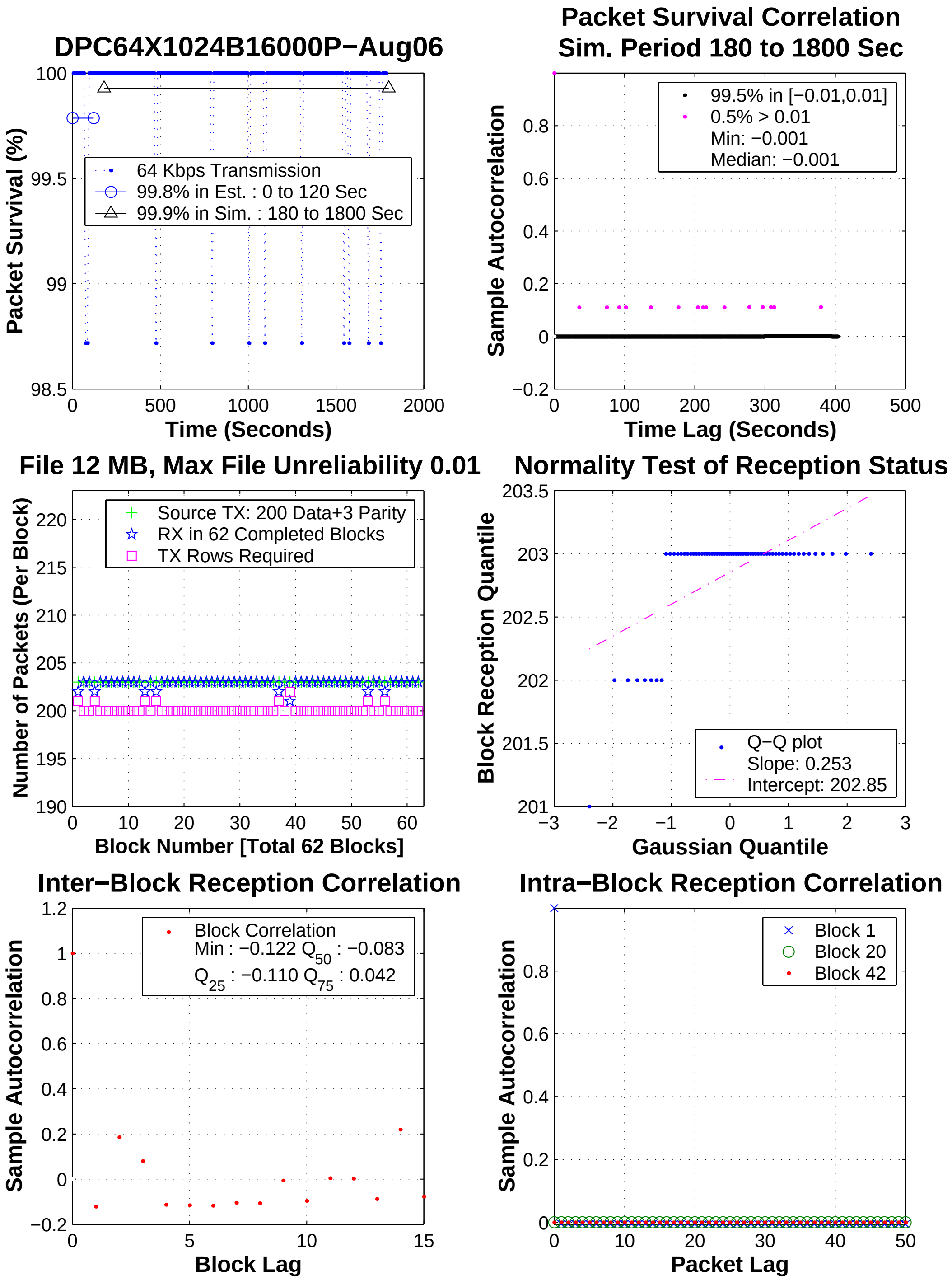


Figure B.10: Trace-driven simulation : August 06, 2001 : HST 64 Kbits/sec transmission rate

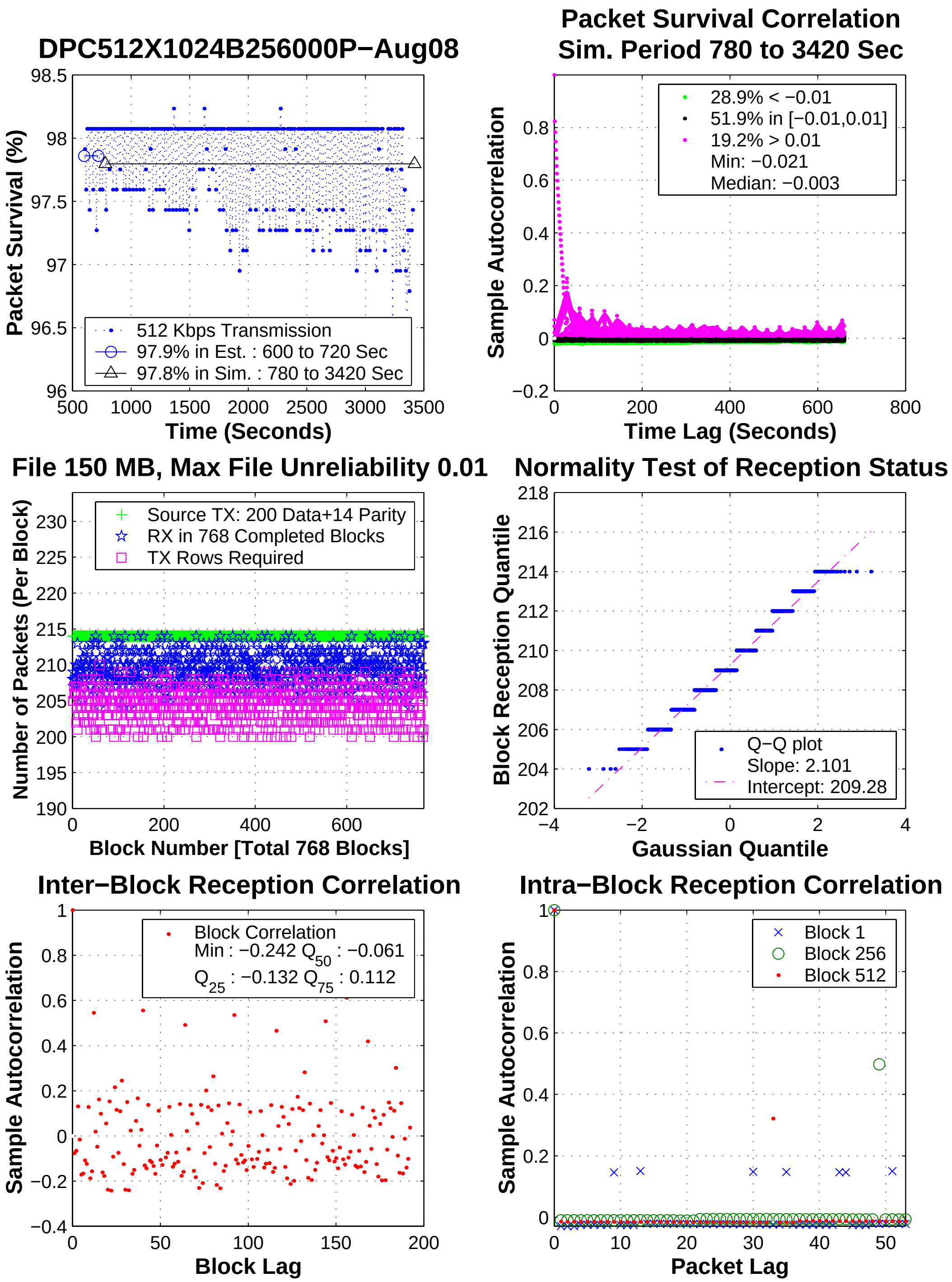


Figure B.11: Trace-driven simulation : August 08, 2001 : HST 512 Kbits/sec transmission rate

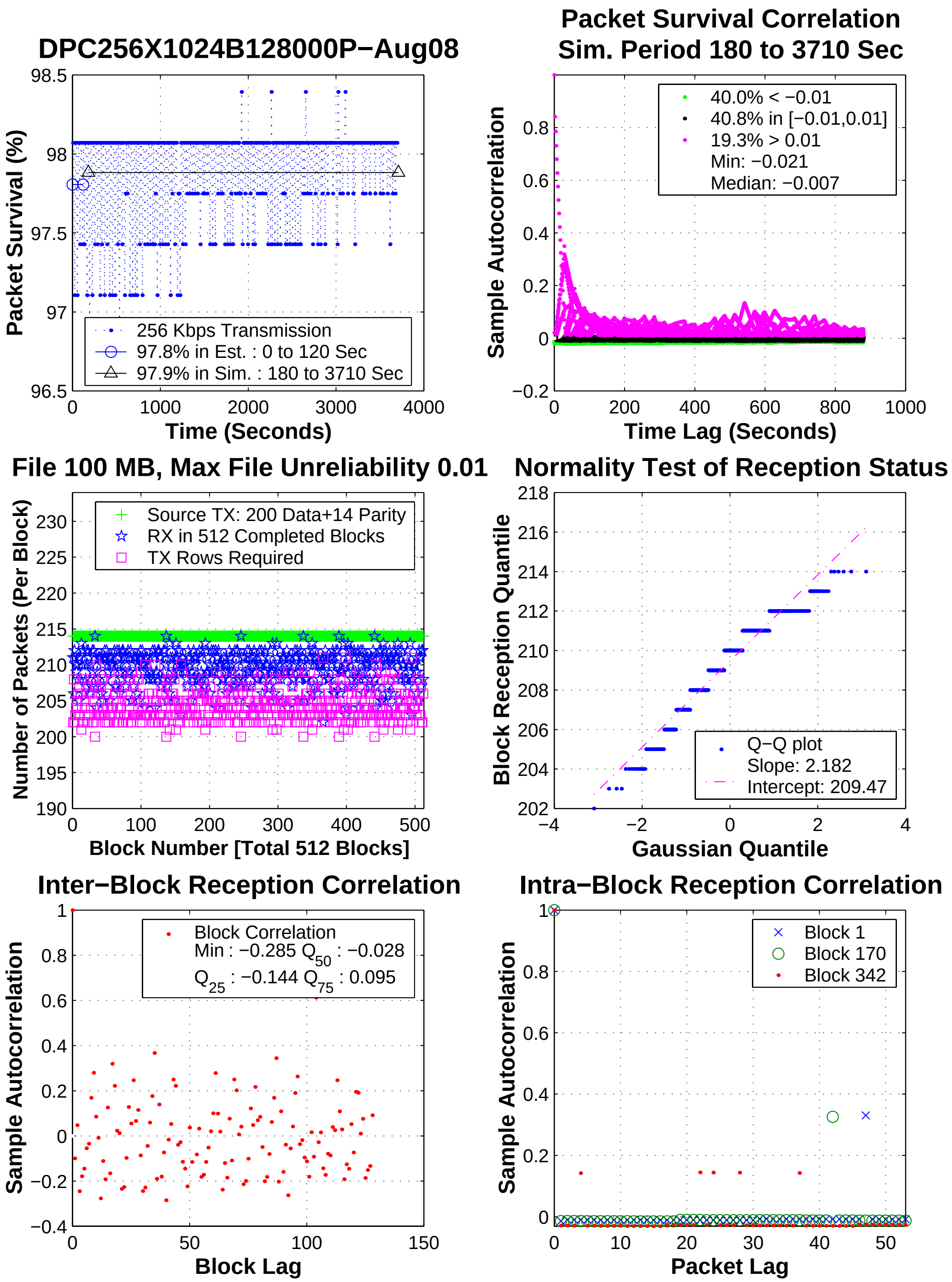


Figure B.12: Trace-driven simulation : August 08, 2001 : HST 256 Kbits/sec transmission rate

