EFFICIENT CRL DISTRIBUTION USING MULTICASTING AND UNICASTING

by

HANSEN MIN HENG WANG

B.A.Sc. (Electrical and Computer Engineering), University of British Columbia, 1995

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

March 2001

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Electrical + Computer Engineering

The University of British Columbia
Vancouver, Canada

Date 2001/04/16

Abstract

Communication costs in providing certificate status information to those who wish to validate public key certificates have been cited as the most expensive component of operating a large scale Public Key Infrastructure. One mechanism for providing certificate status information is a Certificate Revocation List (CRL). This thesis proposes a system for cost effective distribution of CRLs using a combination of multicasting and unicasting. The proposed system for CRL distribution calls for periodic and aperiodic multicasting of Delta CRLs to reduce network bandwidth requirements and peak CRL request rates in unreliable networks. An analytical model and a simulation model are used to compare the network bandwidth requirements of the proposed system against a system which uses only unicasting for CRL distribution. Results show that the proposed MCA system which multicasts Delta CRLs aperiodically requires significantly less network bandwidth and reduces peak CRL request rates. For an example network, the communication cost of the MCA system is 89% less than that of the system which only uses unicasting. The communication costs for the MCA system is also less sensitive to the location of the CRL Repository. The MCA system may be retrofitted to legacy client programs which may only obtain CRLs using unicasting.

Table of Contents

List of Tables

List of Figures

Acknowledgment

I would like to thank my thesis advisor, Dr. Cyril Leung, for his guidance, suggestions, sound reasoning, and thoroughness in his questioning throughout this work.

I would also like to thank the many friends that I've made during my time at the Department of Electrical and Computer Engineering as both a Masters student and research engineer for the many stimulating intellectual and technical discussions.

Finally, I would like to thank my parents for their encouragement, understanding and support through my years in university.

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grant OPG0001731.

Chapter 1 Introduction

The notion of a paperless society has been heralded by researchers, members of the Internet community and proponents of e-Commerce as the next great step in streamlining everyday transactions [1-4, 28]. In a paperless society, everyday transactions are streamlined using bits and bytes over networks or the Internet rather than utilizing the postal system or couriers. People or computers may authorize transactions with digital signatures [5] and have the documents arrive at their destinations in a matter of milliseconds. Document handling will be much faster, more cost effective and often more accurate and consistent through electronic means than with human intervention. At the core of implementing a paperless society is the Public Key Infrastructure (PKI). The PKI embodies the hierarchy of a multilevel system of trust using certificates. These certificates are held by entities residing in the PKI hierarchy. A certificate consists of among other objects, the public key portion of the public/private key pair belonging to the entity. Each certificate has a validity period attached to it stating times during which the certificate may be used. From time to time, the issuer of the certificate will want to revoke a certificate before the certificate's expiration date if the confidentiality of the private key of the certificate has been reported to be possibly compromised. The certificate issuer will also want to revoke a certificate before the certificate's expiration date if the holder of the certificate is no longer entitled to use the certificate or affiliated with the certificate issuer. In [1], the communication cost of providing certificate revocation information, or more generally certificate status information, to a large number of users is estimated to be the most expensive part of running a large scale PKI such as a Federal Government PKI. Other costs of operating a large scale PKI are: staffing and equipment costs. The goal of this thesis is to study methods for reducing the cost of conveying certificate status information in a large scale PKI.

1.1 Motivation for Research

With the legalization of digital signatures, increasing popularity of e-Commerce, the trend towards e-Everything, and the need for streamlining high volumes of transactions will create the demand for a large scale PKI. The transition towards a digital society will be of trial and error. Potential cheaters will always be on the lookout for new loopholes and opportunities for personal gain. Cheaters in a digital society have one key ability which was not readily available previously: the ability to automate. Cheaters may use automation to attack the smallest window of opportunity. For this reason, certificate status information must be issued frequently to ensure up-to-date information. However, more frequent updates result in increased communication costs. Thus there is a need to balance the cost of supplying certificate status information against acceptable levels of risk for service providers (such as banks, creditors, etc.) and/or end-users (bank card users, smart card users, etc.). The risk of loss is always present and cannot be easily mitigated. However risk may be controlled and reduced by more frequent certificate status updates at the expense of increase in communications costs. In light of the Mitre Report [1] which identified communications cost as largest component of operating a large scale PKI, researchers [6-13, 28, 30] have studied methods for reducing the communication cost of conveying certificate status information. The recommendations include Certificate Revocation Status [9], Certificate Revocation Tree [10], OCSP [13] and Certificate Revocation Lists [6, 7, 11, 12, 30]. The work in this thesis is based on CRLs and a cost effective system of distributing CRLs using a combination of multicasting and unicasting or otherwise known as CRL push and pull.

1.2 Research Goals and Contributions

Most works [6-11] on reducing the cost of conveying certificate status information have

been based on using unicasting as the transport method. A novel approach to distributing certificate status information using Windowed Revocation was proposed in [30]. Windowed Revocation uses unicasting and multicasting to provide CRLs to Relying Parties (RPs) that need to validate certificates. This thesis proposes a system which uses multicasting and unicasting to distribute Sliding Window Delta CRLs [7]. It is mentioned in Section 2.1.3.5 that the main difference between the Sliding Window Delta CRL and Windowed Revocation systems is that in the latter the Certificate Authority (CA) needs to be on-line at all times to respond to certificate requests from the RPs. The system proposed in this thesis uses CRLs that are more closely related to traditional CRLs in that the CA does not need to respond to individual RP requests. The performance of the Hybrid CRL Multicast/Unicast system is studied using computer simulation and compared with that of a system which uses unicasting only. The study will provide the user with information on how much resource is expected to be needed, in a given network environment. One limitation of the study is that it does not account for processing delays, transmission delays, or router queueing delays. However, unlike most previous works, the network environment including network topology, network availability and packet errors are taken into consideration. Since the performance of multicasting is highly dependent on the quality and topology of the network, the inclusion of these aspects is necessary to fairly compare the merits of multicasting and unicasting. The network availability and packet error probability information is based on data extracted from Internet Service Provider's Service Level Agreements (ISP SLA) for their network backbones. The network topologies are generated using the Tiers [22, 23] model. The results demonstrate the effectiveness of the proposed Hybrid Multicast/Unicast system of distributing CRLs over unicast only method. The proposed Hybrid Multicast/Unicast system is also designed so as to retrofit onto existing client (end user) programs through the implementation of a multicast

receiving module at the end user side.

1.3 Thesis Outline

This thesis is organized as follows.

Chapter 2 provides background information on PKI, Certificates and in particular, X.509 standard Certificates and the participants of the PKI. Next, brief descriptions of current methods of providing certificate status information are provided. Chapter 2 concludes with information on the transportation methods, multicasting and unicasting, and how to compare multicasting and unicasting costs.

Chapter 3 describes the proposed Hybrid Multicast/Unicast system of CRL distribution. The analytical model for both the proposed Hybrid Multicast/Unicast system and the unicast only system is developed and some analytical results are derived.

Chapter 4 gives a detailed description of the simulation model used for the Hybrid Multicast/Unicast system and the unicast only system. Information on the inputs and outputs of simulation model program are provided.

Chapter 5 discusses and compares the results of the simulation model with those from the analysis as well as the performance of the Hybrid Multicast/Unicast system against the unicast only system with respect to various network parameters and topologies.

Chapter 6 summarizes and concludes this thesis.

Chapter 2 Background Information

Since the publication of the Mitre report [1] which estimated the high communications cost of running a Federal Government PKI, researchers have been searching for ways of reducing these costs. Information on PKIs such as certificates, the parties involved, and the pros and cons of current methods of providing certificate status information are provided in the following sections. To analyze the Hybrid Multicast/Unicast CRL distribution proposal, additional information on the two data transport methods, unicasting and multicasting, are provided. Finally, the generation of large random networks that are used by the simulation program is discussed.

2.1 Public Key Infrastructures

A Public Key Infrastructure (PKI) is a hierarchical system of trust that helps enable secure electronic transactions. A PKI provides two basic services: certification and validation [14]. Certification is the process of binding pieces of information to an entity called the Certificate Holder (CH). The binding process is done through a trusted third party called the Certificate Authority (CA) using public key cryptography and certificates. Validation is the process of verifying the validity of the certification [14].

2.1.1 Certificates

In broad terms, a certificate is a collection of information that is digitally signed by its issuer (CA) [14]. The CA digital signature acts to bind the collection of information to the CH to form a certificate. Included in the collection of information in the certificate is the public key of the CH. The public keys in the certificates are used by others who may not have had any previous contact with the CH to verify the CH's digital signatures, to send confidential information to the

CH or to create a secure communications channel with the CH. These certificates facilitate the dissemination of public keys with high integrity [2]. One of the current certificate standards is CCITT X.509 [12]. Each X.509 certificate has a serial number, a unique name that identifies the CH, name of the CA which issued the certificate, the time and date of the certificate was issued, the time and date of the certificate expiry, the public key of the CH and the cryptographic algorithm which the public/private key pair may be used for. These certificates may also have other extensions and options. A full list of the fields and extensions for the X.509 certificates may be found in [12]. The entire certificate is then signed with the CA's private signing key. The CA's public key used to verify the authenticity and integrity of the certificate is assumed known to all parties involved in the PKI. The problem of public key distribution is non-trivial [5].

2.1.2 Parties Involved

There are several parties involved in a Public Key Infrastructure: the Certificate Authority (CA), the Certificate Holder (CH), the Repositories, and the Relying Parties (RP) as shown in Figure 2.1. At the top of the PKI are the CAs. The CAs issue and revoke certificates. The Repositories receive certificate status information from the CA and provide this information to the RPs for the purpose of validating certificates presented to it by CHs to initiate electronic transactions.

2.1.2.1 Certificate Authority

The Certificate Authority (CA) holds the highest position in the PKI hierarchy. In practice, the CA may be the national post office, a bank, a corporation, government agency, school or CA company. There may also be many levels of CA but this is not important in this thesis. Information on multilevel CAs are discussed in [15].

The basic role of the CA is to issue and revoke certificates. Before issuing a certificate to

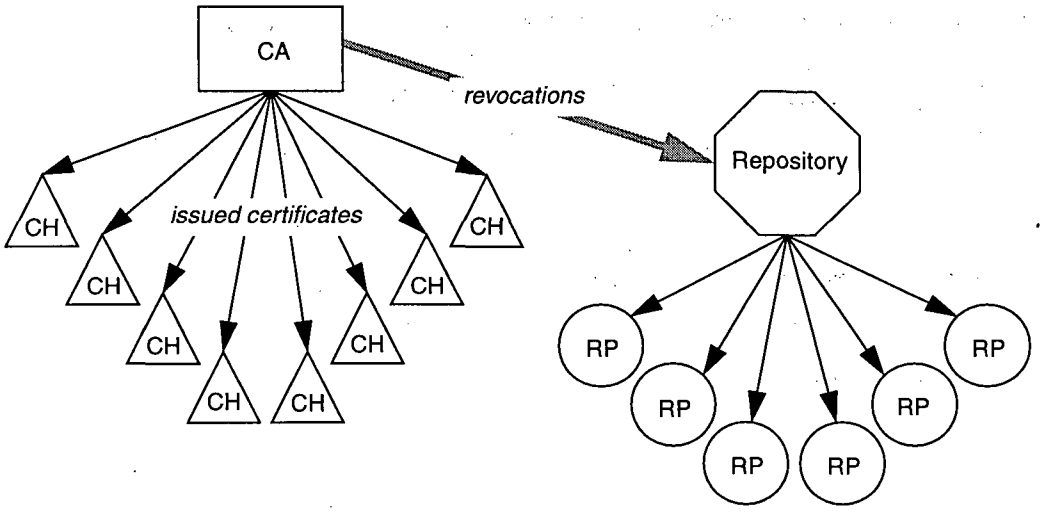

Figure 2.1 The parties involved

an entity, the CA must verify that the entity has rights to the name to be bound to in the certificate and qualification to obtain a certificate [2]. The CA may also verify the legitimacy of the company or require proof of identity if the entity is a person. Other policies, qualifications and restrictions may be enforced by the CA. Policies, qualifications and restrictions are written in the Certificate Practice Statement (CPS) as provided by the CA. The other basic task of the CA is to revoke certificates that it has issued but have not yet expired and provide that information to the RPs. The mechanisms by which certificate status information are supplied is discussed later in this chapter.

Each CA has one or more private keys for signing certificates. It is vital that the CA keep these keys secret. If another entity has a copy of a CA's private key, it may forge new certificates at will. The CA's public key is made available to all who need to use it to verify the integrity of a certificate that the CA has issued. A certificate that passes the integrity check does not guarantee its validity. The certificate may have been revoked. To check the validity, the RP must check the

certificate's status. The RP trusts that the CA has completed the necessary background and qualifications check before issuing the certificate and that the CA will issue timely information on the validity of certificates that it has issued. However, it is up to the RP to verify the validity of the CA's public key.

2.1.2.2 Repositories

The Repository acts as an intermediary in providing RPs with certificate status information in a large scale PKI. These Repositories do not have to be affiliated with the CA and may be a third party entity. The CA may not have the resources or technical expertise to provide a large number of RPs in various locations with certificate status information. These servers would need to have high bandwidth and processing capabilities. A CA which wants to provide faster response times for RPs in a wide geographical range may want to enlist Repository services located at various geographical locations [6]. Many smaller CAs may enlist the services of large Repository services so as to make its certificates validatable in a wide geographical range with less delay and higher reliability without having to bear the entire cost of setting up and maintaining a large number of Repositories in various geographical locations.

In the case of the CRL system of providing certificate status information, the RP does not need to trust the Repository [2, 8]. This is because the CRLs are signed by the CA which issues the CRLs. The RP needs to only verify that the CRL was issued by the correct CA and does not need to verify the identity of the Repository it received the CRL from. If a rogue Repository altered or removed information in the CRL, the CRL will not pass the integrity check performed by the RP unless the CA's private key has been compromised. Without a copy of the CA's private key, the rogue Repository cannot create and sign a fake CRL and make the RP think that the altered version originated from the CA.

The basic task of the Repository is to receive updated copies of certificate status information from one or more CAs and send the certificate status information update to RPs when requested.

2.1.2.3 Certificate Holder

The Certificate Holder (CH) is the entity for which the CA issues the certificate. The CH may be a person, a computer system or host, an e-commerce server which takes credit card information from a person over the Internet, or a corporation. The Certificate Holder is at the bottom of the PKI hierarchy and is trusted by no one [8, 9, 15]. The CH's certificate must be verified for integrity, authenticity and validity each time it is presented to a RP to perform a transaction.

2.1.2.4 Relying Parties

The Relying Parties (RP) are the entities to whom the CH's present their certificates to carry out a transaction. In practice, the RP may be an email client program on a computer used to read secure mail that was signed with a CH's private key, a vending machine which may use certificates stored on smart cards for purchases, an on-line store, government taxation office tax return processing computer or building entrance which uses certificates stored on smart cards. The RP has a copy of the certificate that belongs to a CA that it trusts and need not trust anything or anyone else [9]. The RP's are not trusted by either the Repository, CA nor CHs [8, 9].

The basic operation of the RP is as follows. When a CH presents its certificate to a RP to make a transaction, the RP will need to verify the integrity and authenticity of the certificate presented by the CH. If the certificate is signed by a CA that the RP trust, then the RP will use the CA's public key to verify the integrity and authenticity of the certificate. Next, the RP will check

to see if the certificate is valid if it has not already expired. Certificate validation may be done using various validation check mechanisms such a Certificate Revocation Lists (CRL) [6, 7, 12], Certificate Revocation Status (CRS) [9], Certificate Revocation Tree (CRT) [10], On-line Certificate Status Protocol (OCSP) [13] and various other means under investigation. Some of these mechanisms are described briefly in the following sections. Once the RP has validated the certificate presented by the CH, the electronic transaction may proceed.

2.1.3 Certificate Revocation

From time to time, a CA needs to revoke an unexpired certificate that it had previously issued. Reasons for revoking a certificate include [1]:

- the Certificate Holder has a change of affiliation, and is no longer entitled to services which the certificate helps unlock or
- the Certificate Holder's private key which is paired with the public key bound to the certificate has been compromised.

According to the Mitre report [1], 10% of all certificates are expected to be revoked of which 5% are attributed to each of the two reasons above. If the private key of a corresponding certificate was found to be compromised, the CH has changed jurisdiction or is no longer affiliated with the CA which issued the certificate, the services or accesses to which the certificate and private key are used to access will be vulnerable to cheaters. The period of vulnerability is from the time the private key of the certificate is found to be compromised or when the CH changes affiliation to the expiry time of the certificate [15]. The purpose of the certificate revocation mechanism is to reduce the period of vulnerability to risks of loss.

There are several mechanisms presented by researchers for providing certificate status information. One of the goals is to reduce the transmission costs of providing certificate status information while still providing timely and frequent updates. The mechanisms used will have different levels of risk and cost. The acceptable level of risk needs to be balanced against the cost of operation. Proposed mechanisms are:

- none (not using any revocation mechanism), but use shorter certificate validity periods instead,
- On-line Certificate Status Protocol (OCSP),
- Certificate Revocation System (CRS),
- Certificate Revocation Tree (CRT) and
- Certificate Revocation List (CRL).

The pros and cons and trade-offs of each revocation mechanism are now discussed.

2.1.3.1 None

A PKI which uses short-lived certificates is proposed in [16] to simplify management of certificates. These short-lived certificates do not use a certificate revocation mechanism. The validity period of short-lived certificates are in the range of 10 hours or less. With short certificate validity periods, if the private key has been compromised, the vulnerability period is short. There is often very little evidence that a private key has been compromised. In the case of long-lived certificates, by the time the compromised private key's certificate has been added to the revocation list, the damage may have already been done. Short-lived certificates are suitable for computer login applications. One disadvantage of using short-lived certificates is that they are inconvenient for CHs which need to use the keys for longer than the lifetime of the certificate. In this case the

CHs need to obtain a new certificate. This may not be acceptable for critical applications, e.g. electronic locks which uses certificates to control access hospital equipment or access to a hospital's pharmaceutical storage rooms.

2.1.3.2 On-line Certificate Status Protocol (OCSP)

The On-line Certificate Status Protocol [13, 28] provides the most up-to-date information on the status of a certificate to the Relying Party on a per validation request basis. Whenever a RP has a validation request, it makes a request to the CA for information on the status of the certificate it wishes to validate. The CA sends back the requested status information along with a CA digital signature for that response. Some revocation mechanisms provide a negative response if a certificate is no longer valid, otherwise no response is given. OSCP provides both a negative and a positive response to the status of a certificate. A negative response indicates to the RP that the certificate has been revoked. A positive response indicates to the RP that a certificate is valid. The problem with only providing a negative response is that if a certificate was a fake, such a revocation mechanism would not be able to indicate this. The drawbacks to OCSP is that: it is very expensive in terms of communication and server costs, the RP must be on-line, and the CA server machine which holds the CA's private key used to sign the OCSP responses must be on-line. Each response has a large overhead. Each reply has to be signed by the CA and so the use of a Repository is unfeasible. The CA's server must be fast and powerful to handle the workload of signing the response to each validation request. The high cost of running OCSP permits it to be used only in the most risk adverse and time critical applications such as on-line stock trading transaction processing [13].

2.1.3.3 Certificate Revocation Status (CRS)

The Certificate Revocation Status [9] mechanism is similar to OCSP in that it also

provides both positive and negative responses on the validity of certificates. The disadvantage is that CRS has significantly higher communications cost between the CA and the Repository. On the other hand, CRS is claimed to provide a 900-fold reduction in bandwidth cost over Certificate Revocation Lists for communications between the Repository and RPs using Federal Government PKI estimates and parameters as presented in [1].

In the CRS system, the CA creates new certificates with two additional values Y^I and N for Yes and No respectively. The values, Y^I and N, are 100 bits long each. Y^I and N, are generated using:

$$Y^I = H^I(Y_0) \tag{2.1}$$

and

$$N = H(N_0) \tag{2.2}$$

where I is the number of certificate status update intervals till the expiration of the certificate and H is a one way hash function [5]. Y_0 and N_0 are secret random values that are unique to each certificate and known only to the CA. The value Y^I and N are included in the certificate and signed by the CA with the usual certificate information. At the beginning of each certificate status update interval and for each certificate, the CA calculates and publishes Y^{I-i}, where

$$Y^{I-i} = H^{I-i}(Y_0), \tag{2.3}$$

if the certificate is valid; otherwise, if the certificate has been revoked, the CA publishes the secret

value N_0. The value i is the number of certificate status update intervals since the creation of the certificate. Note that only the CA with the secret values Y_0 and N_0 may generate these messages. All these new values along with their corresponding certificate serial numbers are sent to the Repository. A RP makes a request for certificate status information each time it needs to validate a certificate that has not been previously presented in the same certificate status update interval. The Repository responds to the request with a Y^{R-i} value, indicating a valid certificate, or N_{R0} value, indicating a revoked certificate. Note that the certificate status response is not signed. The RP verifies whether the value is authentically generated by the CA by hashing the value received from the Repository i times if the value is Y^{R-i} as shown in (2.4).

$$H^i(Y^{R-i}) = Y^R \tag{2.4}$$

If $Y^R = Y^I$, where Y^I is the value stored in the certificate, then the response from the Repository is authentic. If the response from the Repository is N_{R0}, the RP will calculate

$$H(N_{R0}) = N_R. \tag{2.5}$$

If $N_R = N$, where N is the value stored in the certificate, then the response from the Repository is authentic. If the RP requests for status on a forged certificate, then the Repository will have no authentic Y or N value in response to the request. In this case, the Repository will send a message saying that there is no such certificate and so the certificate will be rejected by the RP. The CRS scheme is based on the fact that no one else except for the CA can calculate the values $(Y_0, Y^1, Y^2, \ldots, Y^{(I-i)-1})$, while others may calculate $(Y^{(I-i)+1}, Y^{(I-i)+2}, \ldots, Y^I)$ if the

current certificate status update period is i and the CA has published the value $Y^{(l-i)}$. The short responses to single request are what make CRS more efficient than CRL. The ability to serve and cache certificate status information make CRS more scalable than OCSP.

2.1.3.4 Certificate Revocation Tree (CRT)

Certificate Revocation Trees (CRTs) [10] provide a short proof to the requesting RP as to whether or not a particular certificate has been revoked or whether its status is known at all. A CRT consists of a hash tree whose leaves represent revoked certificates identified by their serial numbers. The non-leaf nodes of the tree are concatenated and digitally signed hashes of its child nodes. When a RP requests the Repository for information on the status of a particular certificate, the Repository responds with the subtree which contains the path from a root to an appropriate leaf. Since the node values are created from their child node values, and hashed using a one-way function and signed with a digital signature, the values of the nodes are made difficult if not impossible to forge [8].

By only sending out a subtree with the information needed by the RP, a reduction in transfer cost is achieved over CRL. However, the main disadvantage of CRT is that any certificate addition or removal requires the entire CRT to be recalculated and is thus computationally expensive to operate [8].

2.1.3.5 Certificate Revocation List (CRL)

Certificate Revocation Lists (CRLs) are the simplest of the systems described here. CRLs are basically lists of unexpired certificates that have been revoked. An updated CRL is issued by the CA and signed with the CA's private key periodically. Additionally, each CRL has an issuance time and an expiration time so that the RPs will always use the most up-to-date CRL. Unlike CRS

and OCSP, CRLs only give a negative response as to the status of a certificate. The main problem with traditional CRL is that they are potentially very long when the number of issued certificates that have not yet expired is large. When an RP needs to validate even a single certificate, it would need to have the most recent full CRL. Once the RP has a copy of the current CRL, the RP may cache the CRL for off-line validations until the CRL expires. Researchers [6, 7, 11, 12] have come up with variations on the traditional CRL to reduce communication costs by reducing the need to download the entire CRL. Some of these are segmented CRLs and Delta CRLs. Segmented CRLs divide up the CRL into small segments whereas Delta CRLs list only the changes to the CRL since previous issuance period(s). CRL segments and Delta CRL are much shorter than Full CRLs. There are several variations of segmented and Delta CRLs.

Traditional CRL: When a RP has a validation request and its copy of the CRL has expired, the RP will make a request for a CRL update. The Repository will then send the RP the Full CRL. As the Full CRL can be very large, the traditional CRL method is wasteful and bandwidth intensive [6].

Segmented CRL: One method to avoid sending the Full CRL is to divide up the Full CRL into smaller sections. The division may be done in various ways such as by serial number. Although the segmented CRL [6] reduces bandwidth usage, it does not reduce server loading. In fact, server peak request rates stay the same and average request rates increase with increases in number of segments [6]. The bandwidth requirement peaks at times when new CRLs are issued and drops off exponentially until the next CRL issuance. Although purchasing equipment to handle high peak request rates is a one time cost, leasing communications bandwidth to handle the peak request rates is not.

Segmented CRL with Stagger Issuance: By staggering the CRL issuance times of each CRL

segment [6], the peak request rates may be lowered. However the reduction in peak rates is not directly proportional to the number of segments. There is an optimal number of segments which minimizes the peak rate. As the number of segments increase over the optimal number of segments, the peaks will go back up and tend toward the same peak rate as traditional CRL. Additionally the average request rate will increase to the point where the peak and the average are the same. The equation for the peak CRL request for the segmented and staggered issuance CRL is [6]

$$R_{peak} = \frac{N_{RP}\lambda_{Val}}{S}\sum_{i=0}^{S-1} e^{-\lambda_{Val}\frac{T}{S^2}i}, \tag{2.6}$$

where N_{RP} is the number of RPs, λ_{Val} is the rate of validation requests coming to an RP, S is the number of CRL segments, and T is the time interval of S CRL segment issuance intervals.

Overissued CRL: Rather than dividing the CRL into segments, the Overissued CRL method [6] divides the RPs into groups and stagger the expiration times of CRLs. In this way, there are several different CRLs that are valid at the same time but expire at different times. By this method, peak request rates are effectively reduced.

Traditional Delta CRL: The number of changes between CRLs of consecutive CRL issuance intervals is expected to be small. To reduce bandwidth needs and download time, Delta CRLs have been proposed [12]. Delta CRLs contain incremental changes to the CRL since a previous CRL. To use Delta CRLs, the RP must have a prior copy of a Full CRL and the Full CRL must be recent enough that the Delta CRL may update it. The use of Delta CRLs yield significant reductions in communication costs only if the probability of the RPs requesting Delta CRLs is

much higher than the RPs requesting Full CRLs. As shown in [7], Traditional Delta CRLs have a high probability of RPs not being able to apply the Delta CRL and results in the Full CRL being downloaded. Downloading the Full CRL may be regarded as a penalty as the Full CRL may be several times larger than a Delta CRL. The root of this problem is that Traditional Delta CRLs issue a Full CRL at longer intervals and Deltas CRLs at shorter intervals. Whenever the Full CRL is issued, a Delta CRL is also issued along to update the previous Full CRL to the current Full CRL. All subsequent Delta CRLs may be applied only to the latest Full CRL. If a RP has not made any validations in the interval since both the Full and Delta CRL was issued, the RP will be unable to use the current Delta CRLs and thus will need to download the Full CRL.

Sliding Window Delta CRL: To overcome the problem with Traditional Delta CRLs, the Sliding Window Delta CRL was proposed in [7]. The Sliding Window Delta CRL uses a fixed window size whereas the traditional Delta CRL uses a varying window size as shown in Figure 2.2. It is

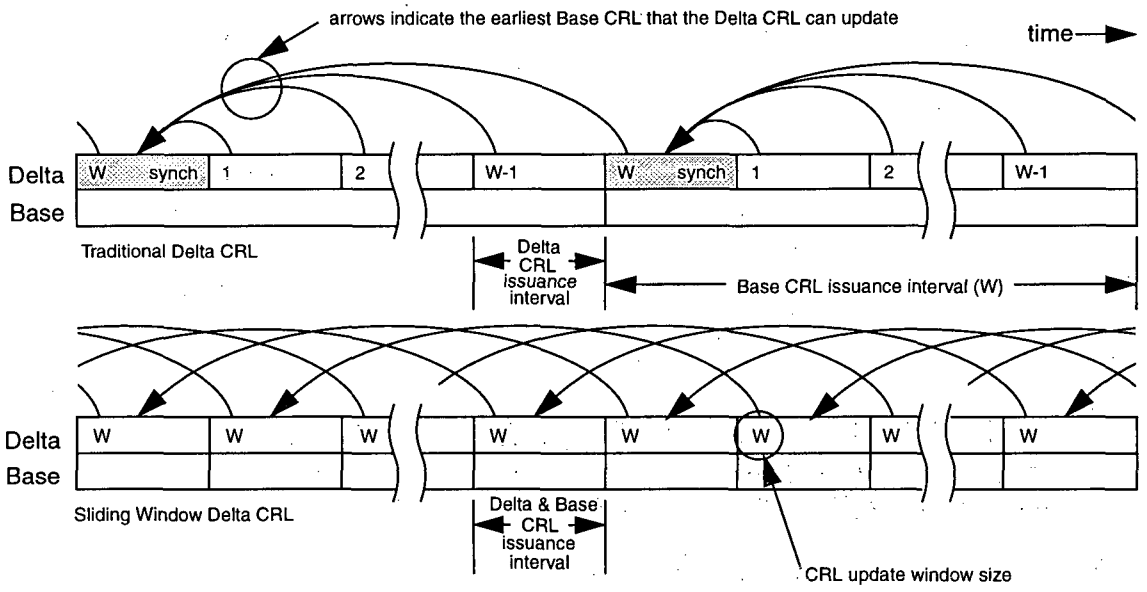

Figure 2.2 Traditional Delta CRL vs. Sliding Window Delta CRL time line

stated in [7] that the traditional Delta CRL system issues Base CRLs and Delta CRLs at different frequencies. Delta CRLs are issued more frequently than Base CRLs. In the traditional Delta CRL system, a "synch" Delta CRL is the only Delta CRL that can update the previous Base CRL to the current Base CRL. If a RP did not have a validation request during the interval in which the "synch" Delta CRL is valid, it will have to download both a Base CRL and a Delta CRL to complete its next validation request. Additionally, if the RP did not have a validation request during the previous Base CRL issuance interval, it will also have to download both a Base CRL and a Delta CRL to complete its next validation request. It is shown in [7] shows that the traditional Delta CRL system does not provide a significant reduction in Base CRL requests over not using Delta CRLs. By issuing both a Base CRL and a Delta CRL at each CRL issuance interval and applying a fixed CRL update window size, the Sliding Window Delta CRL system can yield a significant reduction in Base CRL requests [7].

