A transport layer protocol for cognitive radio networks

by

Aminu Muhammad Musa

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
July 2012

Thesis advisor

Rasit Eskicioglu and Attahiru Alfa

Author

Aminu Muhammad Musa

A transport layer protocol for cognitive radio networks

Abstract

For years, TCP has been the first choice for data transportation for the Internet. It provides reliable data delivery with the help of its flow control and congestion control mechanism. In order to improve TCP performance many modifications were proposed to the TCP congestion control mechanisms. However, some of the features of cognitive radio networks make TCP perform poorly in terms of throughput. Some of these features of cognitive radio networks are frequent bandwidth variation, licensed user interruption, and disconnections due to spectrum sensing. These features cause packet losses and time-outs which are mistakenly categorized as congestion losses by TCP. In this thesis, the TCP framework was modified to work efficiently in cognitive radio networks. Markov model that captures the behaviour of TCP is developed and used to evaluate the performance of the proposed protocol.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to begin by thanking almighty God for giving me the strength to carry out this research.

They say it takes a village to raise a child. In my case, it took a group of people like my professors, family and friends that got me through this phase of my life.

My research project would have not been possible without the support, time and effort put in by my two supervisors Dr. Rasit Eskicioglu and Dr. Attahiru Alfa. They constantly communicated with me through emails, and finding time in their busy schedules to meet with me at moments notice. They have my utmost gratitude. I also appreciate my examination committee members Dr. Neil Arnason and Dr. Dean McNeill for their suggestions that forced me to dig deeper into my research. Thank you all.

I will also want to express my gratitude to all my professors that their classes help prepare me for my research project. To all my fellow graduate students, especially Hesham Elsawy.

A big thank you to all my family and friends that prayed for me and gave me the space and time to concentrate on my work without any distractions. A big hug to my room-mates, Aliyu Zango, Ismail Aliyu, and Abdullahi Shehu. Last but not in anyway the least, I will like to thank Dr. Abba Gumel, Miss Fiona Clarke, and Dr. Sule Mundi, you all were my biggest support away from campus.

This thesis is dedicated to my parents, Mrs. Hadiza Imam and Mr. Muhammad Musa Gumel. There is no way I can pay you back for all you did to me. I appreciate it all from the bottom of my heart.

Chapter 1

Introduction

1.1 Overview

Network access for wireless devices is mostly limited to the unlicensed spectrum bands [9]. Unlicensed bands include the 900 MHz band and the 2.4 GHz Industrial Scientific Bands (ISM). Due to increasing number of wireless devices, the unlicensed spectrum bands are becoming congested. On the other hand, research by the Federal Communication Commission (FCC) reveals that the licensed spectrum bands, such as TV bands, are underused. To ease the overuse of the unlicensed bands, the FCC approved the use of licensed bands by unlicensed devices [9]. However, whenever a licensed device wants to make use of the licensed band, the unlicensed devices have to immediately stop using the licensed band. Therefore, the unlicensed devices must have the ability to detect unused licensed bands, the ability to detect the arrival of a licensed user, and the ability to dynamically modify their antenna settings. Cognitive radio techniques were designed for this environment.

A cognitive radio is a transceiver that has the ability to sense its environment for available channels and also the ability to re-configure its transmission or receiving parameters to operate on the channel to be used. A cognitive radio network [13] is a wireless network of cognitive radio nodes. On detection of a licensed device by a cognitive radio node, the cognitive radio node switches to a new spectrum band. Unlicensed devices are equipped with cognitive radios, so that they can use the licensed bands opportunistically.

To apply cognitive radio techniques, we need to modify almost all the protocols in the TCP/IP Internet stack [5]. This is because the existing protocols in the TCP/IP stack were not designed to capture the sensing feature of cognitive radios. The lower layers (physical layer and the data link layer) deal with spectrum sensing and decision. Whereas, the transport layer handles end-to-end data transmission. The focus of this work is on the transport layer protocols of the TCP/IP stack.

Transmission Control Protocol (TCP) [20] is the most widely-deployed transport layer protocol on the Internet. TCP connects two applications running on two hosts. TCP is considered to be a reliable transport layer protocol because it provides flow control and congestion control services [20]. Flow control [20] is the mechanism used by TCP to prevent the sender from overwhelming the receiver with segments. On the other hand, congestion control [3] is the mechanism used by TCP to respond to congestion in the path between the sender and the receiver. Therefore, a TCP sender starts sending segments to the receiver at a very slow rate, and then gradually increases the sending rate as it receives acknowledgements from the receiver. This technique is called the slow-start feature of TCP. On receiving a segment, the receiver

sends an acknowledgement message to the sender and the sender resets a timer called the retransmission time-out timer. When this timer expires, the sender will assume there is a possible congestion in the path and responds by reducing the rate at which it sends segments to the receiver.

The reliability of TCP in general made it the natural choice for use in cognitive radio networks. However, in cognitive radio networks, the arrival of a licensed user interrupts data transmission. Also, spectrum sensing causes temporary disconnections. Unfortunately, TCP assumes that these interruptions are caused by congestion in the path. Hence, a TCP sender unnecessarily decreases the rate at which it sends segments to the receiver.

TCP variations differ in how they update their sending rate. TCP Reno [15, 16] is one of the most commonly-used variation of TCP. Arvidsson and Krzesinski [4] modelled TCP Reno as an embedded Markov chain. They computed performance measures such as segment loss probability, throughput rate, carried load, and average download time.

Slingerland et al. [23] and Issariyakul et al. [14] evaluated the performance of TCP Reno on cognitive radio networks. Felice et al. [10] evaluated the performance of TCP Reno on cognitive radio ad hoc networks. The common result obtained by all these performance evaluations is that existing variations of TCP yield low throughput in cognitive radio networks.

Chowdhury et al. [8] designed a new transport protocol for cognitive radio ad hoc networks. However, their protocol requires explicit feedbacks from intermediate nodes and destination nodes. Sarkar and Narayan [22] designed rules for transportation in

cognitive radio infrastructure networks. Their proposed rules are augmented with TCP and TCP Westwood [7]. However, Sarkar and Narayan did not capture the effect of the arrival of licensed users.

We propose to modify TCP Reno [15, 16] to work efficiently in cognitive radio networks. We will maintain the connection set-up, flow control, and congestion control features of TCP Reno. However, we will take into consideration licensed user interruption, and temporary disconnection due to spectrum sensing.

For performance evaluation, we will modify the model by Arvidsson and Krzesinski [4] to capture the behaviour of cognitive radio networks. This modified model will be used to evaluate the performance of the proposed modification to TCP Reno.

1.2 Cognitive Radio Networks

A cognitive radio network [13] is a wireless network of cognitive radio nodes. Cognitive capability and reconfigurability are the two main characteristics of cognitive radios. Cognitive capability is the ability of a cognitive radio to sense its environment for currently unused licensed bands, perform an analysis on the Licensed User (LU) activity on the band, and finally make a decision on whether to access the band or not. On the other hand, reconfigurability is the ability of cognitive radios to reconfigure their antenna settings dynamically to adapt to the newly selected spectrum band [13].

The tasks required for a cognitive capable radio are categorized into three groups: spectrum sensing, spectrum analysis, and spectrum decision. These tasks form a cycle known as the cognitive cycle [13].

Spectrum sensing is the process of trying to detect unused licensed bands by

cognitive radio nodes. Spectrum sensing usually is a binary result, 0 if the channel is vacant, or a 1 if the channel is in use by an LU. Spectrum sensing is a difficult task and has many challenges [13]. Some of the spectrum sensing techniques are, energy detection, matched filter, cyclostationary detection, wavelet detection, decision fusion, data fusion, and multi-user diversity.

Spectrum analysis is carried out based on the spectrum sensing information gathered. It is used to categorize the currently unused licensed bands. In addition to the LU activity on the bands, interference level, channel error rate, path-loss, link layer delay and holding time are also taken into consideration [1]. History is also taken into consideration by some spectrum analysis models.

Deciding on which band to access based on spectrum sensing results and spectrum analysis model is referred to as spectrum decision. Decisions are made taking into consideration the quality of service requirement of the user application. Parameters considered include data rate and acceptable error rate.

Cognitive radio networks access techniques can be classified as: an overlay access technique or an underlay access technique [14]. In an overlay access technique, the Unlicensed Users (UU) are allowed access to the licensed bands only when the bands are not currently being used by any LU. This technique is also known as the opportunistic spectrum access model [24]. On the other hand, in an underlay access technique, the UUs are allowed access to the licensed bands even in the presence of LUs. Therefore, UUs transmissions may interfere with the LUs transmissions. However, in this model, UUs are allowed to interfere with LUs transmission only up to a certain level [24].

Our focus in this thesis is on the overlay spectrum access technique. Therefore, we assume that unlicensed users access the licensed bands only when the bands are not currently being use by any LU.

We also assume that cognitive radio nodes are equipped with a single transceiver. Therefore, cognitive radio nodes alternate between two modes, sensing mode and transmission mode. In sensing mode, the cognitive radio node will disconnects itself from all other connections and listens to the environment for signals from licensed devices. When in transmission mode, the cognitive radio node can engage in data transmission activities. Also, the cognitive radio node does not listen for signals from licensed devices. Therefore, the more time a cognitive radio node spends in transmission mode, the more data is transmitted. On the other hand, the more time a cognitive radio node spends in sensing mode, the more protection is given to licensed devices. Hence, we need to find a workable balance between sensing mode time and transmission mode time. It was shown that the optimal channel sensing time and transmission time pair is $(0.2s, 1.0s)$ [10, 17].

1.3 TCP

TCP [20] is the most widely-deployed transport layer protocol on the Internet. TCP connects two applications running on two hosts. The host that establishes the connection is called the sender, and the other host is called the receiver. The sender gets data from the application process and stores it in a buffer. The sender periodically takes some chunk of data from the buffer, encapsulates the chunk into what is called a segment, assigns a sequence number to the segment, and then sends the segment

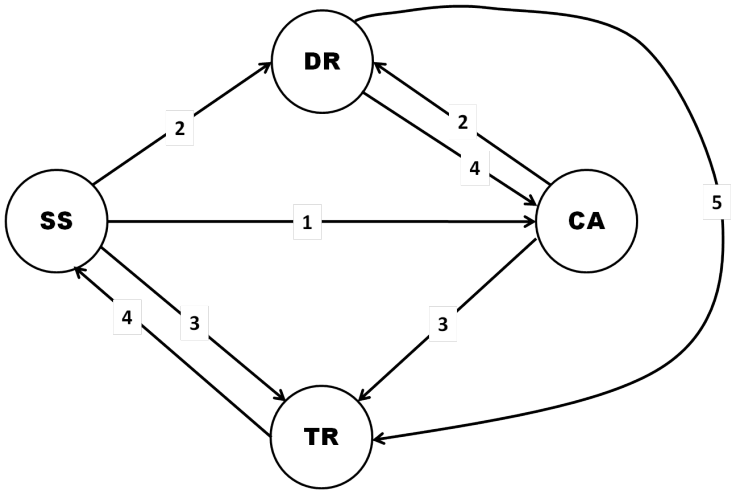


Figure 1.1: TCP Reno Transition Diagram

to the receiver. On receiving each segment, the receiver sends an acknowledgement message to the sender. The receiver uses the sequence numbers to reorder the received segments correctly.

TCP is considered to be a reliable transport layer protocol because it provides flow control [20] and congestion control [20, 3]. In particular, to control congestion, TCP requires the sender to keep a variable called the congestion window (w) [3]. The value w is the maximum number of unacknowledged segments the sender can send to the receiver. Therefore, the sender has to make sure that the amount of unacknowledged data in the network is always less than the w value. Hence, to control the sending rate, the TCP sender either increases or decreases the value of w. All TCP variations are based on the w reduction and increment technique. However, TCP variations differ in how they update the w value. TCP Reno [15, 16] is one of the most commonly-used variation of TCP.

Arvidsson and Krzesinski [4] explain TCP Reno transmission using four states;

Slow Start (SS), Congestion Avoidance (CA), Duplicate acknowledgement Retransmission (DR), and *Time-out Retransmission (TR)*. Figure 1.1 shows a transition diagram for TCP Reno. The conditions for the transitions in the diagram are represented by numbers on each link in the figure. Descriptions for the numbers are as follows:

1: Congestion window reaches the threshold ($w_n = w_{thresh}$).

2: segment loss detected by DA.

3: segment loss detected by TO.

4: Acknowledgements for retransmitted packets are received.

5: Acknowledgements for retransmitted packets are not received and timer expires.

When a connection is established, the sender begins in the SS state, sending segments with w of size 1 Maximum Segment Size (MSS). The sender increases the value of w by 1 after each acknowledgement received from the receiver. In addition, the sender keeps a threshold w_{thresh}. When the w value reaches w_{thresh}, the transmission moves to the CA state. In the CA state, when the sender perceives that the network is congestion free, the sender continues to increase its rate, but slower than the increment in the SS state. In particular, the sender increases the w value by $1/w$ for each acknowledgement received. On the other hand, when the sender detects congestion while in the CA state, the sender multiplicatively decreases its sending rate. This technique is known as Additive-Increase Multiplicative-Decrease technique.

TCP Reno relies on segment loss to detect congestion in the network. There are two ways a sender can detect a segment loss. One way is by the use of a Time-Out (TO) timer. Here, the sender resets a timer called the *timeout-interval* when it receives an acknowledgement from the receiver. Therefore, when the *timeout-interval* timer expires, it means the segments in transit are not yet received by the receiver or the acknowledgement message gets lost. Hence, the sender concludes that there is congestion in the path. The other way of detecting segment loss is by the use of Duplicate Acknowledgements (DA). The receiver sends three duplicate acknowledgements to the sender when the receiver detects a missing segment.

When there is a segment loss, the threshold w_{thresh} is set to half of the current w, but it has to satisfy the inequality $2 \leq w_{thresh} \leq w_{rcv}$, where w_{rcv} is the maximum number of segments the receiver can receive at a time.

If a segment loss is detected by DA, then transmission enters the DR state. In DR, the missing segment is retransmitted. If the sender received an acknowledgement for the retransmitted segment, then the state changes to CA and w is set equals to w_{thresh}. If the retransmitted segment was not acknowledged and the TO timer expires, then the state changes to TR.

If a segment loss is detected by TO, the transmission enters the TR state. In TR, the scaling factor that is used to set the time-out timer is doubled, and the lost segment is retransmitted. If the sender received an acknowledgement for the retransmitted segment, then the state changes to SS and w is set equals to 1, otherwise the state remains in TR and the process is repeated.