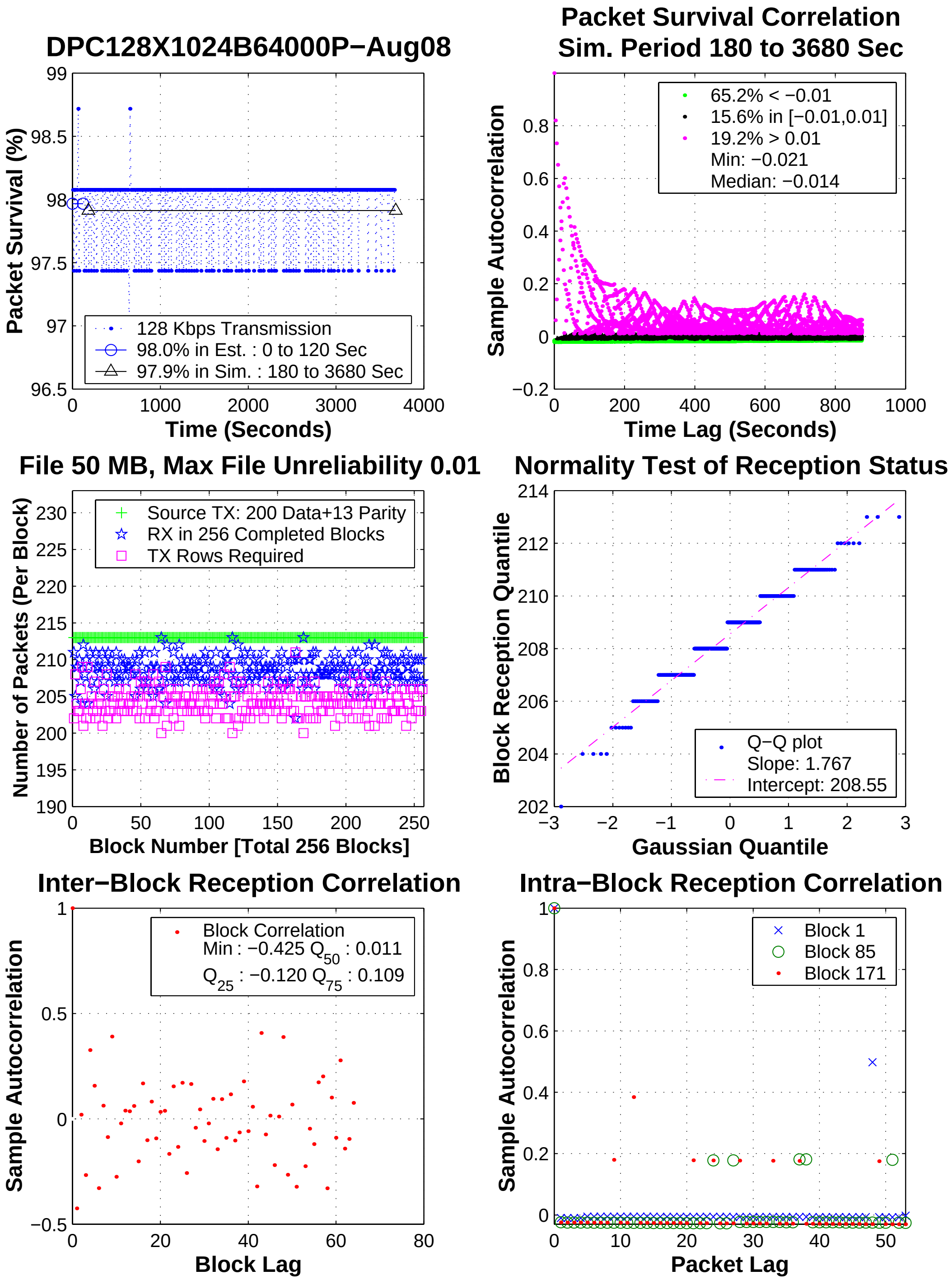


Figure B.13: Trace-driven simulation : August 08, 2001 : HST 128 Kbits/sec transmission rate

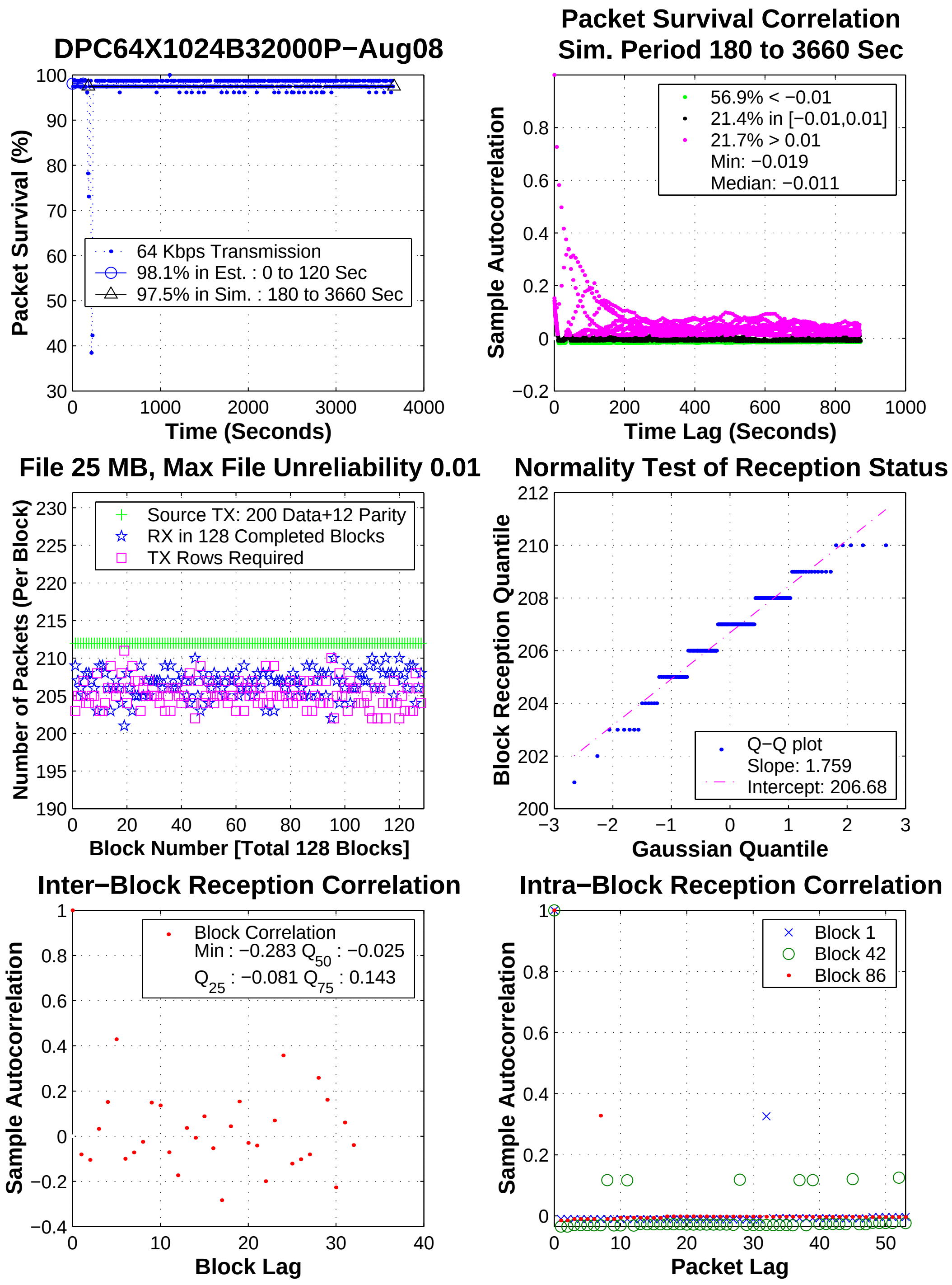


Figure B.14: Trace-driven simulation : August 08, 2001 : HST 64 Kbits/sec transmission rate

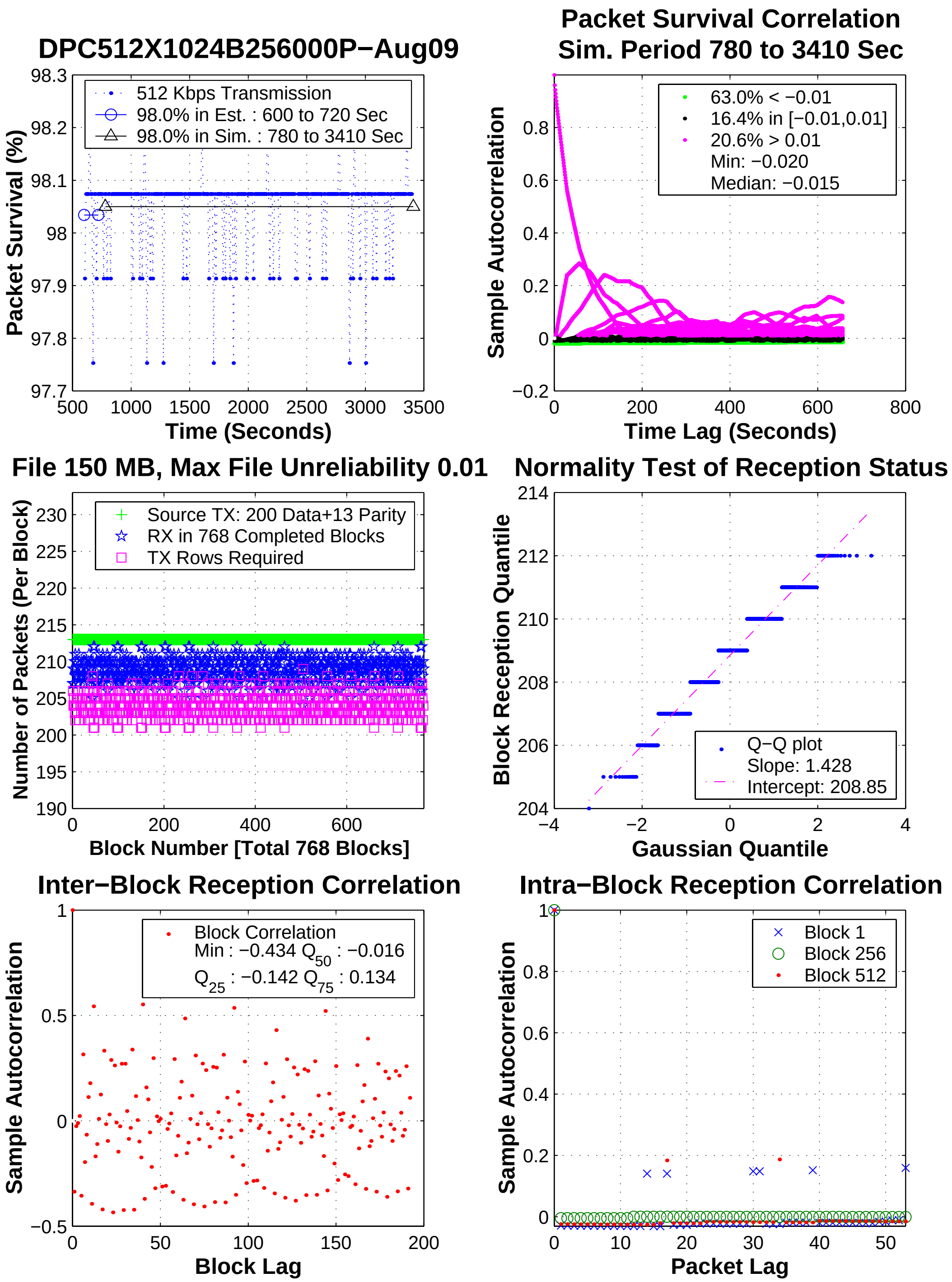


Figure B.15: Trace-driven simulation : August 09, 2001 : HST 512 Kbits/sec transmission rate