Windowed Revocation: Rather than reducing the probability of Base CRL requests, Windowed Revocation [30] eliminates the Base CRLs entirely. By shortening the time in which a revoked certificate needs to appear in the CRL and requiring the RPs to obtain certificates from the CA, Windowed Revocation effectively shortens the length of the CRL. There are two ways of providing certificate status information in Windowed Revocation: explicit and implicit (Figure 2.3). Explicit revocation refers to the use of CRLs. Implicit revocation refers to the requirement for RPs to obtain certificates they wish to validate from the CA. If the CA does not provide the requested certificate, then the certificate has been revoked. If the RP receives the requested certificate from the CA, then the certificate is valid. The received certificate is cached by the RP and is valid for Time-to-Live (TTL) as shown in Figure 2.3. If a RP needs to validate a certificate with an expired TTL that is in its cache, it will need to re-request for the certificate or request for the

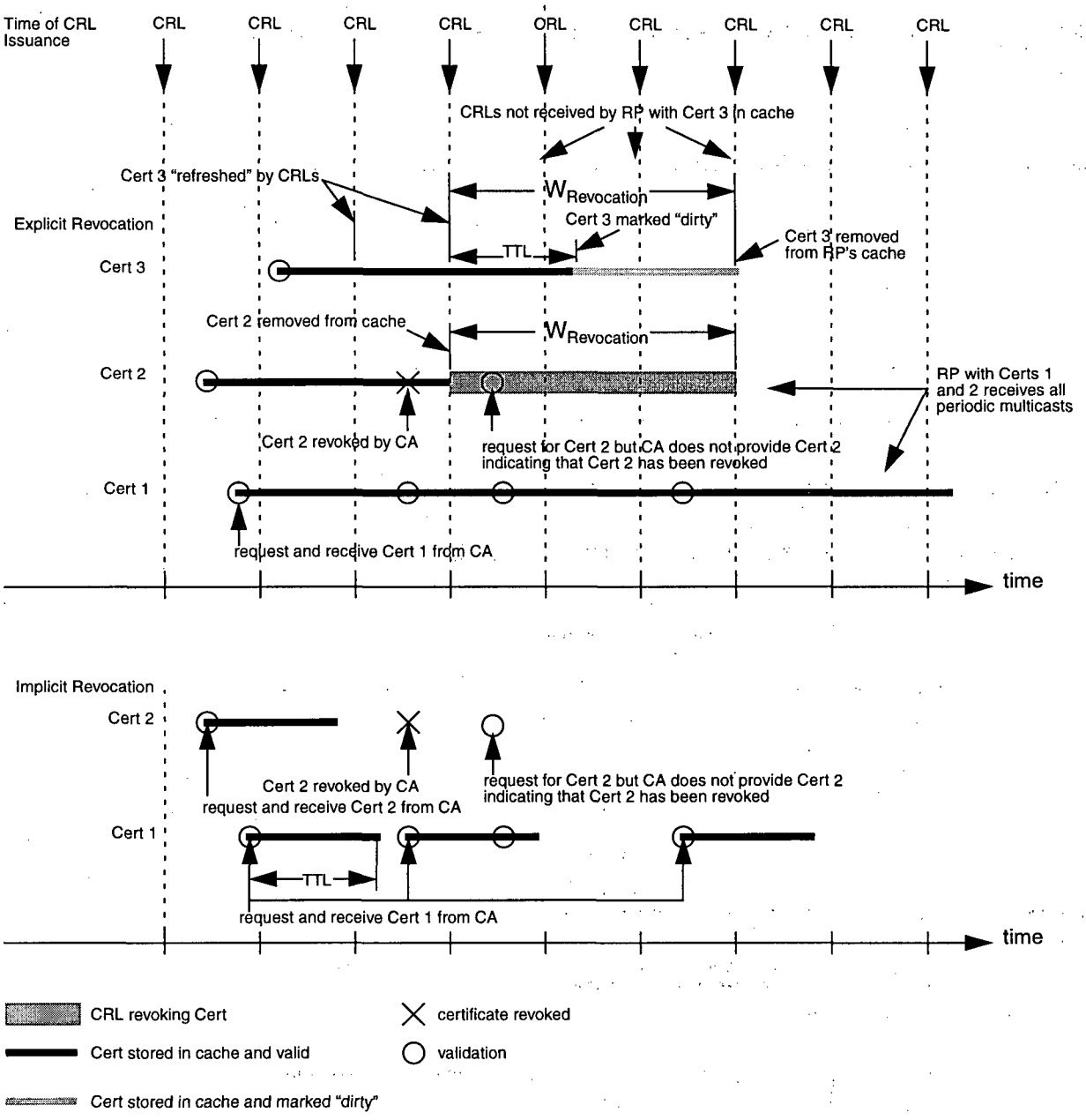

Figure 2.3 Windowed Revocation

current CRL. If the RP wishes to know the most up-to-date status of a certificate, it can request for the certificate from the CA as shown in Figure 2.3 with implicit revocation. Each RP maintains a cache of certificates it had received from the CA. The RP keeps a certificate in its cache for

Revocation Window, $W_{Revocation}$, from when the RP received the certificate or the last received CRL as shown in Figure 2.3. As shown in Figure 2.3, a certificate may be in a RP's cache but may not be valid. A cached certificate is marked "dirty" when its TTL expires. Certificates in a RP's cache can be "refreshed" with a CRL. If the current CRL does not list a certificate that is in the RP's cache, then the certificate is valid. If the current CRL does list a certificate that is in the RP's cache, then the certificate has been revoked and the RP must remove it from its cache. In Windowed Revocation, a revoked certificate need only be listed in the CRL for a duration of $W_{Revocation}$. $W_{Revocation}$ is much shorter than the lifetime of the certificate thus resulting in a much shorter CRL. To protect the PKI from CA spoofing attacks, each certificate request must be signed by the CA to guarantee freshness and authenticity [30]. To improve scalability of Windowed Revocation, the CRLs are periodically multicasted to the RPs in order to reduce the probability of requesting for previously cached certificates or requesting for a CRL. One drawback of Windowed Revocation is that the CA's private keys must be on-line to sign certificate requests. Other methods allow the certificate to be presented to the RP by anyone. Another drawback of Windowed Revocation is that the RP must go on-line whenever it needs to obtain certificates that are not cached. Unlike the other CRL systems, the RP may not work off-line until the next CRL issuance interval once the latest CRL has been obtained.

The work in this thesis is based on Sliding Window Delta CRLs. By leveraging the cost savings of using multicasting to provide RPs with Delta CRLs, significant communications cost savings can be achieved.

2.2 Internet Transport Protocols

The conventional way of providing certificate status information to RPs is by unicasting.

Unicasting is a point to point connection between two parties. Unicasting is suitable for most applications. However, in the case of CRLs, where all the RPs may potentially need the same information, unicasting leads to significant wastage in bandwidth as the same information will be carried over the same links multiple times. Distribution of Delta CRLs lends itself well to multicasting. Multicasting is a point to multi-point transport method which aims to reduce the number of redundant transfers over each link. In multicasting, ideally the same data will go over each link only once for any number of recipients (Figure 2.4). In unicasting to a number of recipients, the number of times a packet must pass through the link is equal to the number of recipients that are dependent on that link to connect to the server as shown by the numbers in parentheses in Figure 2.4.

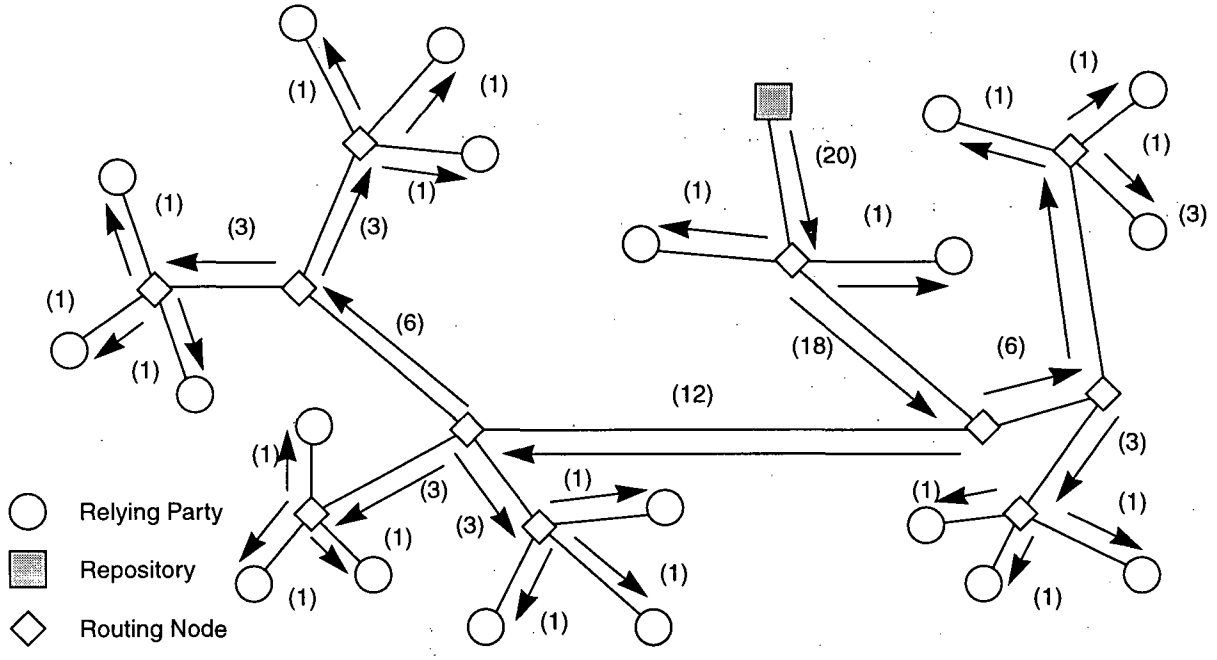

Figure 2.4 Spanning tree

In this study, the cost of employing a Hybrid Multicasting/Unicasting approach to CRL distribution is compared against the unicasting only approach. To accurately and fairly gauge the

improvements of the multicasting approach against the unicasting only approach to CRL distribution, the unit called the $packet \cdot link$ as suggested by [17] is applied. The $packet \cdot link$ refers to one packet carried over one link on the network. Analysis in this study assumes that each attempt to transfer one packet over any one network link incur one $packet \cdot link$ charge. In reality, some links may cost nothing and some links may be very expensive. A pricing model for multicasts based on either sparse or dense mode multicasting is proposed in [17]. In sparse mode multicasting, the number of multicast subscribers is small compared to the total number of end or leaf nodes in a large network such as the Internet. Therefore, for sparse mode multicasting, the cost of a multicast should be determined by the number of subscribers. On the other hand, in dense mode multicasting, the number of multicast subscribers is large and thus the likelihood of the packets traversing a high percentage of the total links in the network is high enough to saturate the entire network. In dense mode multicasting, [17] suggests that a fixed price be charged for each multicast. In this study, the sparse mode multicasting cost model is adopted for comparing costs. Dense mode multicast pricing may be easily adapted to the model.

2.2.1 Unicasting

The standard method of transporting information between a server and a client is unicasting. The server and the clients are represented by the Repository and the RPs. In unicasting CRLs, a bidirectional connection is created between the Repository and a single RP upon the request of a RP. The number of links and nodes between the Repository and the RPs varies and is determined by the network topology. When a RP makes a request for the current CRL from the Repository, the Repository responds by sending the CRL back. Each packet of the CRL file must traverse several links and routers to reach the destination RP which made the request. In a realistic

network, not all packets will be received by the requesting RP. Some packets will have bit errors and some will be lost due to congestion causing buffer overflows at the routers [20, 26]. Intermediate nodes such as routers perform a CRC check on the packets before sending the packets to the next node or RP. If the packet does not pass the CRC, then it will be rejected and not sent to the next node or RP. When the RP receives a CRL and finds that some packets are in error or missing, the RP by way of an Automatic Repeat Request (ARQ) mechanism will send a request to the Repository for the missing packets. The RP will continue to request for missing packets or packets that are received in error until all the packets of the CRL are received correctly. In this way, unicasting may be said to be reliable in the sense that when a RP makes a CRL request, it will eventually receive the CRL update to complete a pending validation request. Unicasts are priced according to the packet size of the CRL and the number of links each packet traverses to be received completely by the RP. This will include costs incurred as a result of erroneous packets as well.

2.2.2 Multicasting

The other method of network data transport is multicasting. Multicasting is a point to multi-point data transport method. The general idea of multicasting is to send the data over each link once in a spanning tree to reach all the recipients on a network. Ideally, the server will need to only send the data once, while the routers in the network do the work in routing copies of the data to the multicast subscribers. Multicasting is not useful for all applications. However, it is well suited for the multicasting of Delta CRLs because of the relatively small size of the Delta CRL file. The main problem with multicasting is that it cannot be considered a reliable means of data transfer in its basic form. Basic unicast style ARQ schemes may not be applied to multicasting to improve its reliability. Attempting to use unicast style ARQ may cause an implosion of requests

from recipients who miss packets or have packet errors. A variation of a multicast ARQ scheme in which a nearby receiver who has received the multicast data correctly would multicast the data to an affected subnet have been devised [18] but is not suitable for this application due to the fact that the RP would have to be more complex. Rather, a modified memory ARQ scheme is applied to the Hybrid Multicast/Unicast method to improve the success rate of multicasted Delta CRLs. Proposals in multicast reliability based on Forward Error Correction (FEC) [19-21] may also work for the distribution of Delta CRLs and may be the subject of future work in this area.

To analyze the cost savings of using a Hybrid Multicast/Unicast system of CRL distribution in comparison to a unicast only system, we need to consider the topology of the network. Previous works in CRL distribution methods which used unicast only methods of distribution need not consider the network topologies as the cost of unicasting is based on the capacity of the Repository's network connection. However, in multicasting, the network topology will affect the total cost of CRL distribution. The cost of a multicast depends on the number of RPs and the number intermediate nodes in the network.

For multicasting analysis in Chapter 3, a spanning tree that provides a path from the Repository to each RP is assumed to have already been found. The number of links in a spanning tree with N nodes is $N-1$ links. Thus the number of $packet \cdot link$ for a 100% successful multicast is $N-1$ for each packet leaving the Repository.

At the time of writing, multicasting is only available to networks that have multicasting routers in service. However it is foreseen that multicasting services will be widely available in the future [29]. The work-around is to implement IP encapsulation to carry multicast data over networks that are not multicast capable, e.g., as in MBone [29].

2.3 Network Topology

In the unicast model, only the number of links between the Repository and each individual RP needs to be considered. In a real network, the number of links between the Repository and each RP may vary. The distribution of link distances between the Repository and the RPs can be considered for analysis of CRL distribution costs. However, in the multicast model, intermediate links are shared with other RPs. Links closer to the Repository in the spanning tree have a larger number of dependent RPs. An error or failure on such a link will cause data reception failure to a larger number of RPs than a link that is farther away from the Repository (Figure 2.5).

The cost of a multicast depends on the total number of links in the spanning tree rather than the distribution of link distances between the Repository and each RP in the spanning tree as in the case of unicasting. The analysis of the cost of unicast only and the Hybrid Multicast/ Unicast CRL distribution requires the simulation of a large number of RPs. Simulating a large number of RPs that reside on a real world network topology is difficult as corporations do not make their intranet topologies readily available to the general public. Thus for simulation work, the Tiers random network topology model [22, 23] is used. A Tiers random network topology generator is available as free software from the Internet. The Tiers random network topology generator provides a means of generating large realistic random hierarchical network topologies based on user parameters. The Tiers model consists of three broad categories of nodes that are linked together using a set of rules that are based on observations of real world networks [22, 23]. The three categories of nodes are LAN, MAN and WAN nodes. These nodes are used to represent the bridges, routers, switches and gateways in a large network. Although Tiers also model redundant links, network link redundancies will not be considered in our model for reasons of complexity. To model a network with redundant links, a dynamically changing network needs to

be modeled. Network topologies used in this study are simple static spanning trees consisting of LAN, MAN and WAN nodes. Details for generating Tiers random network topologies are described in further detail in [22, 23].

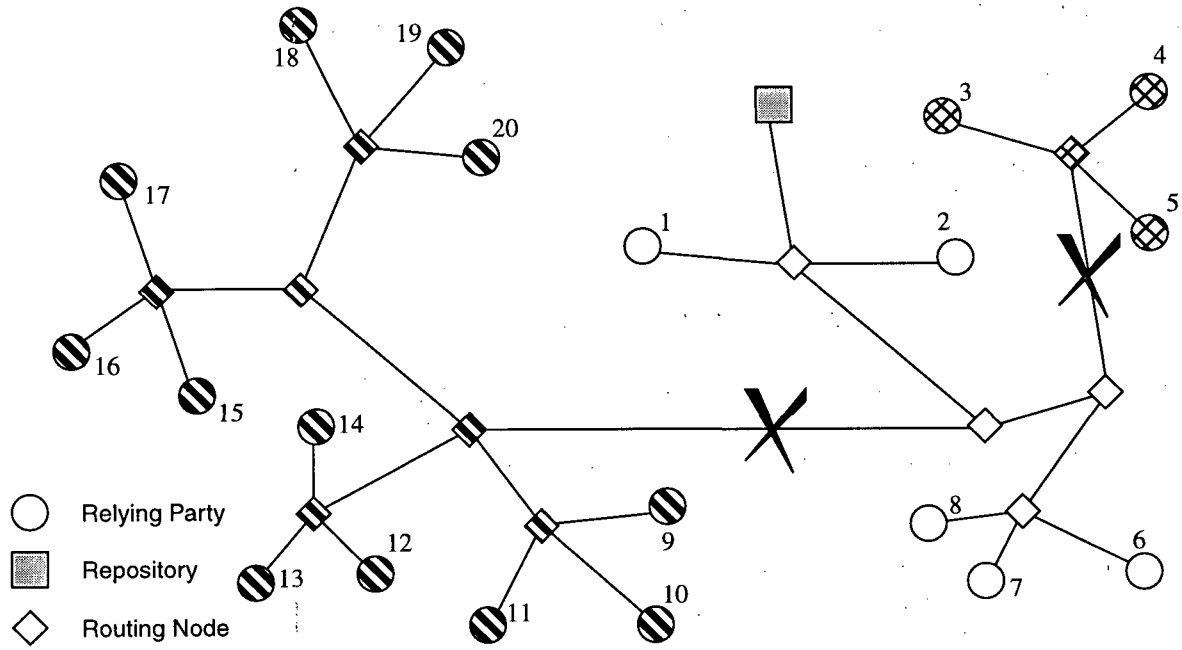

Figure 2.5 Link dependency

Chapter 3 Hybrid Multicast/Unicast CRL Distribution

In this chapter, a method is proposed with the aim to significantly reduce the data transmission cost of providing CRL updates to a large number of RPs. CRLs are first discussed. The proposed Hybrid Multicast/Unicast CRL distribution method is then described, followed by a detailed description of the Hybrid Multicast/Unicast CRL distribution model.

3.1 Certificate Revocation Lists

CRLs are akin to the "black list" of bad credit cards that credit card companies supply to stores that accept their cards. CRLs contain a list of certificates that have been revoked by the CA which issued them. In a large PKI, the number of unexpired certificates in circulation is expected to be in the range of 3 million per CA [1]. Thus, the expected size of the CRL is large. Figure 3.1 shows the basic fields contained in a X.509v2 CRL. The `TBSCertList` or "To Be Signed" list consists of the following fields [12]:

`version`: (optional) describes the version of the encoded CRL,

`signature`: the identifier for the algorithm used to sign the CRL,

`issuer`: identifies the CA who signed and issued the CRL,

`thisUpdate`: indicates the issuance time of this CRL,

`nextUpdate`: (optional) indicates the time of the next CRL issuance,

`revokedCertificates`: list of revoked certificates that are uniquely identified by their serial numbers, the time at which the certificate was revoked, and optional extensions,

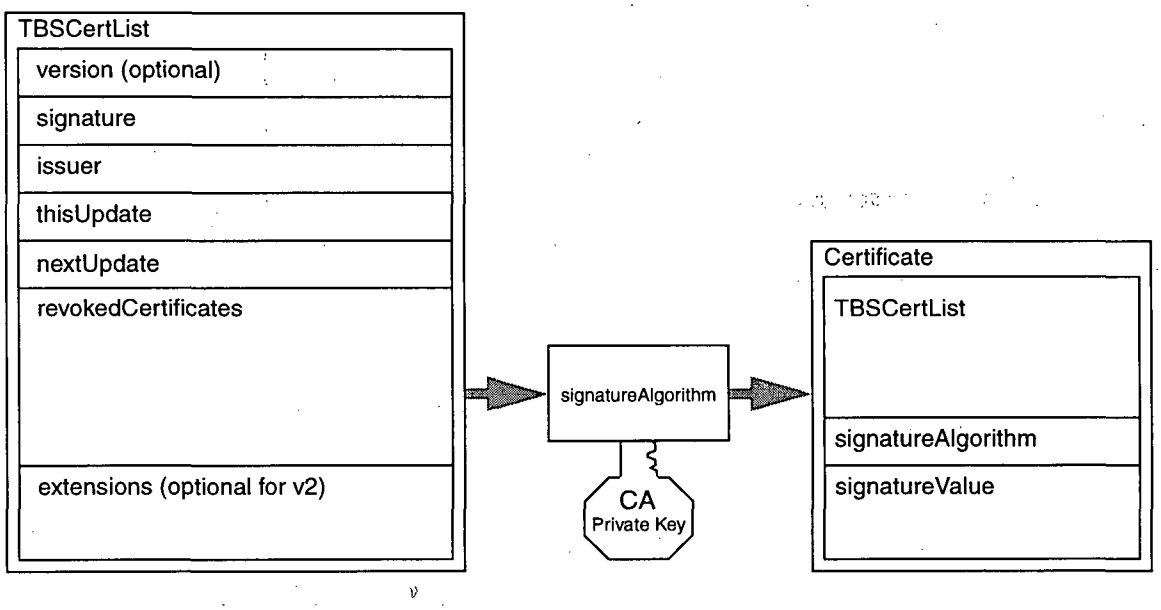

Figure 3.1 X.509v2 CRL

`extensions`: (optional for version 2): is a field containing a sequence of one or more CRL extensions.

The list of extensions available for X.509v2 CRLs [12] are:

`AuthorityKeyIdentifier`: identifies the public key corresponding to the private key used to sign a CRL. This extension may be used when the issuing CA has a number of public/private key pairs,

`IssuerAlternativeName`: allows additional identities to be associated with the CA which issued the CRL,

`cRLNumber`: sequence number assigned to each CRL issued by a CA. The sequence number of each CRL is one greater than that of the previous CRL,

`deltaCRLIndicator`: identifies the current `cRLNumber`. This extension is used to distinguish this CRL as a Delta CRL,

`BaseCRLNumber`: identifies the `cRLNumber` of the earliest CRL to which the Delta CRL may update.

The `TBSCertList` is then signed by the issuing CA using its private CA key to form the CRL with the following construct (Figure 3.1):

`signatureAlgorithm`: is the signature algorithm identifier that is exactly the same as that in the `TBSCertList` signature field,

`signatureValue`: is a bit string representing the digital signature formed by the issuing CA using the algorithm in the `signatureAlgorithm` field.

There are two types of CRLs: Base CRL (or Full CRL) and Delta CRLs. In the Sliding Window Delta CRL system [7], which is used in the proposed Hybrid Multicast/Unicast CRL distribution system, both a Base CRL and a Delta Base CRL are issued by the CA at the same time. Upon receiving a CRL update request, the Repository will send either a Base CRL or a Delta CRL. Note that there must be at least one valid CRL[1] at any point in time. The period in which a CRL is valid is from `thisUpdate` to `nextUpdate`. Therefore the time difference between `thisUpdate` and `nextUpdate` is the CRL issuance interval, T_{CRL}.

3.1.1 Base CRL

The Base CRL contains all of the serial numbers of unexpired certificates that have been

[1] Note that the terms CRL, Base CRL and Full CRL are interchangeable and all refer to the entire CRL.

revoked by the CA which originally issued the certificates. The size of the Base CRL depends on the number, N_{Cert}, of unexpired certificates that were issued by the CA, the lifetime, T_{Cert}, of a certificate, the expected fraction, P_{Revoke}, of revocations per certificate lifetime, the length, L_h, of the CRL header, the length, L_r, of each certificate revocation record and the number, L_{packet}, of bytes per data packet. The average length of the Base CRL is given by [6, 7]

$$L_{B\ \text{bytes}} = L_h + \frac{1}{2} L_r N_{Cert} P_{Revoke} \tag{3.1}$$

in terms of bytes and

$$L_B = \left\lceil \frac{L_h + \frac{1}{2} L_r N_{Cert} P_{Revoke}}{L_{packet}} \right\rceil \tag{3.2}$$

in terms of packets. The length, L_h, of the CRL header and length, L_r, of each certificate revocation record used in this study are 51 bytes and 9 bytes respectively as given in [1]. The certificate lifetime, T_{Cert}, is one year. The estimated percentage of certificate revocations is 10% [1]. The number of unexpired certificates issued is 3 million [1]. Applying (3.1), the size of the Base CRL is 1,350,051 Bytes. Using $L_{packet} = 1$ KBytes, the size of the Full CRL is 1351 packets.

3.1.2 Delta CRL

The Delta CRLs contain only revocations that have been added since the CRL indicated in the `BaseCRLNumber` field of the Delta CRL. The Delta CRLs are generally much smaller than the Base CRLs. The inherently small sized Delta CRLs are more suitable for multicasting than Base CRLs in an error prone network such as the Internet. The estimated size of a Delta CRL is

given by [6, 7]

$$L_{\Delta \text{ bytes}} = L_h + L_r N_{Cert} P_{Revoke} W_U \left(\frac{T_{CRL}}{T_{Cert}} \right) \tag{3.3}$$

in terms of bytes and

$$L_{\Delta} = \left\lceil \frac{L_h + L_r N_{Cert} P_{Revoke} W_U \left(\frac{T_{CRL}}{T_{Cert}} \right)}{L_{packet}} \right\rceil \tag{3.4}$$

in terms of packets. The Delta CRL update window, W_U, specifies the range of previous CRLs which the Delta CRL can update. Therefore, a large W_U size will yield a longer Delta CRL. A formula for finding the optimal window size for minimizing server bandwidth requirements is detailed in [7] for the unicast only system.

3.2 General Description of Hybrid Multicast/Unicast CRL Distribution

This section provides a general description of the proposed Hybrid Multicast/Unicast CRL distribution system for reducing the cost of CRL distribution to a large number of RPs. The system uses a combination of multicasting and unicasting. Multicasting is suitable for applications in which there are large numbers of receivers that require the same information. Multicasting cuts down on communications costs by removing the redundant data transfers over the same link. Ideally, each packet is sent over each link in the spanning tree once in multicasting.

In the unicast only system, a RP makes a request for a CRL update if and only if it needs to validate a certificate and its current CRL has expired; otherwise, the RP uses the copy of the CRL that it has cached. The RP has enough storage to cache a complete Base CRL with no losses,

i.e. a perfect cache. An ARQ system is used in which only the packets that are missing or received in error by the RP are resent by the Repository. The RP continues to re-request missing packets until it receives the CRL update completely. If a RP makes a request to the Repository for a CRL update and the Repository is unreachable due to one or more failed links between it and the Repository, the RP's request will time-out and the RP will postpone its request to a later time.

The Hybrid Multicast/Unicast Delta CRL distribution method adds a multicast receiving module to the RP which uses traditional unicasting only. The RP can continue to operate in the same manner as the unicast only RP except that it has an additional module which listens for and receives multicasts from a specified Repository. The multicast receiving module may be used with legacy programs which reside on the RP side that cache their copy of the CRL in a file that is accessible for reading and writing by other programs. When the Repository multicasts a Delta CRL to the RPs, each RP's multicast receiving module listens for the packets of the multicast Delta CRL. Since packets errors and link failures are possible, it is not guaranteed that all or any of the multicast Delta CRL packets will be received by the RPs. Each RP's multicast receiving module stores the error-free packets of the CRL that it has received until a new Delta CRL is multicast. If the Repository multicasts the same Delta CRL more than once, the error-free packets from each multicast repetition may be used by the RPs to reconstruct a complete copy of the Delta CRL so that no multicast is completely wasted. If the RP's multicast receiving module has received a complete Delta CRL, it will check to see if the Delta CRL can be used to update its local CRL. Whether or not a RP can use the Delta CRL to update its local CRL depends on how old its copy of the CRL is. If the Delta CRL's `BaseCRLNumber` refers to a CRL that is equal or earlier than what the RP has, then the Delta CRL can be used to update the RP's local CRL to the current one. In the Hybrid Multicast/Unicast CRL distribution method the Repository will always

multicast the new Delta CRL at the beginning of each issuance interval. The Repository may also multicast the same Delta CRL multiple times aperiodically and as determined to be needed. A means by which the Repository can determine whether a Delta CRL is needed is to monitor the rate of incoming CRL requests. Using the rate of incoming CRL requests, the number of RPs that have yet to make a first validation request within the current CRL issuance interval may be estimated. The estimate of how many RPs have yet to make a validation request will indicate whether a multicast would be costlier compared to the total cost of unicasting to the estimated number of remaining RPs that still need a CRL update. A break-even point based on the expected cost of a Delta CRL multicast and the estimated equivalent cost of a number of unicast CRL requests may be found to set the threshold CRL request rate for triggering aperiodic multicasts. The goal of the multicast(s) is to reduce the population of RPs that are expected to make a CRL request as a result of a first validation request in the CRL issuance interval. The number of RPs that successfully receive a complete copy of the Delta CRL from the multiple multicasts depends on the network topology, the packet error probability of the links and the availability statistics of the network links. Note that a RP receiving a complete Delta CRL via multicasts does not mean that it can use the Delta CRL. If the RP cannot use the multicast Delta CRL, it will need to make a request for a Base CRL when its next validation request arrives. Each network link may have a probability of a failure and recovery from failure. Should one of the primary links go down for a long period, there may be a large number of pent-up validation requests at the RPs that are dependent on the failed link. These links may not have successfully received prior multicast Delta CRLs due to packet errors, network link failures, or may not have been able to update its CRL using the multicast Delta CRL. If a primary link were to return from failure, there will be a peak in CRL update requests created by the pent-up validation requests at the RPs. Should this occur,

the rate of incoming requests may exceed the threshold limit that is set at the Repository to trigger an aperiodic Delta CRL multicast. The Repository continues to send Delta CRL multicasts until the incoming CRL request rate falls below the threshold limit. An analytical model for both the Hybrid Multicast/Unicast system and the unicast only system is discussed in the following section.

3.3 Detailed Description of Multicast/Unicast CRL Distribution

A model which can be used to analyze the Sliding Window Delta CRL which incorporates the unicast only system and the proposed Hybrid Multicast/Unicast CRL distribution system is derived in this section. The unicast only system is referred to as NMC. The Hybrid Multicast/Unicast system is divided into Hybrid Periodic Multicast/Unicast (MCP) and Hybrid Aperiodic Multicast/Unicast (MCA). The limitation of the analytical model is that it only accounts for packet errors at the links and does not account for network link availability as the model would becomes much more complex. However, network link availability is accounted for in the simulation model and program.

3.3.1 Measures of Cost for Determining the Efficiency of Multicasting CRL

In previous works [6-9], cost comparisons for various systems proposed for supplying certificate status information are based on the number and the rate of bits or packets leaving the Repository. This provided a cost estimate for an error-free and reliable network which does not need to consider the multicasting option. With the multicast option, the number of packets leaving the Repository does not give an accurate assessment of the true cost of the multicasts where the cost is also dependent on the network topology. In this study, cost comparisons are based on the $packet \cdot link$. A packet may need to go over several links to reach its destination. Note also that a

$packet \cdot link$ cost is incurred whenever a packet is sent over a link, regardless of whether the packet is received correctly by the receiving router, switch or RP at the other end of the link. In the case of the routers and switches, packets that are received in error are rejected rather then transmitted over the next link.

In determining the bandwidth of the lines required for the Repository, the peak bandwidth requirements needs to be estimated for the system. Note that ISPs generally charge more for higher bandwidth lines and so a system with low peak data rates are preferred to cut communications costs [6, 7, 27].

The following are the measures used in this thesis to compare system performance.

Total System Packets (T_{sP}): the total $packet \cdot link$ cost of supplying CRL updates to a network of RPs during one CRL issuance interval. This measure gives a more accurate assessment of the cost of operating the Hybrid Multicast/Unicast CRL distribution and the unicasting only CRL distribution method in a given network. In reality, links may have different link charges which depends on the type of medium and service providers. Analysis in this study assumes that all links incur an equal charge. The simulation model and analytical models may be modified to account for different link charges.

Total Server Packets (T_{servP}): the total number of packets served by the Repository during one CRL issuance interval.

Server Packet Rate (R_{servP}): the rate at which CRL packets leave the Repository. The Server Packet Rate gives information on the bandwidth of the network connection needed for the Reposi-

tory.

Total Server Requests (T_{servR}): the total number of requests received by the Repository during one CRL issuance interval. Note that each RP request for missing packets counts as a request.

Server Request Rate (R_{servR}): the rate of incoming CRL requests as measured by the Repository. The Server Request Rate gives information on the loading on the Repository. Note that each RP request for missing packets counts as a request.

Total Server Delta CRL (T_{servDC}): the total number of Delta CRLs served by the Repository during one CRL issuance interval.

Total Server Base CRL (T_{servBC}): counts the total number of Base CRLs served by the Repository during one CRL issuance interval.

3.3.2 Analytical Model for Performance Evaluation

The derivation of the analytical model for the distribution of CRLs begins with the rate at which validation attempts are expected to arrive at each RP. The interarrival times of incoming validation attempts at each RP are assumed to be independent of each other and modeled by an exponential probability density function (pdf) [6, 7]. The probability of a RP making at least one validation request after time t [6, 7]

$$P_{Val}(\lambda_{Val}, t) = 1 - e^{-\lambda_{Val}t} \tag{3.5}$$

where λ_{Val} is the rate of validation requests at a single RP.