Some other variations of TCP include TCP New Reno [11], TCP Westwood [7],

TCP SACK [19], TCP CUBIC [12], and TCP Vegas [6].

TCP New Reno [11] and TCP SACK [19] are very much similar to TCP Reno. These newer protocols also relied on segment losses to detect congestion. They behave exactly like TCP Reno when segment loss is detected by TO. However, they react different from TCP Reno when segment loss is detected by duplicate acknowledgements. TCP Westwood [7] uses end-to-end bandwidth estimate to update its congestion window. However, similar to TCP Reno, it sets the congestion window to 1 when segment loss is detected by TO. On the other hand, TCP Vegas does not rely only on duplicate acknowledgements or time-out to detect congestion. It can detect congestion even before a segment is lost by calculating the expected traffic and the actual traffic and then set the congestion window size based on the difference. One of the disadvantages of TCP Vegas is that it is not as aggressive as TCP Reno. When in the SS state, it stops the increase in its congestion window sooner than TCP Reno [6]. TCP CUBIC is the most recent among the TCP variations mentioned above. It was designed specifically for high-speed networks. In network set-ups where TCP Reno performs reasonably well, it was shown that TCP CUBIC does not give any significant improvement [12].

TCP Reno, TCP New Reno, TCP SACK, and TCP Westwood all reduce their congestion window to 1 when they detect a time-out event. We chose TCP Reno primarily because it is the most commonly-used among them all. In addition, the other variants result from modifications to TCP Reno, therefore, the same modifications can easily be made to our proposed modifications to TCP Reno to work in cognitive radio networks.

From now on when we write just TCP we are referring to TCP Reno.

1.4 TCP in Cognitive Radio Networks

Here, we will describe how TCP behaves when employed in cognitive radio networks. In cognitive radio networks, the arrival of a licensed user interrupts data transmission. Also, spectrum sensing causes temporary disconnections that leads to TCP time-outs. Unfortunately, TCP assumes that these time-outs are caused by congestion in the path. Hence, a TCP sender unnecessarily decreases the rate at which it sends segments to the receiver.

Suppose we employ TCP to work in cognitive radio networks. In addition to the events that lead to the reduction of sending rate, as we saw in the previous section, there are other new events. Suppose the sender is transmitting at rate μ_c, and an LU arrived. The sender then has to look for another channel and resume transmission. However, TCP always starts transmission at a slower rate say μ_1, and $\mu_1 \leq \mu_c$. Hence, arrival of an LU leads to the reduction of the sending rate. Therefore, frequent arrival of LUs will lead to frequent reduction of the sending rate.

Another event that leads to reduction of sending rate is when the receiver is engaged in spectrum sensing. Here, we assumed that the receiver is using a single transceiver, therefore, it alternates between sensing mode and transmission mode. Therefore, when in sensing mode, it cannot participate in transmission, hence, the sender will experience a temporary disconnection which will lead to timer expiration. Unfortunately, the TCP sender will assume that the timer expired because of congestion and it will reduce its sending rate.

In our design, we will assume a cross layer design. Therefore, we assume that information flows from one layer to another. We will assume that TCP relies on the lower layers for spectrum sensing and spectrum decision. The lower layers notify TCP about which channel to use for transmission and which channel to move to when an LU arrives on the current channel. Therefore, TCP does not have to make any decision regarding which channel to sense.

In our design, we have to prevent the reduction of sending rate due to LU arrival and due to engagement of the receiver in spectrum sensing. The sender can also go into sensing mode, therefore, we have to prevent the sender from reducing its sending rate when transmission resumes.

1.5 TCP as a Queueing Model

We can capture the behaviour of the TCP described above as a queueing model. Let us represent the path (channel) that connects the sender and the receiver as a server. Therefore, the rate at which segments are sent can be represented as the service rate. However, we have seen that, in order to control congestion in the network, TCP keeps on varying the sending rate. Therefore, the service rate in the model is not fixed. If the server starts serving at rate μ_1, then the service rate will keep increasing to $\mu_2, \ldots, \mu_c$ where $\mu_2, \ldots, \mu_c \geq \mu_1$, and μ_c is the current service rate. When the TCP sender perceives that the path is congested, then it drops the service rate to μ_p, where $\mu_p < \mu_c$. The TCP sender continues to decrease the service rate until when it perceives that the path is less congested, then it starts increasing the service rate again. Therefore, we can see that the service rate varies based on the level of

congestion in the server.

Similarly, we can model TCP in cognitive radio networks as a queueing model. Suppose we have two channels to use for transmission. We can model this as a single server that serves two queues one at a time. We set the probability of the server attending to the first queue Q_1 to be p_1, where p_1 is the probability that the first channel is idle. Similarly, we set the probability of the server attending to the second queue Q_2 to be p_2, where p_2 is the probability that the second channel is idle. Since the server can remain idle, then $p_1 + p_2 \leq 1$. Therefore, if the server is attending to Q_1 at service rate μ_1 and an LU arrives, then the server switches to attend to Q_2 but at a rate μ_2 and $\mu_2 \leq \mu_1$. On the other hand, suppose the server is attending to Q_1, and it experiences temporary disconnection due to sensing by the receiver. Here, when connection resumes, the server continues to attend to Q_1, but at a slower rate. In a situation where TCP is waiting for notification about the new channel to move to, the server remains idle.

1.6 Summary

The concept of cognitive radio networks was introduced to solve the spectrum shortage problem. It allows unlicensed users to access the licensed spectrum opportunistically. The main features of cognitive radio networks are spectrum sensing, spectrum analysis, and spectrum decision. To incorporate these features into the existing TCP/IP protocol stack, some modifications are needed. Our focus is on the transport layer protocol of the TCP/IP stack, TCP Reno in particular.

Features of cognitive radio networks such as frequent bandwidth variation, inter-

ruptions due to licensed user arrival, and temporary disconnections due to spectrum sensing were not captured by the existing transport layer protocols for wireless networks. Therefore, the existing transport layer protocols perform poorly in cognitive radio networks, in terms of throughput. We need an efficient transport layer protocol that will capture all the features of cognitive radio networks and achieves a reasonable throughput.

In this thesis, we made some modifications to TCP Reno to capture some of the features of cognitive radio networks. The features we captured are licensed user interruptions and spectrum sensing.

Chapter 2

Related Work

The existing research work that evaluated the performance of TCP in cognitive radio networks are summarized. From this research, we identified characteristics that are needed to modify the standard TCP in order for it to work efficiently in cognitive radio networks. Some of the proposed modifications to TCP and other transport protocols for cognitive radio networks will be discussed later. How they differ from our proposed modification to the TCP Reno protocol will also be discussed. Finally, we discussed a proposed scheme to model TCP Reno as an embedded Markov chain and how it is related to our proposed scheme.

2.1 Performance of TCP on Cognitive Radio Networks

Slingerland et al. [23] evaluated the performance of TCP on cognitive radio networks. They considered different TCP variations including TCP NewReno [11] and

TCP Vegas [6]. They performed simulations with the NS-2 simulator using the source code of the Linux TCP stack. They concluded that time taken by cognitive radio nodes in sensing mode affects the throughput of the TCP connections.

Akildiz et al. [2] in their survey discussed the research challenges for developing a transport layer protocol for cognitive radio ad hoc networks. They concluded that modifications can be made to TCP [20] to obtain higher throughput.

Issariyakul et al. [14] evaluated the performance of TCP [20] on cognitive radio networks. They only considered licensed-user interruption. Their simulation results showed that the throughput of the unlicensed users increases as the number of channels increases. However, when the number of channels reaches some threshold, the TCP throughput starts to drop. Hence, they concluded that there is an optimum number of channels for unlicensed users to achieve maximum aggregate throughput. The optimum number of channels depends on the number of licensed users and also the number of unlicensed users on the network.

Felice et al. [10] evaluated the performance of TCP over cognitive radio ad hoc networks. They considered different TCP variations including TCP Reno [15, 16], TCP NewReno [11] and TCP Vegas [6]. They extended the standard NS-2 simulator to support unique features of cognitive radio ad hoc networks. They used the extended version of the NS-2 to evaluate the TCP performance. In summary, their results showed that TCP experiences the highest throughput when the duration of licensed user presence is long and it is followed by long absence of a licensed user. Also, they showed that $(0.2s, 1.0s)$ is the optimal choice of sensing period and transmission period that yields higher throughput.

2.2 Transport Layer Protocols for Cognitive Radio Networks

Chowdhury et al. [8] designed a transport protocol for cognitive radio ad hoc networks. They modeled the transport protocol as a six-state system. Some of the events that cause the system to change state are route failure, congestion notification, node mobility, and spectrum change. However, their protocol requires explicit feedbacks from intermediate nodes and the destination.

Luo et al. [18] designed a scheme that optimizes TCP throughput without making changes to TCP. The variation of TCP they used is the TCP Reno [15, 16]. To achieve higher TCP Reno throughput, their scheme optimized the low-layer parameters of the network, such as modulation and coding scheme in the physical layer, and frame size in the data-link layer. However, they did not modify the congestion control mechanism in TCP Reno. Their scheme treats all type of losses as congestion losses. Hence, sometimes a TCP Reno sender unnecessarily decreases the speed at which it transmits data.

Sarkar and Narayan [22] designed, implemented, and evaluated a transport protocol for cognitive radio networks. They augmented their protocol with TCP [20] and TCP Westwood [7]. Their protocol handles temporary disconnections caused by spectrum sensing. Also, their protocol handled frequent bandwidth variation as the connection moves from one channel to another. However, they did not handle disconnection caused by licensed user arrival.

2.3 Embedded Markov Chain for TCP Reno

Arvidsson and Krzesinski [4] modelled TCP Reno [15, 16] as an embedded Markov model (See Section 1.3). Their model is based on the following assumptions and representations [4]:

- The lengths of the files to be transmitted are assumed to be geometrically distributed with parameter φ.
- The minimum file size is one packet and the average file size is $1/(1-\varphi)$.
- $q = 1 - p_f$ denotes the probability that a segment is received successfully, where p_f is the probability that a segment is lost in the forward direction.
- $r = (1-p_f)(1-p_b)$ denotes the probability that a segment is received and acknowledged successfully, where p_b is the probability that a segment is lost in the backward direction.
- $s = 1 - r$ denotes the probability that a segment is not successfully received and acknowledged.
- To represent successful transmission in the SS and the CA states, q is used. In the TR and DR state, the chance of losing acknowledgements is higher (since the network is already congested). Therefore, for these states, r is used to represent successful transmission.
- The first visit to the SS state is treated as a different state denoted by SS'.

- $N_{SS'} = \sum_{n=1}^{\infty} nq^{*(n-1)}(1-q^*) = 1/(1-q^*)$ denotes the number of segments transmitted when in the SS' state, where $q^* = \varphi(1-p_f)$ is the probability that there is a segment and it is successfully transmitted.

- $N_{SS} = min(N_{SS'}, w_{thresh}-1)$ denotes the number of segments transmitted when in the SS state.

- $N_{CA} = 1/(1-q^*)$ denotes the number of segments transmitted when in the CA state.

- $N_{DR} = 1$ denotes the number of segments transmitted when in the DR state.

- $N_{TR} = \sum_{n=1}^{\infty} ns^{*(n-1)}r = 1/r$ denotes the number of segments transmitted when in the TR state.

The state space for their Markov model is the set $\{SS', SS, CA, DR, TR, OK\}$, where OK is the final state that represents file transmission completion. The transition probability matrix is given below:

$$\begin{bmatrix} 0 & 0 & 0 & \pi_{SS',DR} & \pi_{SS',TR} & \pi_{SS',OK} \\ 0 & 0 & \pi_{SS,CA} & \pi_{SS,DR} & \pi_{SS,TR} & \pi_{SS,OK} \\ 0 & 0 & 0 & \pi_{CA,DR} & \pi_{CA,TR} & \pi_{CA,OK} \\ 0 & 0 & \pi_{DR,CA} & 0 & \pi_{DR,TR} & \pi_{DR,OK} \\ 0 & \pi_{TR,SS} & 0 & 0 & 0 & \pi_{TR,OK} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The arrangement of the columns corresponds to the states order: SS', SS, CA, DR, TR, and OK. The state transition probabilities are denoted by $\pi_{i,j}$. Where i and j are

states in the Markov model.

The transition probabilities are defined by Arvidsson and Krzesinski [4] as follows:

$$\pi_{SS',OK} = \sum_{n=1}^{\infty} \varphi^{n-1}(1-\varphi)q^n = q(1-\varphi)\frac{1}{1-q\varphi},$$

$$\pi_{SS,OK} = \sum_{n=1}^{w_{thresh}-1} \varphi^{n-1}(1-\varphi)q^n = q(1-\varphi)\frac{1-(q\varphi)^{w_{thresh}-1}}{1-q\varphi},$$

$$\pi_{CA,OK} = \sum_{n=1}^{\infty} \varphi^{n-1}(1-\varphi)q^n = q(1-\varphi)\frac{1}{1-q\varphi},$$

$$\pi_{DR,OK} = (1-\varphi)r,$$

$$\pi_{TR,OK} = (1-\varphi),$$

$$\pi_{SS,CA} = \sum_{n=w_{thresh}}^{\infty} \varphi^{n-1}(1-\varphi)q^{w_{thresh}-1} = (q\varphi)^{w_{thresh}-1},$$

$$\pi_{SS',DR} = (1-\pi_{SS',OK})(1-\rho_{TO}(W_{SS'}))\ ,$$

$$\pi_{SS',TR} = (1-\pi_{SS',OK})\rho_{TO}(W_{SS'})\ ,$$

$$\pi_{SS,DR} = (1-\pi_{SS,OK}-\pi_{SS,CA})(1-\rho_{TO}(W_{SS}))\ ,$$

$$\pi_{SS,TR} = (1-\pi_{SS,OK}-\pi_{SS,CA})\rho_{TO}(W_{SS})\ ,$$

$$\pi_{CA,DR} = (1-\pi_{CA,OK})(1-\rho_{TO}(W_{CA}))\ ,$$

$$\pi_{CA,TR} = (1-\pi_{CA,OK})\rho_{TO}(W_{CA})\ ,$$

$$\pi_{DR,CA} = \varphi r,$$

$$\pi_{DR,TR} = s,$$

$$\pi_{TR,SS} = \varphi,$$

where ρ_{TO} is the probability that a lost segment is detected by TO,

$\rho_{TO}(W_{SS'}) = min(1, 3/(min(N_{SS'}, w_{rcv})\varphi))$,

$\rho_{TO}(W_{SS}) = min(1, 3/(min(N_{SS}, w_{rcv})\varphi))$,

$\rho_{TO}(W_{CA}) = min(1, 3/((w_{thresh} + T_{CA})\varphi))$,

w_{rcv} is the maximum number of segments the receiver can receive at a time, and T_{CA} is the average time spent in CA.