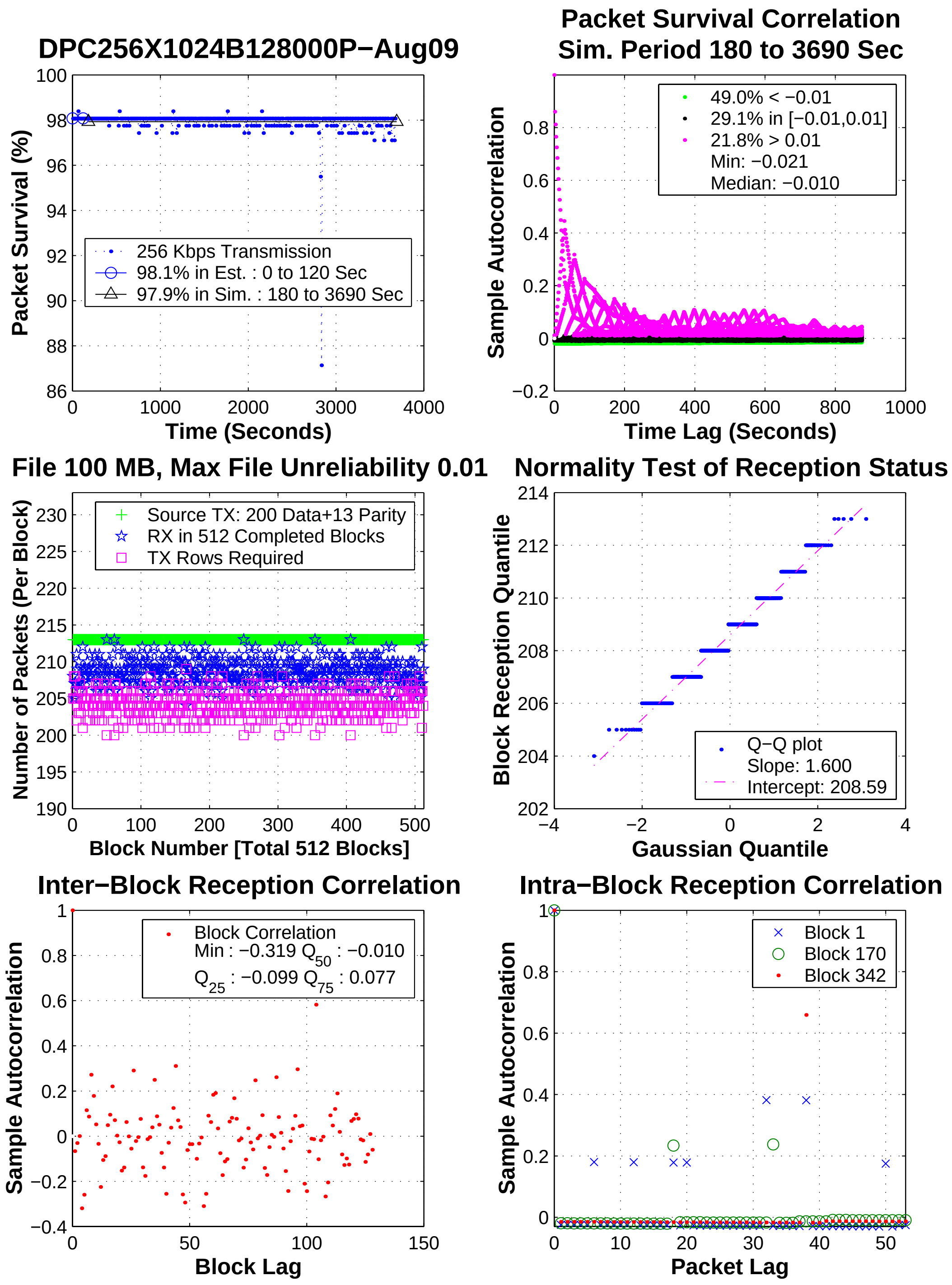


Figure B.16: Trace-driven simulation : August 09, 2001 : HST 256 Kbits/sec transmission rate

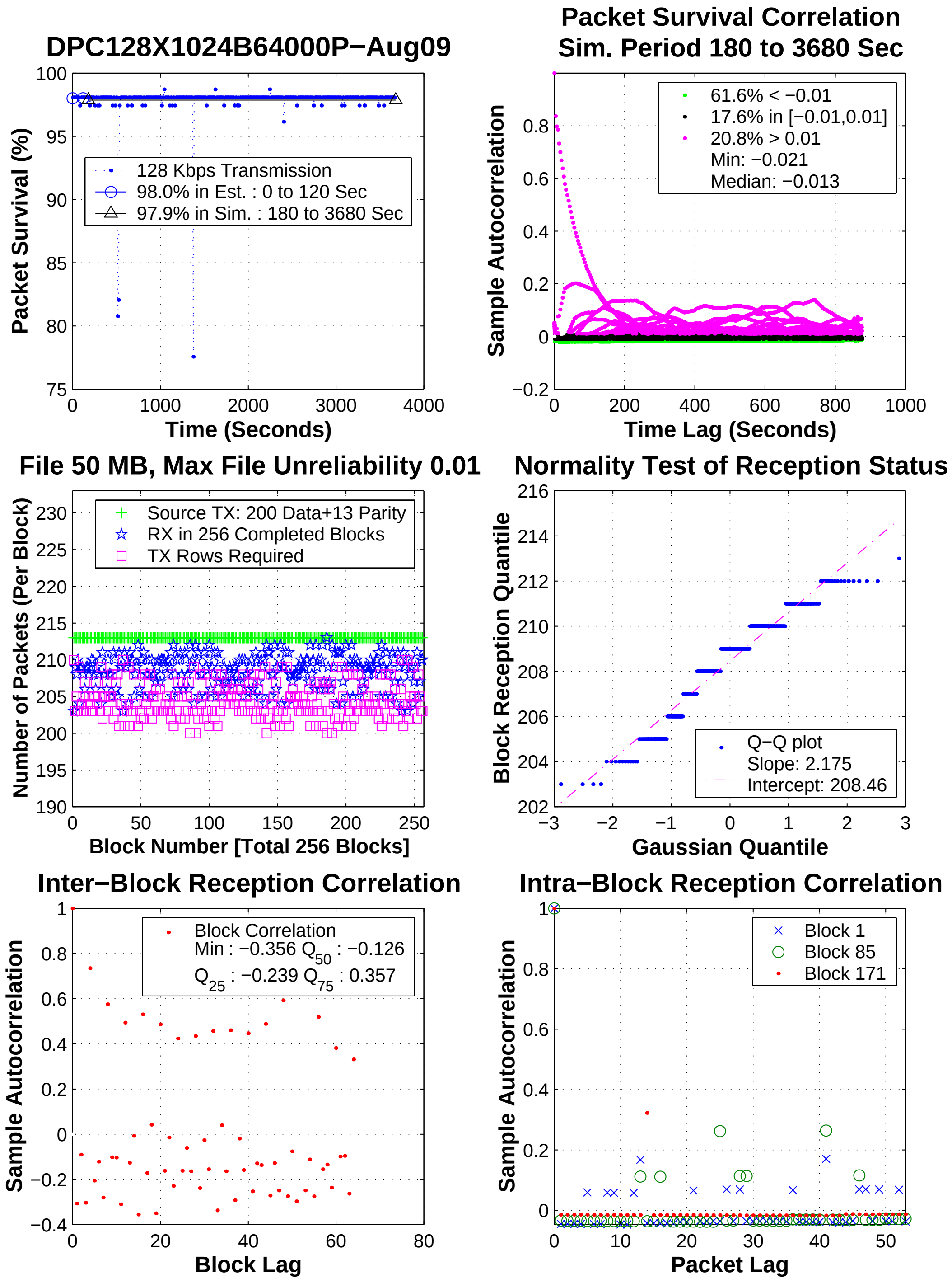


Figure B.17: Trace-driven simulation : August 09, 2001 : HST 128 Kbits/sec transmission rate

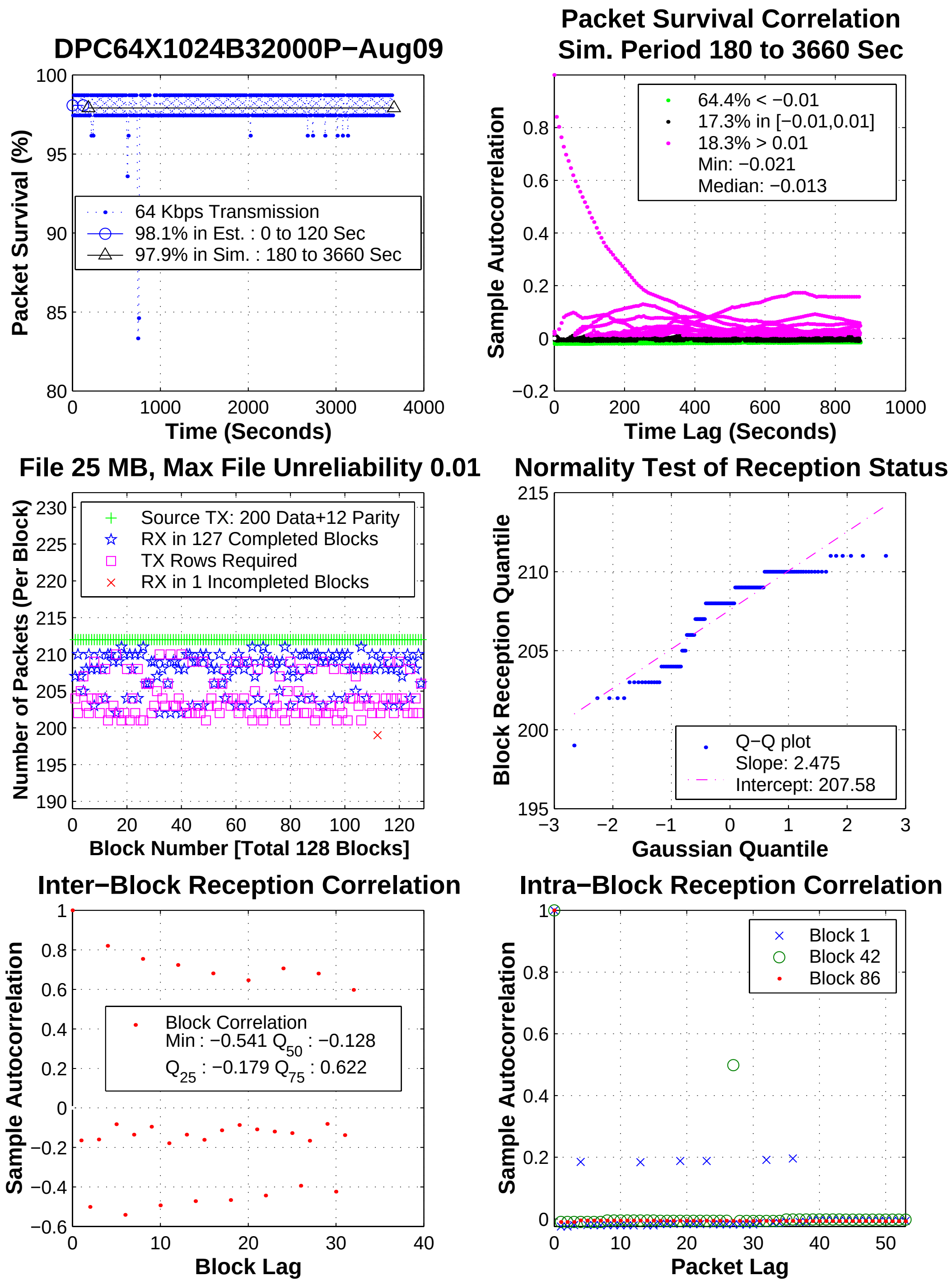


Figure B.18: Trace-driven simulation : August 09, 2001 : HST 64 Kbits/sec transmission rate