The probability of a RP making at least one validation request during a CRL issuance interval, T_{CRL}, is given by

$$\begin{aligned} P_{Val}(\lambda_{Val}, T_{CRL}) &= 1 - e^{-\lambda_{Val} T_{CRL}} \\ P_{Val} &= P_{Val}(\lambda_{Val}, T_{CRL}) \ . \end{aligned} \tag{3.6}$$

The probability of a RP not making any validation request during a CRL issuance interval is given by

$$P_{NVal} = 1 - P_{Val} \tag{3.7}$$

The rate at which a RP make its first validation request in a CRL issuance interval is given by

$$\begin{aligned} R_{Val}(\lambda_{Val}, t) &= \frac{d}{dt} P_{Val}(\lambda_{Val}, t) \\ &= \frac{d}{dt}(1 - e^{-\lambda_{Val} t}) \\ &= \lambda_{Val} e^{-\lambda_{Val} t} \ . \end{aligned} \tag{3.8}$$

To reach its destination RP, a packet must traverse a number of links. The probability of a packet being successfully received by the requesting RP is given by

$$P_s(d) = P_c^d \tag{3.9}$$

where d is the number of links between the Repository and the requesting RP and P_c is the probability of a packet being received correctly over a single link. The probability of a packet not received successfully by the requesting RP is given by

$$P_f(d) = 1 - P_s(d). \tag{3.10}$$

The minimum cost in terms of $packet \cdot links$ for a single packet to be received successfully by the requesting RP is the number of links d between the Repository and the requesting RP. The average cost is given by

$$\begin{aligned} n_{tx}(d) &= \text{E[number of packet transmit attempts till success]} \times \text{E[cost of each attempt]} \\ &= \left(\sum_{n=1}^{\infty} nP_f^{n-1}(d)P_s(d) \right) \left(dP_c^d + P_e \sum_{n=1}^{d} nP_c^{n-1} \right) \\ &= \frac{1}{P_S(d)} \frac{P_f(d)}{P_e} \end{aligned} \tag{3.11}$$

where $P_e = 1 - P_c$.

The expected number of times that the Repository has to send a packet before an RP at distance d links from the repository receives it correctly is

$$\begin{aligned} n_{stx}(d) &= \text{E[number of packet transmit attempts till success]} \\ &= \sum_{n=1}^{\infty} nP_f^{n-1}(d)P_s(d) \\ &= \frac{1}{P_s(d)} . \end{aligned} \tag{3.12}$$

The expected number of requests that the Repository will receive from a requesting RP for missing packets for the same CRL is

$$
\begin{aligned}
n_{sr}(L, d) &= \text{E[number of requests for packets of CRL of length } L] \\
&= P_s^L(d) + 2\left(P_s^L(d)\sum_{j=1}^{L}\binom{L}{j}P_f^j(d)\right) + 3\left(P_s^L(d)\sum_{j=1}^{L}\binom{L}{j}P_f^j(d)\left(\sum_{m=1}^{j}\binom{j}{m}P_f^m(d)\right)\right) \\
&\quad + 4\left(P_s^L(d)\sum_{j=1}^{L}\binom{L}{j}P_f^j(d)\left(\sum_{m=1}^{j}\binom{j}{m}P_f^m(d)\left(\sum_{k=1}^{m}\binom{m}{k}P_f^k(d)\right)\right)\right) + \ldots \\
&= \lim_{n\to\infty}\left(n - \sum_{i=1}^{n-1}(1-P_f^i)^L\right)
\end{aligned}
\tag{3.13}
$$

where L is the length of the CRL in packets. (3.13) shows that $n_{sr}(L, d)$ is the sum of the probabilities of a RP making n requests to receive all the packets of a CRL of length L.

The expected cost of multicasting a single Delta CRL is given by

$$
\begin{aligned}
C_{MC}(L_\Delta) &= L_\Delta \sum_{d=1}^{D_{\text{Link}Max}} n_d P_c^{d-1} \\
C_{MC} &= C_{MC}(L_\Delta)
\end{aligned}
\tag{3.14}
$$

where L_Δ is the packet length of the Delta CRL in packets, n_d is the number of links at distance d from the Repository and $D_{\text{Link}Max}$ is the distance of the link that is furthest away from the Repository for a given network topology. (3.14) shows that the expected cost is the sum of the probabilities of transmitting the Delta CRL over a link that is d distance away from the Repository. Note that the expected $packet \cdot links$ cost is maximized when all RPs receive the Delta CRL correctly. This is so because packets that are received in error by intermediate nodes (i.e. the routing nodes) will be rejected and not transmitted over the next link.

The probability, $P_{MCc}(L_\Delta, d, r)$, of a RP being able to reconstruct a good copy of the

Delta CRL from r repetitions of the same multicast Delta CRL and the probability, $P_{MCe}(L_\Delta, d, r)$, of not being able to reconstruct a good copy are given by

$$\begin{aligned} P_{MCc}(L_\Delta, d, r) &= (1-(1-P_c^d)^r)^{L_\Delta} \\ P_{MCe}(L_\Delta, d, r) &= 1-P_{MCc}(L_\Delta, d, r) \ . \end{aligned} \tag{3.15}$$

The number of multicast Delta CRL repetitions, r, depends on the system. In the MCP and NMC systems, the number, r, of repetitions is fixed at 1 and 0 respectively. The number, r, of repetitions varies in the MCA system. Each CRL issuance interval is identified by the index i and the corresponding number of multicast repetitions in that interval is r_i. Then $\boldsymbol{r}$ is the sequence of variables r_i, $i = 1, 2, 3, 4, \ldots$. To test if r_i is an independent random variable, random sequences of r_i were generated using the pdf of $\boldsymbol{r}$ found through simulation and substituted into analytical model. Results indicate that r_i is a statistically independent discrete random variable. Figure 3.2[2] shows the pdf of r for various P_e values. The results in Figure 3.2 were obtained with parameter settings shown in Table 3.1. Figure 3.2 shows that the number of aperiodic multicasts sent out by the Repository increases with P_e.

In the following derivations, it is assumed that the network is always available and that requests are always received correctly by the Repository. Therefore, peak request rates only occur at the beginning of a CRL issuance interval. This allows for the assumption that all multicasts will occur at the beginning of the CRL issuance interval. The next simplification is that the unicast

---

[2] In Figure 3.2, the discrete pdf's of r is shown as continuous curves to better discern the pdf's of different P_e values.

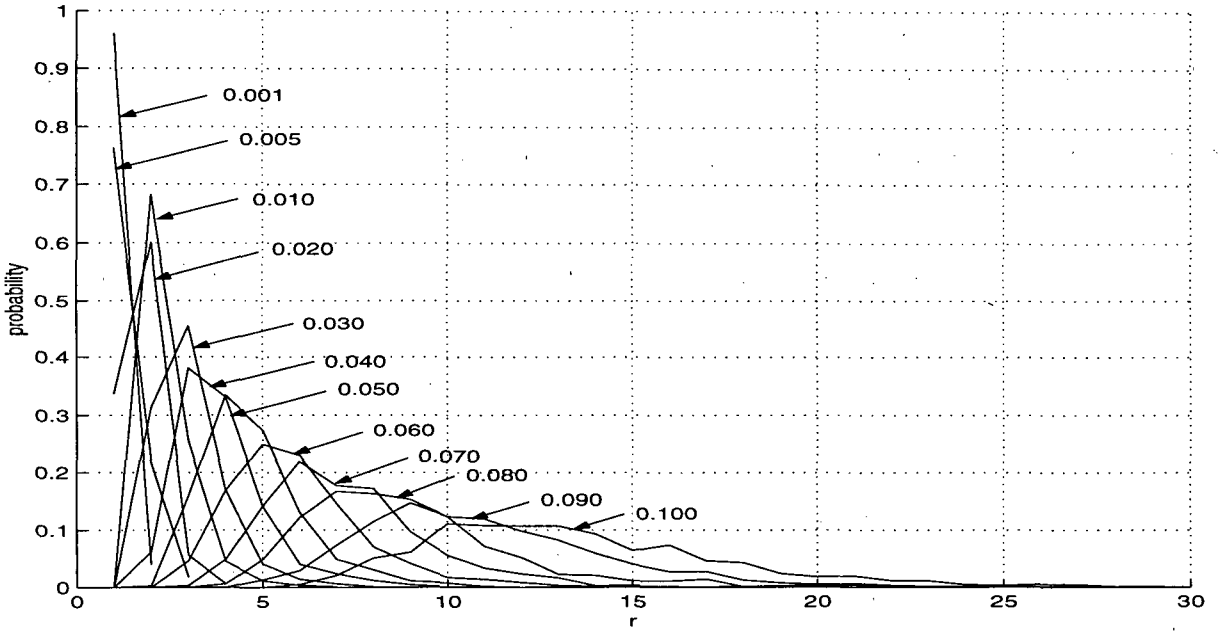

Figure 3.2 Pdfs of r with respect to P_e

Table 3.1 Pdf of r with respect to P_e parameters

Parameter	Value	Additional Notes
Network	#2	from Table 5.8
Packet error probability, P_e	0.001…0.1	various increments
Network Availability	100%	
CRL issuance interval, T_{CRL}	2 hours	
CRL update window size, W_U	3	
Delta CRL length, L_Δ	2 packets	calculated from (3.4)
Base CRL length, L_B	1351 packets	calculated from (3.2)
Request rate threshold, $R_{Threshold}/T_{RRW}$	30 requests/50sec	

cost incurred in measuring the request rate is small and negligible. A rough estimate of the cost due to the Repository unicasting CRLs during the time delay in measuring the request rate is on order of 3% for parameters used in this study.

The probability of a RP requesting a Base CRL during CRL issuance interval i is

$$P_B(L_\Delta, W_U, d, i, \boldsymbol{r}) = P_{Val} P_{NU}(L_\Delta, W_U, d, i, \boldsymbol{r}) \tag{3.16}$$

where $P_{NU}(L_\Delta, W_U, d, i, \boldsymbol{r})$ is the probability that the Delta CRL at CRL issuance interval i may not be used to update the RP's local CRL. $P_{NU}(L_\Delta, W_U, d, i, \boldsymbol{r})$ is given by the sum of:

(1) the probability of the RP not having any validation requests in the previous W_U CRL issuance intervals and none of multicast Delta CRLs were received correctly during those intervals,

$$P_{NVal}^{W_U}(P_{MCe}(L_\Delta, d, r_{i-1})P_{MCe}(L_\Delta, d, r_{i-2})\ldots P_{MCe}(L_\Delta, d, r_{i-W_U})) \tag{3.17}$$
$$= \prod_{j=i-W_U}^{i-1} P_{NVal} P_{MCe}(L_\Delta, d, r_j) \ ,$$

(2) the probability of not receiving a validation request during the previous CRL issuance interval and receiving a complete multicast Delta CRL for the previous interval which could not be used to update the CRL,

$$P_{NVal} P_{MCc}(L_\Delta, d, r_{i-1}) P_{NU}(L_\Delta, W_U, d, i-1, \boldsymbol{r}), \tag{3.18}$$

(3) the probability of not receiving any validation requests from CRL issuance interval $i-1$ to $j = \{i-2, i-3, \ldots, i-W_U\}$ and the multicast Delta CRLs from CRL issuance interval $i-1$ to $j+1$ were not correctly received and a complete multicast Delta CRL was received during CRL issuance interval j which could not be used to update the CRL,

$$P_{NVal}^2 P_{MCe}(L_\Delta, d, r_{i-1}) P_{MCc}(L_\Delta, d, r_{i-2}) P_{NU}(L_\Delta, W_U, d, i-2, \boldsymbol{r}) \tag{3.19}$$

$$+ P_{NVal}^3 P_{MCe}(L_\Delta, d, r_{i-1}) P_{MCe}(L_\Delta, d, r_{i-2}) P_{MCc}(L_\Delta, d, r_{i-3}) P_{NU}(L_\Delta, W_U, d, i-3, \boldsymbol{r})$$

$$+ \ldots$$

$$+ P_{NVal}^{W_U} \left(\prod_{k=i-W_U+1}^{i-1} P_{MCe}(L_\Delta, d, r_k) \right) P_{MCc}(L_\Delta, d, r_{i-W_U}) P_{NU}(L_\Delta, W_U, d, i-W_U, \boldsymbol{r})$$

$$= \sum_{j=i-W_U}^{i-2} P_{NVal} P_{MCc}(L_\Delta, d, r_j) P_{NU}(L_\Delta, W_U, d, j, \boldsymbol{r}) \prod_{k=j+1}^{i-1} P_{NVal} P_{MCe}(L_\Delta, d, r_k) \ .$$

Combining (3.17), (3.18) and (3.19) gives

$$P_{NU}(L_\Delta, W_U, d, i, \boldsymbol{r}) = \prod_{j=i-W_U}^{i-1} P_{NVal} P_{MCe}(L_\Delta, d, r_j) \tag{3.20}$$

$$+ P_{NVal} P_{MCc}(L_\Delta, d, r_{i-1}) P_{NU}(L_\Delta, W_U, d, i-1, \boldsymbol{r})$$

$$+ \sum_{j=i-W_U}^{i-2} P_{NVal} P_{MCc}(L_\Delta, d, r_j) P_{NU}(L_\Delta, W_U, d, j, \boldsymbol{r}) \prod_{k=j+1}^{i-1} P_{NVal} P_{MCe}(L_\Delta, d, r_k) \ .$$

If r is fixed, then $P_{NU}(L_\Delta, W_U, d, i, \boldsymbol{r})$ simplifies to

$$P_{NU}(L_\Delta, W_U, d, r) = \frac{(P_{NVal} P_{MCe}(L_\Delta, d, r))^{W_U}}{1 - P_{MCc}(L_\Delta, d, r) \sum_{j=1}^{W_U} P_{NVal}^j P_{MCe}^{j-1}(L_\Delta, d, r)} . \tag{3.21}$$

The probability of a RP requesting a Delta CRL is given by the probability that there is at least one validation request in the current CRL issuance interval, that the RP's CRL can be updated with the current Delta CRL and that the RP has failed in receiving a complete copy of the current multicasted Delta CRL after all multicast repetitions. The equation for the probability of

requesting for a Delta CRL at CRL issuance interval i is given by

$$P_{\Delta}(L_{\Delta}, W_U, d, i, \boldsymbol{r}) = P_{Val} P_{MCe}(L_{\Delta}, d, r_i)(1 - P_{NU}(L_{\Delta}, W_U, d, i, \boldsymbol{r})) . \quad (3.22)$$

The expected number of Base CRLs sent out by the Repository during one CRL issuance interval is given by

$$T_{servBC}(L_{\Delta}, W_U, d, i, \boldsymbol{r}) = \sum_{d=1}^{D_{\mathrm{RP}Max}} n_{rpd} P_B(L_{\Delta}, W_U, d, i, \boldsymbol{r}) \quad (3.23)$$

where n_{rpd} is the number of RPs that are d distance away from the Repository, and $D_{\mathrm{RP}Max}$ is the distance of the RPs that are furthest away from the Repository. The rate, $R_{servBC}(L_{\Delta}, W_U, d, i, t, \boldsymbol{r})$, at which Base CRLs are sent by the Repository after the start of a new CRL issuance interval is found by differentiating $P_{Val}(\lambda_{Val}, t)$ with respect to t in the expression for $P_B(L_{\Delta}, W_U, d, i, \boldsymbol{r})$. The expression for the rate of Base CRL requests is

$$
\begin{aligned}
R_{servBC}(L_\Delta, W_U, d, i, t, \boldsymbol{r}) &= \frac{d}{dt}(T_{servBC}(L_\Delta, W_U, d, i, \boldsymbol{r})) \\
&= \frac{d}{dt}\left(\sum_{d=1}^{D_{\mathrm{RPMax}}} n_{rpd} P_B(L_\Delta, W_U, d, i, \boldsymbol{r})\right) \\
&= \frac{d}{dt}\left(\sum_{d=1}^{D_{\mathrm{RPMax}}} n_{rpd} P_{NU}(L_\Delta, W_U, d, i, \boldsymbol{r}) P_{Val}(\lambda_{Val}, t)\right) \\
&= \sum_{d=1}^{D_{\mathrm{RPMax}}} n_{rpd} P_{NU}(L_\Delta, W_U, d, i, \boldsymbol{r}) \frac{d}{dt}(P_{Val}(\lambda_{Val}, t)) \\
&= \sum_{d=1}^{D_{\mathrm{RPMax}}} n_{rpd} P_{NU}(L_\Delta, W_U, d, i, r) R_{Val}(\lambda_{Val}, t) \\
&= \sum_{d=1}^{D_{\mathrm{RPMax}}} n_{rpd} P_{NU}(L_\Delta, W_U, d, i, \boldsymbol{r}) \lambda_{Val} e^{-\lambda_{Val} t} \ .
\end{aligned}
\tag{3.24}
$$

The expected number, $T_{servDC}(L_\Delta, W_U, d, i, \boldsymbol{r})$, of Delta CRLs sent out by the Repository during one CRL issuance interval is given by

$$
T_{servDC}(L_\Delta, W_U, d, i, \boldsymbol{r}) = \sum_{d=1}^{D_{\mathrm{RPMax}}} n_{rpd} P_\Delta(L_\Delta, W_U, d, i, \boldsymbol{r}) . \tag{3.25}
$$

The rate, $R_{servDC}(L_\Delta, W_U, d, i, t, \boldsymbol{r})$, at which Delta CRLs are sent by the Repository after the start of a new CRL issuance interval is found in a similar way to the Base CRL request rate, (3.24), and is given by

$$
\begin{aligned}
&R_{servDC}(L_\Delta, W_U, d, i, t, \boldsymbol{r}) \\
&= \sum_{d=1}^{D_{\mathrm{RPMax}}} n_{rpd} P_{MCe}(L_\Delta, d, r_i)(1 - P_{NU}(L_\Delta, W_U, d, i, \boldsymbol{r})) \lambda_{Val} e^{-\lambda_{Val} t} \ .
\end{aligned}
\tag{3.26}
$$

The expected number, $T_{servR}(L_\Delta, W_U, d, i, \boldsymbol{r})$, of CRL requests received by the Repository during one CRL issuance interval is given by

$$T_{servR}(L_\Delta, W_U, d, i, \boldsymbol{r}) = \sum_{d=1}^{D_{\mathrm{RPMax}}} n_{rpd}(P_\Delta(L_\Delta, W_U, d, i, \boldsymbol{r}) n_{sr}(L_\Delta, d) + P_B(L_\Delta, W_U, d, i, \boldsymbol{r})) n_{sr}(L_B, d) \quad . \tag{3.27}$$

The rate, $R_{servR}(L_\Delta, W_U, d, i, t, \boldsymbol{r})$, of incoming CRL requests after the start of a new CRL issuance interval is found in a similar way to the Base CRL, (3.24), request rate and is given by

$$R_{servR}(L_\Delta, W_U, d, i, t, \boldsymbol{r}) = \sum_{d=1}^{D_{\mathrm{RPMax}}} n_{rpd}(P_{MCe}(L_\Delta, d, r_i)(1 - P_{NU}(L_\Delta, W_U, d, i, \boldsymbol{r})) n_{sr}(L_\Delta, d) + P_{NU}(L_\Delta, W_U, d, i, \boldsymbol{r}) n_{sr}(L_B, d)) \lambda_{Val} e^{-\lambda_{Val} t} \quad . \tag{3.28}$$

The expected number, $T_{servP}(L_\Delta, W_U, d, i, \boldsymbol{r})$, of packets sent out by the Repository during one CRL issuance interval is given by

$$T_{servP}(L_\Delta, W_U, d, i, \boldsymbol{r}) = r_i L_\Delta + \sum_{d=1}^{D_{\mathrm{RPMax}}} n_{rpd}(P_B(L_\Delta, W_U, d, i, \boldsymbol{r}) L_B + P_\Delta(L_\Delta, W_U, d, i, \boldsymbol{r}) L_\Delta)) n_{stx}(d) \quad . \tag{3.29}$$

The rate, $R_{servSP}(L_\Delta, W_U, d, i, t, \boldsymbol{r})$, of packets leaving the Repository after the start of a new

CRL issuance interval is found in a similar way to the Base CRL, (3.24), request rate and is given by

$$R_{servSP}(L_\Delta, W_U, d, i, t, \boldsymbol{r}) = \sum_{d=1}^{D_{\mathrm{RP}Max}} n_{rpd}(P_{MCe}(L_\Delta, d, r_i)(1 - P_{NU}(L_\Delta, W_U, d, i, \boldsymbol{r})) \quad (3.30)$$

$$+ P_{NU}(L_\Delta, W_U, d, i, \boldsymbol{r}))n_{tx}(d)\lambda_{Val}e^{-\lambda_{Val}t} \ .$$

The expected total cost, $T_{sP}(L_\Delta, W_U, d, i, \boldsymbol{r})$, of supplying CRL updates to RPs during one CRL issuance interval is given by

$$\begin{aligned} & T_{sP}(L_\Delta, W_U, d, i, \boldsymbol{r}) \\ &= r_i C_{MC}(L_\Delta) + C_{UC}(L_\Delta, W_U, d, i, \boldsymbol{r}) \\ &= r_i L_\Delta \sum_{d=1}^{D_{\mathrm{Link}Max}} n_d P_c^{d-1} + \sum_{d=1}^{D_{\mathrm{RP}Max}} n_{rpd}(P_B(L_\Delta, W_U, d, i, \boldsymbol{r})L_B + P_\Delta(L_\Delta, W_U, d, i, \boldsymbol{r})L_\Delta)n_{tx}(d) \ . \end{aligned} \quad (3.31)$$

3.3.3 CRL Update Window W_U

A Delta CRL with CRL update window size W_U can be used to update a CRL that was issued W_U periods prior to the current CRL. Larger window sizes will reduce the probability of an RP requesting a Base CRL at the expense of a longer Delta CRL. As Delta CRLs are requested more often than Base CRLs, a small increase in the length of the Delta CRL will cause a significant increase in T_{servP} and T_{sP}. Thus an optimal value for the CRL update window is needed to minimize costs.

For the NMC system, the optimal CRL update window size, W_U, is found by minimizing

the bandwidth requirements of the Repository's local link with respect to W_U is given by [7]

$$\begin{aligned} BW &= L_B R_{servBC}(t=0) + L_\Delta R_{servDC}(t=0) \\ \frac{d}{dW_U} BW &= 0 \ . \end{aligned} \tag{3.32}$$

The resulting equation for the optimal size for the NMC system is given by [7]

$$W_U = \left(\frac{1}{R_{Val}}\right) \ln\left(\frac{(L_h + (L_r N_{Cert} P_{revoke})/2) R_{Val}}{(L_r N_{Cert} P_{revoke})/T_{Cert}}\right). \tag{3.33}$$

Expressions for W_U for the MCP and the MCA systems are difficult to obtain. The optimal values for W_U in this study are found through simulation. The optimal values for W_U for the NMC system found through simulation matches those values found using (3.33).

3.3.4 The Request Rate Threshold $R_{Theshold}/T_{RRW}$

The request rate threshold, $R_{Theshold}/T_{RRW}$, only applies to the MCA system. To determine the need for a multicast, the Repository monitors the rate of new CRL update requests by counting the number of recent requests, R_{Recent}, in the last T_{RRW}. If R_{Recent} meets or exceeds the requests threshold, $R_{Theshold}$, the Repository will multicast a Delta CRL in an attempt to reduce the population of RPs that are expected to need a CRL update in the current CRL issuance interval. The R_{Recent}/T_{RRW} rate represents a sliding time average request rate with an averaging time window of T_{RRW}. The R_{Recent}/T_{RRW} rate is used by the Repository as a feedback to estimate the number of RPs that have not yet updated their CRLs. Since the pdf for the CRL update request rate is exponential as shown in Figure 3.3, using the calculation

$$n_{rp}(t) = \int_{0}^{(T_{CRL}-t)} \left(\frac{R_{recent}}{T_{RRW}}\right) e^{-\lambda_{Val}t} = \left(\frac{R_{recent}}{T_{RRW}}\right)\left(\frac{1}{\lambda_{Val}}\right)\left(1 - e^{-\lambda_{Val}(T_{CRL}-t)}\right) \tag{3.34}$$

gives the number, $n_{rp}(t)$, of RPs that are expected to make a CRL request given the rate, R_{Recent}/T_{RRW}, of incoming CRL requests and the remaining time in the CRL issuance interval $T_{CRL}-t$.

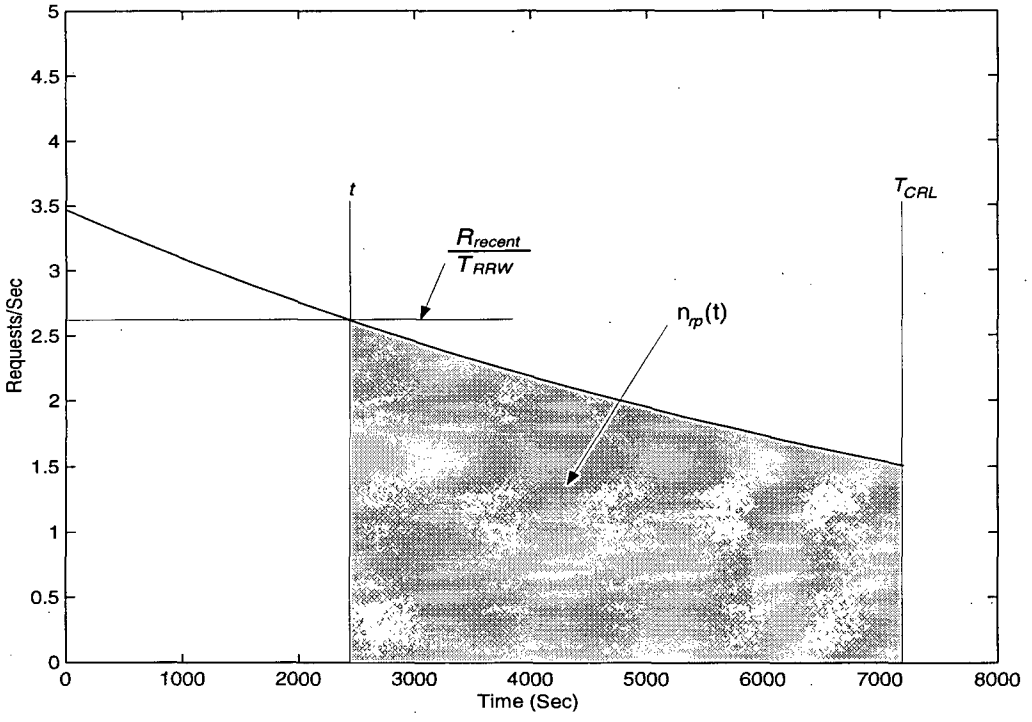

Figure 3.3 Estimating the number of RPs yet to make a CRL request

In a network which does not have any network link failures, the request rate will be at its peak at the beginning of each CRL issuance interval. The Repository continues to multicast at the beginning of the CRL issuance interval until the request rate is at or below $R_{Theshold}/T_{RRW}$.

Assuming that the number of unicast CRL requests used by Repository to measure the request rate, R_{Recent}/T_{RRW}, is negligible (Section 3.3.2), the formula for the expected number of RPs that have yet to make a CRL request in the remainder of the CRL issuance interval becomes

$$\begin{aligned} n_{rp} &= \left(\frac{R_{recent}}{T_{RRW}}\right)\left(\frac{1}{\lambda_{Val}}\right)(1 - e^{-\lambda_{Val}T_{CRL}}) \\ &= \frac{R_{recent}}{T_{RRW}}\frac{P_{Val}}{\lambda_{Val}} \end{aligned} \quad . \tag{3.35}$$

Using the expected number, n_{rp}, of RPs that have yet to make a CRL request, the Repository can make a decision based on the expected $packet \cdot link$ cost of multicasting and the expected $packet \cdot link$ cost of unicasting CRL updates to those RPs that have yet to make a CRL update request. The break-even point is given by

$$\begin{aligned} C_{MC} &= (n_{rp\ \text{break even}})c_{UC}(r) \\ &= (n_{rp\ \text{break even}})\left(\frac{C_{UC}(r)}{\sum_{d=1}^{D_{\text{RP}Max}} n_{rpd}(P_{\Delta}(L_{\Delta}, W_U, d, r) + P_B(L_{\Delta}, W_U, d, r))}\right) \end{aligned} \tag{3.36}$$

where $n_{rp\ \text{break even}}$ is the number of RPs needed to make a unicast CRL request to match the cost of multicasting a Delta CRL, $c_{UC}(r)$ is the expected cost of a single unicast CRL request after r multicast repetitions, $C_{UC}(r)$ is the expected cost of unicasting to RPs that are expected to make a CRL request after r multicast repetitions during a CRL issuance interval and the expression in the denominator is the number of RPs that are expected to make a CRL request after r multicast repetitions during a CRL issuance interval. Note that the average distance of a RP that may

require a CRL update will increase as the number of multicasts during that interval increases. This is due to that fact that RPs that are closer to the Repository have a higher probability of receiving the multicast completely.

By setting $R_{Recent} = R_{Threshold}$ and $n_{rp} = n_{rp\ \text{break even}}$, a "break-even" point is found where the expected cost of doing a multicast is the same as the cost of unicasting to all the remaining RPs that are expected to make a CRL request during the remainder of the CRL issuance interval, i.e.

$$\frac{R_{Threshold}}{T_{RRW}}\frac{P_{Val}}{\lambda_{Val}} = \frac{C_{MC}}{c_{UC}(r)}$$

$$\frac{R_{Threshold}}{T_{RRW}} = \frac{\sum_{d=1}^{D_{RPMax}} n_{rpd}(P_{\Delta}(L_{\Delta}, W_U, d, r) + P_B(L_{\Delta}, W_U, d, r))}{\left(\frac{C_{UC}(r)}{C_{MC}}\right)\left(\frac{P_{Val}}{\lambda_{Val}}\right)} \quad (3.37)$$

Finding an analytic expression for the optimal values for $R_{Threshold}$ and T_{RRW} which minimize CRL distribution costs is an open problem. In this study, optimal values for $R_{Threshold}$ and T_{RRW} are found through simulation. Equation 3.37 indicates that the total communication costs can be minimized by appropriate choice of the rate $R_{Theshold}/T_{RRW}$. However this derivation is based on instantaneous and accurate knowledge of the rate. The simulation results in Section 5.4 shows that this is not always valid.

Chapter 4 Description of Simulation Model

A computer simulation[1] model was used to validate the mathematical models developed in Chapter 3. Computer simulation was used because high cost and time do not permit us to demonstrate and compare the performance of the Hybrid Multicast/Unicast system of CRL distribution in a real network. The simulation model was implemented in C++ (Gnu G++) with pre and post processing programs written in Perl [25] and Matlab scripts. Object Oriented C++ is ideally suited for developing this simulation model. C++ allows the different objects and parties (`ServerNode`, `RouteNode`, `Edge` and `LanNode`) to be packaged neatly into classes, making development of the simulation model easy to structure and organize. The simulations were run on Intel P3-450MHz machines each with 384MB running Linux 2.0.x.

The next sections provide an overview of the simulation model and a detailed description of the simulation program, its program objects, input file parameters and output files.

4.1 Overview of Simulation Model

The simulation program is an event driven program that simultaneously models the entire spanning tree network. The triggering events are RP validation requests and periodic Repository multicasts. These events are stored on an event queue. At any one time there are $N_{RP} + N_S$ events in the event queue. Each RP and Repository may have only one event in the queue at a time. When the RP or Repository's event has been serviced, the program will generate the time for the RP's or Repository's next event and sort it chronologically onto the time queue (Figure 4.1).