For performance evaluation, Arvidsson and Krzesinski [4] begin with an estimate of offered load to compute the segment loss probability. They use the segment loss probability to compute the state transition probabilities, which are in turn used to compute the average number of visits to each state.

They compute the average number of visits (V) to each state as follows:

$V_{SS'} = 1$,

$V_{OK} = 1$,

$V_{SS} = V_{TR}\pi_{TR,SS}$,

$V_{CA} = V_{SS}\pi_{SS,CA} + V_{DR}\pi_{DR,CA}$,

$V_{DR} = V_{SS'}\pi_{SS',DR} + V_{SS}\pi_{SS,DR} + V_{CA}\pi_{CA,DR}$,

$V_{TR} = V_{SS'}\pi_{SS',TR} + V_{SS}\pi_{SS,TR} + V_{CA}\pi_{CA,TR} + V_{DR}\pi_{DR,TR}$.

They compute the following performance measures: RTT, total number of packets sent during an average file transmission, total time to complete an average file transmission, throughput rate, and average download time.

In this thesis, we made some modifications to TCP Reno, to capture some of the features of cognitive radio networks. Features we captured are licensed user interruptions and spectrum sensing. To evaluate our work, we extended the TCP model by Arvidsson and Krzesinski [4] to capture licensed user interruptions and spectrum sensing.

Chapter 3

Modifications to TCP Reno

Here, we use difference equations to show how TCP Reno updates the value of it's congestion window w. We show how licensed user arrival and spectrum sensing affect the window size w. Finally, we propose some modifications to TCP and show how the proposed modifications could help prevent TCP from unnecessary reduction of the value of w.

TCP provides a logical connection between two processes running at two end systems. The sending application passed a stream of data to its TCP module. TCP establishes a connection with the receiver and sends the data to the receiver. To establish a connection, TCP initializes some variables and buffers, which are released when data transmission is completed. Applications experience delay, when TCP takes a long time before it completes transmission. This delay is caused by the congestion and flow control mechanisms used by TCP to ensure reliable data delivery.

To ensure reliable data delivery, TCP uses the window principle when transmitting data. A window w_n is used to specify the amount of data bytes the sender can

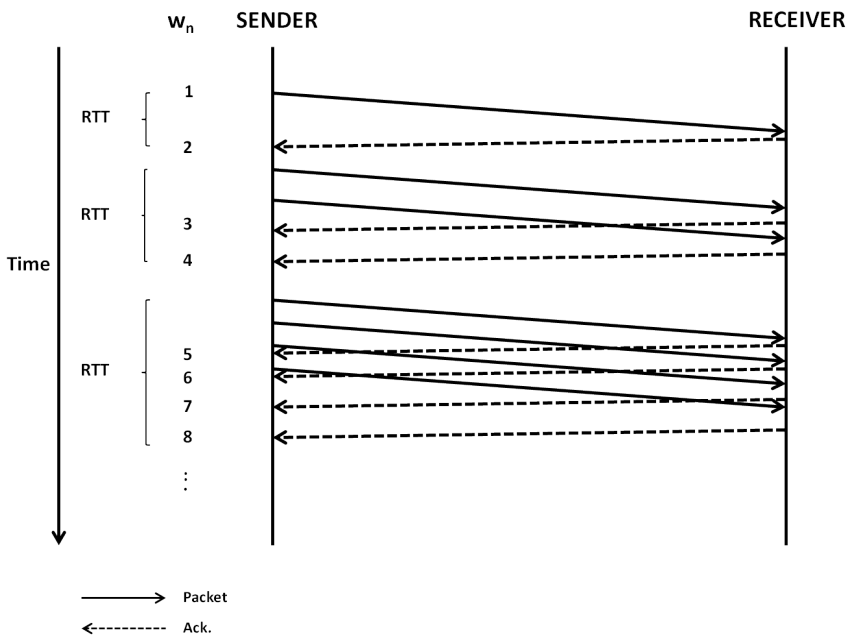


Figure 3.1: Congestion window increment in the SS state

send to the receiver at time n, and then waits for acknowledgement. To control the congestion in the network, TCP keeps on varying the size of w_n depending on the level of congestion in the network at time n.

Suppose a TCP connection is established between a sender and a receiver. Let the discrete random variable X_n denote the number of segments in the TCP sending buffer at a given time n. Our interest is to determine when data transmission will be completed. In other words to determine when the value of X_n will become zero. Let us define an equation for X_n as:

$$X_{n+1} = max\{0, X_n - w_n\}, \tag{3.1}$$

where $w_n \geq 0$.

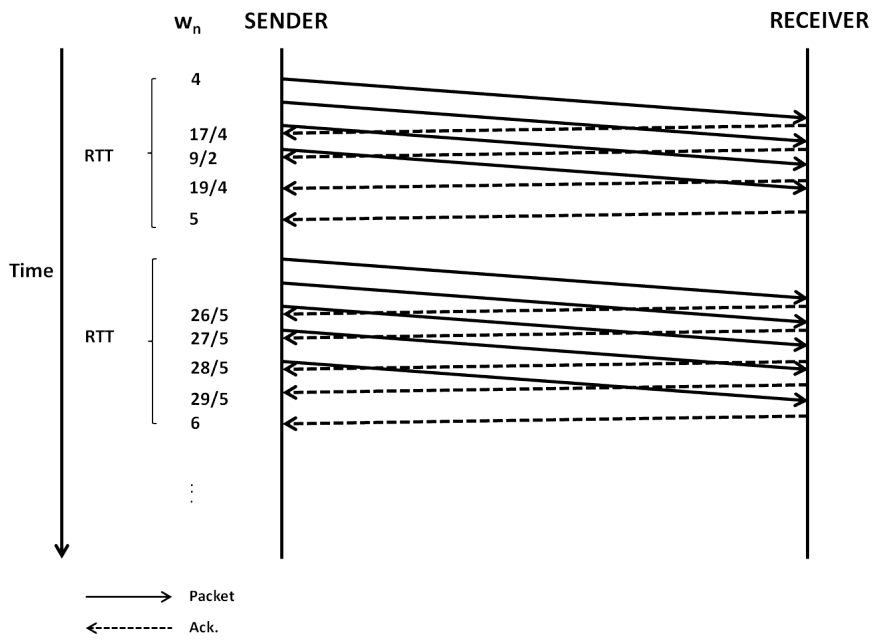


Figure 3.2: Congestion window increment in the CA state

Now, we can see that the rate at which X_n becomes zero depends on w_n, and w_n depends on the congestion level in the network. When TCP detects congestion in the network at time n, it decreases the value of w_n. On the other hand, if no congestion is detected by TCP at time n, then it increases the value of w_n. By how much exactly TCP increases or decreases the value of w_n depends on the current state of transmission, and also how the congestion was detected.

When in the SS state and the network is perceived to be free of congestion, TCP increases the value of w_n by 1 MSS after each acknowledgement received from the receiver. MSS is the maximum size a segment can be. Therefore, after every 1 Round-Trip Time (RTT), the value of w_n will be doubled. This can be seen more clearly with the aid of an example.

Figure 3.1 shows an example where transmission is in the SS state. The sender

begins sending data to the receiver with $w_n = 1$ MSS. Upon receiving the 1 segment, the receiver sends an acknowledgement for it. On receiving the acknowledgement, the sender increases the value of w_n by 1 MSS, now $w_n = 2$ MSS. This completes the first RTT. Now, the sender sends two segments back-to-back to the receiver. The receiver sends an acknowledgement for each of the segments. On arrival of each acknowledgement, the receiver increases the value of w_n by 1 MSS. RTT becomes complete when all the packets sent are acknowledged. Therefore, at the end of this RTT, w_n will be set to 4 MSS. The process continues until a segment get lost or w_{thresh} is reached.

When in the CA state and the network is perceived to be free of congestion, TCP increases the value of w_n by $1/w_n$ for each acknowledgement received. Therefore, after every 1 RTT, the value of w_n will be increased by 1 MSS. This also can be described more clearly with the aid of an example.

Figure 3.2 shows an example where transmission is in the CA state. The value of w_n is set at 4 MSS. The sender sends four segments back-to-back to the receiver. The receiver sends an acknowledgement for each of the segments. On arrival of each acknowledgement, the receiver increases the value of w_n by $1/w_n$. Therefore, at the end of this RTT, w_n will be set to 5 MSS. The process continues until a segment get lost.

As we have seen earlier, TCP relies on segment loss to detect congestion in the network. There are two ways a sender can detect a segment loss. One way is by the use of a Time-Out (TO) timer. The other way of detecting segment loss is by the use of Duplicate Acknowledgements (DA). If a segment loss is detected by DA, then

transmission enters the DR state. In DR, the missing segment is retransmitted. If the sender received an acknowledgement for the retransmitted segment, then state changes to CA and w_n is set equals to $w_n/2$. If the retransmitted segment was not acknowledged and the TO timer expires, then state changes to TR. On the other hand, if a segment loss is detected by TO, the transmission enters the TR state. In TR, the scaling factor that is used to set the time-out timer is doubled, and the lost segment is retransmitted. If the sender received an acknowledgement for the retransmitted segment, then the state changes to SS and w_n is set equal to 1 MSS, otherwise the state remains in TR and the process is repeated. Therefore, we can now define an equation for the value of w_n at a given time n. Here, we will assume that time is in units of RTT. Therefore, the time difference between time n and time $n+1$ is 1 RTT. Hence, we define the w_n equation as follows:

$$w_{n+1} = \begin{cases} 2w_n, & \text{if there is no congestion \& in the } SS \text{ state;} \\ w_n + 1, & \text{if there is no congestion \& in the } CA \text{ state;} \\ w_n/2, & \text{if congestion was detected by DA;} \\ 1MSS, & \text{if congestion was detected by TO.} \end{cases}$$

3.1 Delay caused by Cognitive Radio Network Features

Cognitive radio network features introduce an extra delay due to LU interruption and spectrum sensing. Licensed user arrival interrupts transmission and spectrum sensing caused time-outs. TCP considers time-outs to be caused by congestion in the

network. Therefore, whenever there is a time-out, the value of w_n is decreased to 1 MSS. Hence, we now have a new equation for the value of w_n as follows:

$$w_{n+1} = \begin{cases} 2w_n, & \text{if LU is off \& no sensing \& no congestion \& in } SS \text{ state;} \\ w_n + 1, & \text{if LU is off \& no sensing \& no congestion \& in } CA \text{ state;} \\ w_n/2, & \text{if LU is off \& no sensing \& congestion detected by DA;} \\ 1, & \text{if LU is off \& no sensing \& congestion detected by TO;} \\ 1, & \text{if LU is off \& sensing \& no congestion;} \\ 1, & \text{if LU is off \& sensing \& congestion detected by DA or TO;} \\ 1, & \text{if LU is on.} \end{cases}$$

3.2 Proposed Scheme

We assume that the transport layer can exchange information with the lower layers. Therefore, the lower layers can notify the transport layer when an LU arrives. Moreover, the transport layer can get the spectrum sensing schedule from the lower layers. We also assume that, each cognitive radio node uses a single transceiver, therefore, each node (sender or receiver) alternates between sensing mode and transmission mode.

The main idea behind our proposed scheme is to prevent the reduction of the value of w_n, caused by either LU interruption or spectrum sensing. We know that TCP is a connection-oriented protocol. During connection set-up, information like initial sequence number and maximum segment size are exchanged between the sender and the receiver. We proposed to add other information to the connection set-up messages.

Spectrum sensing is periodic and its duration is constant, therefore end systems can share their spectrum sensing schedules during connection set-up. We will add spectrum sensing schedule information to the messages exchanged during connection set-up. The information can be of the form $< t_1, t_2, t_3, t_4 >$, where t_1 is the current time, t_2 is the time at which the node will begin its next sensing, t_3 is the sensing time (duration of each sensing), and t_4 is the transmission time. Similar idea was used by Chowdhury et al. [8]. They designed their protocol for cognitive radio ad hoc networks and to exchange sensing schedule they modified the way TCP NewReno set up a connection. Their protocol requires that the end nodes exchange their ids. However, our model do not require any id exchange. We add the sensing information on the TCP segment header in the Option field.

Therefore, the sender now has the knowledge about when and for how long the receiver will be in sensing mode. Hence, instead of dropping the value of w_n to 1 MSS, we now freeze the value of w_n and resume transmitting with it when sensing is over.

When a LU arrives, transmission has to stop until the LU leaves the channel or a new vacant channel has been found. To prevent the TCP sender from setting the value of w_n to 1 MSS, our TCP model will continue to measure the Expected Available Bandwidth (EAB) of the channel. Therefore, when an LU leaves the channel or a new available channel is found, our model will use EAB as the new sending rate.

Therefore we now have a modified equation for the value of w_n:

$$w_{n+1} = \begin{cases} 2w_n, & \text{if LU is off \& no sensing \& no congestion \& in } SS \text{ state;} \\ w_n + 1, & \text{if LU is off \& no sensing \& no congestion \& in } CA \text{ state;} \\ w_n/2, & \text{if LU is off \& no sensing \& congestion detected by DA;} \\ 1, & \text{if LU is off \& no sensing \& congestion detected by TO;} \\ w_n, & \text{if LU is off \& sensing \& no congestion;} \\ w_n/2, & \text{if LU is off \& sensing \& congestion detected by DA;} \\ 1, & \text{if LU is off \& sensing \& congestion detected by TO;} \\ w_n, & \text{if LU is on.} \end{cases}$$

Note that, when LU is on, the value of w_n does not change. However, in this case, it does not really matter what we set the value of w_n to, since, transmission has to stop and when connection resumes, w_n *will be set to a value computed based on the measured EAB.*

We consider LU on to be the dominant case over the following cases: LU on & there is congestion, LU on & there is no congestion, LU on & no sensing in progress, and LU on & sensing in progress.

Modifications of the w_n value equation leads to a little a modification to equation (1), therefore, Equation (1) becomes:

$$X_{n+1} = max\{0, X_n - w_n * k\}, \tag{3.2}$$

where $w_n \geq 0$ and $k = 0$ or 1.