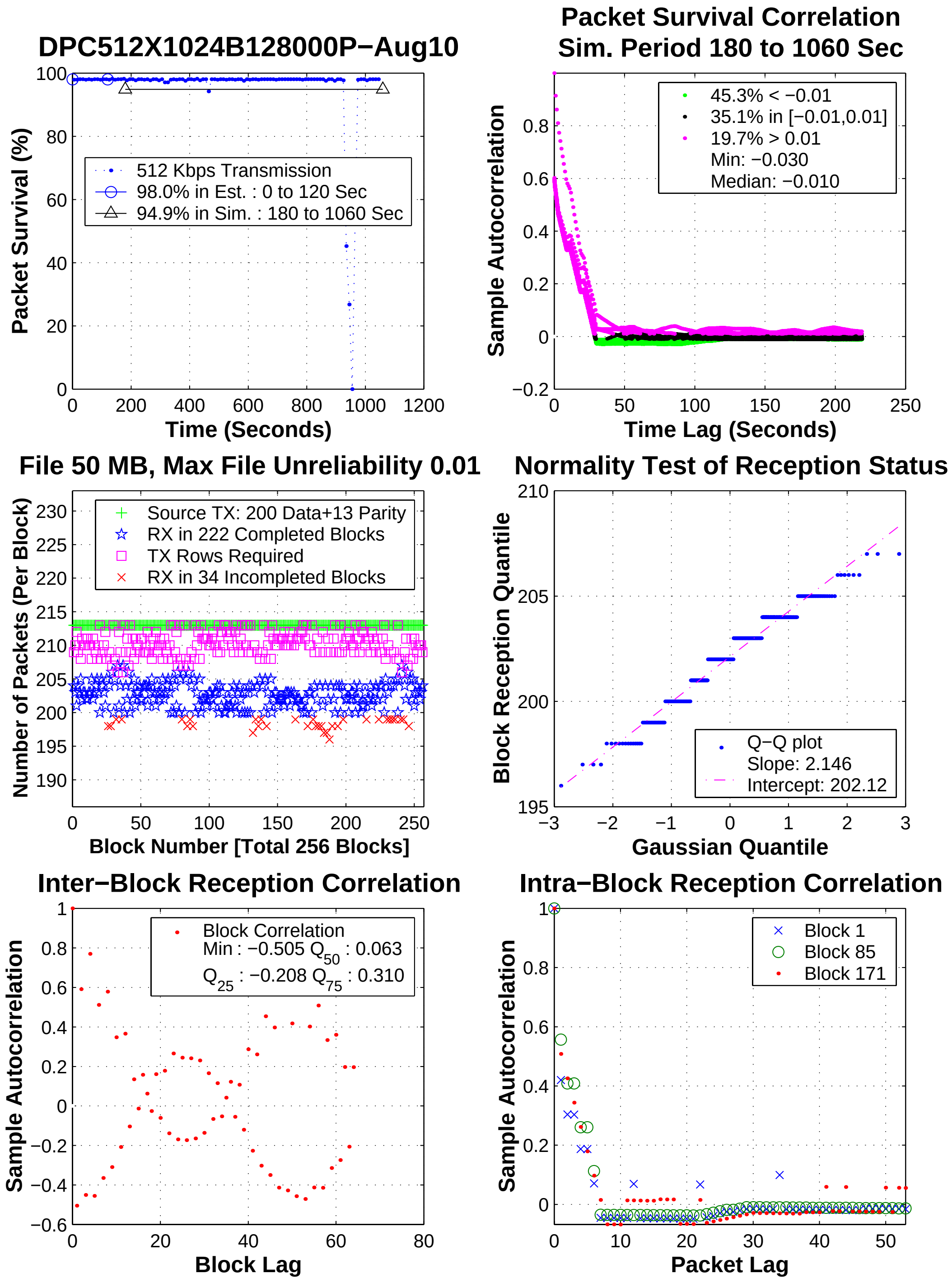


Figure B.19: Trace-driven simulation : August 10, 2001 : HST 512 Kbits/sec transmission rate

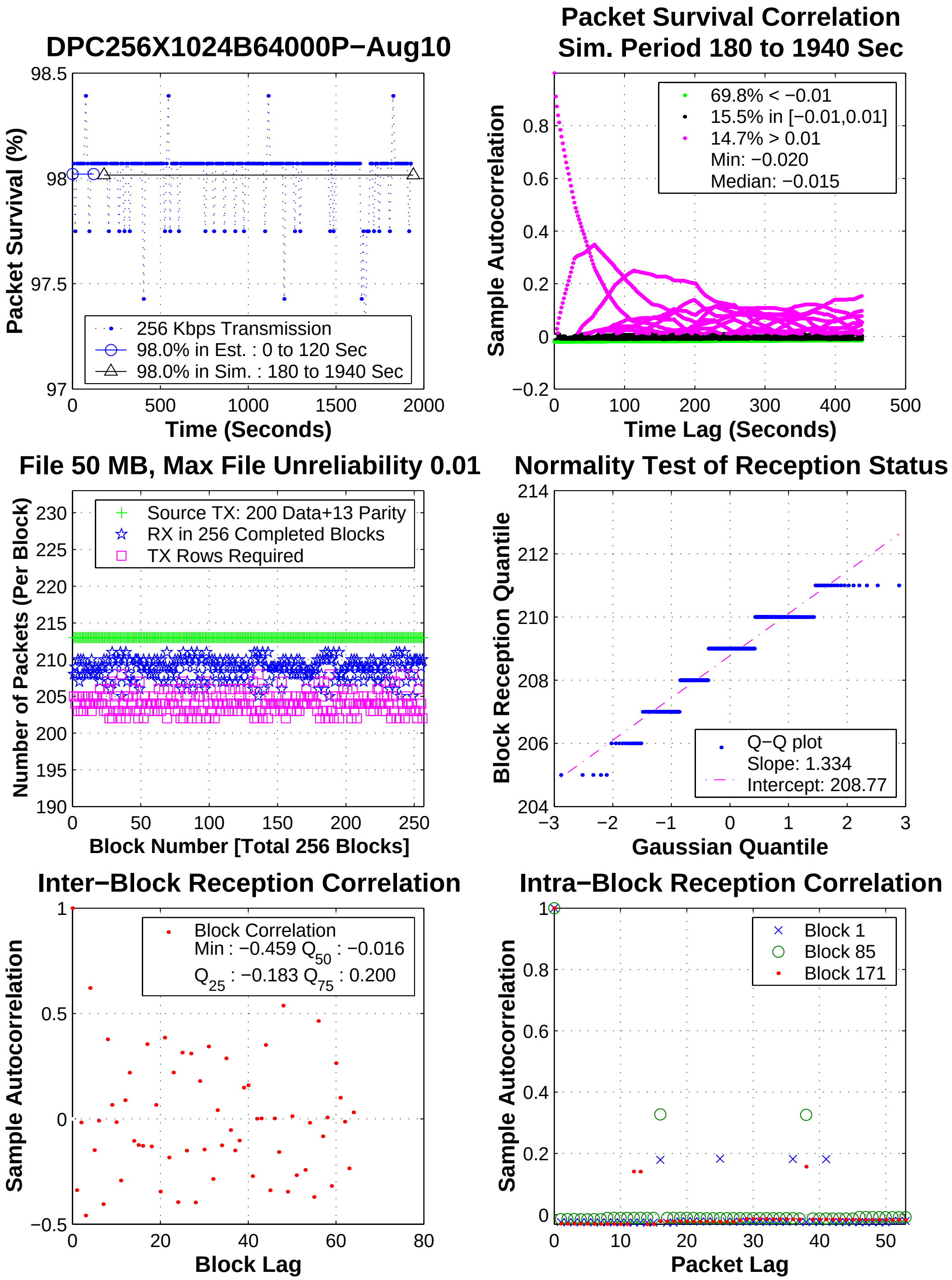


Figure B.20: Trace-driven simulation : August 10, 2001 : HST 256 Kbits/sec transmission rate

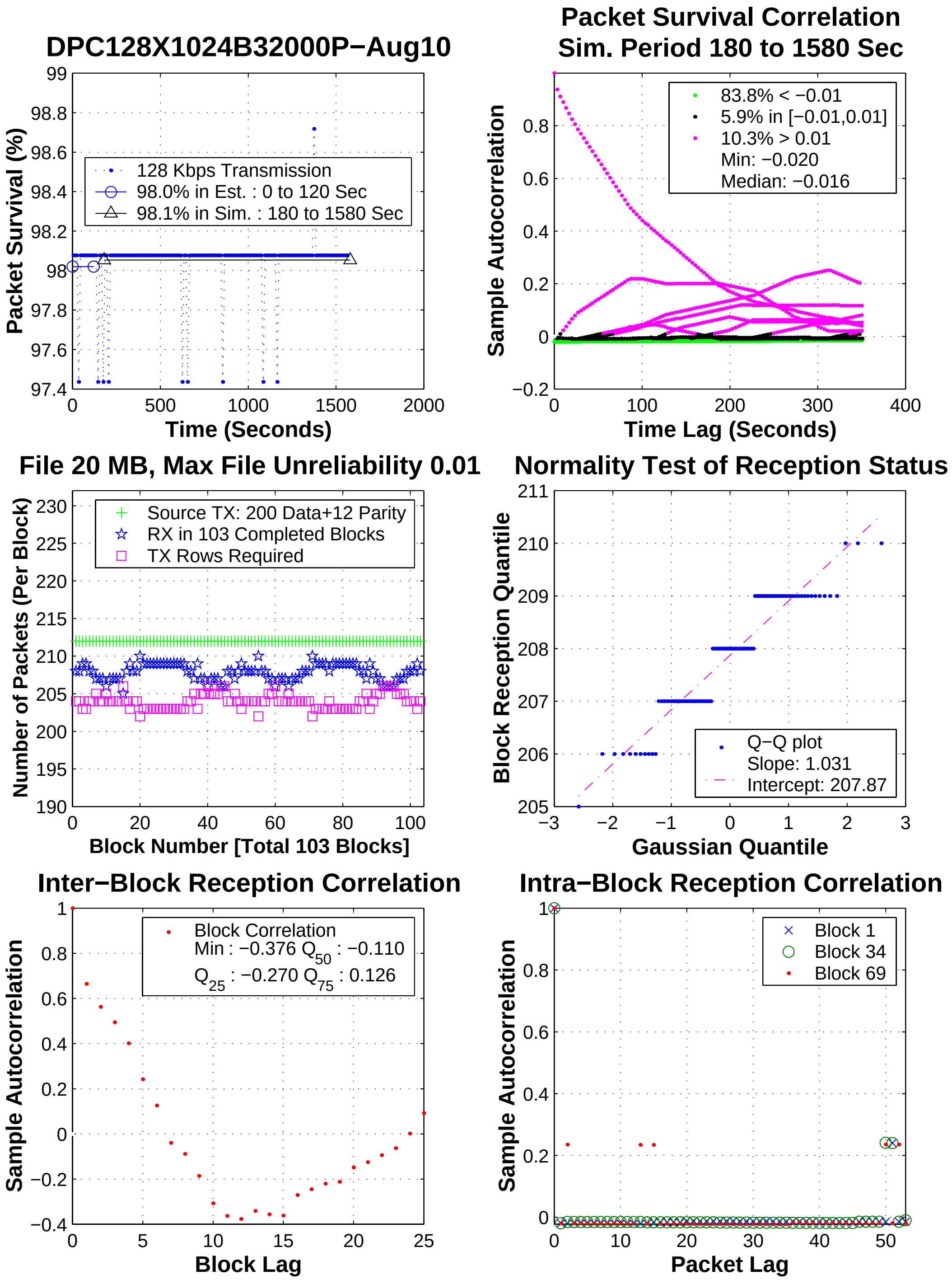


Figure B.21: Trace-driven simulation : August 10, 2001 : HST 128 Kbits/sec transmission rate