The spanning tree network is composed of several classes of objects linked together. These

[1] The source code may be downloaded from: `http://www.ece.ubc.ca/~hansenw/mcrlsim/`

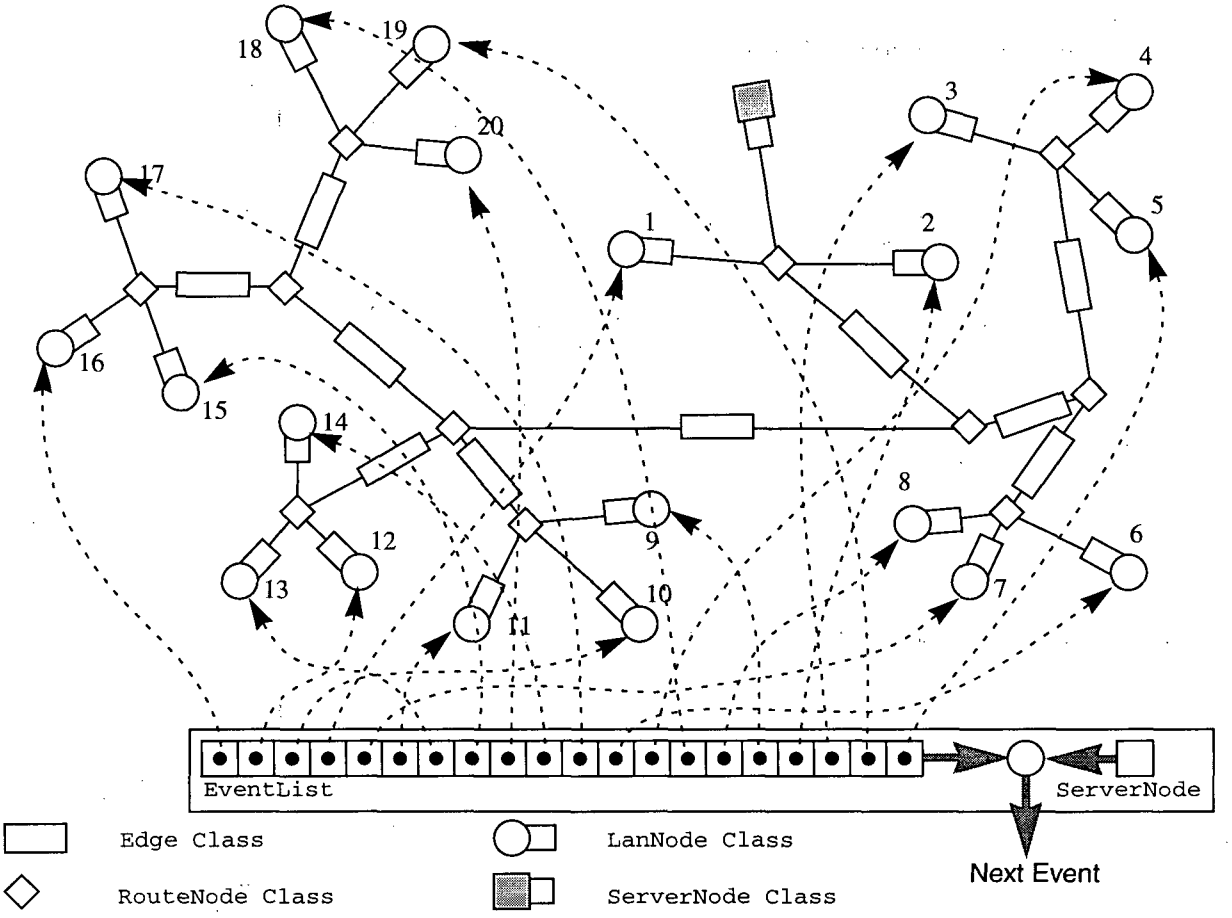

Figure 4.1 Network objects

classes are: `ServerNode`, `Edge`, `RouteNode` and `LanNode`. The `ServerNode` class object models the Repository. The `Edge` class object models the network links. The `RouteNode` class object models the switches and routers of the network. The `LanNode` class object models the RP. Each of the instances of these objects are independent. However they can share the same personalities. The network topology is generated using the Tiers [22] program. The output from the Tiers program is used as one of the input files to the simulation program.

To simulate independent network link failures, each of the `ServerNode`, `Edge` and `LanNode` class objects have a list of link status change times. These lists are generated at the

start of the simulation program during the setup phase. These link failure lists are reproducible with same the seed and network so that the different systems may be compared fairly with the exact same network conditions. Network link availability is modeled as a 2-state Markov chain (Figure 4.2). The state transition rates from on-line to off-line and off-line to on-line are λ_F and

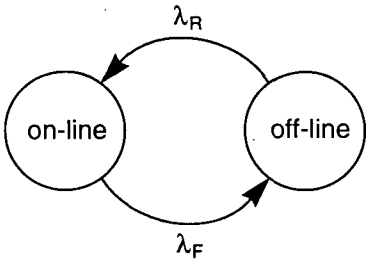

Figure 4.2 2 state Markov chain for link status

λ_R respectively. These values are based on ISP Service Level Agreements (SLA) numbers for guaranteed network availability. Values given by ISP SLA are in Table 4.1.

Table 4.1 ISP Service Level Agreement statements

ISP	Network Availability
"AT&T"	99.9% uptime with no more than 10 min downtime for each occurrence
"MCI"	99% uptime with no more than 1 hour downtime for each occurrence

The values of λ_F and λ_R are calculated using (4.1) and (4.2),

$$P_{up} = \frac{\lambda_R}{\lambda_R + \lambda_F} \tag{4.1}$$

$$F = 1 - e^{\lambda_F T_{down}} \tag{4.2}$$

where P_{up} is the probability of the link being on-line listed in the ISP SLA, T_{down} is the duration of the link downtime listed in the ISP SLA, F is a value between 0 and 1 representing the probability of a link downtime being less than T_{down}. In this study, the value was arbitrarily chosen to be 0.95 so that 95% of the downtimes will be within the ISP's SLA claim.

There are two input files to the simulation program. One of the files is the Tiers1.2 network topology file, the other is the simulation parameters file. The simulation parameters file contains the name of the Tiers1.2 network topology file, the location of the Repository, the names of the output files and defines the characteristics of the RPs, Edges, Repository, CRL attributes, simulation seed and simulation duration. The simulation parameters file will be detailed later in this chapter.

There are 3 output files from the simulation program: the `RouteFile`, `RawFile` and `SimOut` file. The `RouteFile` contains the neighbors listing for each node in the network. The `RawFile` contains a verbose accounting of all that happens during the simulation run and is used for debugging. The `SimOut` file contains a list of the input parameters, the network topology information including Matlab formatted equations that represent the network topology simulated, and the contents of the arrays storing accounts of the packet transfers and a summary of the packet transfers over network links and Repository CRL transfers, for the simulation run.

Each of the `ServerNode`, `Edge` and `LanNode` class objects include a link to a personality. Each object can theoretically have a different personality. These personalities consist of object parameters, to be detailed later in this chapter, such as RP validation rate, link availability,

link packet error probability and Repository multicasting period. Different personalities for links maybe assigned to specific links to represent different link types such as satellite links, wireless local-loops, ATM etc. Different personalities for RPs may also be assigned to accommodate different RP types such as RPs that are always on-line or RPs that shut off their computers at each closing, connect as needed RPs, etc. In this study, all links are assumed to have the same personality and all RPs have the same personality. Although the simulation model may accommodate more than one Repository, the current simulation program can only accommodate one Repository.

One limitation of the simulation program is the inability to account for delays such as queueing delays at the routers, processing delays at the Repository and transfer delays at the links. Another limitation is the inability to provide sliding window averaging peak request and packet transfer rates as mentioned previously. The simulation program does however provide the user with information about the expected bandwidth requirement at the Repository's local link, the $packet \cdot link$ cost of multicasting and unicasting, the expected loading on the server equipment and the estimated performance for variations in server parameters for supplying Base and Delta CRL updates in a given network topology.

4.2 Detailed Description of the Simulation Program

The simulation program consists of four main classes to represent the four different network objects: `ServerNode`, `Edge`, `RouteNode` and `LanNode`. Other important classes are the `Traffic` class and the `EventList` class. The `RouteNode` class is used to carry the structure of the network topology. The other three main network object classes, `ServerNode`, `Edge`, and `LanNode` simulate the system by passing `CRL_Dat` and `CRL_Request` messages.

4.2.1 Data Messages

There are two message structures in the simulation program that are passed between ServerNode, Edge, and LanNode class objects. They are CRL_Request and CRL_Dat.

4.2.1.1 CRL_Request Message

The CRL_Request message is a unidirectional message that originates from a LanNode class instance. The destination of the CRL_Request message is a ServerNode class object. The LanNode class object creates this message when it needs to update its local CRL. The content of the CRL_Request message is contained in Table 4.2.

Table 4.2 CRL_Request message

Field	Description
time	the time of the request
CRLnum	the serial number of the CRL currently held by the requesting RP

The CRLnum field contains the serial number of the requesting LanNode class object's current CRL. With knowledge of the requesting LanNode class object's current local CRL number, the ServerNode class object can determine whether the requesting LanNode class object may use the current Delta CRL or needs a Base CRL. The time field contains the time of the request. The CRL_Request message is assumed to be error-free and of negligible size. This message when sent by the LanNode class object will reach the destination ServerNode class object if and only if all the links between the requesting LanNode class object and the ServerNode class object are available at the time of the request.

4.2.1.2 CRL_Dat Message

The CRL_Dat message is a bidirectional message that represents the CRL file or a re-request for missing packets of a particular CRL file. The CRL_Dat message may originate from

the ServerNode class object responding to a CRL_Request message or multicasting a Delta CRL or the LanNode class object re-requesting missing packets of CRL file that was initiated with a CRL_Request message. The fields of the CRL_Dat message are shown in Table 4.3. The meanings of these fields are different depending on whether the message originates from a ServerNode class object or LanNode class object.

Table 4.3 CRL_Dat message

Field	Description
time	issuance time of the CRL by the CA
time_expire	expiration time of the CRL
CRLnum	serial number of the CRL
baseCRLnum	the earliest CRL which the Delta CRL may be applied to if CRL is a Delta CRL; set to be the same as the CRLnum if CRL is a Base CRL
packets	number of packets in the CRL
packets_received	number of packets that have been received correctly
packets_file	bit string representing the packets of the Delta CRL (1: correct, 0: error)

When the CRL_Dat message originates from a ServerNode class object, the time field contains the time that the CRL file was issued by the CA. The time_expire field contains the expiration time of the CRL. The CRLnum field contains the serial number of the CRL file which increases monotonically with each subsequent issuance. If the ServerNode class object sends a Delta CRL to the requesting LanNode class object, the baseCRLnum field will contain the CRLnum of the earliest CRL that the Delta CRL may update. If the ServerNode class object sends a Base CRL, the baseCRLnum field will contain the same value as the CRLnum field indicating to the LanNode class object that the CRL_Dat message contains a Base CRL. The packets field contains the size of the CRL file in terms of packets. The packets_received field contains the number of packets received by each link correctly. This

value is only allowed to stay the same or decrease as the CRL_Dat message is passed through each link towards the LanNode class object. The final field is the packets_file field used to represent each packet of the Delta CRL only. The size of the packets_file field is 32 bits to represent up to 32 packets. Each bit is used to represent the receive status of a particular packet in the Delta CRL file. The size limitation applies to this version of the simulation program and not the simulation model.

If the CRL_Dat message originates from a LanNode class object, the time field will contain the time of the re-request. The time_expire is not used here. The CRLnum contains the CRL serial number of the CRL which the LanNode needs. The baseCRLnum is the same as that from the ServerNode class object originated CRL_Dat message to the requesting RP. The packets value contains the number of packets still needed, missing or received incorrectly by the requesting RP. The packets_received and the packets_file fields are not used.

4.2.2 EventList Class

The EventList class manages the sequence of events for the simulation. The list of events in the EventList class are stored in a linked list. In the simulation model and Program, events may only originate from either ServerNode class objects or LanNode class objects. The source of the next event is provided by a pointer to either a LanNode class object or a ServerNode class object. At all times, there will be one event on the EventList class object for each ServerNode and LanNode class object on the spanning tree network. In other words, the length of the EventList class object linked list is the sum of the number of ServerNode and LanNode class objects. After an object's event has been processed, the object generates its next event. The object's next event time is then sorted chronologically into the EventList.

4.2.3 `Traffic` Class

The `Traffic` class object handles the recording of the packet transfers, incoming requests and CRLs sent by the `ServerNode` class object. These events are recorded for N_{slots} time slots each of duration T_{slot}. The product of the number of timeslots and the timeslot duration T_{slot} is equal to the duration of the simulation run T_{simrun}, i.e.

$$T_{simrun} = N_{slot} T_{slot} \tag{4.3}$$

For transfer rate information, the timeslot duration determines the time window of averaging for the simulation output only but does not determine the granularity of the event times of the simulation. Peak rates data provided by the simulation results may not necessarily represent the actual peak rates of the simulation. The transfer rates provided by the simulation results are transfers rates calculated over periodic time intervals. Actual peak rates are more appropriately calculated using a sliding averaging window. Each network object, except for the `RouteNode` class object, has one or more instance of `Traffic` to store packet transfers, number of requests, number of CRLs sent.

4.2.4 `ServerNode` Class

The `ServerNode` class object is the first of the main model classes and represents the Repositories. `ServerNode` class objects must be located on a leaf LAN node. Redundant connections to the `ServerNode` class objects are not available in the current version of the simulation program but are possible in the simulation model. The current version of the simulation program only allows the `ServerNode` class object to be located on leaf LAN nodes which are restricted by the Tiers program to having only one link connection to the network. The

`ServerNode` class object local link has associated with it a set of availability and reliability parameters λ_R, λ_F and P_e. The local link is connected to a `RouteNode` class object which connects the `ServerNode` class object to the rest of the spanning tree.

There are four `Traffic` class objects in a `ServerNode` class object.

- `packets_sent`
- `requests`
- `deltaCRLs_sent`
- `baseCRLs_sent`

The `packets_sent Traffic` class object records the number of packets sent out by the `ServerNode` class object. The `requests Traffic` class object records the number of incoming requests from the `LanNode` class objects. Note that re-requests for missing packets count as requests as well. The `deltaCRLs_sent` and the `baseCRLs_sent Traffic` objects record the number of Delta CRLs and Base CRLs requested by the `LanNode` class objects.

Additionally, the `ServerNode` class object records the times of recent CRL requests received by the `ServerNode` class object in the last T_{RRW} time window for determining whether to perform an aperiodic multicast in the MCA mode.

The `ServerNode` class object is assumed to receive an error-free copy of the most recent CA signed Delta and Base CRLs at every new CRL issuance interval. This is simulated by the `ServerNode` class object incrementing the `CRLnum` by one, setting the `time` of the new CRL

to the new CRL issuance time and setting the `time_expire` of the new CRL to the time of the next CRL issuance time.

The `ServerNode` class object basically does two jobs (Figure 4.3): respond to requests for CRL updates via unicasting (NMC, MCP and MCA modes) and optionally multicast Delta CRLs (MCP and MCA mode). For unicast requests, when the `ServerNode` class object receives

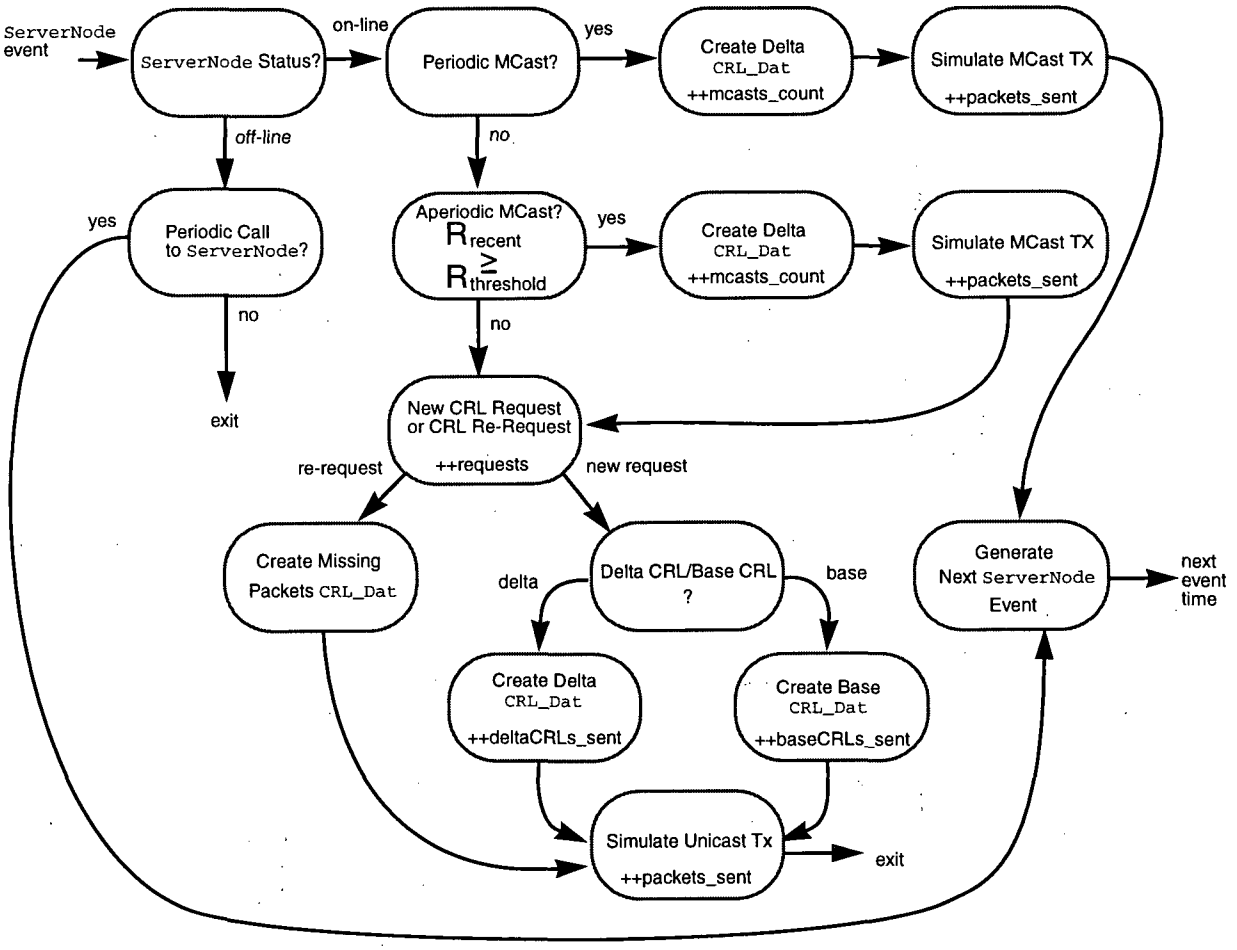

Figure 4.3 `ServerNode` class flow chart

the `CRL_Request` message originating from a `LanNode` class object, the `ServerNode` class object increments the `requests` `Traffic` class object by one. Next, the `ServerNode` class

object determines whether the requesting LanNode class object may use a Delta CRL or needs a Base CRL. If the CRLnum of the CRL held by the requesting LanNode class object is greater than or equal to the current CRL's CRLnum less the CRL update window W_U, then the ServerNode class object sends a Delta CRL; otherwise the requesting LanNode class object's local CRL is too outdated to be able to use the Delta and a Base CRL must be sent. The ServerNode class object then increments by one either the deltaCRLs_sent Traffic class object or the baseCRLs_sent Traffic class object. The ServerNode class object then creates the appropriate CRL_Dat message to represent a Delta or Base CRL to be sent to the requesting LanNode class object. The size of the CRL in terms of packets (which depends on whether the CRL to be sent is a Delta or a Base CRL) is added to the packets_sent Traffic class object. Next, the ServerNode class object simulates the transmission of each packet of the CRL over its local link as a Bernoulli trial. For each packet, the ServerNode class object randomly generates a 0 or a 1 with probabilities P_e and P_c respectively. P_e is the packet error probability parameter for the ServerNode class object's local link and $P_c = 1 - P_e$. A 1 means that the packet is sent successfully and a 0 means that the packet is sent unsuccessfully. The number of packets successfully received by the RouteNode class object that is on the other end of the ServerNode class object's local link is set in the outgoing CRL_Dat message packets_sent field.

If the ServerNode class object receives a CRL request from a LanNode class object for missing packets through a CRL_Dat message, the ServerNode class object increments by one the requests Traffic class object. The packets field in the CRL_Dat message received from the requesting LanNode class object is the number of missing packets that the

LanNode class object needs in order to get a complete copy of the CRL previously requested. The packets value is added to the packets_sent Traffic class object. The ServerNode class object then simulates the sending of the requested missing packets by simulating each CRL packet as a Bernoulli trial as previously described. The number of packets successfully received by the ServerNode class object's local router is then written onto the CRL_Dat message received from the requesting LanNode class object and sent back to the requesting LanNode class object.

There are two multicasting modes that the ServerNode class object can be in: MCP and MCA. In MCP mode, the ServerNode class object multicasts the current Delta CRL at regular times. In MCA mode, the ServerNode class object, in addition to sending multicasts at regular times, sends a multicast whenever the number of requests received in the last time window, T_{RRW}, exceeds a threshold, $R_{Threshold}$.

4.2.5 Edge Class

The Edge class represents a bidirectional data link between two RouteNode class objects. Each Edge class object consists of two pointers, one for the uplink direction towards the ServerNode class object and one for the downlink direction away from the ServerNode class object.

As with the ServerNode class local links, the Edge class also has a pre-generated failure and recovery linked list and a packet error probability parameter, P_e. Optionally, each Edge class object may have a Traffic class object to record all the packets that have been transmitted over the Edge class object. The link status transition rates, λ_R and λ_F, and packet

error probability, P_e, parameters and option for packet transfer recording are assigned to the Edge class objects as personalities. Different personalities may be assigned to different Edges class objects on the network to represent different link types such a wireless networks, PPP links over PSTN, and ethernet. In this study, all Edge class objects have the same personality.

Figure 4.4 shows the flow chart for an Edge class object. The input to an Edge class

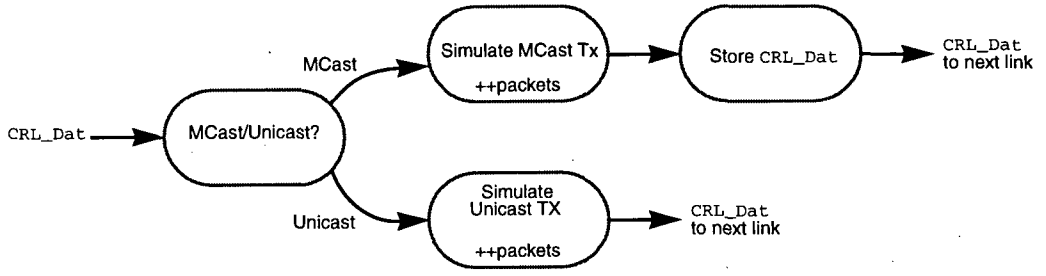

Figure 4.4 Edge class flow chart

object is a CRL_Dat message. The CRL_Dat message may be for either a Base CRL or a Delta CRL. When an Edge class object receives a CRL_Dat message, it simulates the transmission of packets of the CRL indicated in the packets_received field. If the optional Traffic class object was specified for the Edge class object, the value in the CRL_Dat packets_received field is added to the Traffic class object to keep track of how many packets traverse the Edge class object. The packet transmissions are simulated and the number of packets successfully received is then rewritten into the CRL_Dat packets_received field. If the CRL_Dat is a Delta CRL, the packets are simulated and results are stored in the CRL_Dat packets_file field in bit-wise format where each of the 32 bits represents the receive status of each packet in the Delta CRL. Additionally, if the CRL_Dat is a Delta CRL, the Edge class

object stores a copy of the `CRL_Dat` message for the purpose of simulating multicasting.

4.2.6 LanNode Class

The `LanNode` class objects represent the Relying Parties (RPs). `LanNode` class objects, like `ServerNode` class objects, are leaf nodes and are only connected to `RouteNode` class objects. Each `LanNode` class object represents a single RP. Like the `ServerNode` and `Edge` class objects, `LanNode` class objects may have different personalities to allow for RP with different characteristics on the same network. For simplicity, all the `LanNode` class objects are assumed to have the same personality in this study.

As with the `ServerNode` and `Edge` class objects, the `LanNode` class object also has the link failure and packet error probability parameters associated with its local link to represent the last mile connection or the local loops. As with the `Edge` class, the `LanNode` class also has an optional `Traffic` class to record packets going over its local link.

Each `LanNode` class instance is capable of caching a copy of the entire CRL which is referred to as the local CRL. The `LanNode` class object's function may be subdivided into two modules: the client program and the multicast receiving module.

4.2.6.1 Client Program

The client program represents the program which receives validation requests from a user and performs the validation of the certificate presented to it by the user. This client program can be an email reader program or an Internet browser. The flow chart for the client program is shown in Figure 4.5. Validation requests are assumed to arrive to each `LanNode` class object at a rate of `val_req_rate`, λ_{Val}. When the client program receives a validation request, it checks to see if

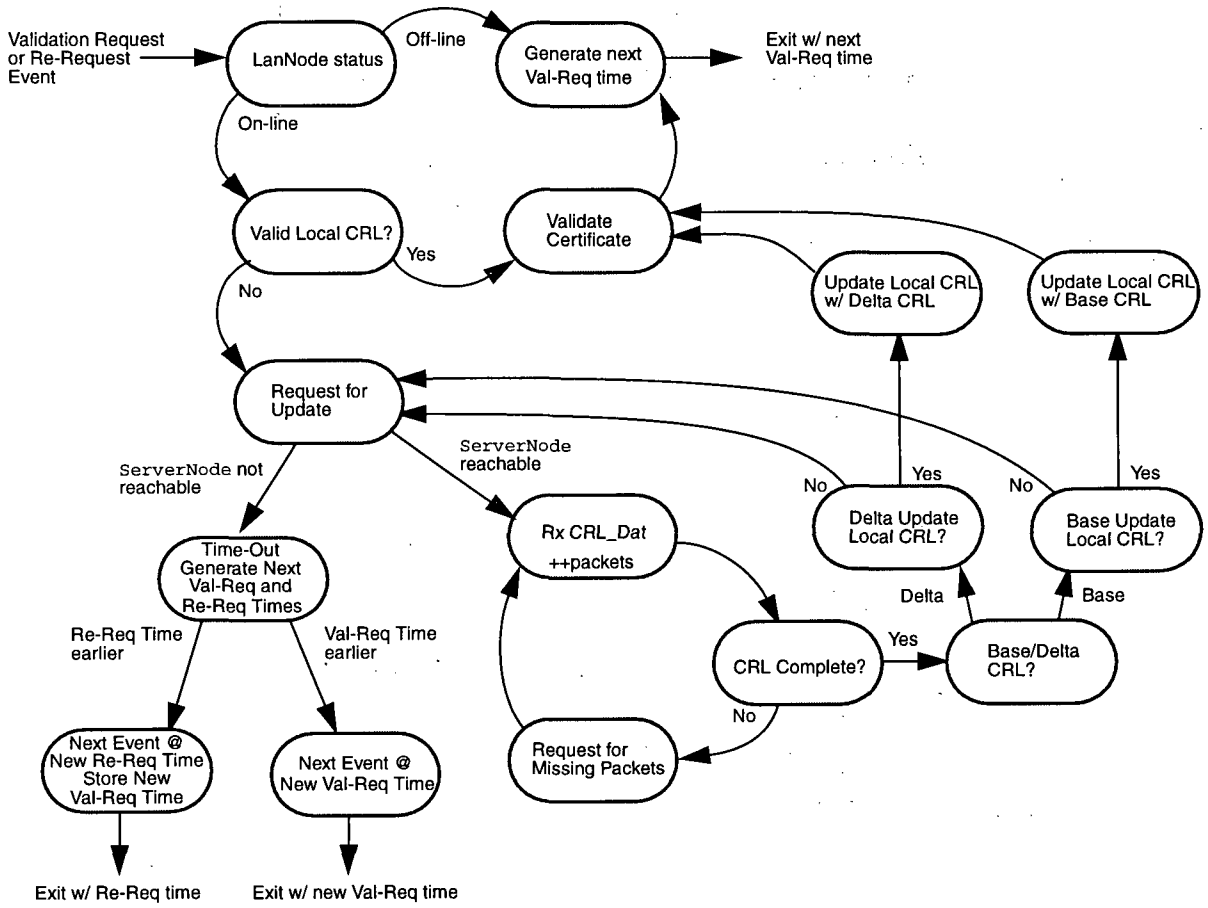

Figure 4.5 `LanNode` class client program flow chart

the its local copy of the CRL is current before performing the validation. If the `LanNode` class object's local CRL has expired, it makes a unicast request for a CRL update by creating a `CRL_Request` message. If the `ServerNode` class object is unreachable due to a network link failure, the `LanNode` class object's request is assumed to have timed-out. In the case of a time-out, the `LanNode` class object will back-off its pending request exponentially according to the `val_req_backoff` parameter, λ_{RR}. This is done by generating two next event times. One event is generated using the λ_{RR} parameter and the other is generated using the λ_{Val} parameter. If

the event generated by λ_{RR} parameter is earlier than that generated by the λ_{Val} parameter then the λ_{RR} parameter generated next event time will be returned as the next event and the event time generated by the λ_{Val} parameter will be stored as a pending validation request. However, if the next event time generated by the λ_{Val} parameter is earlier than the next event time generated by the λ_{RR} parameter, then the next event will be the event generated by the λ_{Val} parameter and the other is discarded. If the current event is a re-request due to a previous time-out, the same actions take place. Two events are generated using the two parameters as when a time-out occurs. This is done so that validation request interarrival times are completely independent from the re-request events. The re-requests however are not independent from the validation request interarrival times because re-request events may only occur if there was a previous validation request that had timed-out.

4.2.6.2 Multicast Receiving Module

The multicast receiving module may be a separate program which may be used in conjunction with legacy programs provided that the legacy program stores its copy of the CRL in a file accessible for reading and writing by the multicast receiving module. The job of the multicast receiving module is to listen for multicasts from a `ServerNode` class object. The flow chart for the multicast receiving module is shown on Figure 4.6. When a `ServerNode` class object multicasts a Delta CRL, the multicasting receiving module at the `LanNode` class object collects the packets of the Delta CRL file. If the received Delta CRL is complete and error-free, the multicast receiving module will check to see if the Delta CRL update the `LanNode` class object's local CRL. If the `baseCRLnum` of the newly received Delta CRL is equal or lower than the `CRLnum` of the local CRL, then the multicast receiving module updates the local CRL with

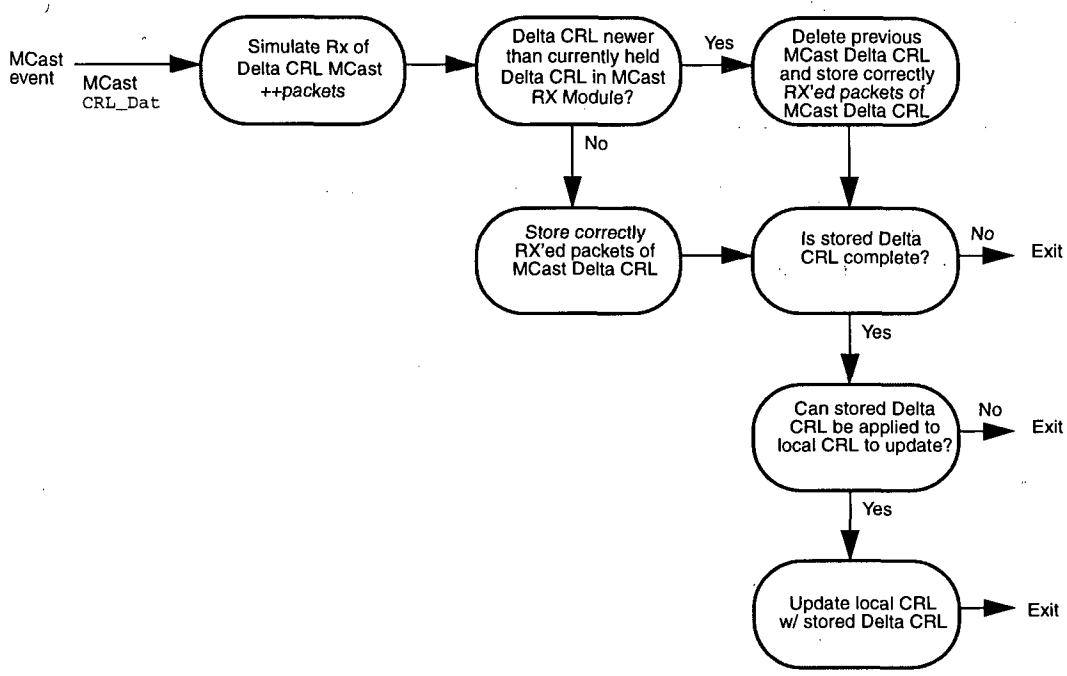

Figure 4.6 `LanNode` class multicast receiving module flow chart

the newly received Delta CRL. Otherwise, if the multicast received complete and error-free Delta CRL may not be applied to the local CRL, then the multicast Delta CRL is discarded. However, if the multicast receiving module does not receive a complete and error-free copy of the multicasted Delta CRL, then it stores all the correctly received packets of the multicasted Delta CRL. If and when the `ServerNode` class object re-multicasts the same Delta CRL, the multicast receiving module will take the correctly received packets of the repeated Delta CRL and attempt to recreate a complete copy of the Delta CRL. The `LanNode` class object continues to use the repeated multicasts to attempt to recreate a complete and error-free Delta CRL until a new Delta CRL is multicasted by the `ServerNode` class object. Note that the multicast receiving module does not share its partially received Delta CRL with the client program. Significant $packet \cdot link$ savings

may be achieved if the client program may copy the error-free packets from the multicast receiving module and request only the missing packets through a unicast request. However, this modification to the system is not analyzed in this study since separating the two programs is more useful for legacy applications. In this way, the client program and the multicast receiving module can be independent of each other.

4.2.7 `RouteNode` Class

The `RouteNode` class represents the routers and hubs of the network. The `RouteNode` class does not distinguish between WAN, MAN or LAN routers and hubs. `RouteNode` class objects have a combination of two or more `Edge`, `ServerNode` or `LanNode` class objects connected to it. Each `RouteNode` class object has a linked list of neighboring `Edge`, `LanNode` or `ServerNode` class objects. Additionally, each `RouteNode` has another link which links to the `Edge` or `ServerNode` class object which leads back to the server. This aspect of the simulation program allows for only one `ServerNode` class object.