The value for the variable k is given by the following formula:

$$k = \begin{cases} 1, & \text{if LU is off \& no sensing \& no congestion;} \\ 1, & \text{if LU is off \& no sensing \& congestion detected by DA or TO;} \\ 0, & \text{if LU is off \& sensing \& no congestion;} \\ 0, & \text{if LU is off \& sensing \& congestion detected by DA or TO;} \\ 0, & \text{if LU is on.} \end{cases}$$

Chapter 4

Performance Evaluation

In Section 2.3, we have presented the embedded Markov model built by Arvidsson and Krzesinski [4] to evaluate the performance of TCP Reno. However, that model did not capture any of the unique features of cognitive radio networks. Here, cognitive radio features are introduced to the model of Arvidsson and Krezesinski and the performance of TCP Reno is observed. In addition, we extended their model to capture separately the effects of LU interruption and spectrum sensing. The performance of our proposed modifications to TCP Reno was evaluated. Finally, we presented some tables and graphs.

4.1 Original TCP with LU Interruption and Spectrum Sensing

LU interruption introduces an extra delay to TCP transmission. In particular, LU interruption causes more packet losses. Let us denote the probability of an LU arrival

on a channel by p. In addition to LU interruption, spectrum sensing also introduces an extra delay to TCP transmission. This is because packets sent when the receiver is in sensing mode are dropped, and hence TCP reduces its sending rate. Also, when the sender is in sensing mode it cannot transmit packets. Let us denote the probability that one of the nodes is in sensing mode by d. Since both p and d contributes to packets loss, they should be included in packet loss probability estimation. To do that, we made the following changes to the model of Arvidsson and Krzesinski [4]. We replace:

- The probability that a segment is received successfully $q = 1 - p_f$ by

  $q = (1 - p_f)(1 - p)(1 - d)$,

  where p_f is the probability that a packet is lost in the forward direction.

- The probability that a segment is received and acknowledged successfully

  $r = (1 - p_f)(1 - p_b)$ by

  $r = (1 - p_f)(1 - p)(1 - p_b)(1 - d)$,

  where p_b is the probability that a packet is lost in the backward direction.

- The probability that there is a packet and it is successfully transmitted

  $q^* = \varphi(1 - p_f)$ by

  $q^* = \varphi(1 - p_f)(1 - p)(1 - d)$.

We assume a cross-layer architecture, therefore the transport layer can communicate with the other layers of the TCP/IP stack. Lowers layers (Physical and Data Link layers) handle spectrum sensing, we assume that the value p and d will be communicated to the transport layer from the lower layers.

4.2 Capturing LU Interruption and Spectrum Sensing in TCP

In order to see explicitly the effects of LU interruption and spectrum sensing, there is need to capture them separately. In other words, there is need to differentiate losses caused by LU arrival and spectrum sensing from losses caused by congestion in the network. To capture separately the effect of LU interruption and spectrum sensing, we modify the transition probability matrix by Arvidsson and Krzesinski [4]. We begin by capturing the effect of LU interruption and later capture the effect of both LU interruption and spectrum sensing.

4.2.1 Capturing LU Interruption in TCP

To capture separately the effect of LU arrival, we modify the transition probability matrix by Arvidsson and Krzesinski [4] by introducing a new state OFF. The OFF state will be the state where the model goes into when a licensed user is detected. Regardless of the current state of the model, whenever a licensed user arrives, the model switches to the OFF state. We now set the transition probability from each of the six states to the OFF state to be equal to p, where p is the probability of licensed user arrival on the current channel. Whenever the model enters the OFF state it will remain in the OFF state until a new channel is found or the current channel has become vacant again.

Another new state MS is introduced. The MS state is where connection goes from the OFF state. When a new channel is found, transmission resumes in the MS state.

The state MS is similar to the DR state. When in the MS state, missing segments are retransmitted and if acknowledgements are received for the retransmitted segments, then the state changes to CA. However, if the retransmitted segments were not acknowledged and the TO timer expires, then state changes to TR.

These modifications lead to a modified transition probability matrix with two additional states shown below:

$$\begin{bmatrix}
0 & 0 & 0 & \bar{p}\cdot\pi_{SS',DR} & \bar{p}\cdot\pi_{SS',TR} & p & 0 & \bar{p}\cdot\pi_{SS',OK} \\
0 & 0 & \bar{p}\cdot\pi_{SS,CA} & \bar{p}\cdot\pi_{SS,DR} & \bar{p}\cdot\pi_{SS,TR} & p & 0 & \bar{p}\cdot\pi_{SS,OK} \\
0 & 0 & 0 & \bar{p}\cdot\pi_{CA,DR} & \bar{p}\cdot\pi_{CA,TR} & p & 0 & \bar{p}\cdot\pi_{CA,OK} \\
0 & 0 & \bar{p}\cdot\pi_{DR,CA} & 0 & \bar{p}\cdot\pi_{DR,TR} & p & 0 & \bar{p}\cdot\pi_{DR,OK} \\
0 & \bar{p}\cdot\pi_{TR,SS} & 0 & 0 & 0 & p & 0 & \bar{p}\cdot\pi_{TR,OK} \\
0 & 0 & 0 & 0 & 0 & p & 1-p & 0 \\
0 & 0 & \bar{p}\cdot\pi_{MS,CA} & 0 & \bar{p}\cdot\pi_{MS,TR} & p & 0 & \bar{p}\cdot\pi_{MS,OK} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}.$$

The arrangement of the columns corresponds to the states arrangement: SS', SS, CA, DR, TR, OFF, MS, and OK. The state transition probabilities are denoted by π. The term p denotes the probability of a licensed user arrival on the current channel and $\bar{p} = 1 - p$. $\pi_{MS,CA} = \pi_{DR,CA}$, $\pi_{MS,TR} = \pi_{DR,TR}$, and $\pi_{MS,OK} = \pi_{DR,OK}$. All other probabilities remain the same as defined by Arvidsson and Krezesinski (see Section 2.3).

4.2.2 Capturing both LU Interruption and Spectrum Sensing in TCP

The transition probability matrix above did not capture the effect of spectrum sensing. We introduce two new states, the spectrum sensing state denoted by SNS and the state where transmission goes after spectrum sensing is over, denoted by FZ. Hence, when spectrum sensing is in progress, transmission switches to the SNS state. In the SNS state, no segments are transmitted. Transmission remains in the SNS state for the duration of spectrum sensing period. When spectrum sensing is over, transmission moves to the FZ state. The FZ state is similar to the DR and the MS states.

In our proposed design (Section 3.2), we let the end systems share their spectrum sensing schedules during connection set-up. The information exchanged can be of the form $< t_1, t_2, t_3, t_4 >$, where t_1 is the current time, t_2 is the time at which the node will begin its next sensing, t_3 is the sensing time (duration of each sensing), and t_4 is the transmission time. The probability that a node is in sensing mode can be represented as $d = \frac{t_3}{t_4+t_3}$.

Let A and B be two nodes connected via TCP. Suppose that the probability that A is in sensing mode is d_A and the probability that B is in sensing mode is d_B. The probability that the connection is in sensing mode (one of the nodes is in sensing mode) can be computed as $d_A + d_B - d_A d_B$.

Below is the new transition probability matrix after the modification above.

$$\begin{bmatrix}
0 & 0 & 0 & \bar{p}\cdot\bar{d}\cdot\pi_{SS',DR} & \bar{p}\cdot\bar{d}\cdot\pi_{SS',TR} & \pi_{SS',OFF} & 0 & \bar{p}\cdot d & 0 & \bar{p}\cdot\bar{d}\cdot\pi_{SS',OK} \\
0 & 0 & \bar{p}\cdot\bar{d}\cdot\pi_{SS,CA} & \bar{p}\cdot\bar{d}\cdot\pi_{SS,DR} & \bar{p}\cdot\bar{d}\cdot\pi_{SS,TR} & \pi_{SS,OFF} & 0 & \bar{p}\cdot d & 0 & \bar{p}\cdot\bar{d}\cdot\pi_{SS,OK} \\
0 & 0 & 0 & \bar{p}\cdot\bar{d}\cdot\pi_{CA,DR} & \bar{p}\cdot\bar{d}\cdot\pi_{CA,TR} & \pi_{CA,OFF} & 0 & \bar{p}\cdot d & 0 & \bar{p}\cdot\bar{d}\cdot\pi_{CA,OK} \\
0 & 0 & \bar{p}\cdot\bar{d}\cdot\pi_{DR,CA} & 0 & \bar{p}\cdot\bar{d}\cdot\pi_{DR,TR} & \pi_{DR,OFF} & 0 & \bar{p}\cdot d & 0 & \bar{p}\cdot\bar{d}\cdot\pi_{DR,OK} \\
0 & \bar{p}\cdot\bar{d}\cdot\pi_{TR,SS} & 0 & 0 & 0 & \pi_{TR,OFF} & 0 & \bar{p}\cdot d & 0 & \bar{p}\cdot\bar{d}\cdot\pi_{TR,OK} \\
0 & 0 & 0 & 0 & 0 & p & 1-p & 0 & 0 & 0 \\
0 & 0 & \bar{p}\cdot\bar{d}\cdot\pi_{MS,CA} & 0 & \bar{p}\cdot\bar{d}\cdot\pi_{MS,TR} & \pi_{MS,OFF} & 0 & \bar{p}\cdot d & 0 & \bar{p}\cdot\bar{d}\cdot\pi_{MS,OK} \\
0 & 0 & 0 & 0 & 0 & \pi_{SNS,OFF} & 0 & \bar{p}\cdot d & \bar{p}\cdot\bar{d} & 0 \\
0 & 0 & \bar{p}\cdot\bar{d}\cdot\pi_{FZ,CA} & 0 & \bar{p}\cdot\bar{d}\cdot\pi_{FZ,TR} & \pi_{FZ,OFF} & 0 & \bar{p}\cdot d & 0 & \bar{p}\cdot\bar{d}\cdot\pi_{FZ,OK} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}.$$

The arrangement of the columns corresponds to the states arrangement: SS', SS, CA, DR, TR, OFF, MS, SNS, FZ, and OK.

$\pi_{FZ,CA} = \pi_{DR,CA}$,

$\pi_{FZ,TR} = \pi_{DR,TR}$,

$\pi_{FZ,OFF} = \pi_{SNS,OFF} = p$, and

$\pi_{FZ,OK} = \pi_{DR,OK}$.

4.3 Capturing Proposed Modifications to TCP

To capture our proposed modifications to TCP, we used the transition probability matrix that incorporates LU interruption and spectrum sensing (the matrix in Section 4.2.2). We chose this matrix because it captured LU interruption and spectrum sensing explicitly.

Recall that in our proposed design (Section 3.2), we use the measured Expected Available Bandwidth (EAB) as the value of w_n whenever connection resumes from LU interruption. To capture this modification, we made the following changes to the w_n value update in the MS state (MS is the state where transmission goes to whenever an LU frees the channel or a new channel is found). When in the MS

state, instead of setting the value of w_n to $w_n/2$ as in the DR state, we set the value of w_n to EAB. This will stop TCP sender from setting its sending window to 1 MSS whenever an LU interrupts transmission, but rather to the measured expected available bandwidth.

Recall that in our proposed design, we freeze the value of w_n whenever the connection goes into sensing mode (Section 3.2). To capture this modification, we made the following changes to the w_n value update in the FZ state (FZ is the state where transmission goes to whenever spectrum sensing is over). When in the FZ state, instead of setting the value of w_n to $w_n/2$ as in the DR state, we keep the value of w_n as it is. This will stop the TCP sender from setting its sending window to 1 MSS whenever one of the nodes goes into sensing mode, but rather keep the current window size.

4.4 Performance Measures

We developed MATLAB programs to compute the entries of the transition probability matrices we discussed in the previous sections (Sections 2.3, 4.1, and 4.2). We assigned hypothetical values to the following parameters: $p_b, p_f, \varphi, w_{rcv}$, and w_{thresh}. For all the programs we fixed the maximum number of segments the receiver can receive at a time to 16 segments ($w_{rcv} = 16$). Loss probabilities p_f and p_b are varied from 0.1 to 0.5. We consider files of three different sizes, a small file of size 2 KB, a little bit larger file of size 20 KB, and a larger file of size 200 KB. Arvidsson and Krzesinski [4] limited their model to geometrically distributed file lengths. The average size of file is $1/(1-\varphi)$ packets, where φ is the parameter for the distribution.

Since $MSS = 536$ octets [21], then there will be an average of $\frac{2\times1024}{536}$, $\frac{20\times1024}{536}$ packets, and $\frac{200\times1024}{536}$ packets in 2 KB file, 20KB file, and 200 KB file respectively. For 2 KB file, $\varphi = 1 - 1/3.82 \approx 0.74$. For 20 KB file, $\varphi = 1 - 1/38.2 \approx 0.97$. For 200 KB file, $\varphi = 1 - 1/382.1 \approx 0.997$. We begin each program with w_{thresh} initially set to the value of w_{rcv} which is equal to 16 ($w_{thresh} = w_{rcv} = 16$). All the programs iterate and update the value of w_{thresh} after each iteration according to the w_{thresh} update formula provided by Arvidsson and Krzesinski [4] as follows:

$$w_{thresh} = \frac{g_{SS}}{g_{SS} + g_{CA}} w_{thresh}^{SS} + \frac{g_{CA}}{g_{SS} + g_{CA}} w_{thresh}^{CA}, \tag{4.1}$$

where g_{SS} is the number of losses in the SS state, g_{CA} is the number of losses in the CA state, and

$$w_{thresh}^{SS} = max(2, W_{SS}/2), \tag{4.2}$$

$$w_{thresh}^{CA} = max(2, W_{CA}/2). \tag{4.3}$$

W_{SS} is the average window size at which the first loss occurs in the SS state and W_{CA} is the average window size at which the first loss occurs in the CA state.

Equation 4.1 is a fixed point equation. It can be solved by a one dimensional root finding method. We used a bisection method to solve it in our programs.

We run our programs iteratively. Iterations stop when the difference between the previous w_{thresh} value and the current is less than 10^{-6}.

Our MATLAB programs compute the entries of the transition probability matrices. The display format used is "long", which rounds and display results up to 15

decimal places. For convenience, the values displayed in the matrices presented later in this chapter are rounded to 4 decimal places. However, the original values (values with 15 decimal places) are used to compute all the performance measures considered in this thesis. The performance measures computed are presented in tables and the values are rounded to 3 decimal places for better visual presentation.