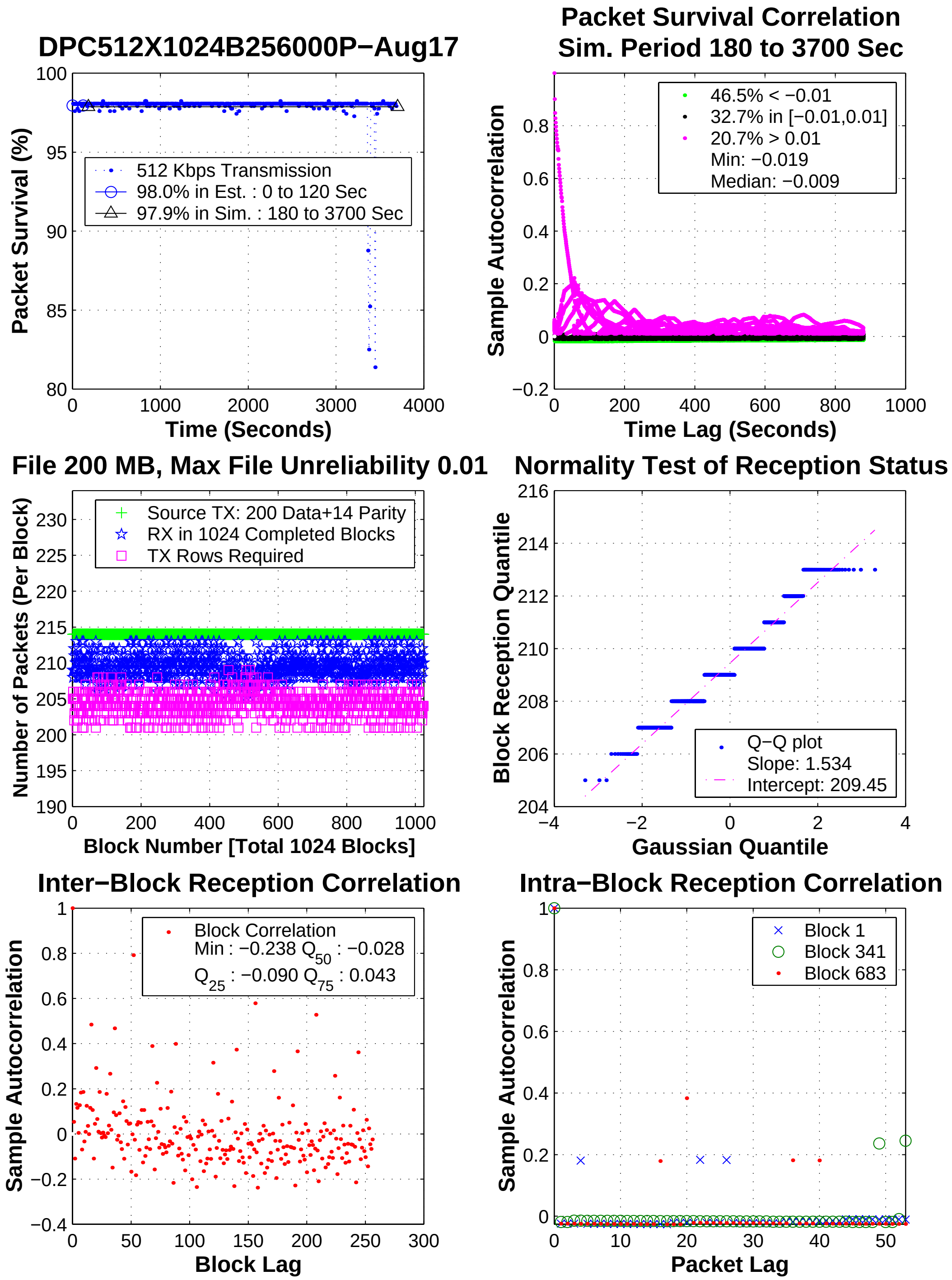


Figure B.22: Trace-driven simulation : August 17, 2001 : HST 512 Kbits/sec transmission rate

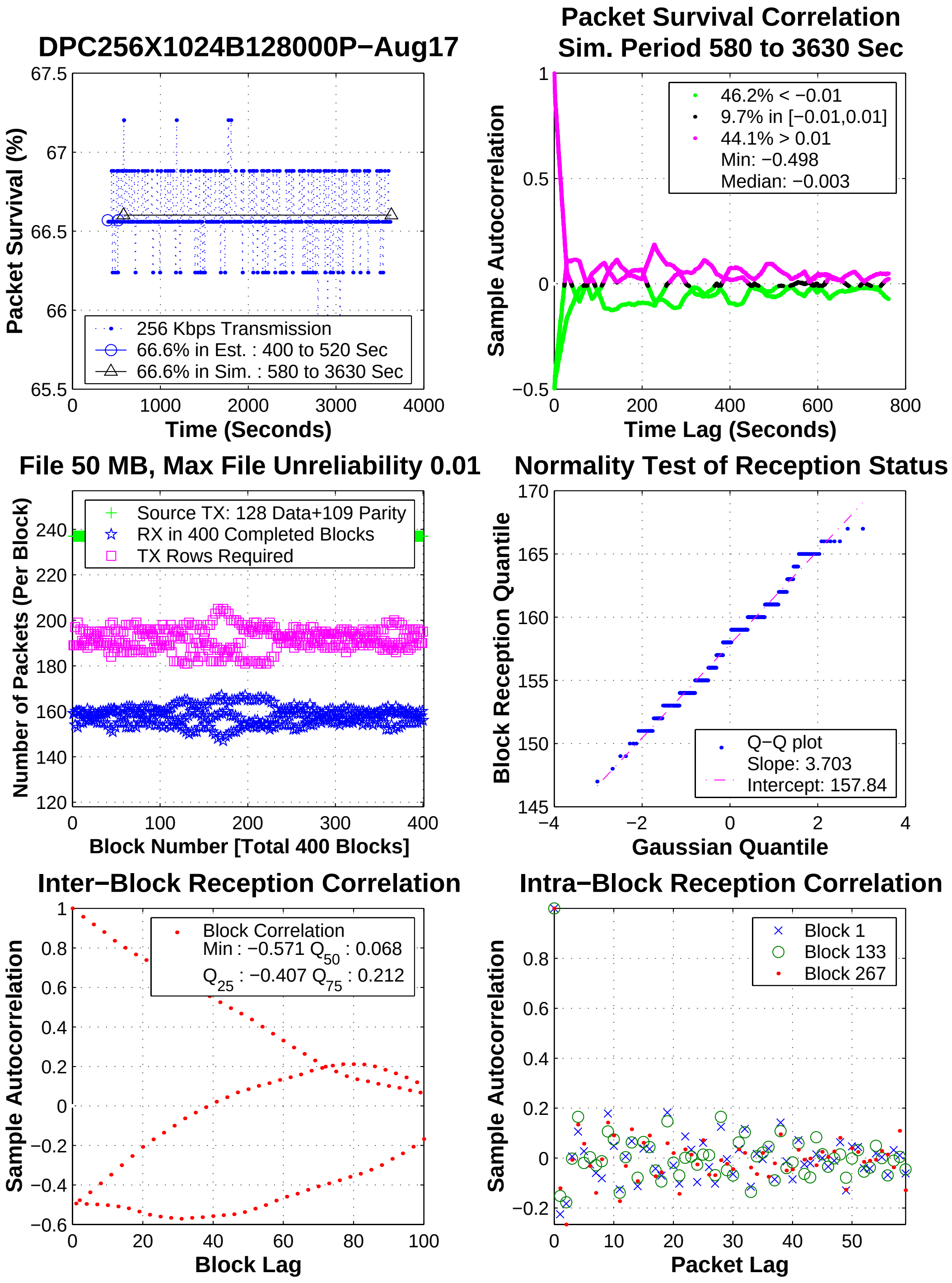


Figure B.23: Trace-driven simulation : August 17, 2001 : HST 256 Kbits/sec transmission rate

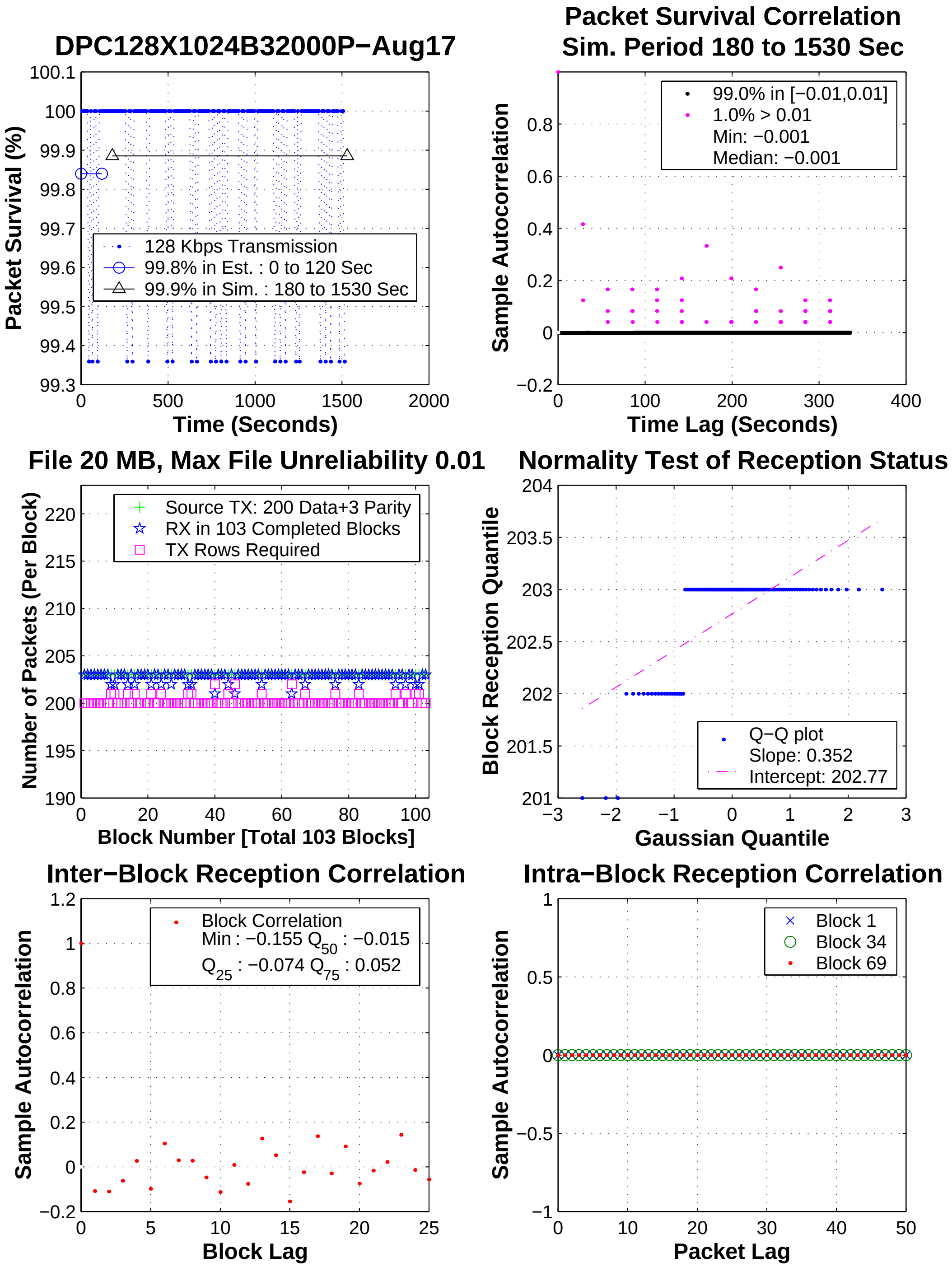


Figure B.24: Trace-driven simulation : August 17, 2001 : HST 128 Kbits/sec transmission rate