4.2.8 Simulation Program Flow

The simulation program flow is detailed here. A state diagram for the overall simulation program is provided in Figure 4.7. At the start of the simulation, the program reads in the simulation parameters from a file and reads in the Tiers file for the network topology file. The Tiers file is converted into a pre-network using the `PreRouteNode` class objects. Details on the `PreRouteNode` class are not important. The pre-network is an intermediate network structure used for the conversion of the Tiers network topology file format to the spanning tree network structure used in the simulation program. The pre-network defines the location of each WAN, MAN and LAN node in relation to other WAN, MAN and LAN nodes but does not define the

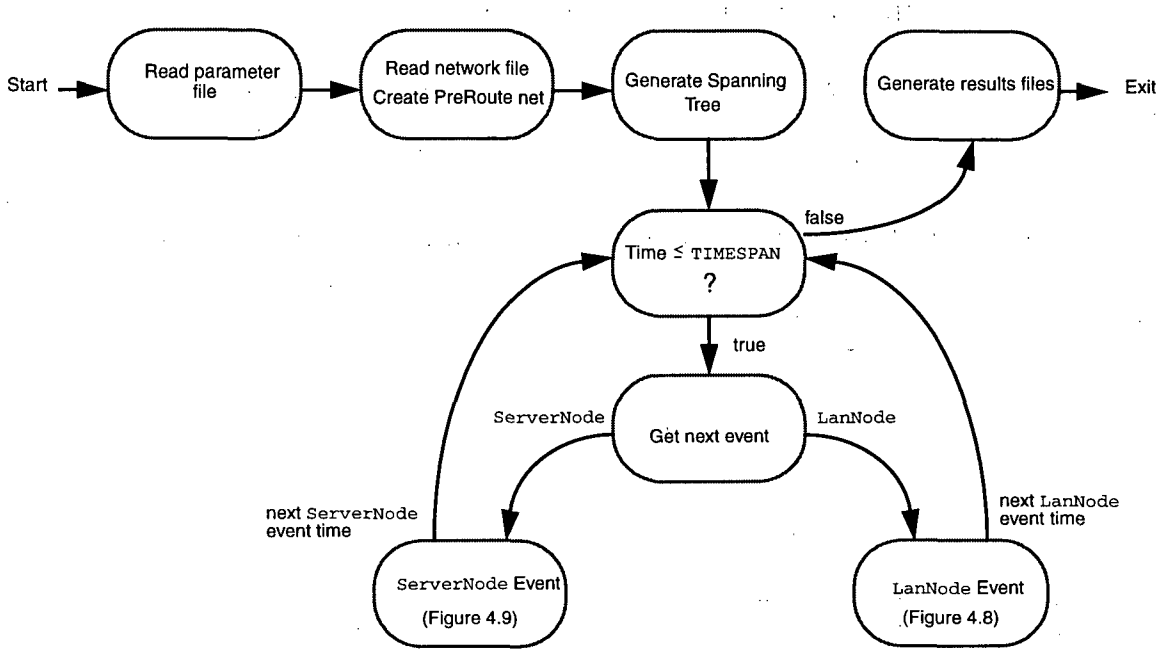

Figure 4.7 Simulation program flow chart.

location of the ServerNode, RouteNode, Edge and LanNode class objects which represents the Repository, routers, edges, and RPs. Each PreRouteNode class object has a list of its neighboring nodes. Another difference between the pre-network and the spanning tree network structure used in the simulation program is that the spanning tree network structure defines a data source and data sinks whereas the pre-network does not. The data source is the ServerNode class object. The data sinks are the LanNode class objects.

Next, the program takes the location of the ServerNode class object as defined in the input parameter file as the starting point and creates a spanning tree composed of ServerNode, RouteNode, Edge, and LanNode class objects with the selected ServerNode class object location as the root of the spanning tree. The ServerNode and each Edge and LanNode class object are initialized as they are created. If the object created is the ServerNode or LanNode

class object, an event time will be generated for the object and sorted chronologically into the `EventList`. When the entire spanning tree has been created, the `EventList` will have the same number of events as there are `LanNode` and `ServerNode` class objects. Additionally, as each `ServerNode`, `Edge` and `LanNode` class object is created, its initial link status is set to on-line or off-line using the respective state transition rates λ_F and λ_R. Additionally, a list of link state transition times is created using the respective state transition rates λ_F and λ_R for each `ServerNode`, `Edge` and `LanNode` class instance. The length of the link state transition times list depends on the duration of the simulation run. The link state transition times are pre-generated to allow for a fair comparison between methods by ensuring the same network conditions. Note that the event times generation of the `LanNode` class objects also uses the same random number generator as the link state transition times list generator function. However, care has been taken to ensure that the number of calls to the random number generator prior to each step in creating and initializing the spanning tree is the same so that link status transition times remain the same and a function of only the simulation's random seed parameter.

After the spanning tree has been created, the simulation clock may start. The first event on the `EventList` is selected. If multicasting has been enabled at the `ServerNode` class object, then the first event will be a multicast. The information provided by the `EventList` is the time of the event, `NodeID`[2], and a link to the `LanNode` class object to which the event belongs to if the event source is an `LanNode` class object. If the event source is a `ServerNode` class object, the `EventList` will only provide the time of the event and the `NodeID` of the `ServerNode` class object. The link to the `ServerNode` class object is made available to the program at all

[2] A `NodeID` is a number given to each WAN, MAN and LAN node on the network which distinctly identifies each node.

times and is not stored on the EventList to conserve computer memory. If the multicasting has been disabled at the ServerNode class object then all events will be from LanNode class objects making validation requests.

If the next event taken from the EventList is a LanNode class object performing a validation request then the process is as shown in the flow chart Figure 4.8. The program first

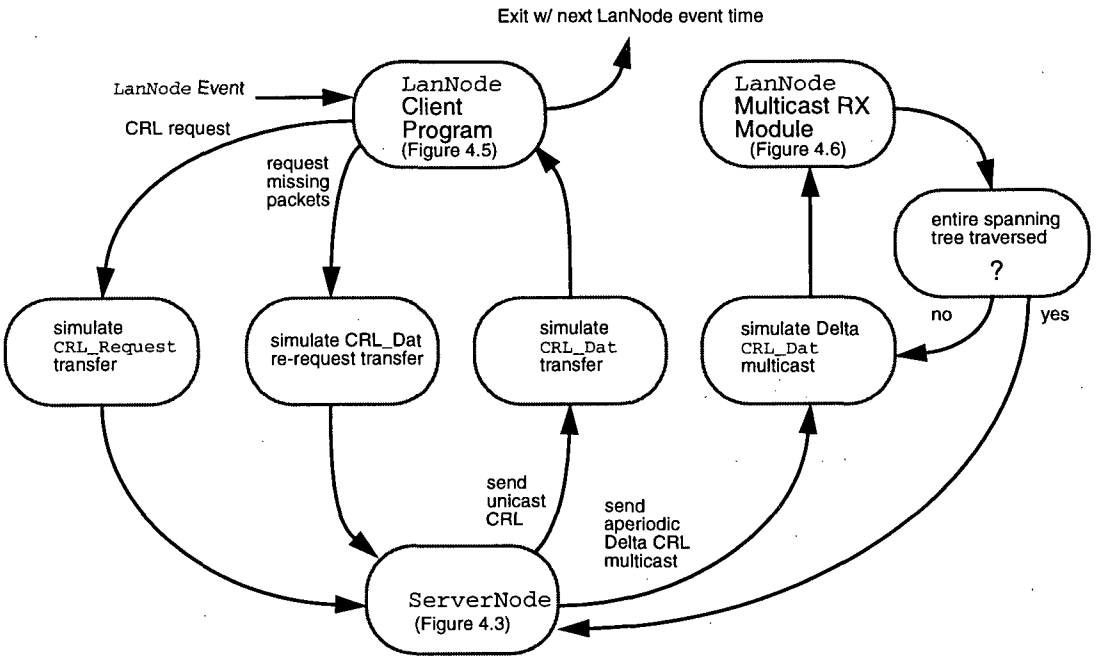

Figure 4.8 LanNode class event flow chart

checks to see if the NodeID does not belong to the ServerNode, if it does not then the event is assumed to come from a LanNode class object. The link attached to the EventList class structure allows the program to go directly to the LanNode class object that the event belongs to. The program checks the status of the LanNode class object needing to make the validation request. If the LanNode class object is down, then the validation request is discarded and the

next event time for that LanNode class object is generated and placed into the EventList sorted chronologically. If the LanNode class object is on-line, then it checks to see if its local CRL is valid to perform the validation request. If yes, then the validation request is performed. Next, the LanNode class object generates the time of its next validation request. This new event time is then added to the EventList and sorted chronologically. If however, the LanNode class object finds that its local CRL has expired and cannot be used to perform the validation request, then it makes a unicast request for a CRL update. The LanNode class object begins by creating a CRL_Request message. The CRL_Request message is assumed to be error-free and of negligible size. The program simulates the sending of the CRL_Request message by checking the status of each link leading to the ServerNode class object. The program finds the route from the LanNode class object leading back to the ServerNode class object by reading the up-route links at each RouteNode class object until it reaches the ServerNode class object. If a link leading to the ServerNode class object is down, the LanNode class object times out and generates its next event time. The next event time is generated using both the λ_{Val} and the λ_{RR} parameters as described in Section 4.2.6.1 for CRL request time-outs. The next event time is then placed on the EventList and again sorted chronologically. If the CRL_Request message reaches the ServerNode class object, the ServerNode class object adds the current CRL request to the recent CRL requests list as described in Section 4.2.4 for aperiodic multicasts. The ServerNode class object then sends out an aperiodic multicast if the number of recent requests in the last T_{RRW} time window meets or exceeds $R_{Threshold}$ CRL requests. Whether or not an aperiodic multicast is sent, the ServerNode class object will send out a unicast CRL_Dat message representing either a Base or Delta CRL in response to the requesting LanNode class object's original CRL request. The ServerNode class object sends the unicast

CRL even if a multicast was sent because the multicasted Delta CRL is not guaranteed to be received by the LanNode class object which made the CRL request. Also, the LanNode class object which made the CRL request may possibly not be able to use the Delta CRL and may require a Base CRL. The unicast CRL_Dat message follows the same path back to the requesting LanNode class object. As the CRL_Dat message is transferred over each link, each packet in the CRL is simulated using the packet error probability parameter attributed to that link (Section 4.2.4). When the CRL_Dat message reaches the LanNode class object that made the request, the LanNode class object checks if the CRL was received completely. If the LanNode class object did not receive the CRL completely, then it will re-request the missing packets with a new CRL_Dat message, rather then a CRL_request message. The CRL_Dat re-request message contains the number of packets needed to complete the CRL in the packets field. The CRL_Dat re-request message like the CRL_Request message is error-free and of negligible size. The ServerNode class object responds by supplying the missing CRL packets in a CRL_Dat message that transfers in the same way as the first CRL_Dat message containing the whole Base or Delta CRL.

If the next event on the EventList is a ServerNode class object multicast, then the program proceed as shown in Figure 4.9. The program first checks the status of the ServerNode class object. If the ServerNode class object is off-line, then the next periodic multicast time will be added to the EventList. If the ServerNode class object is on-line, it will perform a multicast of the current Delta CRL. The ServerNode class object starts by forming a CRL_Dat message which is then transmitted over the ServerNode class object's local link. The program then traverses the entire spanning tree simulating the transmission of the Delta CRL until all the LanNode class object instances have been reached. This is done by the program

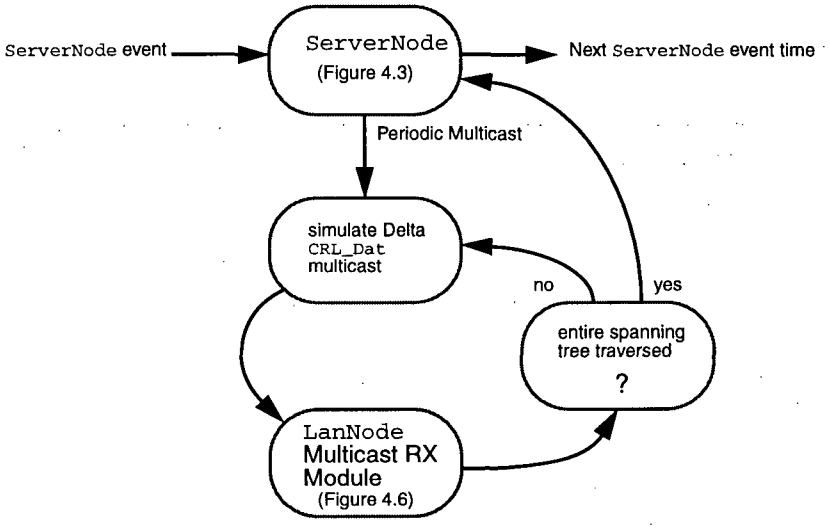

Figure 4.9 `ServerNode` class event flow chart

reading the path lists of each `RouteNode` until all paths on the spanning tree have been exhausted. When the Delta CRL has reached a `LanNode` class object, the `LanNode` class object's multicast receiving module will check to see if the received Delta CRL is complete and error-free. If so, the `LanNode` class object checks if the Delta CRL may update its local CRL. The Delta CRL is discarded if it cannot update the RP's local CRL. The `LanNode` class object will have to request for a Base CRL when it has a validation request to perform. If the `LanNode` class object does not get a complete Delta CRL from the multicast, it may store the good packets of the Delta CRL multicast in the hope that later multicasts of the same Delta CRL may be used to create a good copy.

After an event has been serviced, the program moves to the next event on the `EventList`. This process continues until the time of the event to be serviced next exceeds the duration of the simulation. When the simulation has completed, the program processes the data from the `Traffic` class of each network object and writes them to an output file.

4.3 Parameter Selection for Simulation

The parameters required in the simulation program are detailed in this section. The parameters may be grouped as:

- Simulation Parameters
- `ServerNode` Parameters
- `Edge` Parameters
- `LanNode` Parameters

4.3.1 Simulation Parameters

4.3.1.1 `TIMESPAN`, `SLOTTIME` and `TIMESLOTS` (T_{simrun}, T_{slot}, N_{slot})

The `TIMESPAN` is the duration of the simulation in seconds. This may range from a few minutes to several days. The `TIMESPAN` of the simulation run is divided into a number of equally spaced intervals called `TIMESLOTS`, each with a duration of `SLOTTIME` seconds. The `TIMESLOTS` value is used by the `Traffic` class to define how many array elements to use for recording the packet transfers and CRL requests. The memory requirements for the simulation increases linearly with `TIMESLOTS`. For a fixed `TIMESPAN`, `TIMESLOTS` is inversely proportional to `SLOTTIME`. To reduce memory requirements, SLOTTIME should be long; however this would result in a decrease in the time resolution.

4.3.1.2 `VERBOSE` Mode

The `VERBOSE` Mode is used for debugging. When the `VERBOSE` mode is set to true, all important simulation events, actions and states will be written to a file as the simulation progresses.

4.3.1.3 SEED

The SEED value is used to initialize the random number generator used in the program. Simulation runs with the same SEED, same input parameters and same network topology will yield the same results.

4.3.2 ServerNode Parameters

4.3.2.1 ServerNode Location

The ServerNode Location is defined with the DEFAULT_SERVER_NODE_ID parameter. The ServerNode Location must be a leaf node. If the selection is not a leaf node, the program will write out all possible candidates for the ServerNode class object and exit.

4.3.2.2 CRL_ISSUANCE_PERIOD ($T_{CRLissuance}$)

The CRL_ISSUANCE_PERIOD is the time interval between consecutive issuances of the CRL by the CA in seconds.

4.3.2.3 CRL_LIFETIME (T_{CRL})

The CRL_LIFETIME is the interval between issuance and expiration. This may be set to be the same as the CRL_ISSUANCE_PERIOD or longer but not less than. Longer intervals may be used to demonstrate Overissued CRLs as presented by [6, 7].

4.3.2.4 BASE_CRL_LENGTH (L_B)

The BASE_CRL_LENGTH is the length of the Base CRL in packets. Note that the smallest unit of data in the simulation program is a packet. The size of the Base CRL may be estimated in bytes and rounded up to the next full packets. Only integer values may be used for this parameter.

4.3.2.5 DELTA_CRL_LENGTH (L_Δ)

The DELTA_CRL_LENGTH is the length of the Delta CRL in packets. The size of the Delta CRL may be estimated in bytes and rounded up to the next full packets. Only integer values may be used for this parameter. The length of the Delta CRL is affected by the DELTA_BASE_UPDATE_WINDOW size.

4.3.2.6 DELTA_BASE_UPDATE_WINDOW (W_U)

The DELTA_BASE_UPDATE_WINDOW sets how far back a Delta CRL may be applied to update a RP's local CRL. Changes in the DELTA_BASE_UPDATE_WINDOW size effect the DELTA_CRL_LENGTH. Only Sliding Window Delta CRLs as proposed by [7] may be demonstrated with the current simulation program.

4.3.2.7 MULTICASTING_STATE

The MULTICASTING_STATE enables and disables multicasting for the entire simulation run.

4.3.2.8 MULTICASTING_PERIOD

The MULTICASTING_PERIOD is the time interval between ServerNode class object periodic multicasts and may be set independent of the CRL_ISSUANCE_PERIOD. If the MULTICASTING_PERIOD is set smaller than the CRL_ISSUANCE_PERIOD, the same CRL will be multicasted periodically a number of times.

4.3.2.9 ENABLE_UNSCHEDULED_MULTICASTS

The ENABLE_UNSCHEDULED_MCASTS parameter enables and disables aperiodic multicasting for the whole simulation run.

4.3.2.10 REQUEST_THRESHOLD ($R_{Threshold}$)

The REQUEST_THRESHOLD sets the number of recent requests the ServerNode class object has received in the last REQUEST_RECENT_TIME_WINDOW to trigger an aperiodic multicast. This parameter has no effect when multicast or aperiodic multicasts are disabled. This REQUEST_THRESHOLD is an integer value.

4.3.2.11 RECENT_REQUEST_TIME_WINDOW (T_{RRW})

The REQUEST_RECENT_TIME_WINDOW sets the time window interval to which the ServerNode class object will look back to see how many requests have been received to determine whether an aperiodic multicast is to be called. The REQUEST_RECENT_TIME_WINDOW is a sliding window and is set as an integer value representing seconds.

4.3.2.12 PEP_SERVER (P_e)

The PEP_SERVER parameter sets the packet error probability for the ServerNode class object's local link. This is a value between 0.0 and 1.0.

4.3.2.13 PF_SERVER and PR_SERVER (λ_F, λ_R)

The PF_SERVER and PR_SERVER parameters are the rate of failure and the rate of return from failure respectively for the ServerNode class object's local link.

4.3.3 Edge Parameters

4.3.3.1 EDGE_TRANSFER_LIST

The EDGE_TRANSFER_LIST parameter enables and disables the Traffic class recording option for the Edge class objects.

4.3.3.2 PEP EDGE (P_e)

The PEP_EDGE parameter sets the packet error probability of the Edge class objects and has the same requirements as PEP_SERVER.

4.3.3.3 PF_EDGE and PR_EDGE (λ_F, λ_R)

The PF_EDGE and PR_EDGE parameters are the rate of failure and the rate of return from failure respectively for the Edge class object's link.

4.3.4 LanNode Parameters

4.3.4.1 RELYING_PARTY_TRANSFER_LIST

The RELYING_PARTY_TRANSFER_LIST parameter enables and disables the Traffic class recording option for the LanNode class objects.

4.3.4.2 VALIDATION_REQUEST_RATE (λ_{Val})

The VALIDATION_REQUEST_RATE parameter sets the rate at which validation requests arrive to each LanNode class object.

4.3.4.3 VALIDATION_REQUEST_BACKOFF_RATE (λ_{RR})

The VALIDATION_REQUEST_BACKOFF_RATE parameter sets the rate at which re-requests arrive when there is a pending CRL update request because of a time-out in a prior request attempt.

4.3.4.4 INITIAL_CRL_NUM

The INITIAL_CRL_NUM parameter sets the initial CRL (serial) number of all the RP. This value is overridden when the OVERISSUE_STEADYSTATE parameter has been enabled.

4.3.4.5 OVERISSUE_STEADYSTATE

The OVERISSUE_STEADYSTATE parameter when enabled chooses at random the initial CRL (serial) number of each LanNode class object at initialization. With Overissued CRLs [6, 7], there is a range of CRL's that are valid at any point in time. The range of valid CRLs is calculated with the equation:

$$\left(i, i + \frac{T_{CRL}}{T_{CRLissuance}}\right) \quad \text{(EQ 1)}$$

where i is the CRL number of the current CRL. The initial CRLs of the LanNode class object instances are taken uniformly from the range calculated with [1] with $i = -1$.

4.3.4.6 PEP_RELYING_PARTY (P_e)

The PEP_RELYING_PARTY parameter sets the packet error probability of the LanNode class objects and has the same requirements as PEP_SERVER.

4.3.4.7 PF_RELYING_PARTY and PR_RELYING_PARTY (λ_F, λ_R)

The PF_RELYING_PARTY and PR_RELYING_PARTY parameters are the rate of failure and the rate of return from failure respectively for the LanNode class object's local link.

Chapter 5 Discussion of Results

In this chapter, results obtained using the analytical model and the simulation model are discussed. The simulation parameters are first reviewed, followed by a description of the generation of the random network topologies used in the simulations. Then, the analytical and simulation model results are compared. Thereafter, all results for parameter variations are obtained by simulation. The CRL update window, W_U, parameter may be chosen to minimize the cost of CRL distribution. Various values of W_U are simulated and cost results are compared for the NMC, MCP and MCA systems. The effects of varying the $R_{Threshold}/T_{RRW}$ rate to adjust the multicast decision threshold in the MCA systems are studied. The NMC, MCP and MCA systems are simulated for various link availability levels and packet error probabilities. The three systems are also examined for different network topologies and server locations.

5.1 Review of Simulation Parameters

The NMC, MCP and MCA were simulated under various network conditions and parameters.

Each plotted point is the average of 10 simulation runs using 10 different seed values. The same set of 10 seeds is used for the NMC, MCP and MCA systems to allow for a more controlled comparison. The 95% confidence intervals [31] are shown as error bars in the plots. Some error bars may be difficult to see when the confidence intervals are small.

Failure parameters were set to one of 3 levels labeled 100%, "AT&T" and "MCI" based on their respective Service Level Agreements (SLA). The network availability parameter are

provided in Table 5.1. The values λ_F and λ_R were found based on an arbitrary 95% conformance to SLA values (Section 4.1).

Table 5.1 Network availability levels

Level	Availability	Downtime	λ_F	λ_R
100%	100%	0	n/a	n/a
"AT&T"	99.9%	10 min	0.000005	0.005
"MCI"	99%	1 hour	0.0000084	0.000832

The TIMESPAN of each simulation run is 10 days or 864000 seconds. For the 2 hour, 6 hour and 24 hour CRL lifetimes, there are 120, 40 and 10 CRL issuances respectively. The TIMESPAN is divided into 1440 TIMESLOTS each having a 600sec duration (SLOTTIME) except where indicated otherwise.

The parameters shown in Table 5.2 are used in all analyses and simulations in this study.

Table 5.2 Parameters common to all simulation runs

Parameter	Value	Additional Notes
No. of unexpired certificates, N_{Cert}	3,000,000	from [1], [6]
Certificate Lifetime, T_{Cert}	1 Year	from [1], [6]
No. of RPs, N_{RP}	29,999	30,000 leaf LAN nodes are generated and one was used for the Repository
Validation Request Rate per RP, λ_{Val}	$\frac{10 \text{ Requests}}{86400 \text{ Sec}}$	from [6]
Re-Request Rate per RP, λ_{RR}	$4 \times \lambda_{Val}$	value arbitrarily chosen (Section 4.2.6.1)
Certificate revocation probability	10%	from [1], [6]
Base CRL length, L_B	1351 Packets	calculated from (3.2)

The Delta CRL length, L_Δ, is calculated based on T_{CRL} and W_U according to (3.4).

All simulations were performed using the same network topology and server location except for the simulations which examine variations in network topology and server location. The main network topology used in the simulations was based roughly on the network topologies of various North American universities discovered through the Unix `traceroute` command. The `traceroute` command was used to locate faculty and departmental web servers of various universities with respect to the starting point located in the Communications Group subnet within the Department of Electrical and Computer Engineering at the University of British Columbia. Web servers were targeted as they were expected to be installed by most faculties and departments and distributed over each university's network. The Repository location for the main network topology was selected based on its average distance to all the RPs. The average distance between a computer located in the Communications Group's subnet within the Department of Electrical and Computer Engineering at the University of British Columbia and various web servers of North American universities was found to be 14 links and so a Repository location with similar average RP distance was selected. The Repository to RP distance distribution for the network topology and the server location used for most of the simulation work is shown in Figure 5.1. The Repository to RP distance distribution shows that there is at least one smaller subnetwork with average distance 20 links connected to a larger subnetwork with average distance of 12 links.

5.2 Comparison of Analytical and Simulation Models

The analytical model and the simulation model are described in Chapters 3 and 4 respectively. In this section, the analytical model results are compared against those obtained with the simulation program. Note that the analytical model does not accurately model the MCA system. An equation for the number of multicast repetitions r is not yet available. In the analytical model,

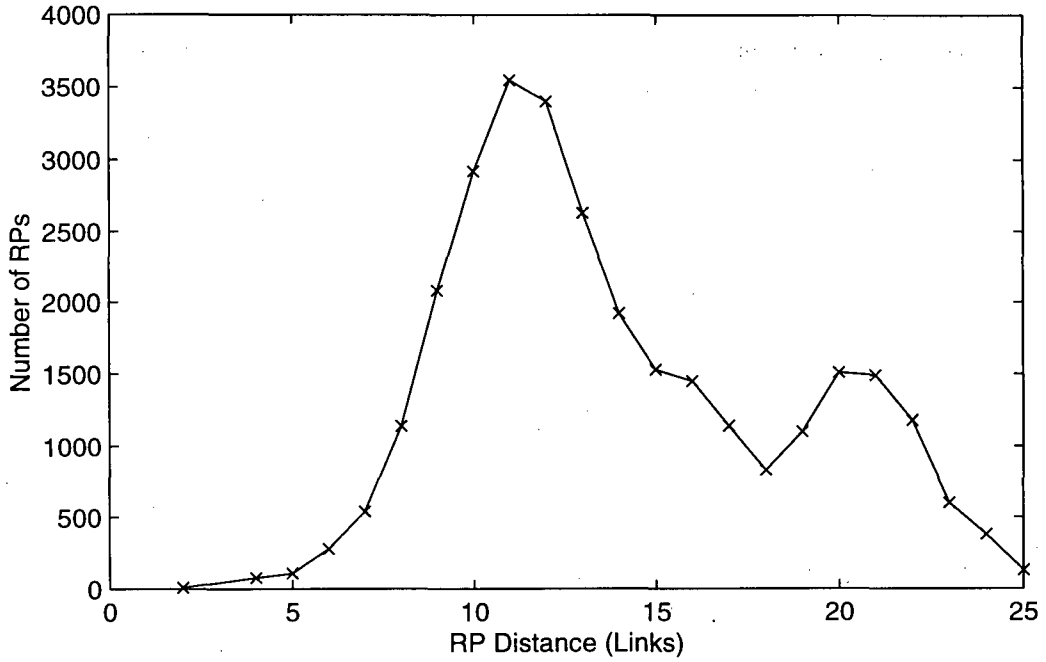

Figure 5.1 Repository to RP distance distribution

r is set to a fixed integer: 0 for the NMC system and 1-3 for the MCP system. The simulation program was set to send a fixed number of multicasts at the beginning of each CRL issuance period. Since the analytical model does not include network link failure and recovery probabilities, comparison is based on 100% network availability. The assumption that all multicast events occur at the beginning of the CRL issuance period is acceptable with 100% network availability. W_U was set to 6, 8 and 9 for the MCP system and 9 for the NMC system. The packet error probability was varied between 0.005 and 0.10. The results are shown in Figure 5.2. The plots show that the analytical model results closely agrees with the simulation results.