Remember that OK state is an absorbing state. Hence, the matrices our programs will compute are going to be absorbing Markov chains. Recall that a transition probability matrix P for an absorbing Markov chain can be written in the form:

$$P = \begin{bmatrix} Q & H \\ 0 & \mathbf{1} \end{bmatrix},$$

where,

Q is a square matrix that contains the transition probabilities within the transient states,

H contains transition probabilities from the transient states to the absorbing state,

$\mathbf{1}$ is a column vector, and

0 is a zero matrix.

Based on the absorbing Markov chains, we computed performance measures that include, average number of visits to the TR state before absorption given that transmission starts from SS', Time to Complete Transmission (TCT), total number of re-transmissions during an average transmission, and average number of visits to the OFF and the SNS states.

To validate the results obtained by running our programs, a comparison can be made with the results obtained by Arvidsson and Krzesinski [4]. This will validate the results obtained by running the unmodified TCP program. Arvidsson and Krzesinski derived formulas to compute average number of visits to TR state. We compute average number of visits to TR state differently by computing the fundamental matrix of the Markov chain. We compare the results obtained by the fundamental matrix with the results obtained by using the formulas derived by Arvidsson and Krzesinski. These two results are almost identical. However, Arvidsson and Krzesinski presented only one performance measure, the average download time. This value is computed based using the Round Time Trip (RTT), which is not applicable in our model. Therefore, we cannot compute the average download time as they did.

4.4.1 Average Number of Visits to TR State

Let us consider the transition probability matrix by Arvidsson and Krzesinski [4] (See Section 2.3). We have seen earlier that transmission experiences the highest delay when it enters the TR state. This is because, whenever transmission enters the TR state, the congestion window is reduced to 1 MSS. Hence, computing the average number of visits to the TR state will give us the average number of times the congestion window (w) is reduced to 1 MSS.

We compute the fundamental matrix for the absorbing transition probability matrix above using the formula:

$$F = (I - Q)^{-1}. \tag{4.4}$$

We assumed that transmission always begins in the SS' state. Therefore, we consider only the (SS', TR) entry of F. This will give us the average number of visits to the TR state since when transmission begins.

4.4.2 Time to Complete Transmission

We have seen that the OK state is the only absorbing state in our transition probability matrices. We also know that the OK state is visited only when transmission is complete. Therefore, if we compute the mean time to absorption, it will give us the mean time for transmission completion. We can compute the mean time to absorption using the formula:

$$\hat{d} = a(I - Q)^{-2}H, \tag{4.5}$$

where a is a vector for which each of its entries a_i represent the probability of starting transmission from the transient state i. Our interest is to compute the mean time to absorption from the moment transmission begins. We assume that transmission always begins in the SS' state. Therefore, in all our programs, we set the first entry of a to 1, and set all the remaining entries to 0. The model we based our model on did not specify any unit for time. The time unit depends on the unit of the sending rate used during implementation.

4.4.3 Total Number of Re-transmissions

Total number of packets retransmission increases as loss probability increases. This is because, more packets will get dropped and hence need to be retransmitted.

Our interest is to see by how much CRN features increase the number of packets retransmitted during a file transfer. In other words, this will give us a measure of the additional packets that get dropped due to CRN features. We use the average number of visits to each state and the average number of packets transmitted in each state to compute the average number of packets transmitted in each state during a file transfer. The total number of packets transmitted successfully N_{pts} can be computed using the formula below:

$$N_{pts} = \sum_i N_i V_i, \ \forall i \in \{SS', SS, CA\}, \tag{4.6}$$

where N_i is the average number of packets transmitted while in state i and V_i is average number of visits to state i. The N_is and the V_is were defined in Section 2.3.

Similarly, the total number of packets retransmitted N_{tpr} using the formula below:

$$N_{tpr} = \sum_i N_i V_i, \ \ \forall i \in \{DR, TR, MS, FZ\}, \tag{4.7}$$

where $N_{MS} = N_{FZ} = N_{DR} = 1$ and $N_{TR} = 1/r$ (defined in Section 2.3).

4.4.4 Average Number of Visits to the OFF and the SNS States

Let us consider the transition probability matrix in Section 4.2.2. Recall that this matrix captured LU interruption and spectrum sensing separately. Computing the average number of visits to the OFF state will give us the average number of times LUs interrupt transmission. Similarly, computing the average number of visits to

the SNS state will give us the average number of times spectrum sensing interrupts transmission.

We compute the fundamental matrix for the absorbing transition probability matrix above using the formula:

$$F = (I - Q)^{-1}. \tag{4.8}$$

We assumed that transmission always begins in the SS' state. Therefore, the (SS', OFF) entry of F is the average number of visits to the OFF state since when the transmission began. The average number of visits to the SNS state since when the transmission began is the (SS', SNS) entry of F.

4.5 TCP

We begin with the transition probability matrix by Arvidsson and Krzesinski [4]. Remember that their model is for TCP without any of the cognitive radio network features. We set $p_f = p_b = 0.01$, $w_{rcv} = w_{thresh} = 16$, and run our MATLAB program designed for normal TCP without any cognitive radio network features. The program ran for files of sizes 2 KB, 20KB, and 200 KB.

4.5.1 Transmission of a 2 KB File

For file of size 2 KB, we have $\frac{2\times1024}{536}$ packets. $\varphi = 1 - 1/3.82 \approx 0.74$. The program computed the transition probability matrix below:

$$P = \begin{bmatrix} 0 & 0 & 0 & 0 & 0.0372 & 0.9628 \\ 0 & 0 & 0.1114 & 0 & 0.0330 & 0.8556 \\ 0 & 0 & 0 & 0.0275 & 0.0097 & 0.9628 \\ 0 & 0 & 0.7236 & 0 & 0.0199 & 0.2565 \\ 0 & 0.7383 & 0 & 0 & 0 & 0.2617 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

From the transition probability matrix P, we can see that,

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0.0372 \\ 0 & 0 & 0.1114 & 0 & 0.0330 \\ 0 & 0 & 0 & 0.0275 & 0.0097 \\ 0 & 0 & 0.7236 & 0 & 0.0199 \\ 0 & 0.7383 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$H = \begin{bmatrix} 0.9628 \\ 0.8556 \\ 0.9628 \\ 0.2565 \\ 0.2617 \end{bmatrix},$$

$$0 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\mathbf{1} = \begin{bmatrix} 1 \end{bmatrix}.$$

Table 4.1: Original TCP: Transmission of a 2 KB File

Loss Probability	**0.01**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
Avg. # of Visits to TR	0.038	0.373	0.733	1.109	1.496	1.888
Time to Complete Tx[1]	1.070	1.722	2.603	3.454	4.203	4.904
Total # of re-Tx	0.039	0.472	1.194	2.339	4.245	7.642

We set $a = [1, 0, 0, 0, 0]$, and compute average number of visits to TR, time to complete transmission, and total number of re-transmissions.

Varying the packet loss probability from 0.01 to 0.5, we obtained the values in Table 4.1. As expected, we can see that as the packet loss probability increases, the average number of visits to the TR state increases as well. Remember that the more the packet losses, the more the number of time-outs and hence more visits to the TR state. The time to complete transmission and the total number of re-transmissions were also computed.

4.5.2 Transmission of a 20 KB File

For file of size 20 KB, we have $\frac{20\times1024}{536}$ packets. $\varphi = 1 - 1/38.2 \approx 0.97$. The program computed the transition probability matrix below:

[1]Transmission is abbreviated as Tx in tables.

Table 4.2: Original TCP: Transmission of a 20 KB File

Loss Probability	**0.01**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
Avg. # of Visits to TR	0.088	2.468	6.587	10.930	15.056	19.061
Time to Complete Tx	1.826	10.552	21.485	31.436	40.138	47.877
Total # of re-Tx	0.390	4.717	11.940	23.393	42.454	76.418

$$P = \begin{bmatrix} 0 & 0 & 0 & 0.2249 & 0.0536 & 0.7215 \\ 0 & 0 & 0.7741 & 0.0352 & 0.0277 & 0.1630 \\ 0 & 0 & 0 & 0.2050 & 0.0734 & 0.7215 \\ 0 & 0 & 0.9544 & 0 & 0.0199 & 0.0257 \\ 0 & 0.9738 & 0 & 0 & 0 & 0.0262 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

We vary the packet loss probability from 0.01 to 0.5 and we obtained the values in Table 4.2. Similarly, as expected, we can see that as the packet loss probability increases, the average number of visits to the TR state increases as well. We also compute the time to complete transmission and the total number of re-transmissions.

4.5.3 Transmission of a 200 KB File

For file of size 200 KB, we have $\frac{200\times1024}{536}$ packets. $\varphi = 1 - 1/382.1 \approx 0.997$. The program computed the transition probability matrix below:

Table 4.3: Original TCP: Transmission of a 200 KB File

Loss Probability	**0.01**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
Avg. # of Visits to TR	0.820	25.449	66.108	109.024	150.799	190.999
Time to Complete Tx	9.441	100.565	210.866	311.703	399.899	477.727
Total # of re-Tx	3.898	47.172	119.403	233.932	424.544	764.179

$$P = \begin{bmatrix} 0 & 0 & 0 & 0.6449 & 0.1493 & 0.2058 \\ 0 & 0 & 0.9151 & 0.0384 & 0.0290 & 0.0175 \\ 0 & 0 & 0 & 0.6375 & 0.1567 & 0.2058 \\ 0 & 0 & 0.9775 & 0 & 0.0199 & 0.0026 \\ 0 & 0.9974 & 0 & 0 & 0 & 0.0027 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

We vary the packet loss probability from 0.01 to 0.5 and we obtained the values in Table 4.3. Similarly, as expected, we can see that as the packet loss probability increases, the average number of visits to the TR state increases as well. The time to complete transmission and the total number of re-transmissions were also computed.

4.6 Original TCP with LU Interruption and Spectrum Sensing

Here, we consider the original TCP model with LU interruption and spectrum sensing (the model in Section 4.1). We set the probability of an LU interrupting transmission to 0.4 ($p = 0.4$), the probability that a node is in sensing mode to 0.167 ($d = 0.306$). The choice of d is based on the optimal sensing time and transmission

time pair [10, 17]. We set $p_f = p_b = 0.1$, $w_{rcv} = w_{thresh} = 16$, and run our MATLAB program designed for normal TCP with LU interruption and sensing effect. The program was ran for a file of size 2 KB, of size 20KB, and of size 200 KB.

4.6.1 Transmission of a 2 KB File

For file of size 2 KB, we set $\varphi = 0.74$. Below is the transition probability matrix computed by the program.

$$P = \begin{bmatrix} 0 & 0 & 0 & 0 & 0.8448 & 0.1552 \\ 0 & 0 & 0.3045 & 0 & 0.5875 & 0.1080 \\ 0 & 0 & 0 & 0.1678 & 0.6769 & 0.1552 \\ 0 & 0 & 0.3015 & 0 & 0.5916 & 0.1069 \\ 0 & 0.7383 & 0 & 0 & 0 & 0.2617 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Varying the packet loss probability from 0.01 to 0.5 we compute the average number of visits to the TR state, the time to complete transmission, and the total number of re-transmissions (See Table 4.4).

Table 4.4: Original TCP with LU Interruption and Sensing: Transmission of a 2 KB File

Loss Probability	**0.01**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
Avg. # of Visits to TR	2.209	2.360	2.527	2.692	2.856	3.019
Time to Complete Tx	5.451	5.689	5.944	6.188	6.423	6.649
Total # of re-Tx	5.497	7.076	9.552	13.256	19.104	29.039

4.6.2 Transmission of a 20 KB File

For file of size 20 KB, we set $\varphi = 0.97$. The program computed the transition probability matrix below:

$$P = \begin{bmatrix} 0 & 0 & 0 & 0 & 0.9820 & 0.0180 \\ 0 & 0 & 0.4017 & 0 & 0.5875 & 0.0108 \\ 0 & 0 & 0 & 0 & 0.9820 & 0.0180 \\ 0 & 0 & 0.3977 & 0 & 0.5916 & 0.0107 \\ 0 & 0.9738 & 0 & 0 & 0 & 0.0262 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Varying the packet loss probability we computed the average number of visits to the TR state, the time to complete transmission, and the total number of re-transmissions (See Table 4.5).

Table 4.5: Original TCP with LU Interruption and Sensing: Transmission of a 20 KB File

Loss Probability	**0.01**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
Avg. # of Visits to TR	22.448	23.881	25.473	27.065	28.657	30.249
Time to Complete Tx	54.089	56.629	59.331	61.907	64.358	66.682
Total # of re-Tx	54.969	70.757	95.522	132.562	191.045	290.388

4.6.3 Transmission of a 200 KB File

For file of size 200 KB, we set $\varphi = 0.997$. The program computed the transition probability matrix below:

$$P = \begin{bmatrix} 0 & 0 & 0 & 0 & 0.9982 & 0.0018 \\ 0 & 0 & 0.4114 & 0 & 0.5875 & 0.0011 \\ 0 & 0 & 0 & 0 & 0.9982 & 0.0018 \\ 0 & 0 & 0.4073 & 0 & 0.5916 & 0.0011 \\ 0 & 0.9974 & 0 & 0 & 0 & 0.0026 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

.

Varying the packet loss probability we computed the average number of visits to the TR state, the time to complete transmission, and the total number of re-transmissions (See Table 4.6).

Table 4.6: Original TCP with LU Interruption and Sensing: Transmission of a 200 KB File

Loss Probability	**0.01**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
Avg. # of Visits to TR	224.478	238.806	254.726	270.647	286.567	302.488
Time to Complete Tx	541.481	567.071	594.251	620.111	644.652	667.872
Total # of re-Tx	549.685	707.573	955.224	1325.617	1910.448	2903.881

4.7 Modified TCP with LU Interruption and Spectrum Sensing

Here, we consider our modified TCP model that captured LU interruption and sensing separately (the model in Section 4.2.2). We set the probability of an LU interrupting transmission to 0.4 ($p = 0.4$), the probability that one of the nodes is in sensing mode to 0.167 ($d = 0.306$), $p_f = p_b = 0.01$, $w_{rcv} = w_{thresh} = 16$, and run

our MATLAB program designed for modified TCP with LU interruption and sensing. The program was ran for a file of size 2 KB, of size 20KB, and of size 200 KB.