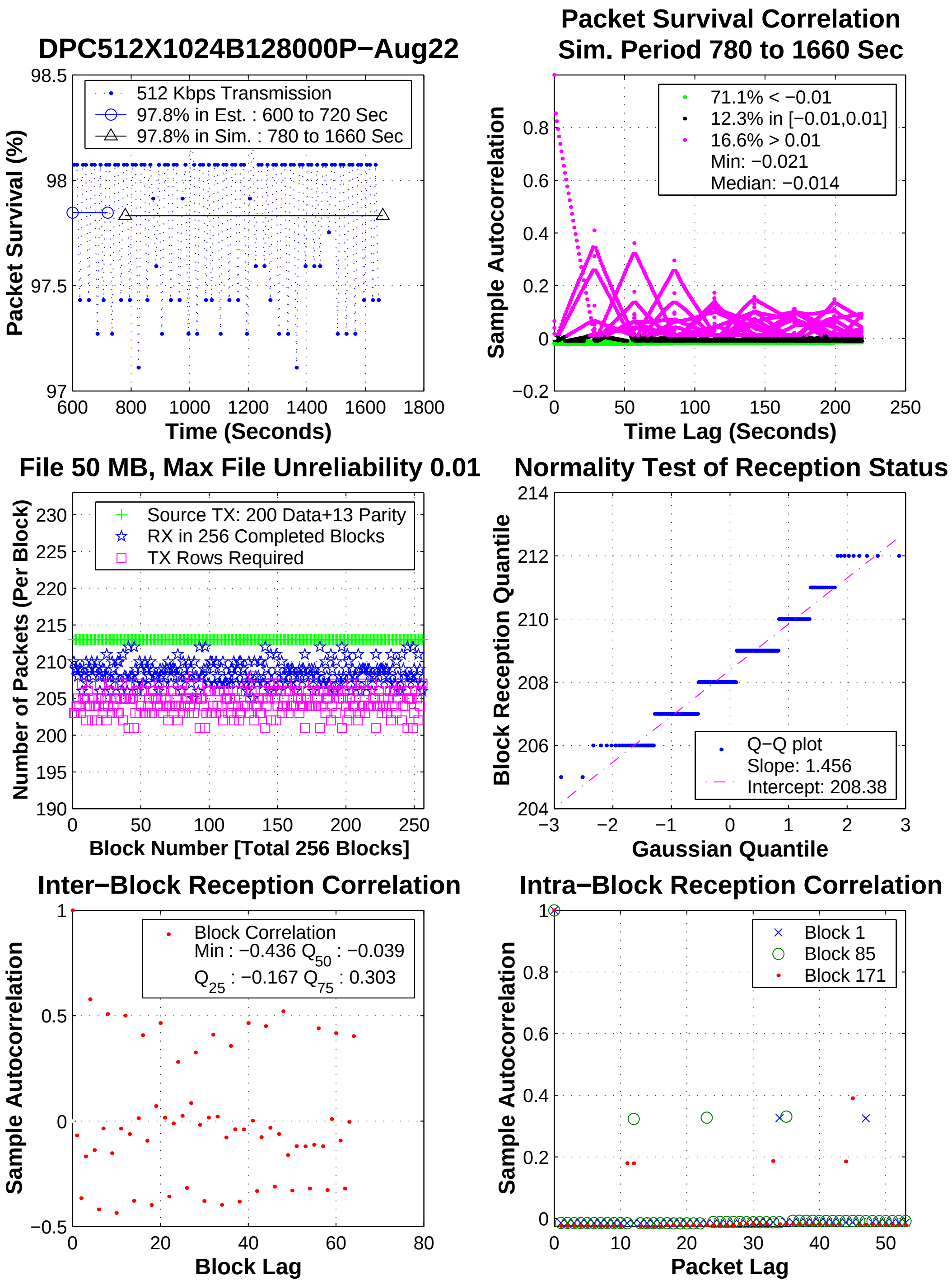


Figure B.25: Trace-driven simulation : August 22, 2001 : HST 512 Kbits/sec transmission rate

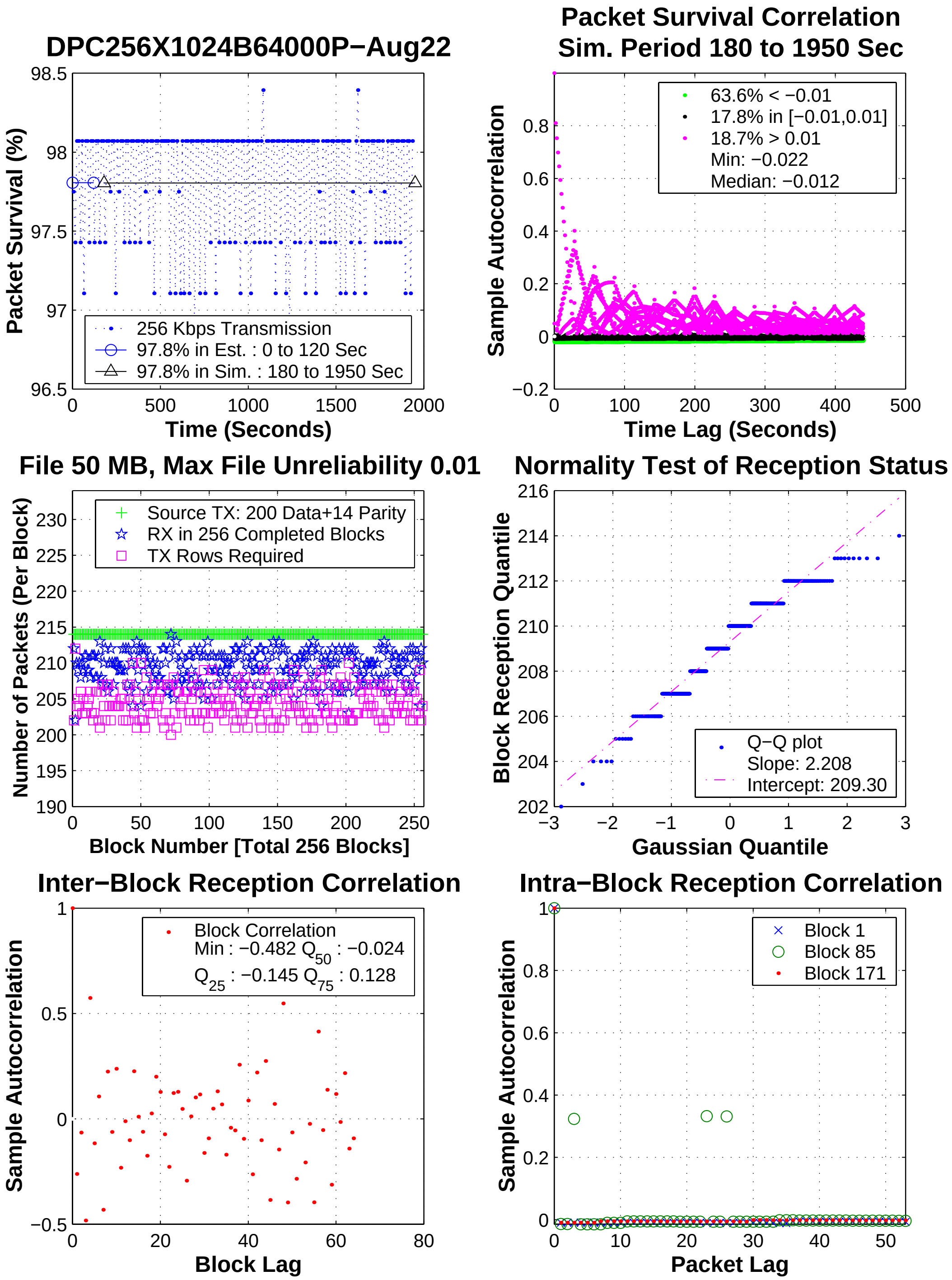


Figure B.26: Trace-driven simulation : August 22, 2001 : HST 256 Kbits/sec transmission rate

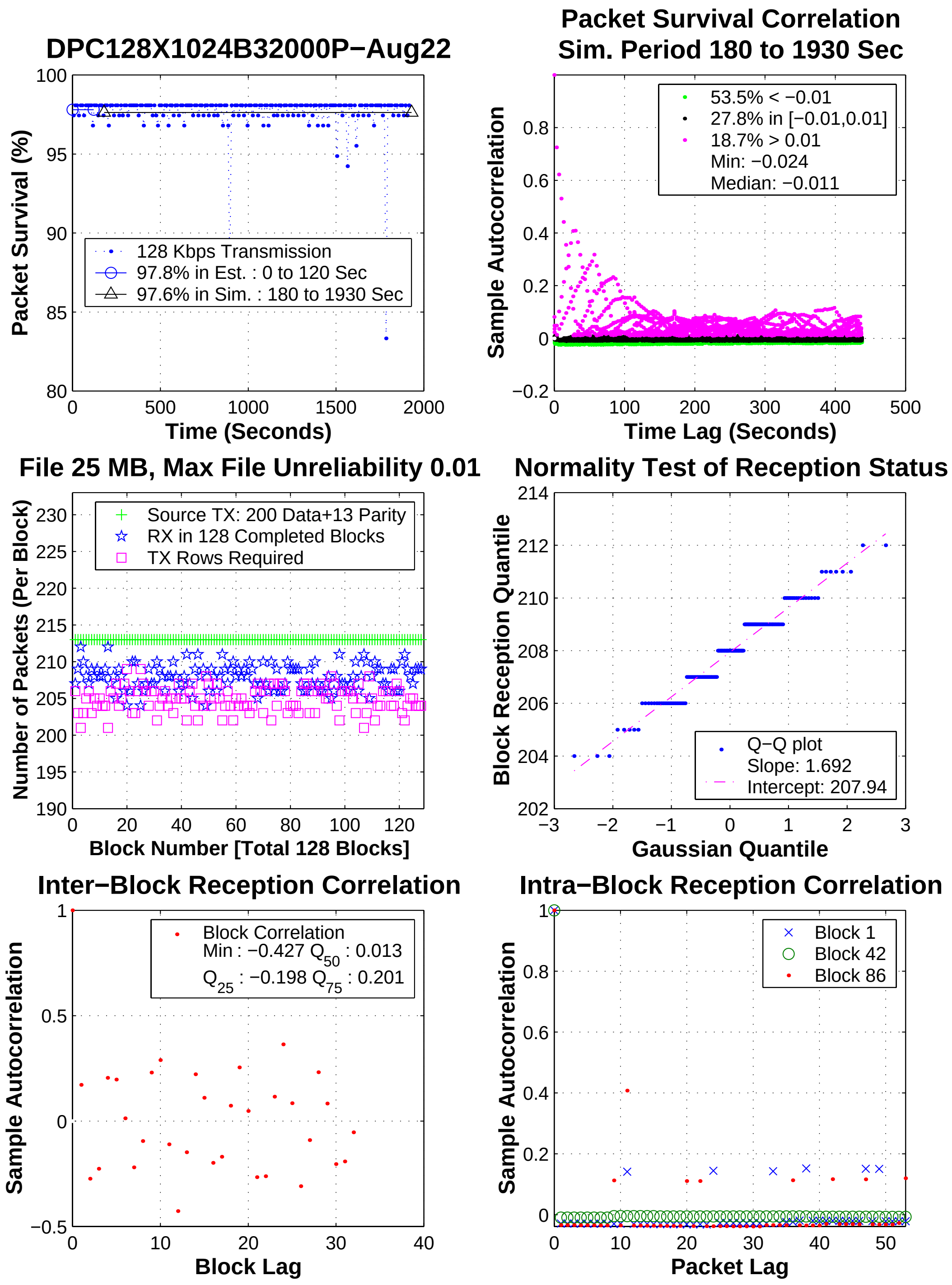


Figure B.27: Trace-driven simulation : August 22, 2001 : HST 128 Kbits/sec transmission rate