5.3 Effects of Varying the Delta CRL Update Window Size W_U

This section compares the effects of varying the CRL update window, W_U, in the Sliding

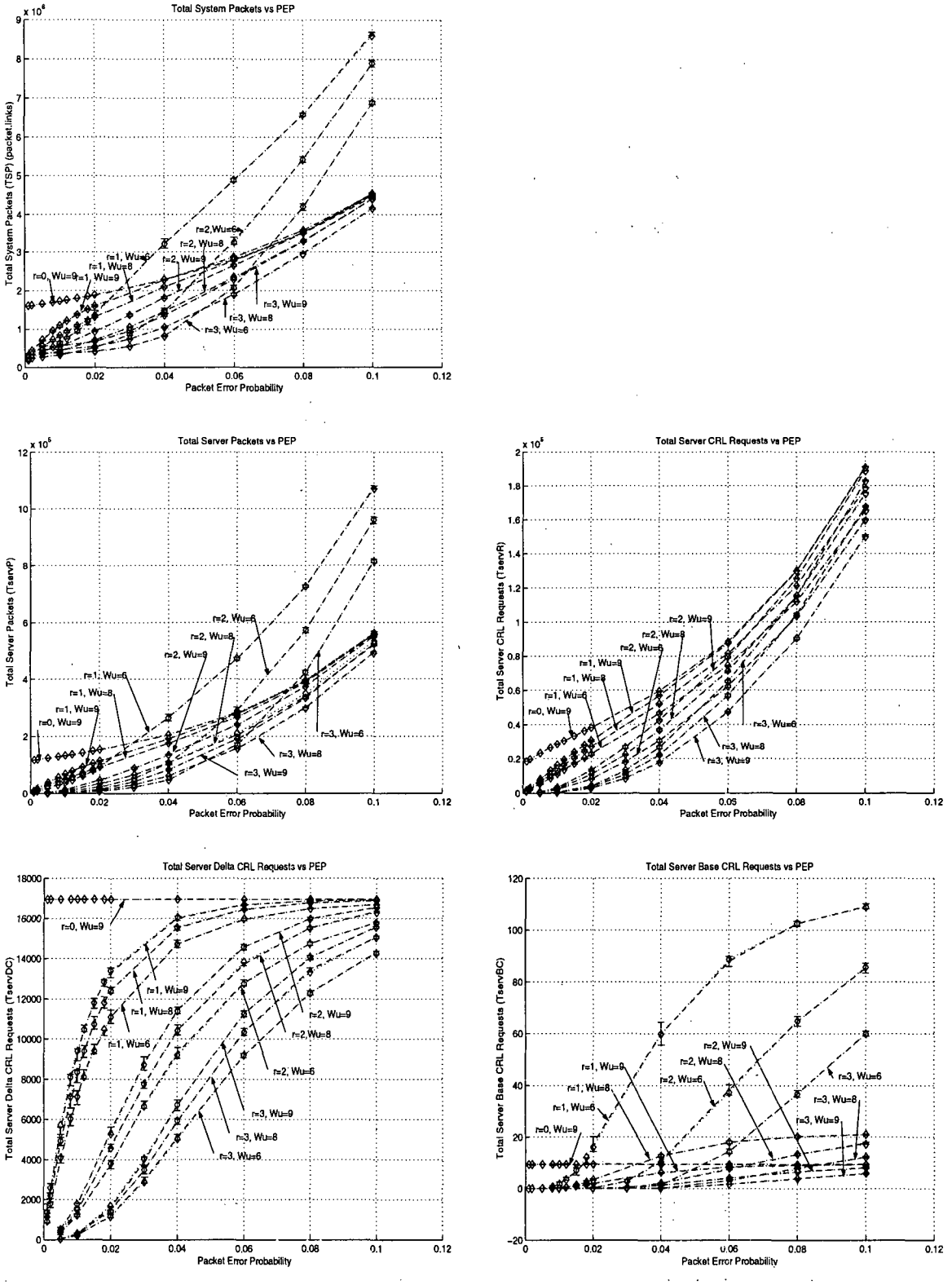

Figure 5.2 Analytical model vs. simulation model results

Window Delta CRL system proposed by [7] for the NMC, MCP and MCA systems. The main purpose of adjusting W_U is to balance the probabilities of a Base CRL request and Delta CRL request to minimize the $packet \cdot link$ costs. The parameters used in this section are shown in Table 5.3

Table 5.3 Effects of varying W_U parameters

Parameter	**Value**	**Additional Notes**
Network	#2	from Table 5.8
Packet error probability, P_e	0.01	from [19-21, 24]
Network Availability	"MCI", "AT&T" and 100%	from Table 5.1
CRL issuance interval, T_{CRL}	2 hours, 6 hours and 24 hours	
CRL update window size, W_U	T_{CRL} = 2 hours : 1...20 T_{CRL} = 6 hours : 1...6 T_{CRL} = 24 hours : 1...4	various increments
Delta CRL length, L_Δ	T_{CRL} = 2 hours : 1...13 packets T_{CRL} = 6 hours : 2...12 packets T_{CRL} = 24 hours : 8...30 packets	calculated from (3.4)
Request rate threshold, $R_{Threshold}/T_{RRW}$	30 requests/50sec	

Short CRL issuance intervals, T_{CRL}, are expected to be used for applications in which high losses are incurred should someone successfully attempt to use a revoked certificate during the vulnerable period. A shorter T_{CRL} increases the likelihood for a Base CRL requests because the number, n_{rp}, of RPs that are expected to make a validation request during a T_{CRL} is smaller according to the (5.4):

$$\begin{aligned} n_{rp}(t_0, t) &= N_{RP}(e^{-\lambda_R t_0} - e^{-\lambda_R t}) \\ n_{rp}(0, T_{CRL}) &= N_{RP}(e^{-\lambda_R 0} - e^{-\lambda_R T_{CRL}}) \\ &= N_{RP}(1 - e^{-\lambda_R T_{CRL}}) \end{aligned} \tag{5.4}$$

The plots in Figure 5.3 show that the minimum T_{sP}, T_{servP}, T_{servR}, T_{servDC} and T_{servBC} values for the MCA system are significantly lower than for the MCP and NMC systems. The NMC system has the highest values for all 3 levels of network availability. However as W_U increases, the T_{sP}, T_{servP}, T_{servR}, T_{servDC} and T_{servBC} curves for MCP approaches those for NMC. This is due to the fact that as W_U increases, the size of the Delta CRL increases. As a result, the likelihood of the multicasted Delta CRL reaching the RPs decreases. In the MCP case, only one multicast is sent by the Repository and thus increasing W_U will increase the need for unicasting CRL updates. As shown in the plots, T_{servBC} drops quickly as the W_U increases and the T_{sP}, T_{servP}, and T_{servR} plots are dominated by Delta CRL requests. The results from the T_{sP} plots show that smaller W_U values may be used for MCA since the likelihood of acquiring a complete Delta CRL increases with multiple Delta CRL multicasts. A reason for using a larger W_U value would be if network reliability is reduced much further below the "MCI" levels. The T_{sP} results for the 3 levels of network availability show that lower network availability increases T_{sP} costs. In a network with lower network availability where portions of the network are more likely to be disconnected due to link failures, the probability of a successful multicast is reduced thereby increasing the need for additional multicasts and unicast CRL update requests. For W_U value smaller than optimum values, the T_{sP} and T_{servP} costs are very high due to the high

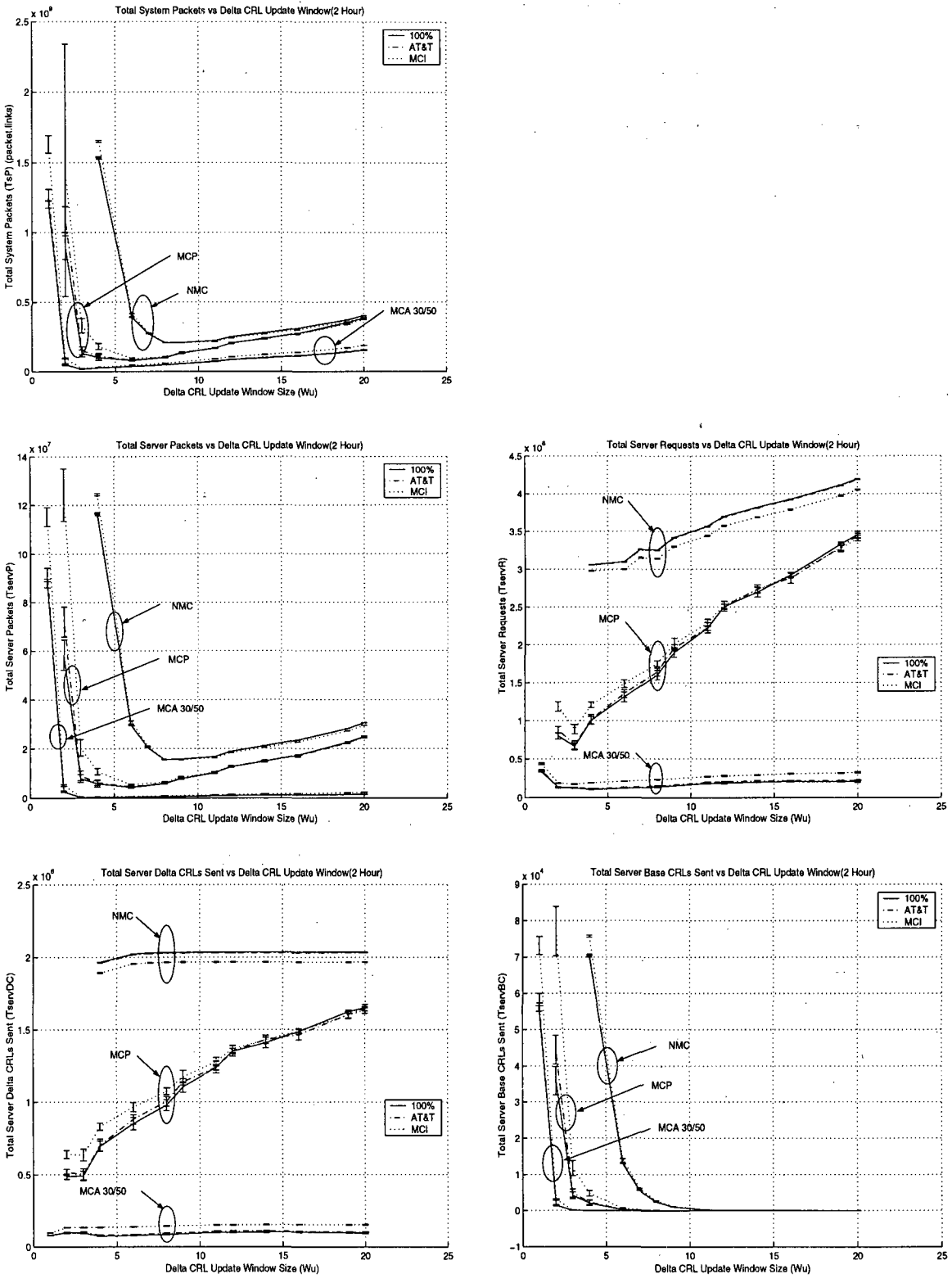

Figure 5.3 Effects of varying W_U (2 hour CRL issuance interval)

request rate and cost of sending Base CRLs.

Results for CRL issuance intervals of 6 hours and 24 hours for the 3 network availability levels are shown in Figures 5.4 and 5.5 respectively. The W_U value which minimizes T_{sP} and T_{servP} decreases when the RP validation rate λ_{Val} is set constant and the CRL issuance interval increases. This is because the probability of a RP making one or more validation requests increases as the CRL issuance interval increases thus decreasing the probability of needing a Base CRL. In the case of the 24 hour CRL issuance interval, the CRL update window size need not be greater than 1 to minimize costs. Additionally, as W_U increases, the T_{sP} cost of the MCP system becomes greater than the cost of the NMC system. This is due to the wasted multicast effort as the number of incomplete multicasted Delta CRLs received by the RPs mounts with increasing W_U and Delta CRL size. The results shows that MCA is beneficial to reducing T_{sP} and T_{servP} cost for various W_U. The optimum W_U values for the NMC, MCP and MCA systems found from simulation are shown in Figure 5.4. The optimum window sizes for the NMC system agrees with those obtained using (3.33) from [7].

5.4 Effects of Varying the $R_{Threshold}/T_{RRW}$ Rate

The $R_{Threshold}/T_{RRW}$ rate is the threshold CRL request rate used by the Repository to determine whether a Delta CRL multicast is needed. This rate may be adjusted to optimize the number of multicasts of the current Delta CRL to minimize T_{sP}. The $R_{Threshold}/T_{RRW}$ rate setting is applicable only to the MCA system. The parameter values are shown in Table 5.5.

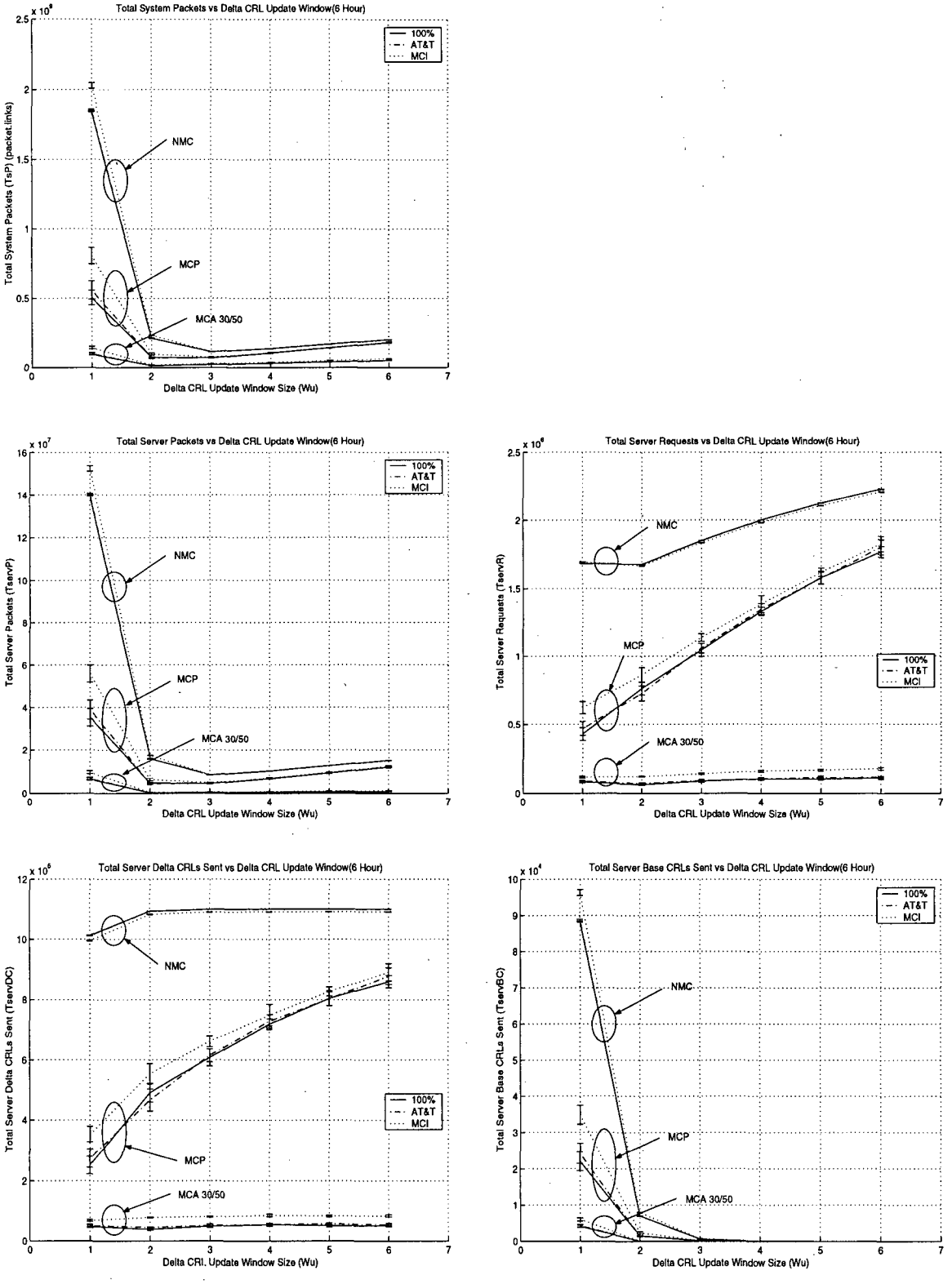

Figure 5.4 Effects of varying W_U (6 hour CRL issuance interval)

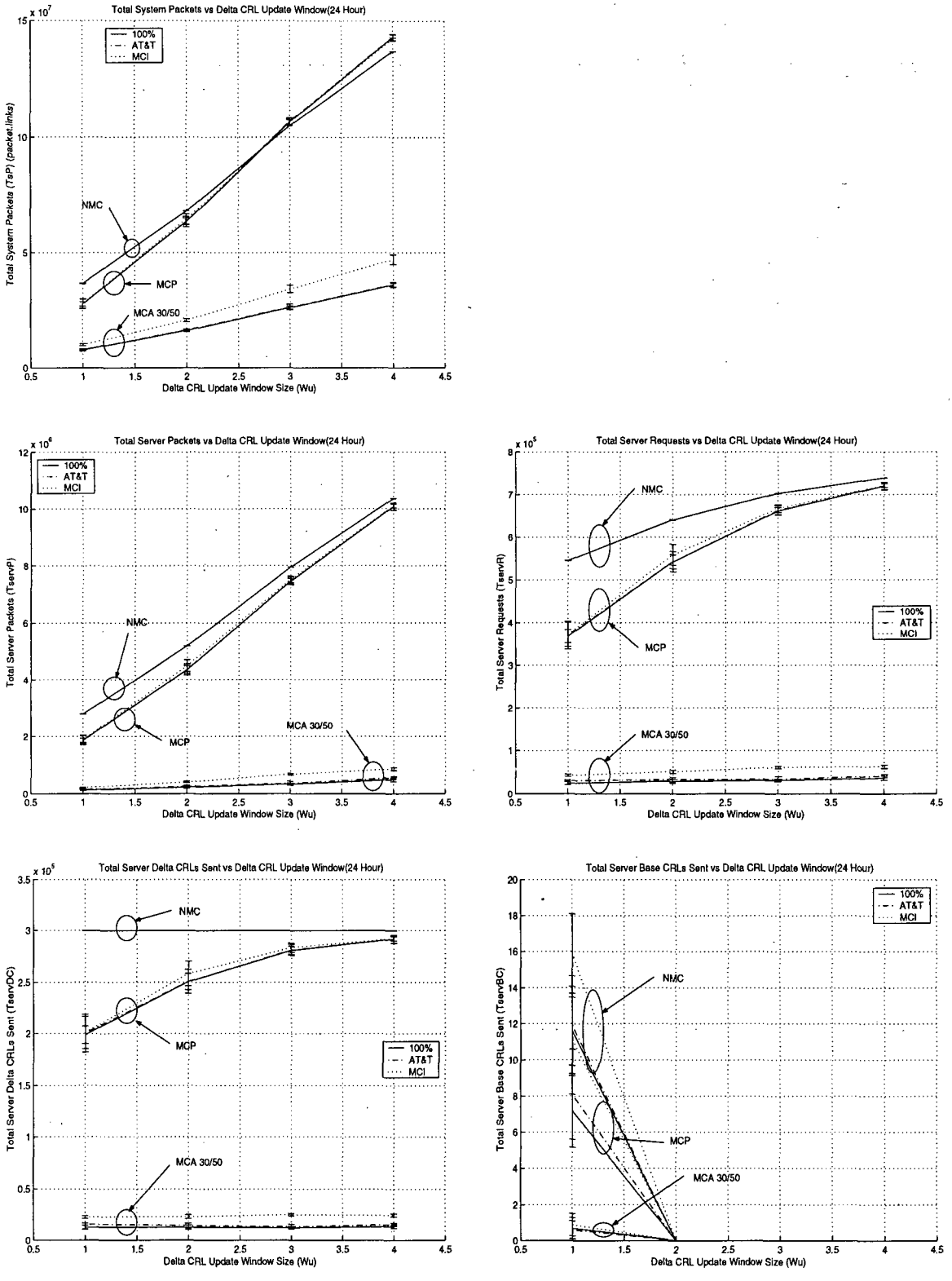

Figure 5.5 Effects of varying W_U (24hour CRL issuance interval)

Table 5.4 Optimal W_U size for minimizing T_{sP} and T_{servP}

System	2 Hours	6 Hours	24 Hours
NMC	9	3	1
MCP	6	2	1
MCA	3	2	1

Table 5.5 Effects of varying $R_{Threshold}/T_{RRW}$ parameters

Parameter	Value	Additional Notes
Network	#2	from Table 5.8
Packet error probability, P_e	0.01	from [19-21, 24]
Network Availability	"MCI", "AT&T" and 100%	from Table 5.1
CRL issuance interval, T_{CRL}	2 hours	
CRL update window size, W_U	3	
Delta CRL length, L_Δ	2 packets	calculated from (3.4)
Recent request threshold, $R_{Threshold}$	5...50 requests	various increments
Recent requests time window, T_{RRW}	50 sec	

The results are shown in Figure 5.6. The T_{sP} curve exhibits a minimum whereas all the other plots increase with $R_{Threshold}$. The shape of the T_{sP} curve is due to the fact that as the CRL request threshold rate, $R_{Threshold}/T_{RRW}$, decreases, the probability of the Repository initiating a Delta CRL multicast increases thus driving up the total $packet \cdot link$ cost for the system. With a large number of Delta CRL multicast repetitions, the probability of the RPs receiving a complete copy of the Delta CRL increases thus reducing the number of unicast requests as shown in the T_{servP}, T_{servR}, T_{servDC} and T_{servBC} plots of Figure 5.6. A high $R_{Threshold}/T_{RRW}$ rate setting can also drive up the T_{sP} cost by reducing the likelihood of multicasting thereby reducing the

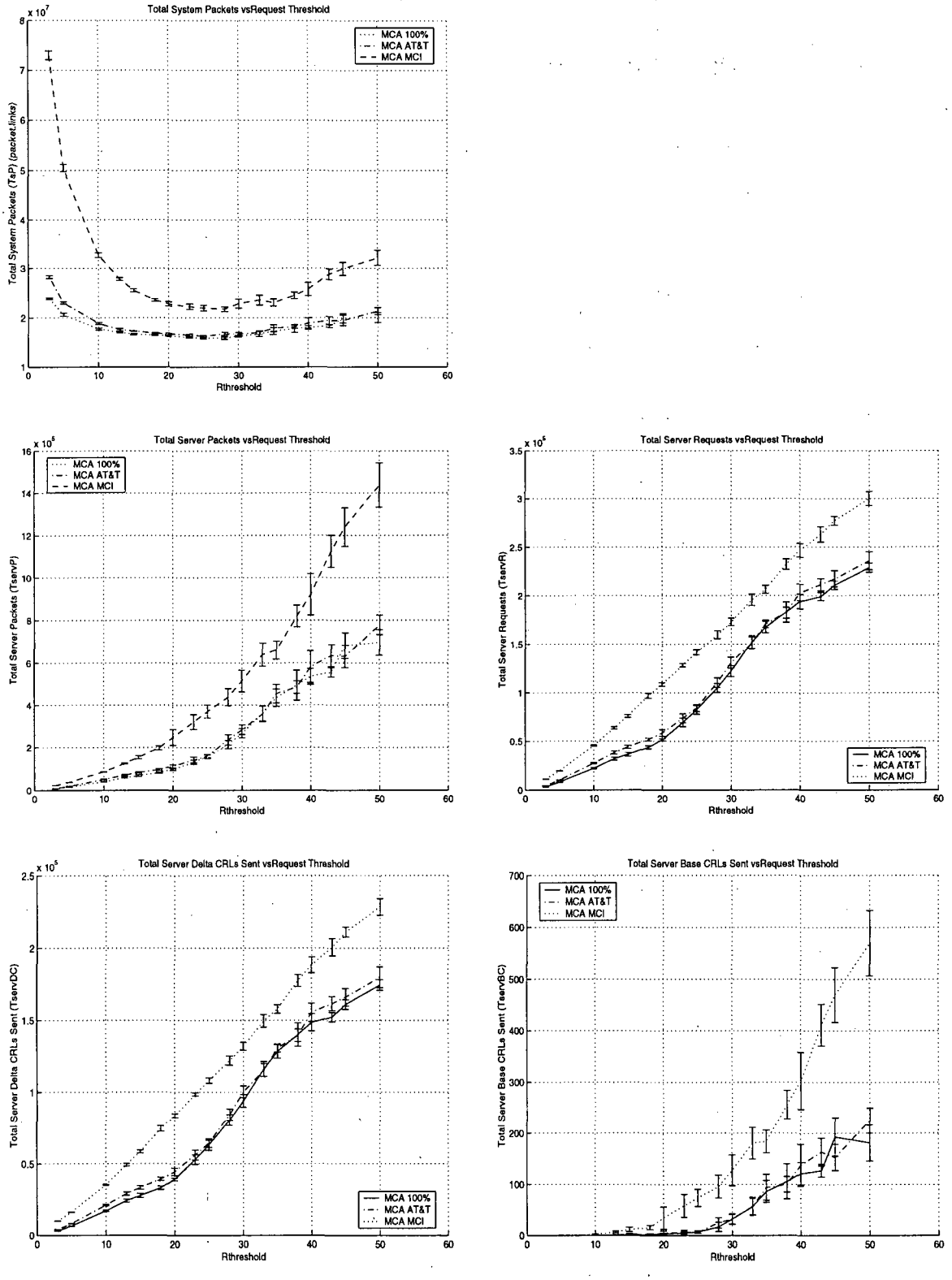

Figure 5.6 Effects of varying $R_{Threshold}$ (T_{RRW} = 50)

probability of the RPs successfully receiving the Delta CRL and thus requiring more unicast CRL updates. The optimal $R_{Threshold}$ value for T_{RRW} = 50 sec found through simulation is 28 requests. This $R_{Threshold}$ value agrees with the optimal rate found by using (3.31) and (3.37) as shown Figure 5.7. The optimal $R_{Threshold}/T_{RRW}$ rate also depends on W_U. By varying W_U

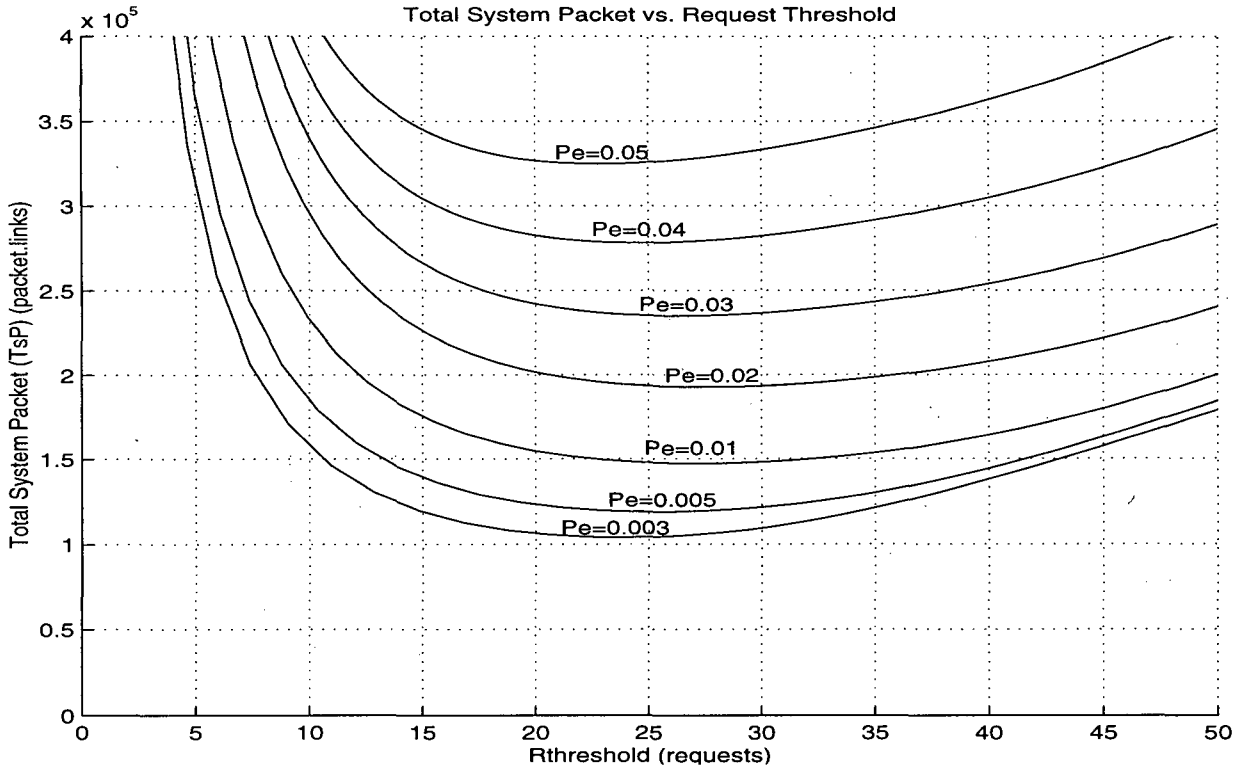

Figure 5.7 T_{sP} vs. $R_{Threshold}$ (T_{RRW} = 50)

and the multicast repetitions variable, r, the optimal $R_{Threshold}/T_{RRW}$ rates for minimizing the T_{sP} can be estimated. A plot of the theoretical optimal $R_{Threshold}$ for minimizing T_{sP} with respect to W_U and P_e is shown in Figure 5.8.

The recent request time window, T_{RRW}, in this study is set to 50 seconds. This window

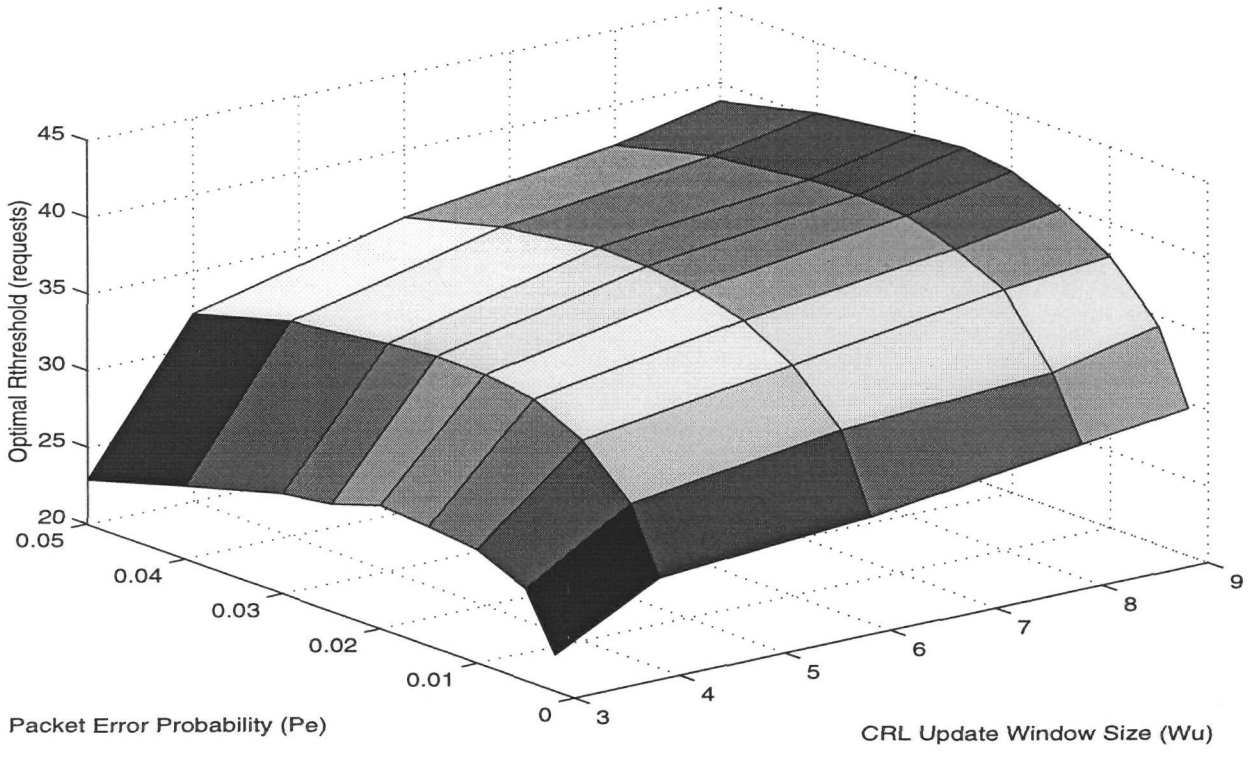

Figure 5.8 Optimal $R_{Threshold}$ as a function of W_U and P_e (T_{RRW} = 50)

size was found through simulation to minimize the T_{sP} for the parameters listed in Table 5.5 with the exception of the $R_{Threshold}$ and T_{RRW}. Figure 5.9 shows T_{sP} as a function of T_{RRW} and the request threshold rate, $R_{Threshold}/T_{RRW}$ obtained using simulation. The plot shows T_{sP} is quite sensitive to changes in T_{RRW}. A difference between the simulation model and the analytical model is that the latter assumes instantaneous and accurate measurement of the CRL request rate whereas the former implements an averaging window to estimate the CRL request rate. Note that the T_{RRW} = 50 seconds setting may not yield the minimal T_{sP} in all cases.

5.5 Network Availability

The performances of the NMC, MCP and MCA systems in a link failure prone network

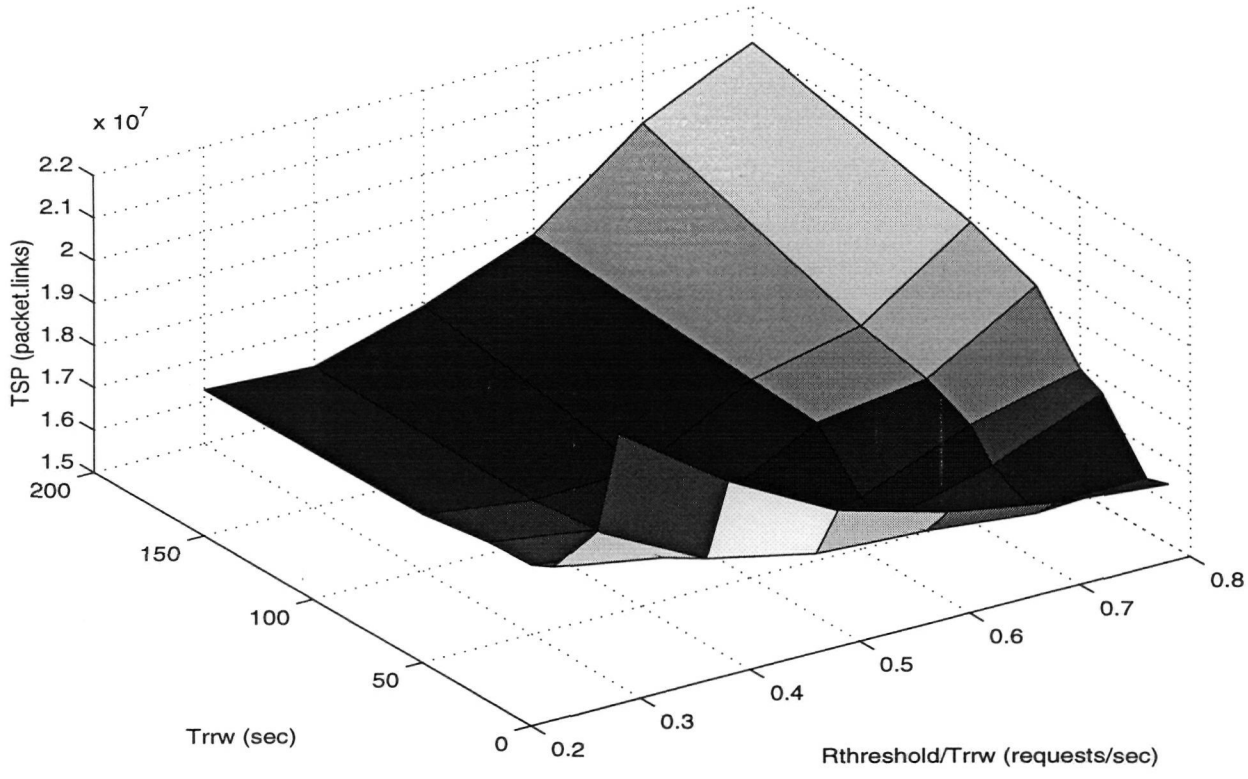

Figure 5.9 T_{sP} as a function of T_{RRW} and $R_{Threshold}/T_{RRW}$ (P_e = 0.01 , 100% network availability)

are studied in this section. A lower network availability reduces the number of RP's that are reachable by the Repository at any point in time thereby reducing the effectiveness of the Delta CRL multicasts. The parameters are shown in Table 5.6.