4.7.1 Transmission of a 2 KB File

For file of size 2 KB, we set $\varphi = 0.74$. The program computed the transition probability matrix below:

$$P = \begin{bmatrix} 0 & 0 & 0 & 0 & 0.0155 & 0.4 & 0 & 0.1833 & 0 & 0.4012 \\ 0 & 0 & 0.0464 & 0 & 0.0138 & 0.4 & 0 & 0.1833 & 0 & 0.3565 \\ 0 & 0 & 0 & 0.0114 & 0.0040 & 0.4 & 0 & 0.1833 & 0 & 0.4012 \\ 0 & 0 & 0.3015 & 0 & 0.0083 & 0.4 & 0 & 0.1833 & 0 & 0.1069 \\ 0 & 0.3076 & 0 & 0 & 0 & 0.4 & 0 & 0.1833 & 0 & 0.1090 \\ 0 & 0 & 0 & 0 & 0 & 0.4 & 0.6 & 0 & 0 & 0 \\ 0 & 0 & 0.3015 & 0 & 0.0083 & 0.4 & 0 & 0.1833 & 0 & 0.1069 \\ 0 & 0 & 0 & 0 & 0 & 0.4 & 0 & 0.1833 & 0.4167 & 0 \\ 0 & 0 & 0.3015 & 0 & 0.0083 & 0.4 & 0 & 0.1833 & 0 & 0.1069 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Varying the packet loss probability we compute the average number of visits to the OFF state, the average number of visits to the SNS state, average number of visits to the TR state, total number of re-transmissions, and the time to complete transmission. The values we obtained are shown in Table 4.7.

Table 4.7: Modified TCP with LU Interruption and Sensing: Transmission of a 2 KB File

Loss Probability	**0.01**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
Avg. # of Visits to OFF	3.610	5.258	7.417	9.779	12.293	14.942
Avg. # of Visits to SNS	0.993	1.446	2.0340	2.689	3.380	4.109
Avg. # of Visits to TR	0.040	0.496	1.201	2.069	3.085	4.226
Time to Complete Tx	9.025	13.146	18.543	24.447	30.732	37.354
Total # of re-Tx	2.630	4.435	7.254	11.293	17.437	27.655

4.7.2 Transmission of a 20 KB File

For file of size 20 KB, we set $\varphi = 0.97$. The program computed the transition probability matrix below:

$$P = \begin{bmatrix} 0 & 0 & 0 & 0.0937 & 0.0223 & 0.4 & 0 & 0.1833 & 0 & 0.3006 \\ 0 & 0 & 0.3226 & 0.0147 & 0.0115 & 0.4 & 0 & 0.1833 & 0 & 0.0679 \\ 0 & 0 & 0 & 0.0854 & 0.0306 & 0.4 & 0 & 0.1833 & 0 & 0.3006 \\ 0 & 0 & 0.3977 & 0 & 0.0083 & 0.4 & 0 & 0.1833 & 0 & 0.0107 \\ 0 & 0.4058 & 0 & 0 & 0 & 0.4 & 0 & 0.1833 & 0 & 0.0109 \\ 0 & 0 & 0 & 0 & 0 & 0.4 & 0.6 & 0 & 0 & 0 \\ 0 & 0 & 0.3977 & 0 & 0.0083 & 0.4 & 0 & 0.1833 & 0 & 0.01069 \\ 0 & 0 & 0 & 0 & 0 & 0.4 & 0 & 0.1833 & 0.4167 & 0 \\ 0 & 0 & 0.3977 & 0 & 0.0083 & 0.4 & 0 & 0.1833 & 0 & 0.0107 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Varying the packet loss probability we compute the average number of visits to the OFF state, the average number of visits to the SNS state, average number of visits to the TR state, total number of re-transmissions, and the time to complete transmission. The values we obtained are shown in Table 4.8.

Table 4.8: Modified TCP with LU Interruption and Sensing: Transmission of a 20 KB File

Loss Probability	**0.01**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
Avg. # of Visits to OFF	7.018	30.694	58.921	87.698	116.887	146.208
Avg. # of Visits to SNS	1.930	8.441	16.203	24.117	32.144	40.207
Avg. # of Visits to TR	0.1312	3.621	10.739	19.900	30.531	42.273
Time to Complete Tx	17.546	76.735	147.302	219.245	292.218	365.520
Total # of re-Tx	5.425	27.834	60.196	104.143	168.793	273.682

4.7.3 Transmission of a 200 KB File

For file of size 200 KB, we set $\varphi = 0.997$. The program computed the transition probability matrix below:

$$P = \begin{bmatrix} 0 & 0 & 0 & 0.2687 & 0.0622 & 0.4 & 0 & 0.1833 & 0 & 0.0857 \\ 0 & 0 & 0.3813 & 0.0160 & 0.0121 & 0.4 & 0 & 0.1833 & 0 & 0.0073 \\ 0 & 0 & 0 & 0.2656 & 0.0653 & 0.4 & 0 & 0.1833 & 0 & 0.0857 \\ 0 & 0 & 0.4073 & 0 & 0.0083 & 0.4 & 0 & 0.1833 & 0 & 0.0011 \\ 0 & 0.4156 & 0 & 0 & 0 & 0.4 & 0 & 0.1833 & 0 & 0.0011 \\ 0 & 0 & 0 & 0 & 0 & 0.4 & 0.6 & 0 & 0 & 0 \\ 0 & 0 & 0.4073 & 0 & 0.0083 & 0.4 & 0 & 0.1833 & 0 & 0.0011 \\ 0 & 0 & 0 & 0 & 0 & 0.4 & 0 & 0.1833 & 0.4167 & 0 \\ 0 & 0 & 0.4073 & 0 & 0.0083 & 0.4 & 0 & 0.1833 & 0 & 0.0011 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Varying the packet loss probability we compute the average number of visits to the OFF state, the average number of visits to the SNS state, average number of visits to the TR state, total number of re-transmissions, and the time to complete transmission. The values we obtained are shown in Table 4.9.

Table 4.9: Modified TCP with LU Interruption and Sensing: Transmission of a 200 KB File

Loss Probability	**0.01**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
Avg. # of Visits to OFF	30.679	279.059	570.554	865.484	1162.575	1459.041
Avg. # of Visits to SNS	8.437	76.741	156.902	238.008	319.708	401.236
Avg. # of Visits to TR	0.947	35.268	106.222	197.978	305.096	422.970
Time to Complete Tx	76.699	697.647	1426.384	2163.709	2906.436	3647.603
Total # of re-Tx	25.902	257.417	587.001	1031.330	1682.255	2734.603

4.8 Discussion

In this section, we discuss the effects of LU interruption and spectrum sensing on TCP based on the performance measures computed in the previous sections. We begin with the overall effects of LU interruption and spectrum sensing on original TCP. We later discuss how exactly LU interruption and spectrum sensing affect TCP explicitly. Discussion on the proposed modifications to TCP follows. Finally, we make an overall comparison of all the TCP models we discussed in this chapter. In the figures presented in this chapter we used the following labels for the TCP models:

- Normal TCP : The original TCP without any cognitive radio network feature.
- Normal TCP with LU interruption and sensing: The original TCP with LU interruption and sensing (the model in Section 4.1).
- Modified TCP: The modified TCP with LU interruption and sensing (the model in Section 4.2.2).

4.8.1 Effects of LU Interruption and Spectrum Sensing

Both LU interruption and spectrum sensing caused packet losses. TCP considered losses caused by these cognitive radio network features as congestion losses. TCP

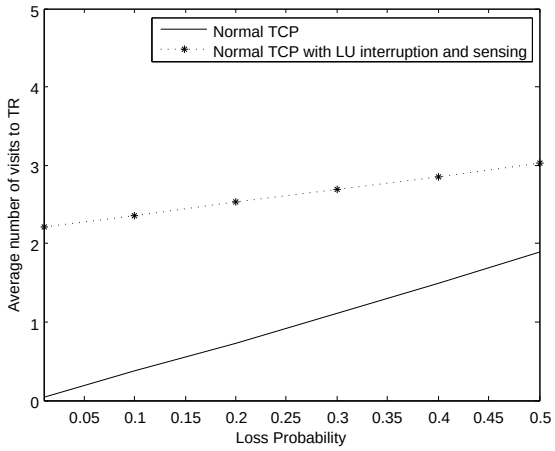
decreases its congestion window in the event of losses caused by either LU interruption or spectrum sensing. Packet losses lead to increase in the average number of visits to the TR state, increase in the time to complete transmission, and also increase in the total number of re-transmissions. Here, we show by graphs the effects of LU interruption and spectrum sensing on the original TCP. We considered files of 2KB size, 20KB size, and 200KB size.

Average Number of Visits to TR

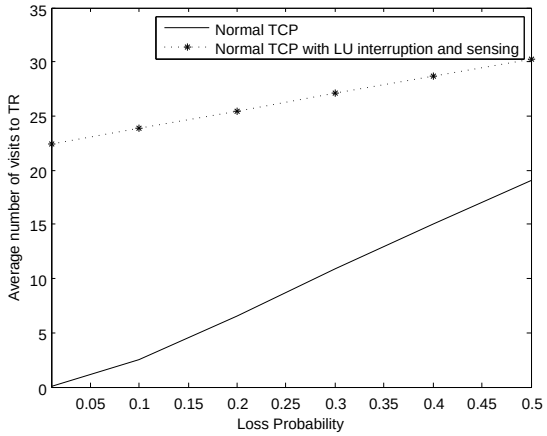
Figure 4.1 shows a comparison in terms of average number of visits to TR state between an original TCP without any of the CRN features and an original TCP with LU interruption and spectrum sensing. From all the graphs (Figure 4.1(a), Figure 4.1(b), and Figure 4.1(c)), we can see how LU interruption and spectrum sensing increase the average number of visits to the TR state. The reason for the increase in average number of visits to TR state is because LU arrival interrupts transmission and spectrum sensing lead to time-outs. Remember that for every time-out TCP transmission switches to the TR state.

Time to Complete Transmission

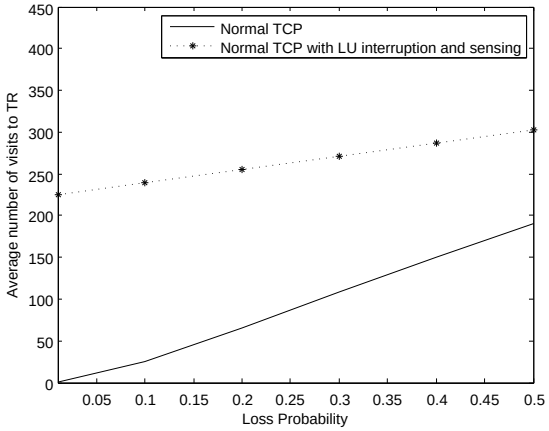
Figure 4.2 shows a comparison in terms of time to complete transmission between an original TCP without any of the CRN features and an original TCP with LU interruption and spectrum sensing. Similar to the case of average number of visits to TR, here also, we can see how LU interruption and spectrum sensing increase the time to complete transmission. This is because interruptions by LUs and spectrum sensing caused more packet losses and hence transmission takes more time to complete.



(a) Transmission of a 2KB file.



(b) Transmission of a 20KB file.

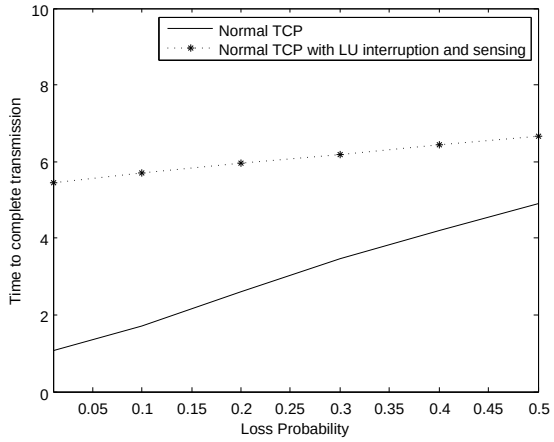


(c) Transmission of a 200KB file.

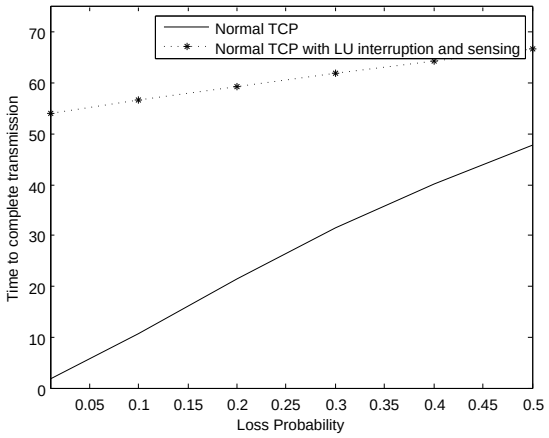
Figure 4.1: Average Number of Visits to TR State: Normal TCP vs TCP with LU interruption and spectrum sensing.

Number of Re-transmissions

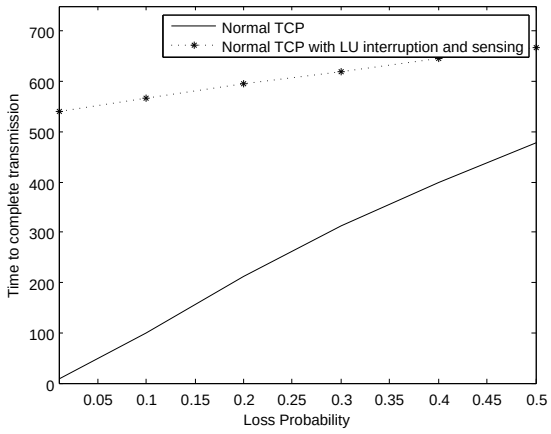
Figure 4.3 shows a comparison in terms of total number of re-transmissions between an original TCP without any of the CRN features and an original TCP with LU interruption and spectrum sensing. Here also, LU interruption and spectrum sensing increase the total number of re-transmissions. This is because these factors caused



(a) Transmission of a 2KB file.



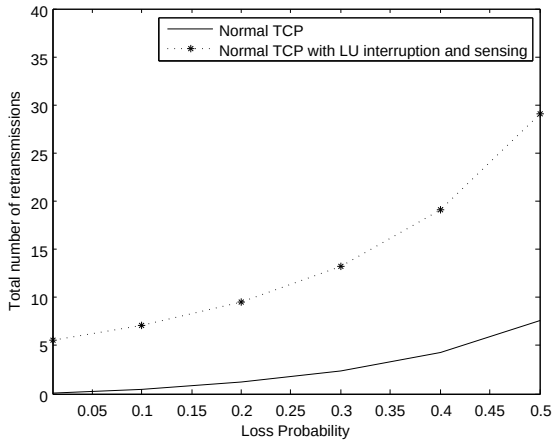
(b) Transmission of a 20KB file.