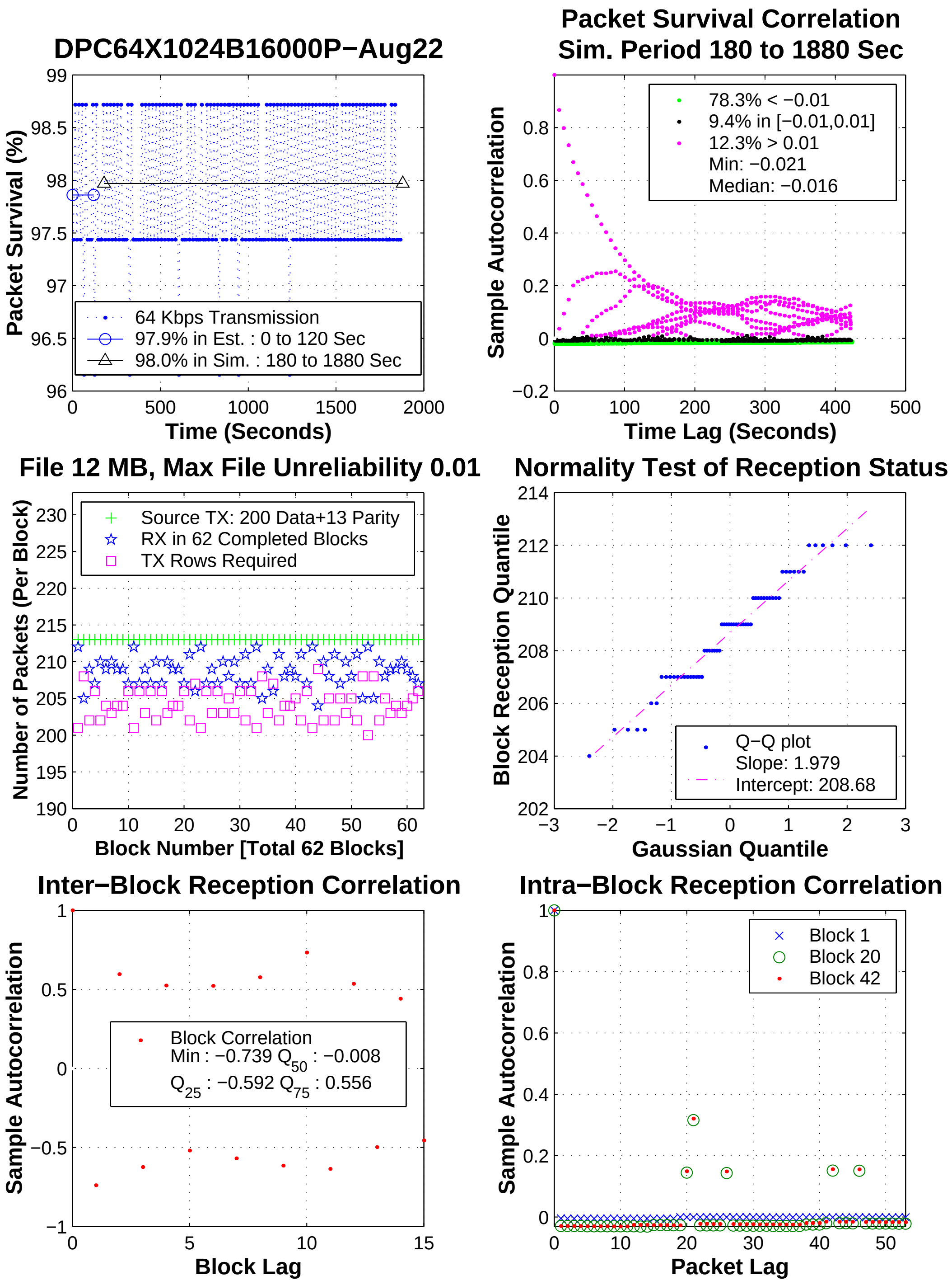


Figure B.28: Trace-driven simulation : August 22, 2001 : HST 64 Kbits/sec transmission rate

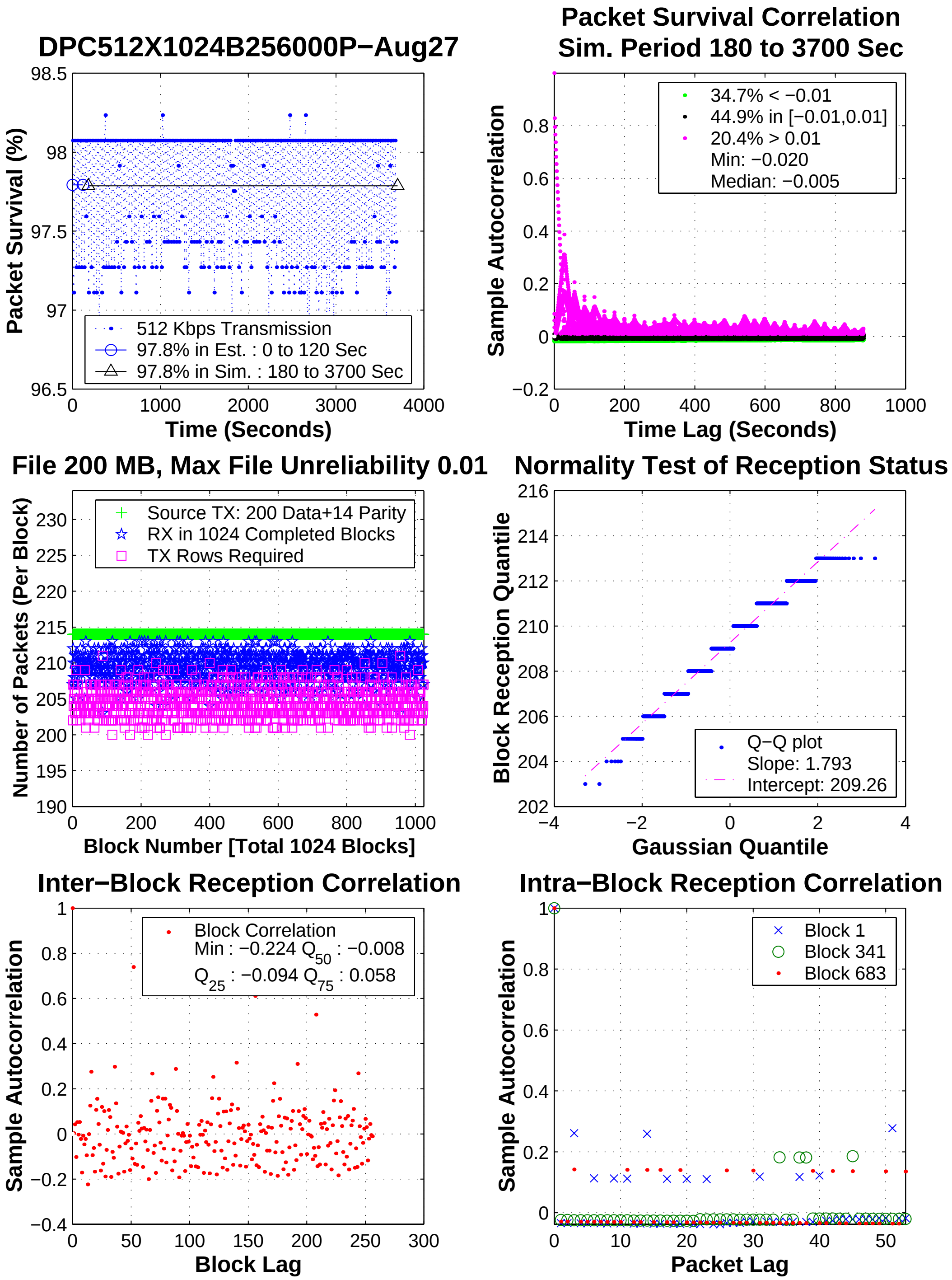


Figure B.29: Trace-driven simulation : August 27, 2001 : HST 512 Kbits/sec transmission rate

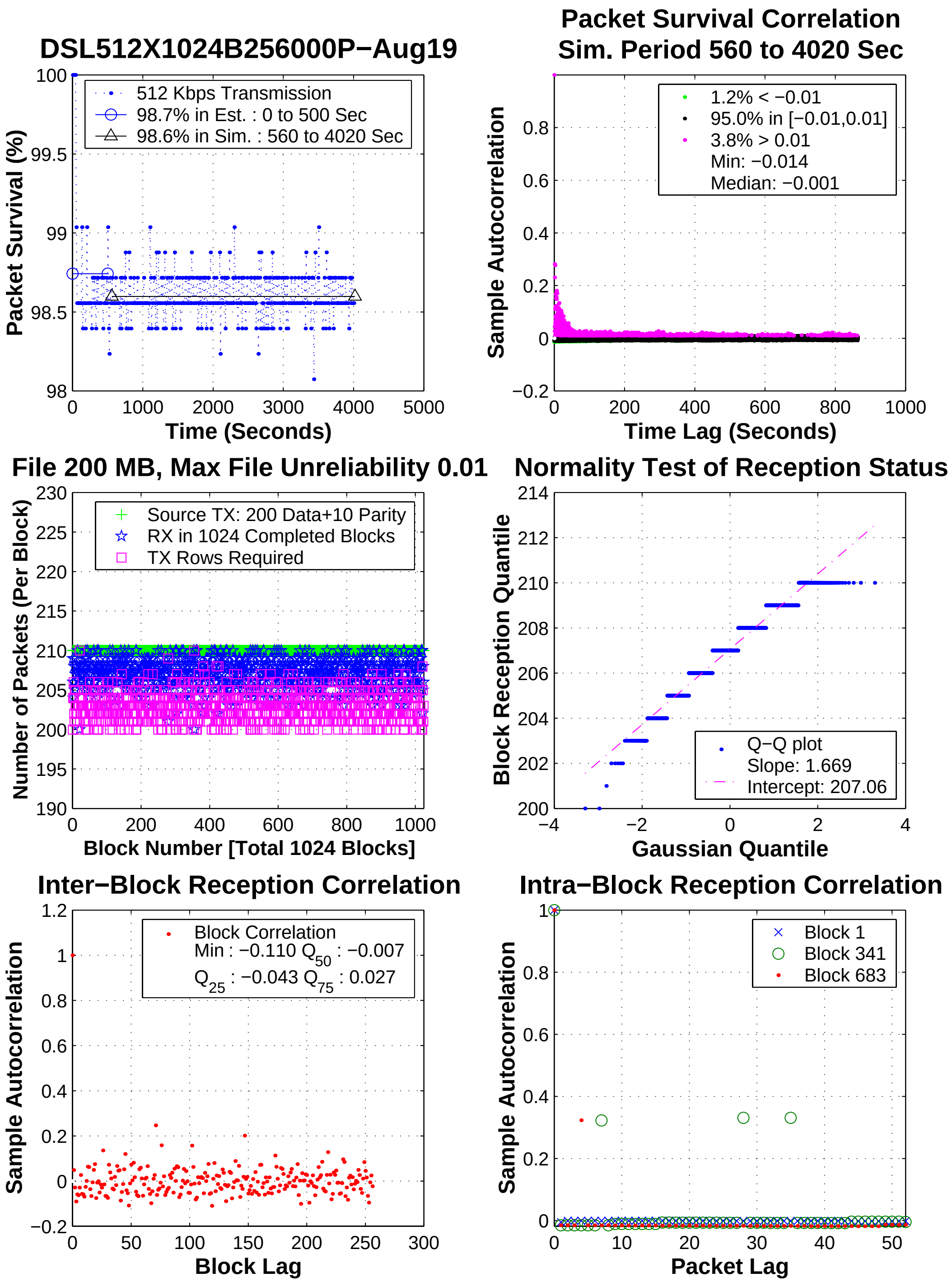


Figure B.30: Trace-driven simulation : August 19, 2001 : DSL 512 Kbits/sec transmission rate