Table 5.6 Effects of network availability parameters

Parameter	**Value**	**Additional Notes**
Network	#2	from Table 5.8
Packet error probability, P_e	0.01	from [19-21, 24]
Network Availability	"MCI", "AT&T" and 100%	from Table 5.1
CRL issuance interval, T_{CRL}	2 hours	
CRL update window size, W_U	MCA: 3 MCP: 6 NMC: 9	

Table 5.6 Effects of network availability parameters

Parameter	Value	Additional Notes
Delta CRL length, L_{Δ}	MCA: 2 packets MCP: 4 packets NMC: 6 packets	calculated from (3.4)
Request rate threshold, $R_{Threshold}/T_{RRW}$	30 requests/50sec	

With the NMC system, when a RP has a validation request and needs a CRL update, it sends a CRL request to the Repository. Should the Repository be unreachable due to a failure of one or more links between the requesting RP and the Repository, the RP will time-out (Section 4.2.6.1) and re-request for the CRL update at a later time to complete the validation request. If the failed link is a link that connects a large number of RPs to the Repository and if that link has been down for a long period of time, there will be a large number of pent-up CRL update requests as shown by the lighter shaded areas of Figure 5.10. When that link comes back on-line, there may be a large number of RPs requesting CRL updates as shown by the darker shaded area of Figure 5.10. This peak in CRL requests may be much higher than the peak request rate at the beginning of the CRL issuance period if λ_{RR} is greater than λ_{Val}, the duration of the link downtime is long and the number of dependent RPs is large.

The Delta CRL request rate plots for the MCP and NMC systems are shown in Figure 5.11 for 4 CRL issuance intervals. At each issuance interval, a single multicast, as shown by the arrow, is sent out by the Repository. Evident in the plot is the fact that the success of the multicast varies greatly from interval to interval where the success of the multicasts are ordered: 4,1,2,3. The success of the multicast depends on the number of RPs that are reachable at the time of the multicast. If a large number of RPs are not reachable at the time of the Delta CRL multicast, then a large number of CRL requests can be expected during the remainder of the CRL issuance

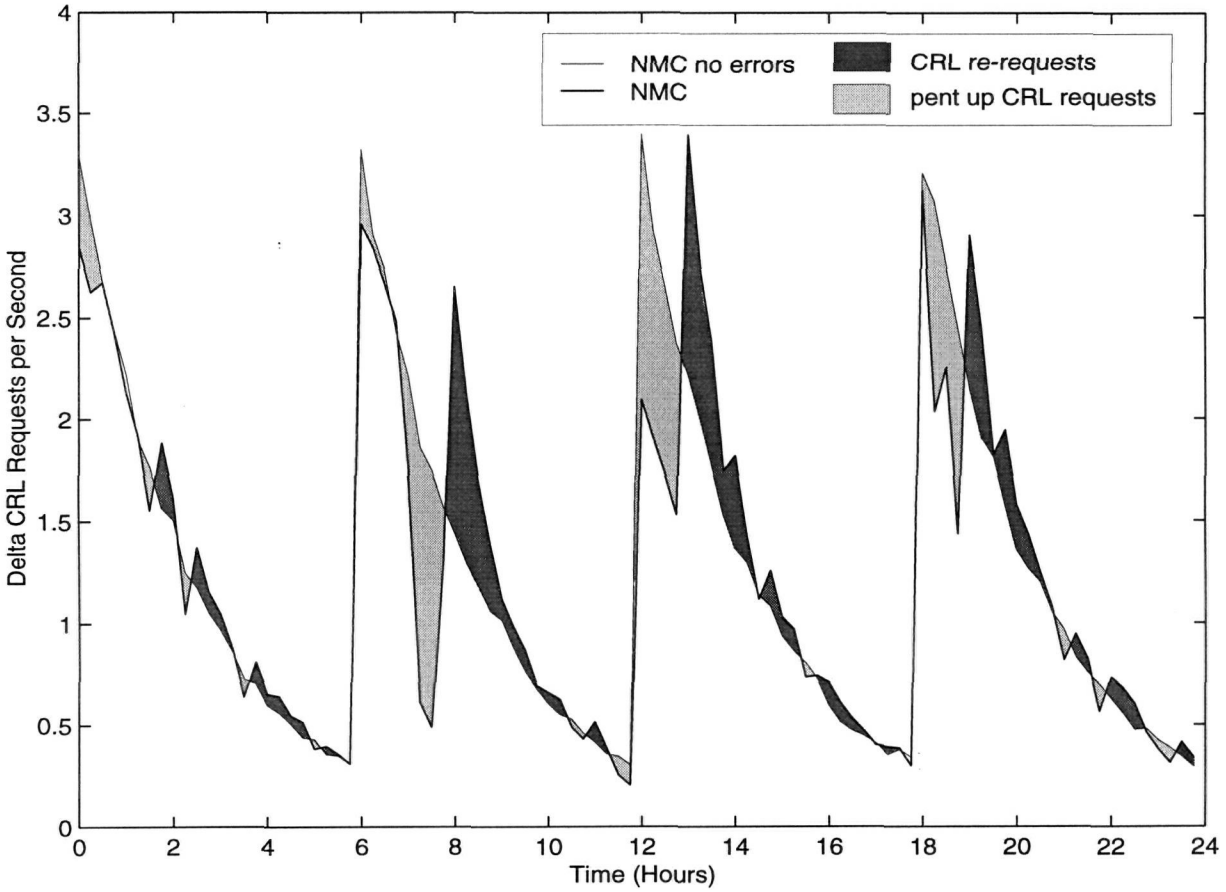

Figure 5.10 Time plot for NMC 100% vs. MCI network availability

interval.

The MCA system uses multicast repetition to reduce the CRL update request rates. This mechanism is suitably applied to reducing peak request rates due to the restoration of a highly depended upon link that has been down for an extended time period. The Delta CRL request rate plots for the MCA and the NMC system are shown in Figure 5.12. The height of the arrows indicate the number of multicasts sent out by the Repository during the time slot recorded by the `Traffic` class. Each `TIMESLOT` is 240 sec. As shown, the MCA system is effective in mitigating high peaks. Shown in CRL issuance interval 2 of Figure 5.12 at about the 13th hour, one or more highly depended upon links have been restored thus causing a high number of Delta CRL

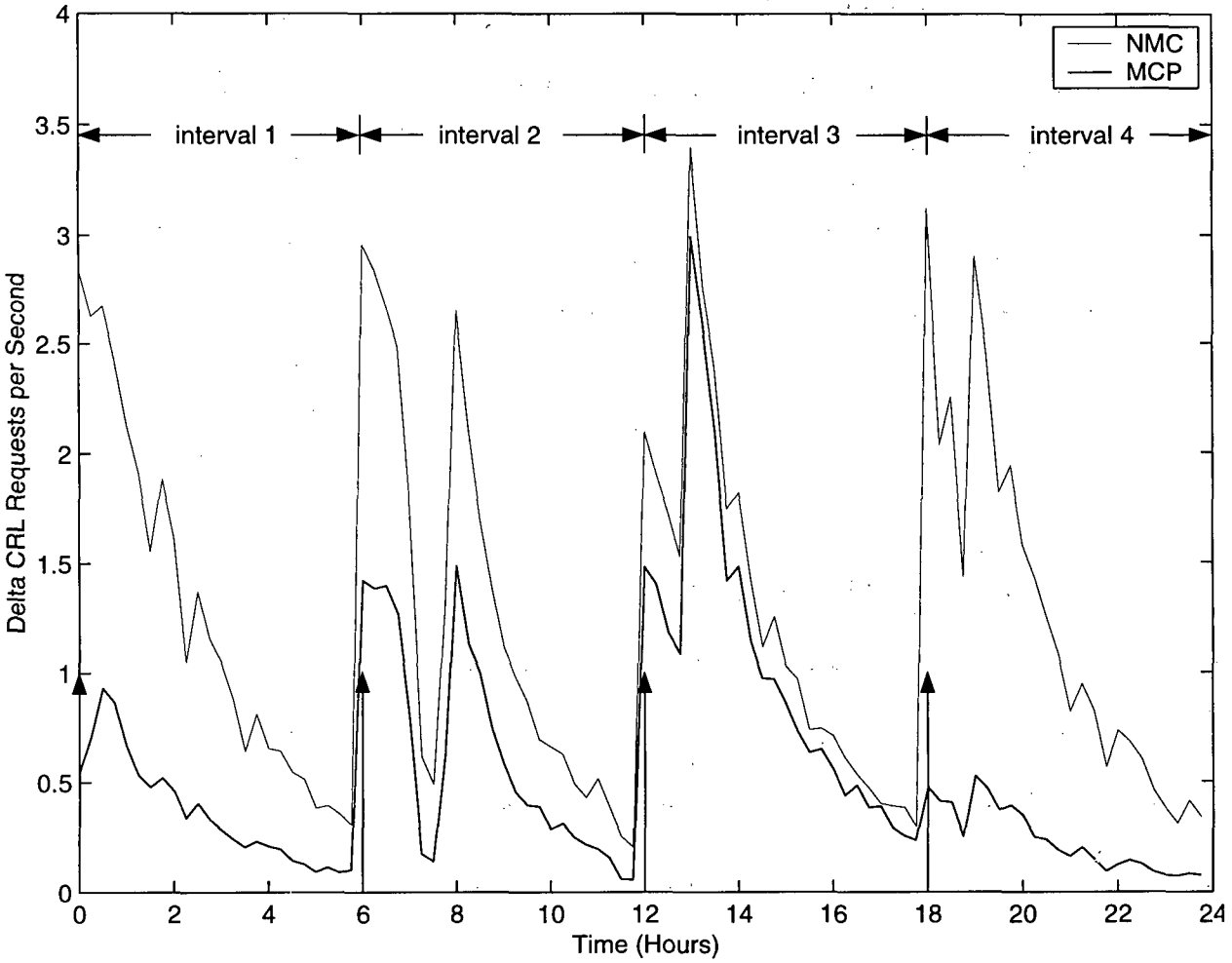

Figure 5.11 Time plot for NMC vs. MCP with "MCI" network availability

requests in the NMC system. The increase in CRL requests triggered the Repository to send out a total of 3 multicasts to mitigate the potentially high number of CRL requests.

A low network availability reduces the effectiveness of multicasts by making segments of the spanning tree unreachable at the time of the multicasts. Therefore more multicasts are needed in the case of the MCA system and more unicasts are needed in the case of the MCP system thus increasing CRL distribution costs. Shown in the T_{sP} plots in Figures 5.3, 5.4 and 5.5, the costs of the MCA system in increasing order are: 100%, "AT&T" and "MCI" network availability. In the

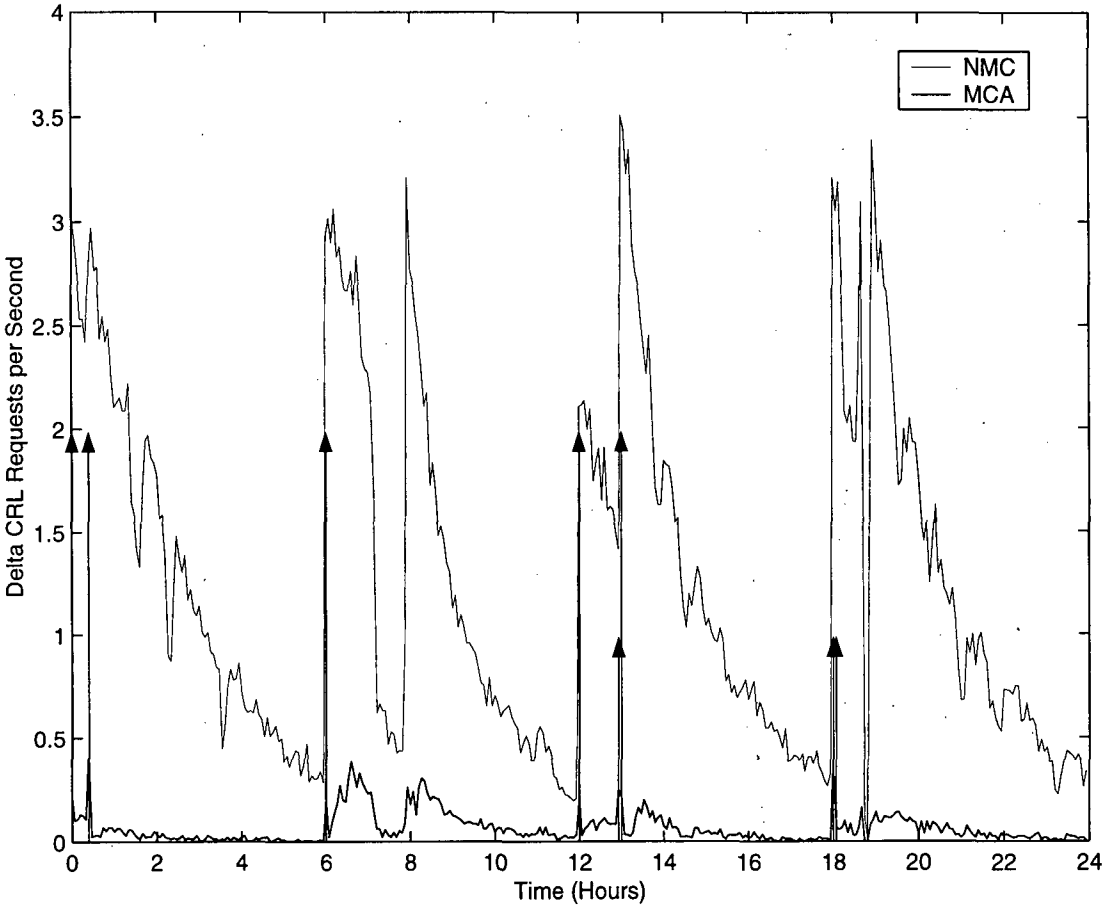

Figure 5.12 Time plot for NMC vs. MCA with "MCI" network availability

case of the NMC system, a low network availability does not increase the total cost of packet transmission but rather displaces it to a later time. The T_{servDC} plots (Figures 5.3, 5.4 and 5.5) for the NMC system with "MCI" network availability is lower than that for the 100% and "AT&T" network availability. This is because pent-up validation requests are more likely to accumulate at disconnected RPs in networks with low network availability. Once the failed link is restored, the RP only needs to make one CRL request to update its local CRL and complete all of its accumulated validation requests.

5.6 Effects of Packet Error Probability P_e

Each packet that is transmitted over a network link is assumed to have a packet error probability, P_e. This section compares the effect of varying P_e in the NMC, MCP and MCA systems. The parameters used in this section are shown in

Table 5.7 Effects of P_e parameters

Parameter	Value	Additional Notes
Network	#2	from Table 5.8
Packet error probability, P_e	0.001...0.1	various increments
Network Availability	"MCI", "AT&T" and 100%	from Table 5.1
CRL issuance interval, T_{CRL}	2 hours	
CRL update window size, W_U	MCA: 3,4,6 MCP: 6,8,9 NMC: 9	
Delta CRL length, L_Δ	MCA: 2,3,4 packets MCP: 4,8,9 packets NMC: 6 packets	calculated from (3.4)
Request rate threshold, $R_{Threshold}/T_{RRW}$	30 requests/50sec	

The results for variations in P_e are shown in Figures 5.13, 5.14 and 5.15. The T_{sP} and T_{servP} results for the MCA case are lower than for the NMC and MCP systems. The plots for the MCP system show that as P_e increases, the MCP $W_U = 9$ curve approaches that of the NMC $W_U = 9$ system. The T_{sP} and T_{servP} curves for the MCP system with W_U size less than the W_U size for the NMC case eventually exceeds that of the NMC case as P_e increases. This is because as P_e increases, the probability of the RPs receiving a complete multicasted Delta CRL decreases. The results for both the MCA and MCP systems show that as the P_e increases, a larger

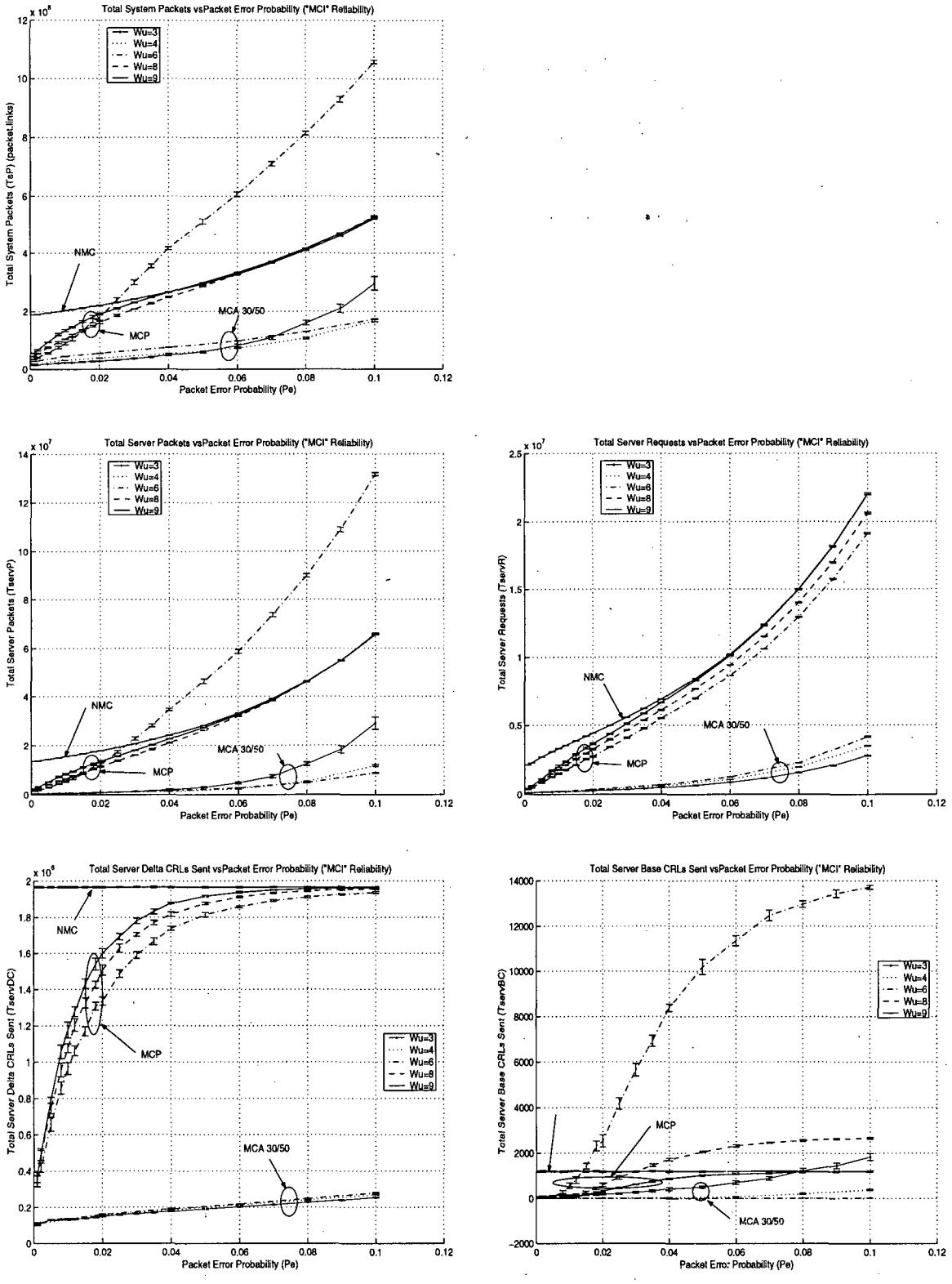

Figure 5.13 Effects of P_e ("MCI" network availability)

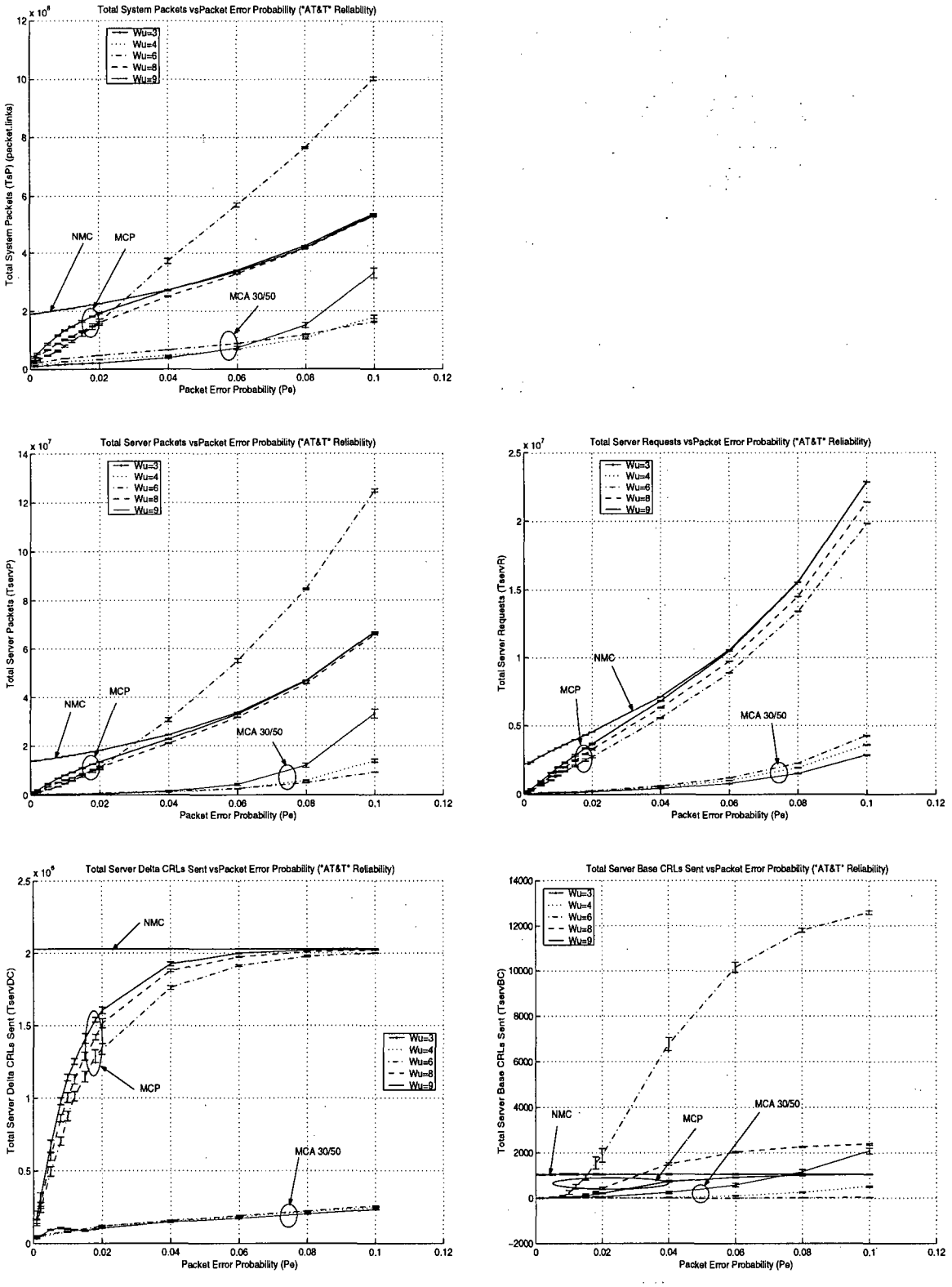

Figure 5.14 Effects of P_e ("AT&T" network availability)

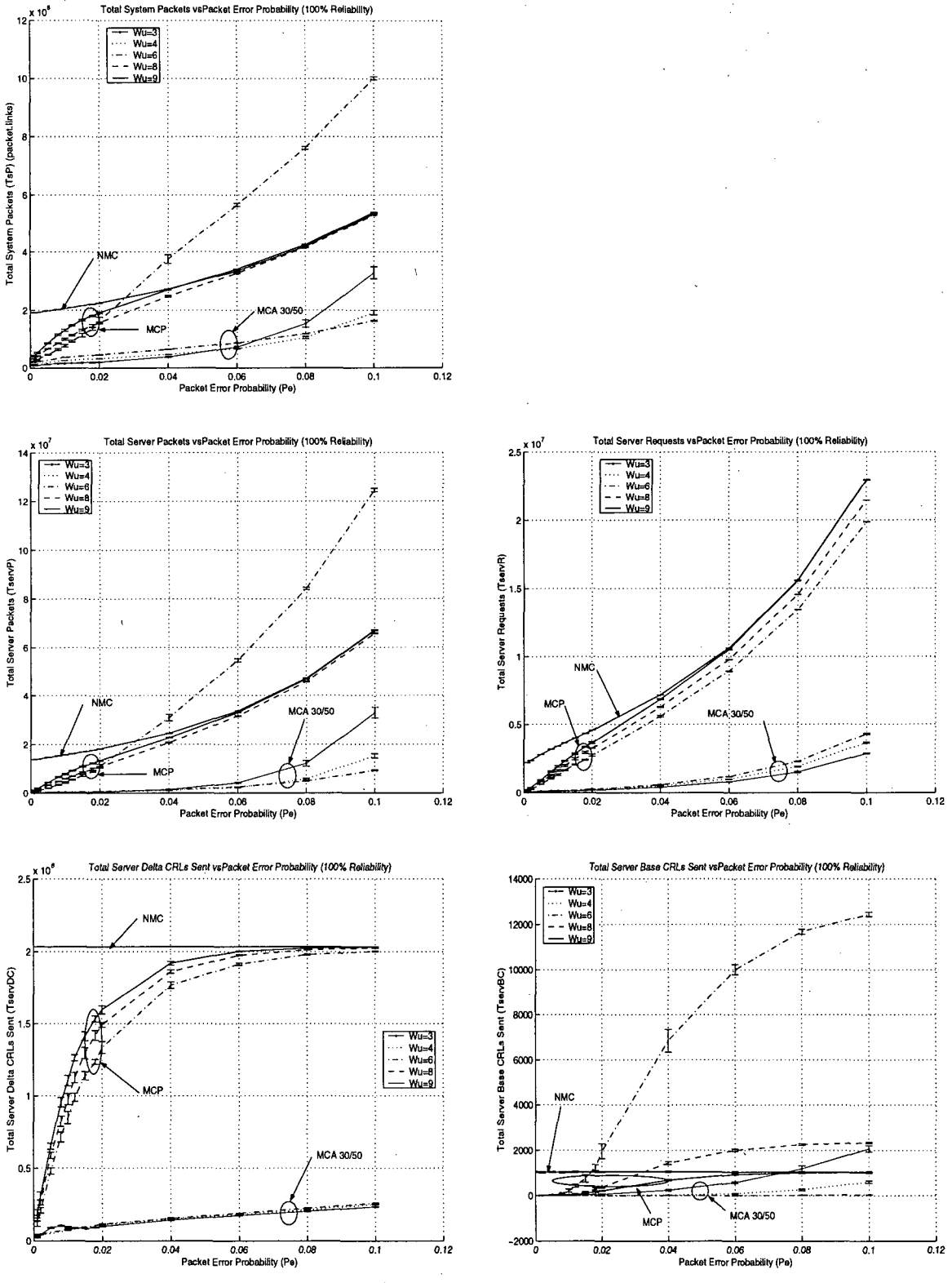

Figure 5.15 Effects of P_e (100% network availability)

W_U is needed to reduce the probability of a Base CRL request as shown in the Base CRL request plots in Figures 5.13, 5.14 and 5.15. A larger W_U helps to reduce Base CRL requests by allowing for a greater number of misses in receiving a complete Delta CRL through multicasting prior to the current CRL. However the larger Delta CRL resulting from the larger W_U decreases the probability of a RP successfully receiving a Delta CRL through multicasting, and thereby limiting the benefits of increasing the W_U size.

5.7 Effects of Repository Location and Network Topology

The previous analyses were all based on the same network topology and the same Repository location. In this section, the network topology and the Repository location are varied and the performances are compared. Five additional network topologies were generated in addition to that used in the previous analysis for a total of 6 network topologies. The network topologies have been generated using the parameters in Table 5.8 with the Tiers program.

Table 5.8 Network topology parameters

Net#	0	1	2	3	4	5
Links	46039	46039	33459	33459	34204	34204
Nodes	46040	46040	33460	33460	34205	34205
RPs	29999	29999	29999	29999	29999	29999
N_W	1	1	1	1	1	1
N_M	200	200	60	60	60	60
N_L	30	30	50	50	50	50
S_W	40	40	40	40	5	5
S_M	50	50	7	7	20	20
S_L	6	6	11	11	11	11

Ten Repository locations for each of the 6 network topologies were chosen at random and sorted according to the average distance bewteen the Repository and the RPs. Each of the 10 Repository locations in each of the 6 network topologies were simulated with the parameters shown in Table 5.9.