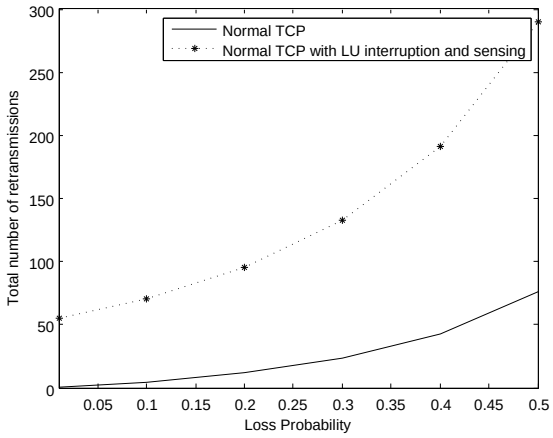
(c) Transmission of a 200KB file.

Figure 4.2: Time to Complete Transmission: Normal TCP vs TCP with LU interruption and spectrum sensing.

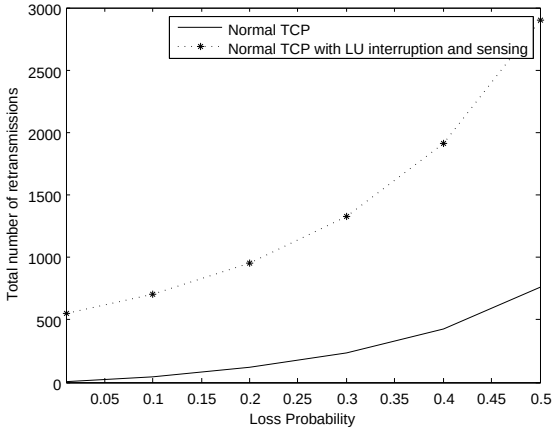
more packet losses and hence more re-transmissions.



(a) Transmission of a 2KB file.



(b) Transmission of a 20KB file.



(c) Transmission of a 200KB file.

Figure 4.3: Number of Re-transmissions: Normal TCP vs TCP with LU interruption and spectrum sensing.

4.8.2 Explicit Effects of LU Interruption and Spectrum Sensing on TCP

Here, we discuss the effects of LU interruption and spectrum sensing on TCP explicitly. The modified TCP model differentiates losses caused by LU interruption and spectrum sensing from losses caused by congestion in the network. We show by

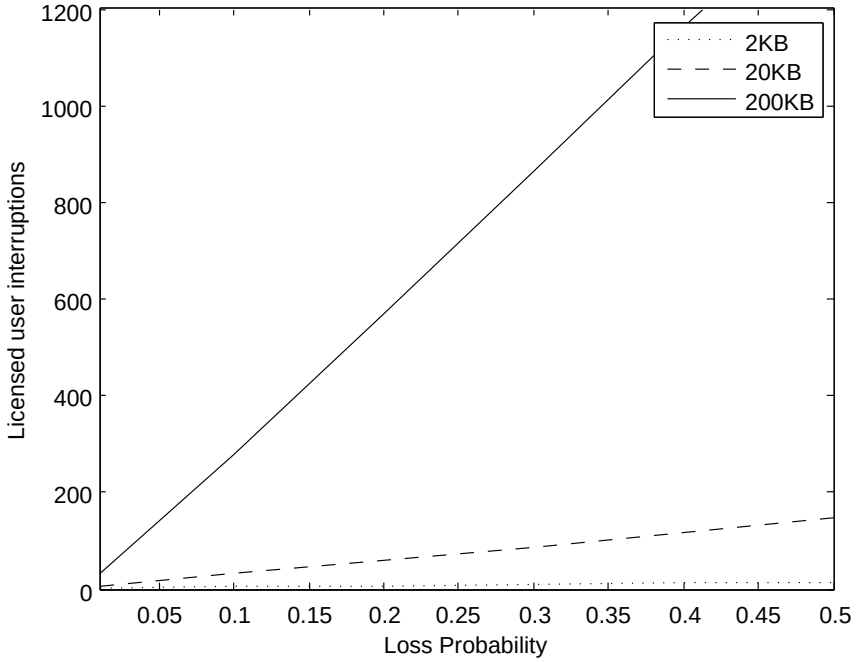


Figure 4.4: Licensed user interruptions

graphs how LU arrivals and spectrum sensing interrupt transmission as loss probability varies. We also compare the original TCP with LU interruption and spectrum sensing to the modified TCP that captured LU interruption and spectrum sensing. We consider three file sizes; a 2KB file, a 20KB file, and a 200KB file.

LU Interruption

Figure 4.4 shows licensed user interruptions as loss probability varies. We can see that as loss probability increases the licensed user interruption increases as well. This is because increase in loss probability leads to increase in the time to complete transmission. As a result, the sender and the receiver need the services of the licensed channels for a longer period of time. The more time transmission remain on the

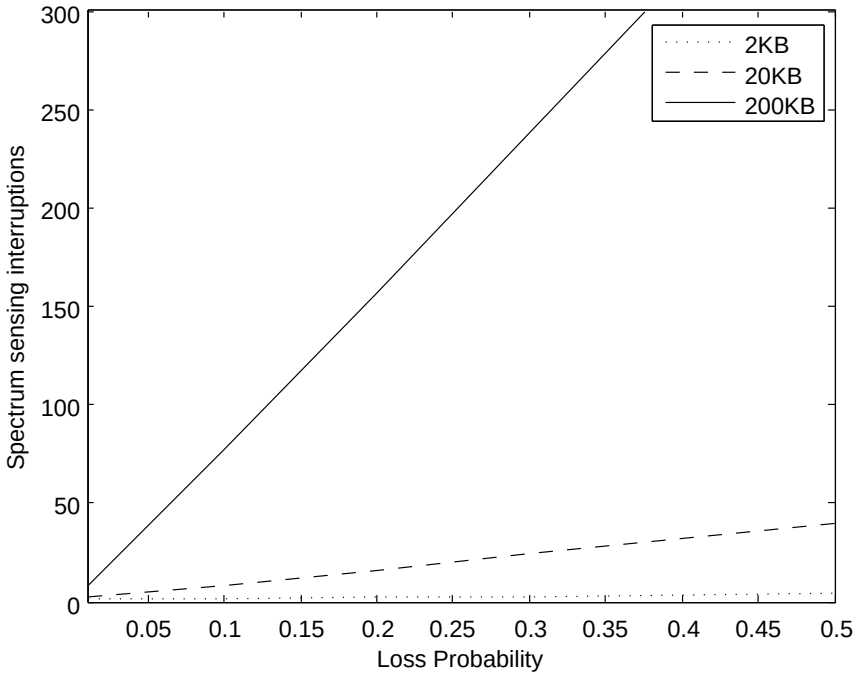


Figure 4.5: Spectrum sensing interruptions

licensed channels, the higher the interruptions by LUs. We can also see that the larger the file size, the higher the licensed user interruption. This is also because larger files require more time to complete transmission.

Spectrum Sensing

Figure 4.5 shows spectrum sensing interruptions as loss probability is varied. We can see that as loss probability increases the spectrum sensing interruption increases as well. Similar to the case of LU arrival, this is because increase in loss probability leads to increase in the time to complete transmission. Since spectrum sensing is periodic then the more time transmission is in progress the higher the interruptions caused by spectrum sensing. Here also, we can see that the larger the file size, the

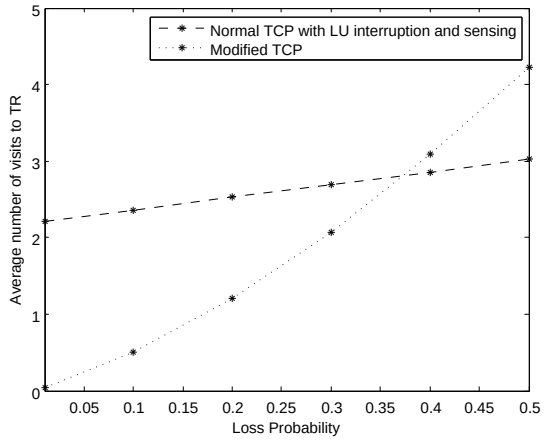
higher the spectrum sensing interruption. This is also because larger files require more time to complete transmission.

4.8.3 Proposed Modifications to TCP

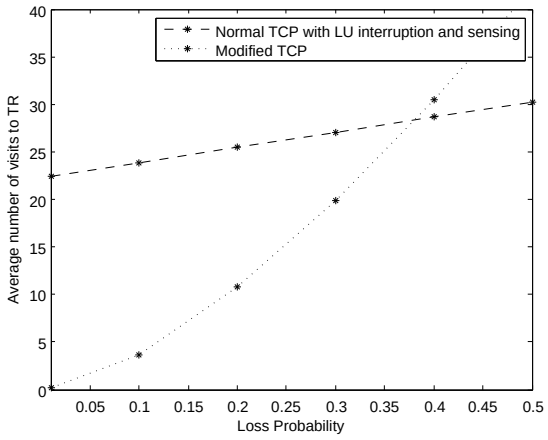
Our proposed modifications to TCP are to make sure that the TCP sender does not reduce it's congestion window value to 1 MSS when either an LU arrived or spectrum sensing interrupts transmission. Remember that congestion window value is set to 1 MSS when transmission goes to the TR state. To get a measure of how much the proposed modifications to TCP reduces the number of visits to the TR state, we compute the average number of visits to the TR state. We compare the average number of visits to the TR state for the original TCP with LU and spectrum sensing and the TCP model that captured the proposed modifications. We also compare the two in terms of number of re-transmissions and the time to complete transmission. Similar to all the evaluations in this work, we consider files of sizes 2KB, 20KB, and 200KB.

Average Number of Visits to TR

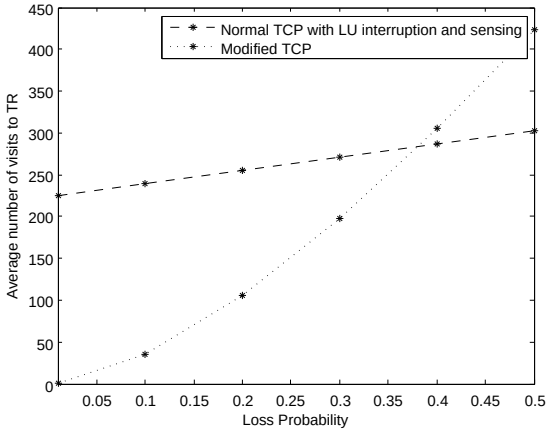
Figure 4.6 shows a comparison in terms of average number of visits to TR between an original TCP with LU interruption and spectrum sensing and the TCP model that captured the proposed modifications. From the figure we can see that the proposed modifications were able to reduce the average number of visits to the TR state compared to the the original TCP with LU interruption and spectrum sensing. However, when loss probability becomes greater than 0.35 (more than 35% packet loss), the



(a) Transmission of a 2KB file.



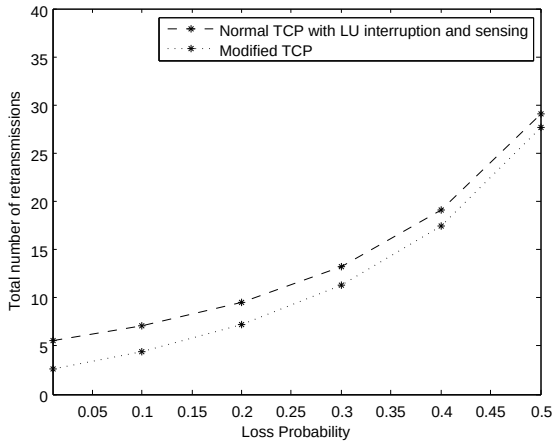
(b) Transmission of a 20KB file.



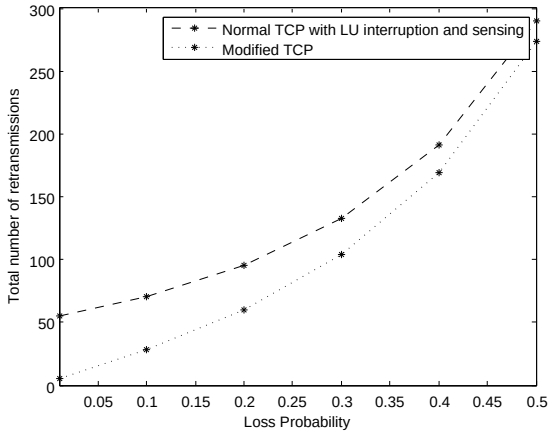
(c) Transmission of a 200KB file.

Figure 4.6: Average Number of Visits to TR State: Normal TCP vs TCP with LU interruption and spectrum sensing.

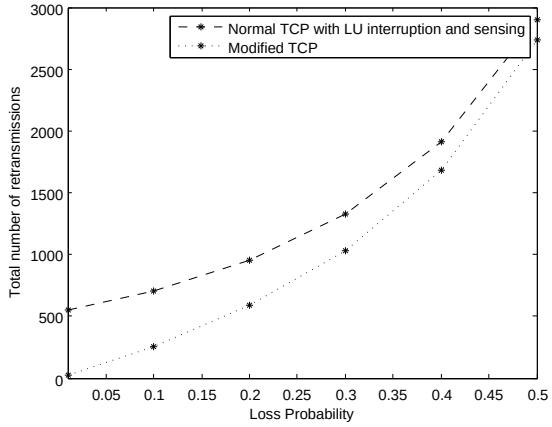
average number of visits to the TR state for the TCP with the proposed modifications becomes higher than that for the original TCP with LU interruption and spectrum sensing.



(a) Transmission of a 2KB file.



(b) Transmission of a 20KB file.

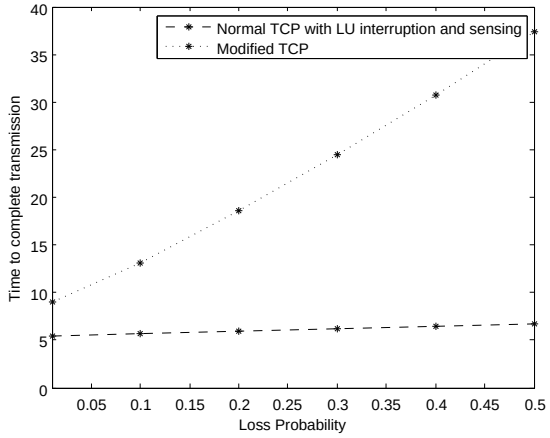


(c) Transmission of a 200KB file.