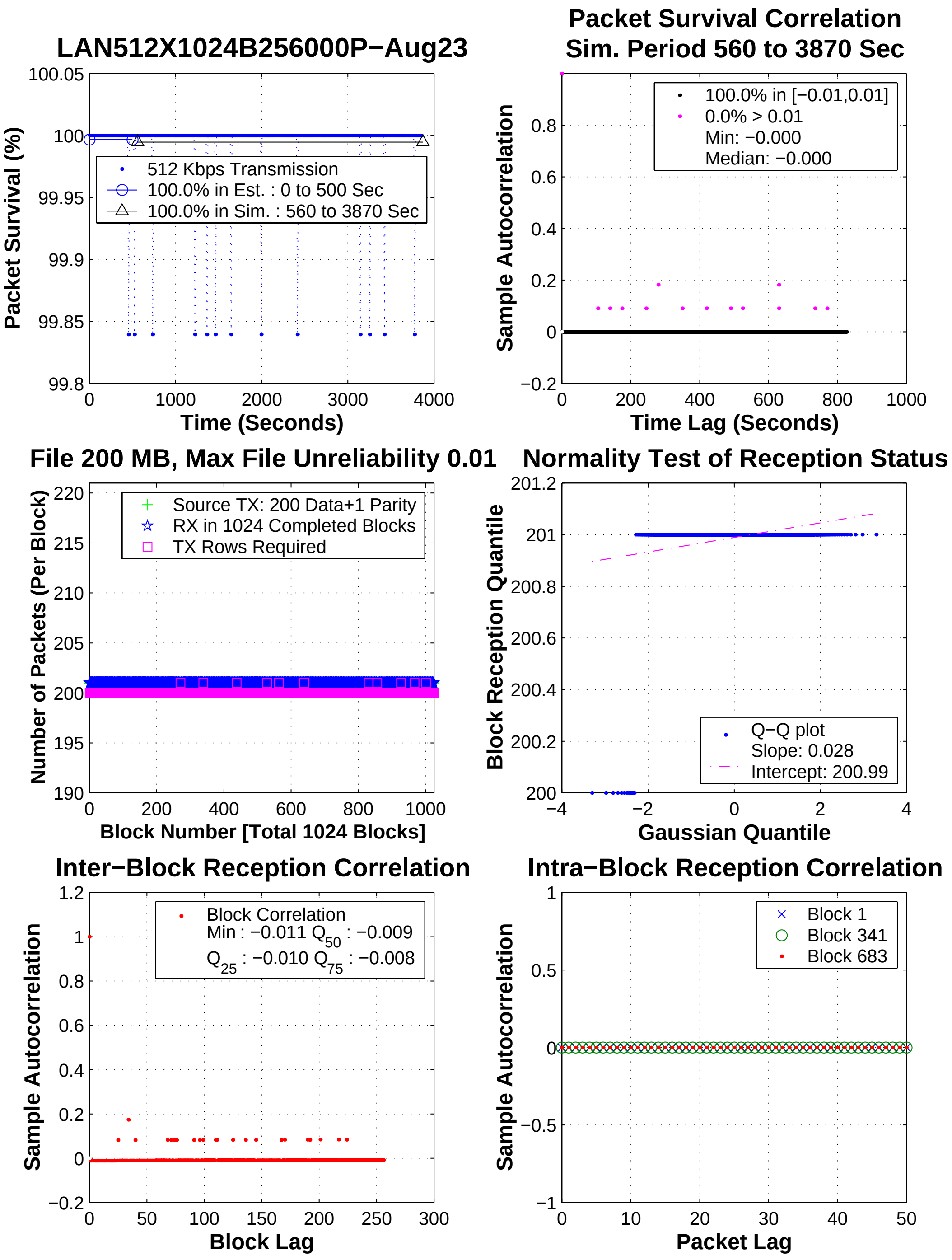


Figure B.31: Trace-driven simulation : August 23, 2001 : campus network 512 Kbits/sec transmission Rate

Bibliography

[Ale00] R. Alexander. Satellite internet architectures, February 2000. CSHCN Advanced Networks Colloquium Series.

[Ash72] Robert B. Ash. *Real Analysis and Probability.* Academic Press, 1972. Section 8.

[BD91] P.J. Brockwell and R.A. Davis. *Time Series: Theory and Methods.* Springer-Verlag New York, second edition, 1991.

[BG92] D.P. Bertsekas and R.G. Gallager. *Data Networks.* Prentice-Hall, Englewood Cliffs, NJ, 1992.

[BKTN98] S. Bhattacharyya, J. Kurose, D. Towsley, and R. Nagarajan. Efficient rate-controlled bulk data transfer using multiple multicast groups. In *IEEE INFOCOM '98*, March 1998.

[BLR98] J.W. Byers, M. Luby, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *ACM SIGCOMM '98*, September 1998.

[BTK99] S. Bhattacharyya, D. Towsley, and J. Kurose. The loss path multiplicity problem in multicast congestion control. In *IEEE INFOCOM '99*, pages 856–863, March 1999.

[CD98] A. Clerget and W. Dabbous. Organizing data transmission for reliable multicast over satellite links. In *Proc. Africom '98*, 1998.

[CD01] A. Clerget and W. Dabbous. Tuf : Tag-based unified fairness. In *IEEE INFOCOM '01*, 2001.

[CMB00] Y. Chawathe, S. McCanne, and E.A. Brewer. Rmx: Reliable multicast for heterogeneous networks. In *IEEE INFOCOM '2000*, March 2000.

[CST99] B. Cain, T. Speakman, and D. Towsley. Generic router assist (gra) building block motivation and architecture, October 1999. RMT Working Group, Internet-Draft (draft-ietf-rmt-gra-arch-00.txt).

[DA01] M.J. Donahoo and S.R. Ainapure. Scalable multicast representative member selection. In *IEEE INFOCOM '01*, 2001.

[DAZ99] M.J. Donahoo, M.H. Ammar, and E.W. Zegura. Multiple-channel multicast scheduling for scalable bulk-data transport. In *IEEE INFOCOM '99*, pages 847–855, March 1999.

[DDC97] C. Diot, W. Dabbous, and J. Crowcroft. Multipoint communication: A survey of protocols, functions, and mechanisms. *IEEE Journal on Selected Areas in Communications*, 15(3):277–290, April 1997.

[DO97] D. DeLucia and K. Obraczka. Multicast feedback suppression using representatives. In *IEEE INFOCOM '97*, 1997.

[FJL$^+$97] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, December 1997.

[GB00] M. Grossglaser and J-C Bolot. On service models for multicast transmission in heterogeneous environments. In *IEEE INFOCOM '2000*, March 2000.

[Gen93] A. Genz. Comparison of methods for the computation of multivariate normal probabilities. In *Computing Science and Statistics 25*, 1993.

[GS99] S. Jamaloddin Golestani and Krishan K. Sabnani. Fundamental observations on multicast congestion control in the internet. In *IEEE INFOCOM '99*, pages 990–1000, March 1999.

[HMMM00] G. Hasegawa, T. Matsuo, M. Murata, and H. Miyahara. Comparisions of packet scheduling algorithms for fair service among connections on the internet. In *IEEE INFOCOM '2000*, March 2000.

[Jai91] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling.* John Wiley & Sons, 1991.

[JNB99] M. Jung, J. Nonnenmacher, and E. W. Biersack. Reliable multicast via satellite: Uni-directional vs. bi-directional communication. In *KiVS '99*, 1999.

[Ker98] R.G. Kermode. Scopped hybrid automatic repeat request with forward error correction (sharqfec). In *ACM SIGCOMM '98*, September 1998.

[KKT98] S. Kasera, J. Kurose, and D. Towsley. A comparison for server-based and receiver-based local recovery approaches for scalable reliable multicast. In *IEEE INFOCOM '98*, March 1998.

[LGT98] L.H. Lehman, S.J. Garland, and D.L. Tennenhouse. Active reliable multicast. In *IEEE INFOCOM '98*, March 1998.

[LNB99] A. Legout, J. Nonnenmacher, and E.W. Biersack. Bandwidth allocation policies for unicast and multicast flows. In *IEEE INFOCOM '99*, pages 254–261, March 1999.

[LNB00] M.S. Lacher, J. Nonnenmacher, and E.W. Biersack. Performance comparison of centralized versus distributed error recovery for reliable multicast. In *IEEE/ACM Transaction on Networking*, April 2000.

[LVS99] M. Luby, L. Vicisano, and T. Speakman. Heterogeneous multicast congestion control based on router packet filtering, June 1999. posted to the Reliable Multicast Research Group (RMRG) mailing list.

[Mac97] J.P. Macker. Reliable multicast transport and integrated erasure-based forward error correction. In *IEEE MILCOM '97*, 1997.

[McA90] A.J. McAuley. Reliable broadband communication using a burst erasure correcting code. In *ACM SIGCOMM'90*, pages 297–306, 1990.

[NBT97] J. Nonnenmacher, E. Biersack, and D. Towsley. Parity-based loss recovery for reliable multicast trasmission. In *ACM SIGCOMM '97*, September 1997.

[NS2] Ucb/lbnl/vint network simulator ns version 2. http://www-mash.cs.berkeley.edu/ns/.

[Pay99] Stephen M. Payne. Reliable multicast via satellite. Master's thesis, Electrical and Computer Engineering, University of Maryland College Park, 1999.

[PR96] J. K. Patel and C. B. Read. *Handbook of the Normal Distribution, 2nd Edition*, volume 150 of *STATISTICS: textbooks and monographs*. Marcel Dekker, Inc., 1996.

[PSLB97] S. Paul, K. Sabnani, C.-H. Lin, and S. Bhattacharyya. Reliable multicast transport protocol (rmtp). *IEEE Jornal on Selected Areas in Communications*, 15(3):407–421, April 1997.

[RJL$^+$00] I. Rhee, S.R. Joshi, M. Lee, S. Muthukrishnan, and V. Ozdemir. Layered multicast recovery. In *IEEE INFOCOM '2000*, March 2000.

[RKT98] D. Rubenstein, J. Kurose, and D. Towsley. Real-time reliable multicast using proactive forward error correction. In *IEEE NOSSDAV '98*, July 1998.

[RKT99] D. Rubenstein, J. Kurose, and D. Towsley. The impact of multicast layering on network fairness. In *ACM SIGCOMM '99*, September 1999.

[RKTK99] D. Rubenstein, S. Kasera, D. Towsley, and J. Kurose. Improving reliable multicast using active parity encoding services (apes). In *IEEE INFOCOM '99*, pages 1248–1255, March 1999.

[Rod01] Dennis Roddy. *Satellite Communications, third edition.* McGraw-Hill, 2001.

[RV98] L. Rizzo and L. Vicisano. Rmdp: an fec-based reliable multicast protocol for wireless environments. In *ACM Mobile Computing and Communications Review*, volume 2, pages 23–31, 1998.

[SSZ98] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks. In *ACM SIGCOMM '98*, September 1998.

[SV95] M. Shreedhar and G. Varghese. Efficient fair queueing using deficit round robin. In *ACM SIGCOMM '95*, 1995.

[TC00a] A. Tunpan and M.S. Corson. Bulk data multicast rate scheduling for hybrid heterogeneous satellite-terrestrial networks. In *ATIRP '2000*, March 2000.

[TC00b] Apinun Tunpan and M. Scott Corson. Bulk data multicast rate scheduling for hybrid heterogeneous satellite-terrestrial networks. In *IEEE ISCC '2000*, July 2000.

[TC01] A. Tunpan and M. Scott Corson. On satellite multicast to heterogeneous receivers. In *IEEE ICC '01*, June 2001.

[TKP97] D. Towsley, J. Kurose, and S. Pingali. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. *IEEE Journal on Selected Areas in Communications*, 15(3):398–406, April 1997.

[Ton90] Y. L. Tong. *The Multivairate Normal Distribution.* Springer-Verlag, 1990.

[VRC98] L. Vicisano, L. Rizzo, and J. Crowcroft. Tcp-like congestion control for layered multicast data transfer. In *IEEE INFOCOM '98*, 1998.

[WHK00] T. Wong, T. Henderson, and R. Katz. Tunable Reliable Multicast for Periodic Information Dissemination. *To Appear Special Issue of the Journal on Special Topics in Mobile Networking and Applications (MONET) on Satellite Based Information Services*, 2000.

[YMKT99] M. Yajnik, S. Moon, J. Kurose, and D. Towsley. Measurement and modelling of the temporal dependence in packet loss. In *IEEE INFOCOM '99*, March 1999.

Index