Table 5.9 Effects of Repository location and network topology parameters

Parameter	Value	Additional Notes
Network	#1-6	from Table 5.8
Packet error probability, P_e	0.01	from [19-21, 24]
Network Availability	"MCI"	from Table 5.1
CRL issuance interval, T_{CRL}	2 hours	
CRL update window size, W_U	MCA: 3 MCP: 6,8,9 NMC: 9	
Delta CRL length, L_Δ	MCA: 2 packets MCP: 4,5,6 packets NMC: 6 packets	calculated from (3.4)
Request rate threshold, $R_{Threshold}/T_{RRW}$	30 requests/50sec	

The results of the simulations are shown in Figures 5.16 to 5.21 with respect to average RP distance from the Repository. In all 6 networks, the T_{sP} and T_{servP} plots for MCA are significantly better than those for the MCP and NMC systems. The T_{sP}, T_{servP}, T_{servR}, T_{servDC} and T_{servBC} plots for MCA system are less dependent on the average RP distance compared to NMC and MCP. The main reason for this is that most RPs are able to update their local CRLs with the multicasted Delta CRLs without needing to make a CRL request. The maximum cost of a multicast of a single packet is equal to the number of links in the spanning tree. For any given network topology, the number of links in the multicast spanning tree is the same regardless of

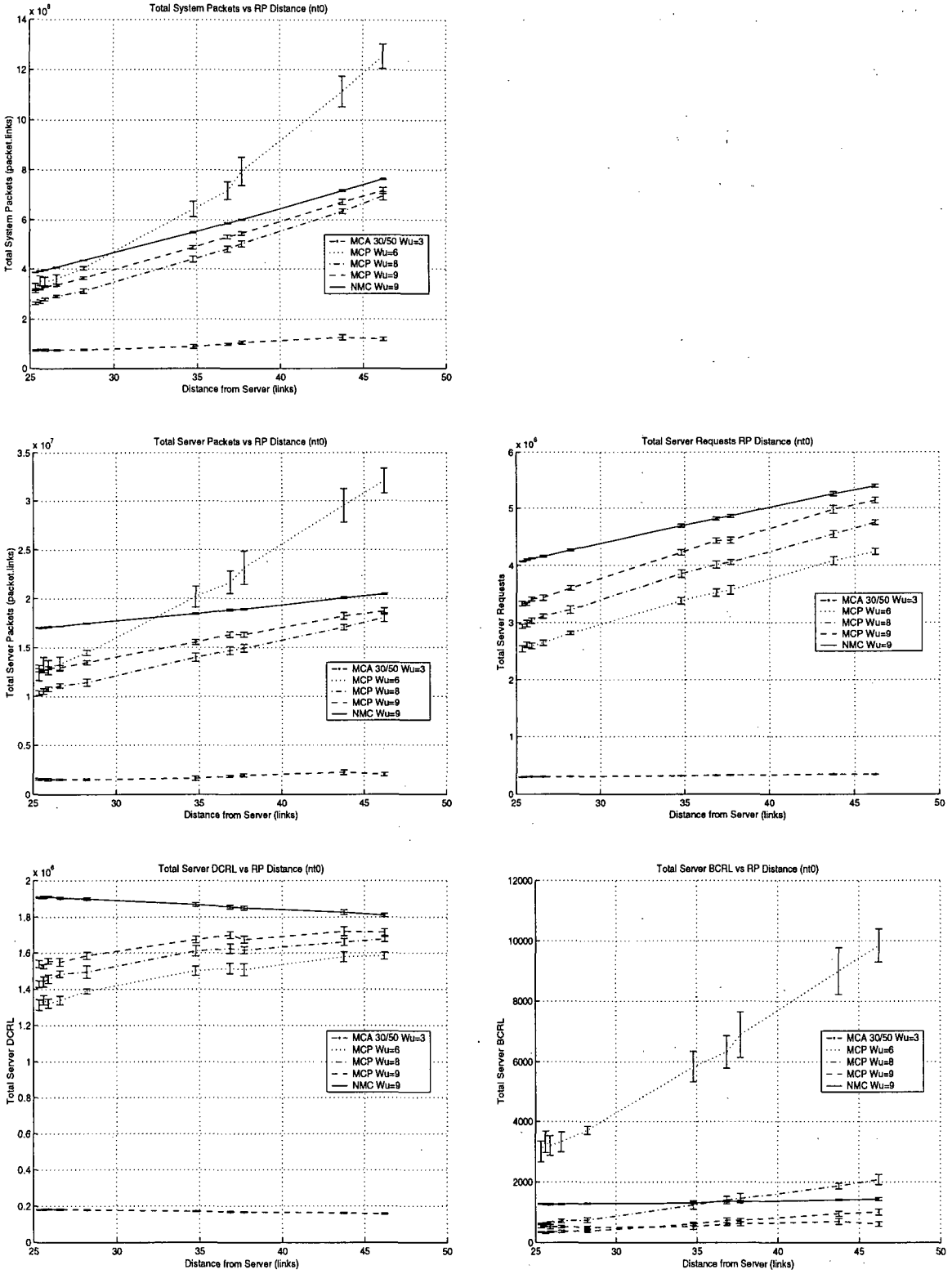

Figure 5.16 Effects of Repository location (Net0)

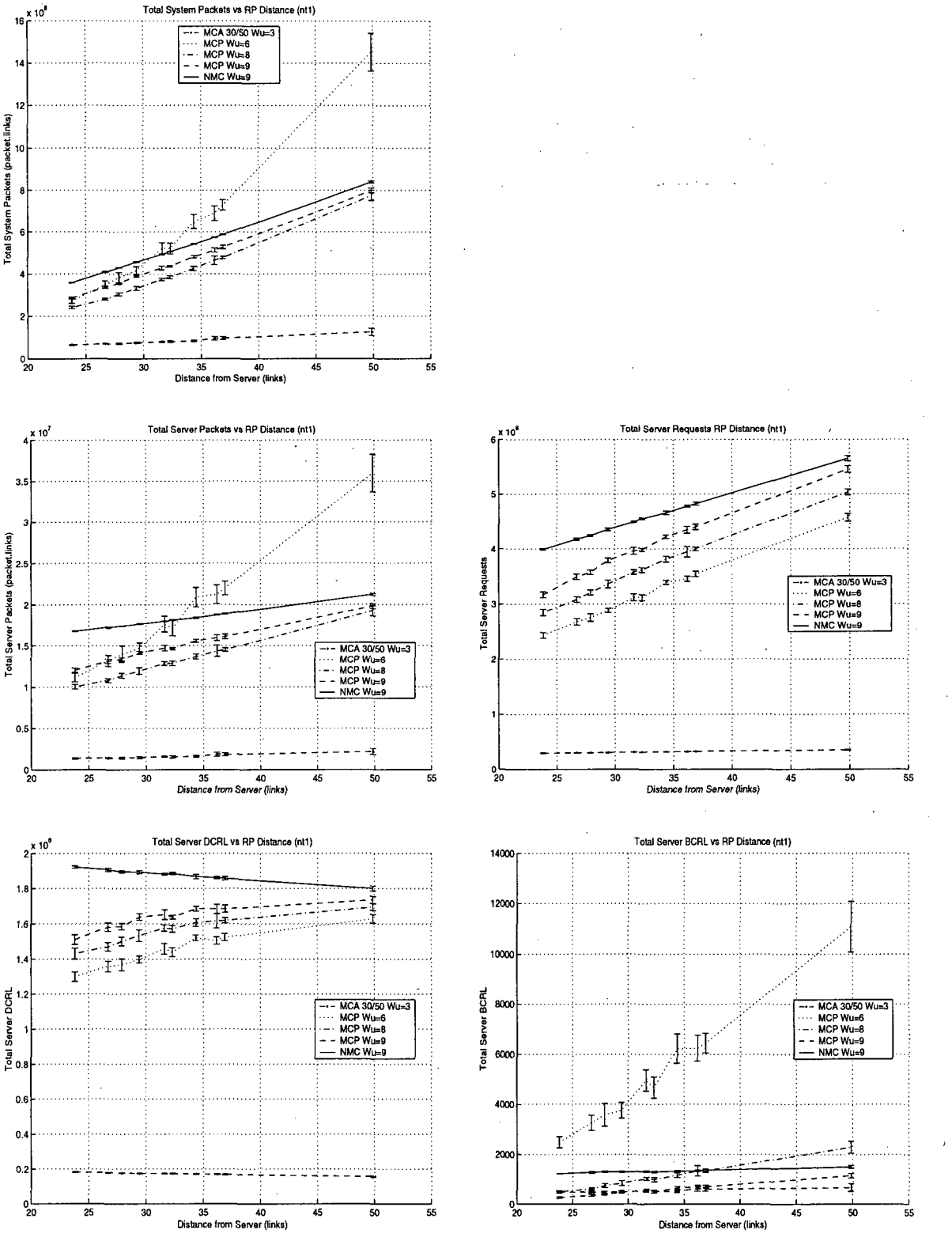

Figure 5.17 Effects of Repository location (Net1)

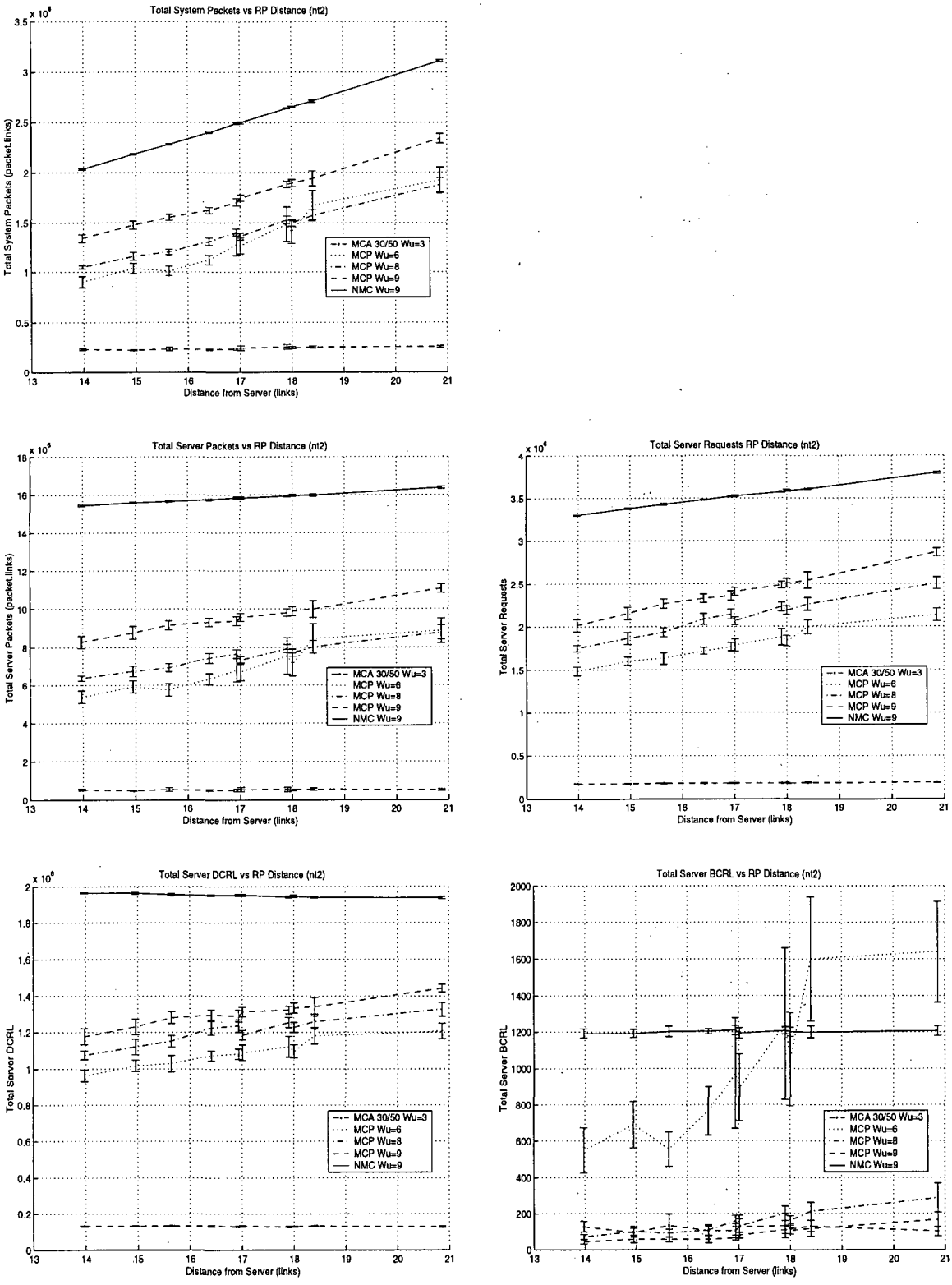

Figure 5.18 Effects of Repository location (Net2)

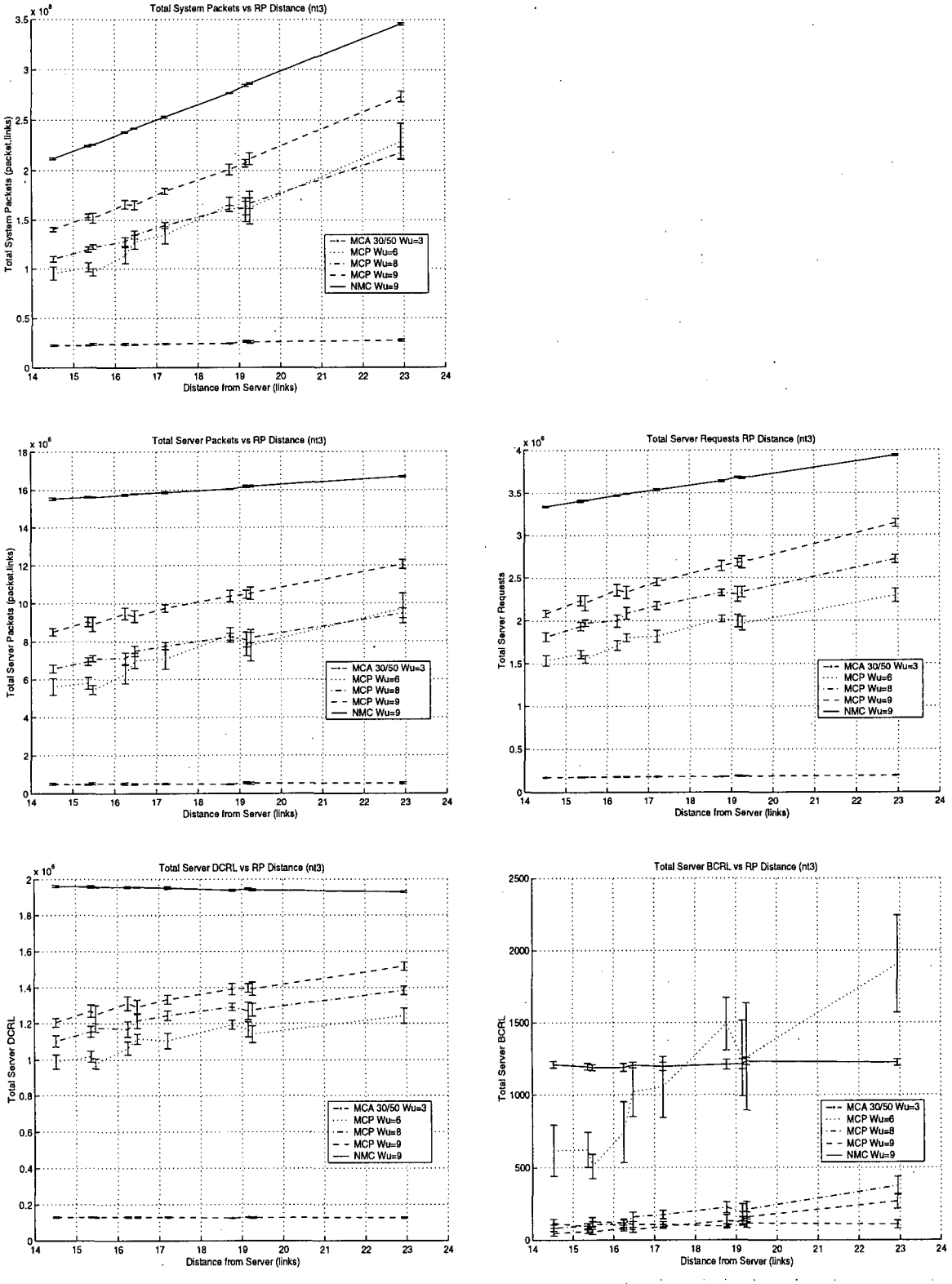

Figure 5.19 Effects of Repository location (Net3)

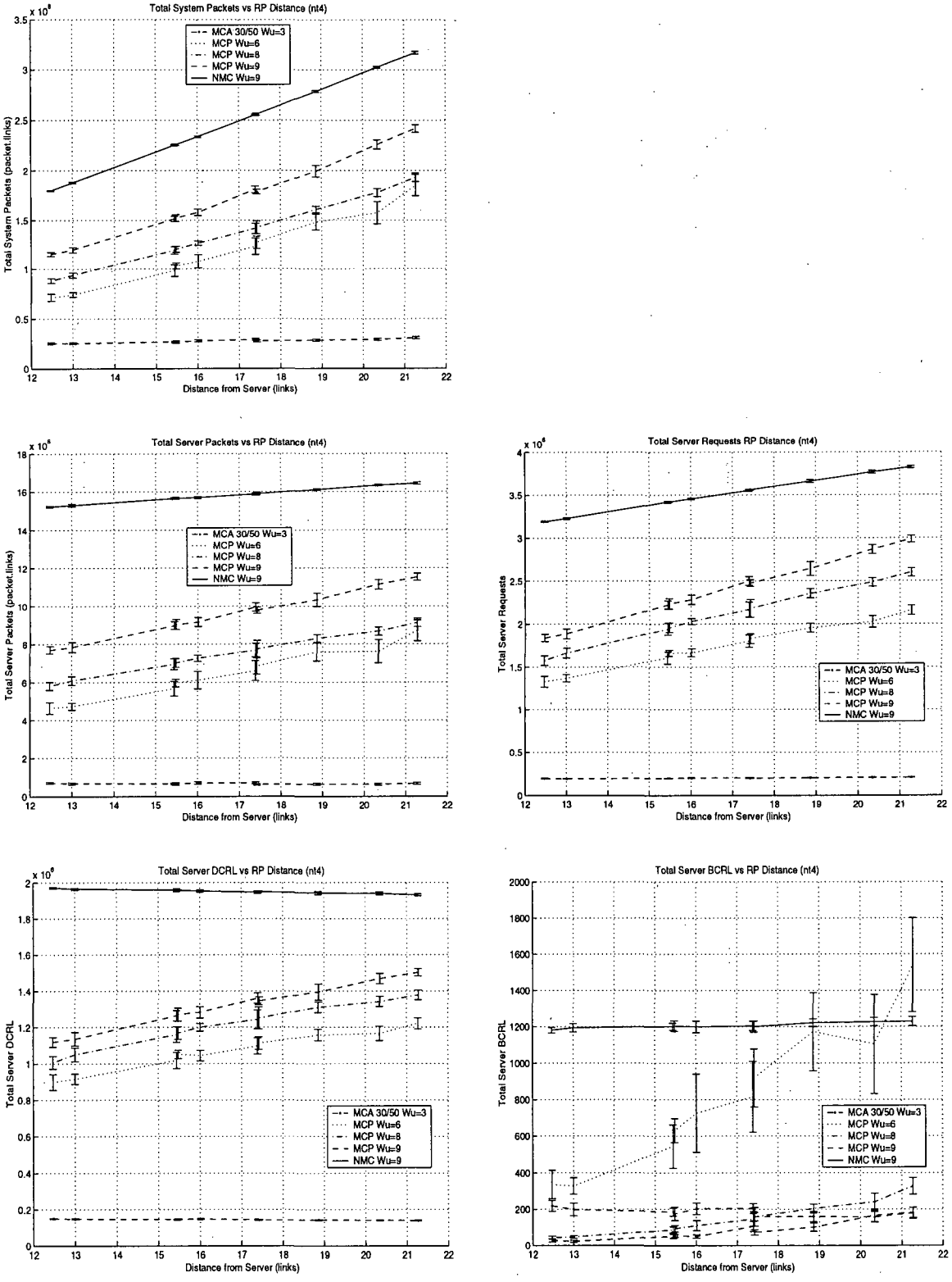

Figure 5.20 Effects of Repository location (Net4)

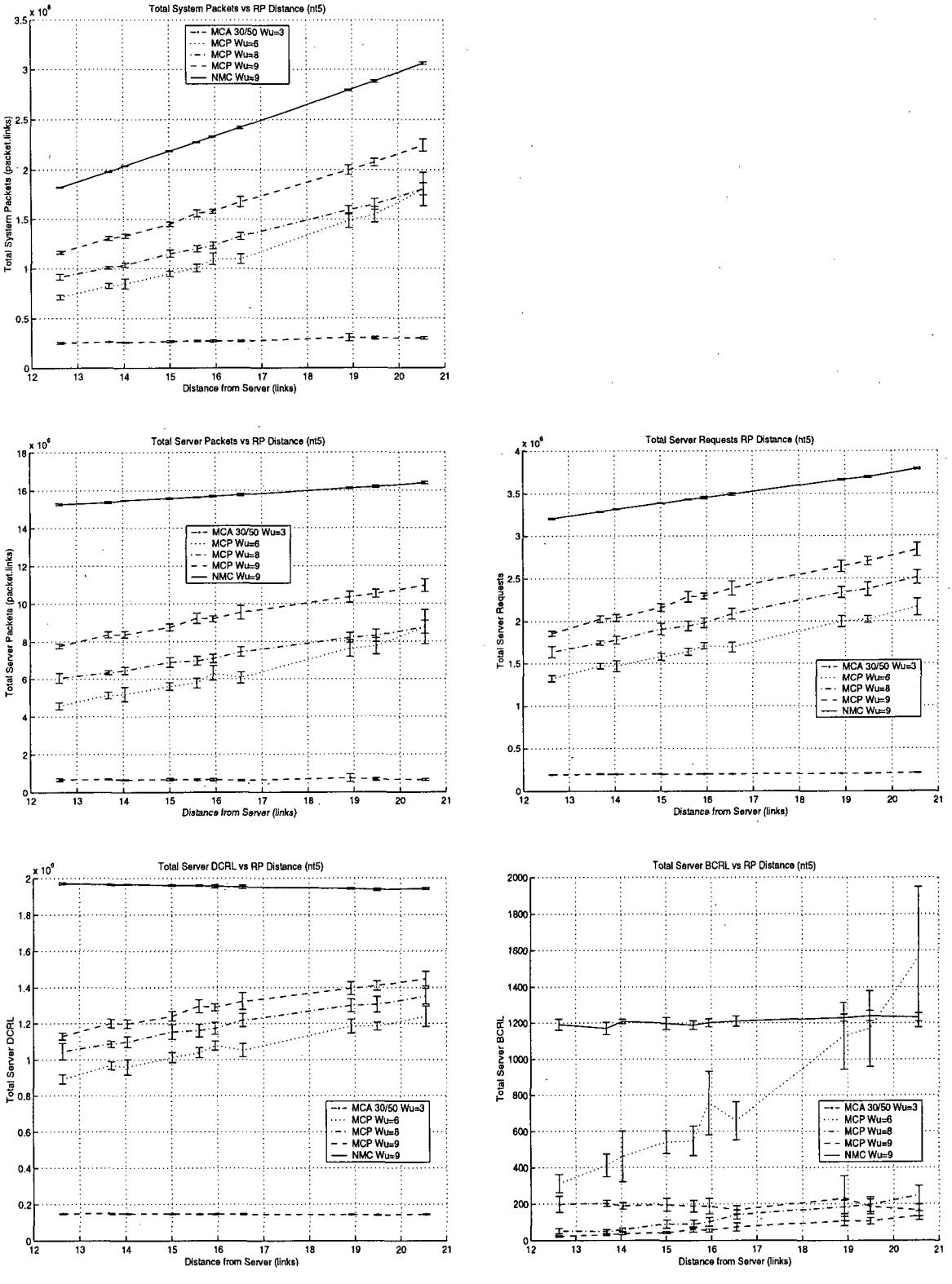

Figure 5.21 Effects of Repository location (Net5)

where the Repository is located and so the cost is the same for 100% success. In the NMC and MCP systems, the dominant T_{sP} and T_{servP} costs are from unicasting where the distances between the Repository and RPs affect the $packet \cdot link$ cost significantly. For the MCP system with a CRL update window that is smaller than that of the NMC system, the T_{sP} and T_{servP} costs exceed that of the optimal value for the NMC systems as the average distance between the Repository and the RP increases. According to the expression for $P_{MCc}(L_\Delta, d, r)$ (3.15), increasing P_e, d, and L_Δ will all decrease the likelihood of a RP receiving the complete Delta CRL multicast. For the MCP systems, the choice of W_U depends on the distance of the RPs from the Repository. The results due to an overly small W_U can be seen in the increasing number of Base CRL requests as the average distance increases. However the optimal value of W_U value should not exceed that for NMC. Note also that the results for MCP are much more variable than for NMC and MCA. This is because the probability of success of a single multicast is highly variable with low network availability and different multicast spanning trees as noted in Section 5.5.

Chapter 6 Conclusions

The main findings and contributions of this thesis are summarized in this chapter. Some areas for further research are also suggested.

6.1 Summary and Contributions

A system was proposed for cost effective distribution of CRLs using a combination of multicasting and unicasting. An analytical model and a simulation model were used in this study. In contrast to almost all previous studies, the analytical model and the simulation model take into account the network topology and link packet errors to estimate the network communication cost of operating each system. In addition, the simulation model takes network link failures into account to compare the robustness of the systems.

The proposed MCA system for distribution of CRLs in a large scale PKI which combines both multicasting and unicasting of CRLs was shown to require significantly less network bandwidth than NMC and MCP. For various network link packet error probabilities and network link failure probabilities, the best results were obtained with MCA, followed by MCP and NMC. In networks with link failures, MCA can reduce CRL request peaks due to pent-up CRL requests caused by long primary network links failures. MCP has a greater performance variability than NMC and MCA for different network packet error and link failure probabilities. Results show that MCP only performs well in networks with low packet error probabilities. At high packet error probabilities, the performance of MCP is close to that of NMC. The results obtained with different Repository locations in a given network shows that the performance of MCA is less sensitive to average link distance between the Repository and the RPs than those of MCP and NMC.

The proposed MCA system may be retrofitted to legacy client programs which may only obtain its CRL using unicasting and cache its copy of the CRL in a file that is accessible for reading and writing by other programs.

6.2 Future Work

Further improvements to the system for the distribution of CRLs proposed in this thesis and refinement of the analytical model are suitable topics for future work.

Additional reductions in network bandwidth requirements may be achieved in two ways. One is to use Forward Error Correction (FEC) [19-21] with the multicast Delta CRLs to reduce packet loss. Another way is to combine the RP's multicast receiving module with the client program such that the RP needs to request for the missing packets of the Delta CRL received by the multicast receiving module only if the RP can use a Delta CRL to update its local CRL to complete a validation request.

The analytical model for the CRL distribution can only be used for networks with 100% availability. This allows all multicast repetitions in the MCA system to occur at the beginning of the CRL issuance interval. A fixed number, r, of multicast repetitions was used to model the MCA system. However, results in Section 3.3.2 indicate that the random variables in the sequence $\boldsymbol{r} = (r_1, r_2, r_3, \ldots)$ are independent. Finding an expression to model the number of multicast repetitions, r, would yield more accurate results. An expression for finding the optimal CRL update window in the NMC system for reducing network bandwidth requirements is provided in [7]. The optimal CRL update window sizes for MCA and MCP were found through simulation. Expressions for the optimal CRL update window sizes for MCA and MCP have not yet been

obtained. Further refinement to the analytical model could account for network availability parameters and redundant network links.

Glossary

This section provides a list of acronyms used in this thesis.

CA	Certificate Authority
CH	Certificate Holder
CPS	Certificate Practice Statement
CRL	Certificate Revocation List
CRS	Certificate Revocation Status
CRT	Certificate Revocation Tree
ISP	Internet Service Provider
LAN	Local Area Network
MAN	Metropolitan Area Network
MCA	Multicasting Aperiodic
MCP	Multicasting Periodic
NMC	No Multicasting
OCSP	On-line Certificate Status Protocol
PKI	Public Key Infrastructure
RP	Relying Party
SLA	Service Level Agreement
WAN	Wide Area Network

List of Symbols

List of symbols used in this thesis.

λ_{RR}	CRL re-request rate at a RP
λ_{Val}	Validation request arrival rate at a RP
C_{MC}	$packet \cdot link$ cost of a single Delta CRL multicast
c_{UC}	$packet \cdot link$ cost of unicasting to a RP for one CRL request
C_{UC}	Total $packet \cdot link$ cost of unicasting to all RPs
$D_{\text{Link } Max}$	Link distance of the furthest network link
$D_{RP\ Max}$	Link distance of the furthest RP
L_B	Length of the Base CRL in packets
L_Δ	Length of the Delta CRL in packets
L_h	Length of the CRL header in bytes
L_{packet}	Length of each packet in bytes
L_r	Length of each certificate revocation record in bytes
N_{Cert}	Total number of unexpired certificates
n_{rp}	Number of RPs that have yet to request for a CRL update
N_{RP}	Total number of RPs
n_{sr}	Average number of requests received by the Repository from a RP for the same CRL until the CRL is received correctly. Pertains to the Repository only.
n_{stx}	Average number of packets sent by the Repository until a RP receives the packet correctly. Pertains to the Repository only.
n_{tx}	Average $packet \cdot link$ cost of successfully sending a packet to a RP. Pertains to T_{sP}.
P_Δ	Probability of a RP requesting for a Delta CRL

P_B	Probability of a RP requesting for a Base CRL
P_c	Probability of a packet received correctly when transmitted over one link
P_e	Probability of a packet received in error when transmitted over one link
P_s	Probability of a packet received correctly when transmitted over multiple links
P_f	Probability of a packet received in error when transmitted over multiple links
P_{MCc}	Probability of a multicast received correctly by a RP
P_{MCe}	Probability of a multicast received in error by a RP
P_{NVal}	Probability of no validation request
P_{NU}	Probability of not being able to use a Delta CRL
P_{Val}	Probability of one or more validation requests
P_{Revoke}	Probability of a certificate being revoked during its lifetime
r, r_i	Number of multicast repetitions
$\boldsymbol{r}$	Sequence of number of multicast repetitions
R_{Recent}	Number of recent (new) CRL requests
$R_{Threshold}$	Threshold request number
R_{servBC}	Rate of Repository Base CRLs sent
R_{servDC}	Rate of Repository Delta CRLs sent
R_{servSP}	Repository out going packet rate
R_{servR}	Repository CRL request rate
R_{Val}	Rate at which a RP make its first validation request
T_{Cert}	Certificate lifetime
T_{CRL}	CRL issuance interval
T_{RRW}	Recent requests time window

T_{servBC}	Total number of Base CRLs sent by the Repository
T_{servDC}	Total number of Delta CRLs sent by the Repository
T_{servP}	Total number of packets sent by the Repository
T_{servR}	Total number of requests received by the Repository
T_{sP}	Total number of packets sent over network links
W_U	CRL update window

References

[1] S. Berkovits, S. Chokhani, J. A. Furlong, J. A. Geiter and J. C. Guild, *Public Key Infrastructure Study: Final Report*, produced by the MITRE Corporation for NIST, April 1994.

[2] S. Chokhani, "Towards a National Public Key Infrastructure," *IEEE Communications Magazine*, vol. 32(9), pp. 70-74, September 1994.

[3] D. G. Masses and A. D. Fernandes, "Economic Modelling and Risk Management in Public Key Infrastructure: The Business Case for a Broadly-based Highly Scalable Public Key Infrastructure," *RSA Data Security Inc. Annual Symposium*, San Francisco, CA, USA, January 1997.

[4] S. Mendes and C. Huitema, "A New Approach to the X.509 Framework: Allowing a Global Authentication Infrastructure without a Global Trust Model," *In Proceedings of the 1995 Internet Society Symposium on Network and Distributed System Security*, pp. 172-189, February 1995.

[5] B. Schneier, *Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, 1996.

[6] D. A. Cooper, "A Model of Certificate Revocation," *In Proceedings of the Fifteenth Annual Computer Security Applications Conference*, pp. 256-264, Phoenix, Arizona, USA, December 1999.

[7] D. A. Cooper, "A More Efficient Use of Delta-CRLs," *In Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pp. 190-202, Oakland, California, USA, May 2000.

[8] M. Naor and K. Nissim, "Certificate Revocation and Certificate Update," *In Proceedings of the 7th USENIX Security Symposium*, San Antonio, Texas, USA, January 1998.

[9] S. Micali, "Efficient Certificate Revocation," Technical Memo MIT/LCS/TM-542b, Massachusetts Institute of Technology, Laboratory for Computer Science, March 1996.

[10] P. Kocher, "A Quick Introduction to Certificate Revocation Trees," ValiCert Inc., `http://www.valicert.com./company/crt.html`.

[11] C. Adams and R. Zuccherato, "A General, Flexible Approach to Certificate Revocation," Entrust Technologies, `http://www.entrust.com/resources/whitepapers.htm`, June 1998.

[12] R. Housley, W. Ford, W. Polk and D. Solo, *Internet X.509 Public Key Infrastructure: Certificate and CRL Profile*, RFC 2459, Internet Engineering Task Force, January 1999.

[13] M. Meyers, R. Ankney, A. Malpani, S. Galperin and C. Adams, *X.509 Internet Public Key Infrastructure: Online Certificate Status Protocol -OCSP*, RFC2560, Internet Engineering Task Force, June 1999.

[14] M. Branchaud, *A Survey of Public-Key Infrastructures, M. S. Thesis*, Department of Computer Science, McGill University, Montreal, Canada, March 1997.

[15] U. Maurer, "Modelling A Public-key Infrastructure," *Fourth European Symposium on Research in Computer Security (ESORICS 96)*, pp. 324-350, Rome, Italy, September 1996.

[16] Y.K. Hsu and S. P. Seymour, "An Intranet Security Framework Based on Short-Lived Certificates," *IEEE Internet Computing*, pp. 73-79, March-April 1998.

[17] J.C.I. Chuang and M.A. Sirbu, "Pricing Multicast Communications: A Cost-Based Approach," *In Proceedings of the INET'98*, Geneva, Switzerland, July 1998.

[18] C. Papadopoulos, G. Parulkar and G. Varghese, "An Error Control Scheme for Large-Scale Multicast Applications," *In Proceedings of IEEE Infocom'98*, San Francisco, CA, USA, March 1998.

[19] P. R. Rodriguez and E. W. Biersack, "Continuous Multicast Push of Web Documents over the Internet," *IEEE Network Magazine*, vol. 12(2), pp 18-31, March-April 1998.

[20] J. Nonnenmacher and E. W. Biersack, "Reliable Multicast: Where to use FEC," In *Proceedings of IFIP 5th International Workshop on Protocols for High Speed Networks (PfHSN'96)*, pp. 134-148, INRIA, Sophia Antipolis, France, October 1996.

[21] J. Nonnenmacher, E. W. Biersack and D. Towsley, "Parity-Based Loss Recovery for Reliable Multicast Transmission," *IEEE/ACM Transactions on Networking*, vol. 6(4), pp. 349-361, August 1998.

[22] M. B. Doar, "A Better Model for Generating Test Networks," *In Proceedings of Globecom'96*, London, UK, November 1996.

[23] K. L. Calvert, M. B. Doar, E. W. Zegura, "Modeling Internet Topology," *IEEE Communications Magazine*, vol. 35, pp. 160-163, June 1997.

[24] J. Nonnenmacher, and E. W. Biersack, "The Impact of Routing on Multicast Error Recovery," *Computer Communications*, vol. 21(10), pp. 867-879, July 1998.

[25] L. Wall, T. Christiansen and R. L. Schwartz, *Programming Perl, 2nd Edition*, O'Reilly, 1996.

[26] J. C. Bolot, "Characterizing End-to-End Packet Delay and Loss on the Internet," in *Journal of High-Speed Networks*, vol.2, no.3, pp 305-323, December 1993.

[27] D. D. Clark, "A Model for Cost Allocation and Pricing in the Internet," *MIT Workshop on Internet Economics*, `http://www.pres.umich.edu/jep/works/sourcefiles/clark.html`, March 1995.

[28] M. Prandini, "Efficient Certificate Status Handling within PKIs: An Application to Public Administration Services," *In Proceedings of ACSAC'99*, Phoenix, AZ, USA, December 99.

[29] Stardust Technologies, *IP Multicast Initiative (IPMI): Implementing IP Multicast in Different Network Infrastructures*, Stardust Technologies Inc, Campbell, CA, USA, January 1997.

[30] P. McDaniels and S. Jamin, "Windowed Certificate Revocation," *In Proceedings of Infocom2000*, Tel-Aviv, Israel, March 2000.

[31] A. Papoulis, *Probability, Random Variables, and Stochastic Processes, Third Edition*, McGraw-Hill Inc., 1991.