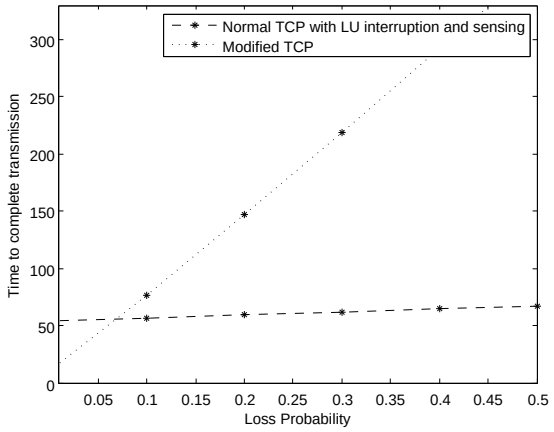
Figure 4.7: Number of Re-transmissions: Normal TCP vs TCP with LU interruption and spectrum sensing.

Number of Re-transmissions

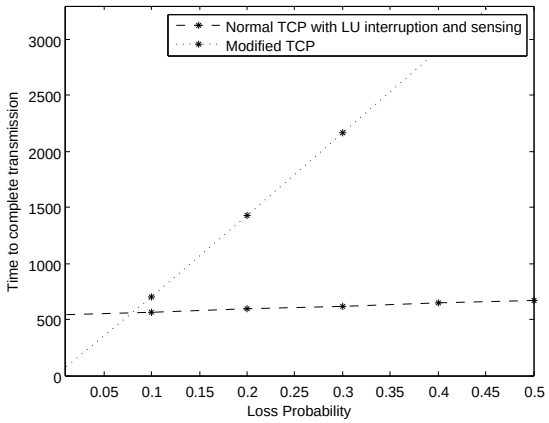
Figure 4.7 shows a comparison in terms of total number of re-transmissions between an original TCP with LU interruption and spectrum sensing and the modified TCP with LU interruption and spectrum sensing. From the figure, we can see that the TCP with the proposed modifications was able to reduce the number of



(a) Transmission of a 2KB file.



(b) Transmission of a 20KB file.



(c) Transmission of a 200KB file.

Figure 4.8: Time to Complete Transmission: Normal TCP vs TCP with LU interruption and spectrum sensing.

re-transmissions compared to the original TCP.

Time to Complete Transmission

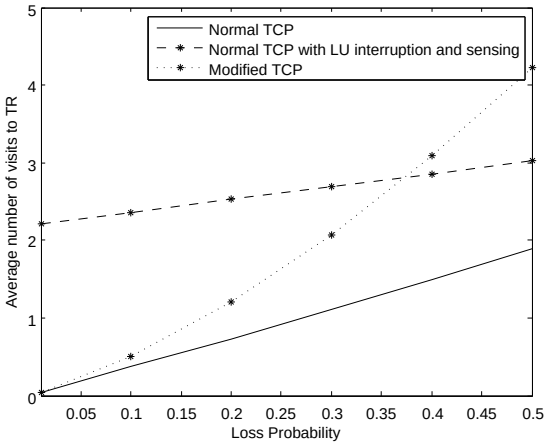
Figure 4.8 shows a comparison in terms of time to complete transmission between an original TCP with LU interruption and spectrum sensing and the modified TCP

with LU interruption and spectrum sensing. From Figure 4.8(b) and Figure 4.8(c) we can see that the modified TCP reduces the time to complete transmission until when loss probability reaches 0.09. This is because, the modified TCP model has to do an extra work to differentiate losses caused by congestion in the network from losses caused by either LU arrival or spectrum sensing.

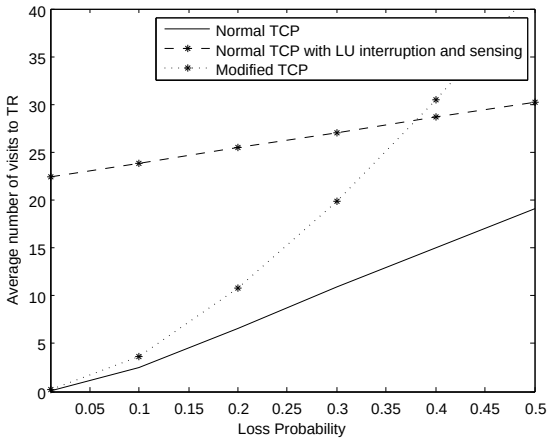
4.8.4 Overall Analysis

Average Number of Visits to the TR State

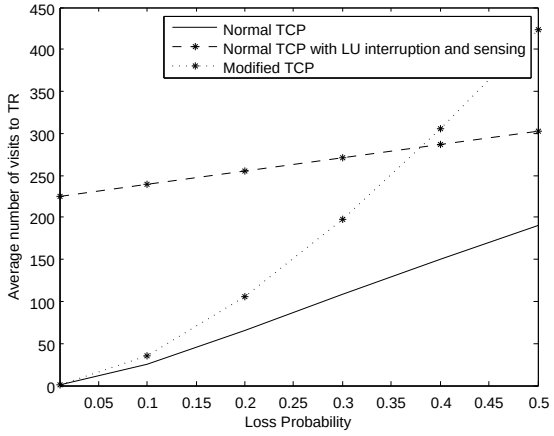
Figure 4.9 shows a comparison in terms of average number of visits to the TR state between an original TCP without CRN features, an original TCP with LU interruption and spectrum sensing, and TCP with the proposed modifications. Figure 4.9(a) is for a 2KB file transmission. From the figure, we can see that for all of the TCPs, the average number of visits to TR increases with increase in loss probability. This is because as loss probability increases, number of timeouts increases, and hence more visits to the TR state. As expected, TCP without any of the CRN features has the lowest average number of visits to the TR state. The TCP with the proposed modifications has lower average number of visits to the TR state than the original TCP with the CRN features. However, when loss probability reaches 0.35, the average number of visits to the TR state for the TCP with the proposed modifications becomes higher than the original TCP with the CRN features. We can also see the similar result in the case of a 20KB file and a 200KB file transmissions from Figure 4.9(b) and Figure 4.9(c), respectively.



(a) Transmission of a 2KB file.



(b) Transmission of a 20KB file.

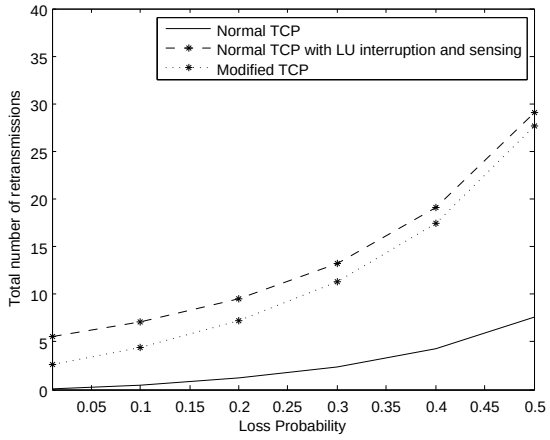


(c) Transmission of a 200KB file.

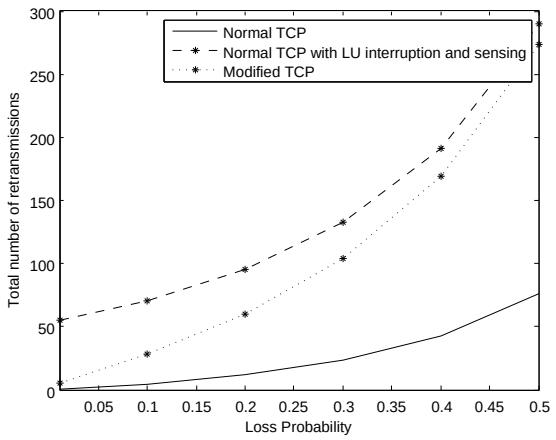
Figure 4.9: Average Number of Visits to TR State: Overall Analysis.

Number of Re-transmissions

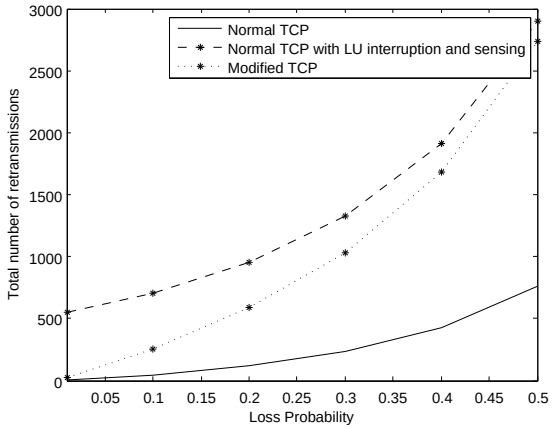
Figure 4.10 shows a comparison in terms of number of re-transmissions between an original TCP without CRN features, an original TCP with LU interruption and spectrum sensing, and TCP with the proposed modifications. For a 2KB file transmission see Figure 4.10(a). From the figure, we can see that for all of them, number of re-



(a) Transmission of a 2KB file.



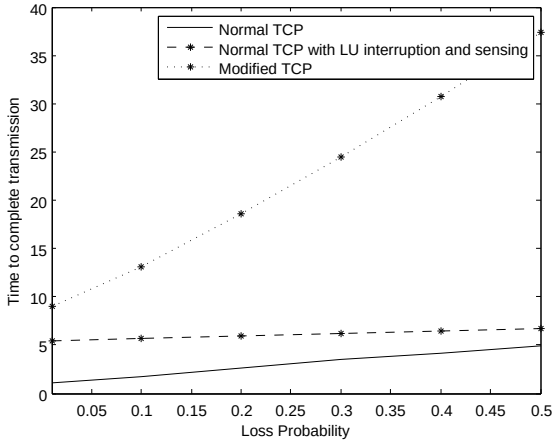
(b) Transmission of a 20KB file.



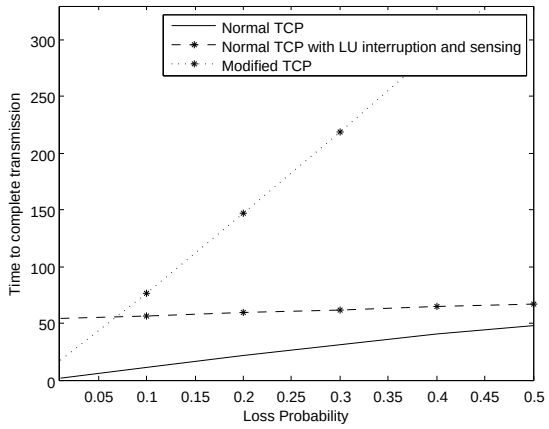
(c) Transmission of a 200KB file.

Figure 4.10: Number of Re-transmissions: Overall Analysis.

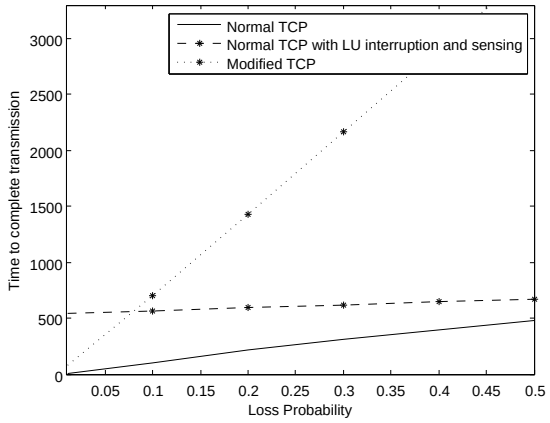
transmissions increases with increase in loss probability. As expected, TCP without any of the CRN features has the lowest number of re-transmissions. The TCP with the proposed modifications has lower number of re-transmissions than the original TCP with LU interruption and spectrum sensing. Figure 4.10(b) and Figure 4.10(c) show similar results for a 20KB file and a 200KB file transmissions, respectively.



(a) Transmission of a 2KB file.



(b) Transmission of a 20KB file.



(c) Transmission of a 200KB file.

Figure 4.11: Time to Complete Transmission: Overall Analysis.

Time to Complete Transmission

Figure 4.11 shows a comparison in terms of time to complete transmission between an original TCP without CRN features, an original TCP with LU interruption and spectrum sensing, and the modified TCP with the LU interruption and spectrum sensing. For a 2KB file transmission, Figure 4.11(a) shows that for for all the TCPs,

the time to complete transmission increases with increase in loss probability. However, the rate at which the time increases for the modified TCP is higher than the other two TCPs. As expected, TCP without any of the CRN features has the lowest time to complete transmission. The modified TCP has the highest time to complete transmission. When transmitting a 20KB file and a 200KB file, the time to complete transmission for the modified TCP is lower than that of the unmodified TCP until when loss probability reaches around 0.09 (See Figure 4.11(b) and Figure 4.11(c)). For most applications the acceptable loss probability is somewhere between 0.01 to 0.02, where the modified TCP was able to reduce the time to complete transmission.

The modified TCP model accounted for changes in window size. The MS state and the FZ state are modelled just like the DR state. They have same transition probabilities with DR state. Therefore, when in the MS state or the FZ state, w_n is set to $w_n/2$. What the model did not account for is our proposal to set w_n to the measured expected window size when in the MS state and to keep w_n as it is when in the FZ state (See Section 4.3). However, the model still improves the window utilization of TCP. This is because the unmodified TCP will set w_n to 1 when an LU arrival or spectrum sensing interrupts transmission, where as the modified TCP will set w_n to $w_n/2$.

Chapter 5

Conclusions and Future Work

Despite the world-wide deployment of TCP in the Internet, TCP cannot be directly deployed in cognitive radio networks. Several research results show how poor TCP performs in cognitive radio networks. To design an efficient transport protocol for cognitive radio networks, a new transport protocol needs to be designed or modification should be made to TCP. In this thesis, we showed how we modified TCP to work better in cognitive radio networks in terms of average number of visits to the TR state, total number of retransmissions, and time to complete transmission.

To evaluate our work, we introduced cognitive radio network features to the TCP Reno model of Arvidsson and Krzesinski [4]. We observed how original TCP Reno performs in the presence of cognitive radio features. We later extended their model to capture our proposed solution and evaluated its performance.

We developed MATLAB programs that compute the entries of the transition probability matrices we used for evaluation. We assigned values to the following parameters: $p_b, p_f, \varphi, w_{rcv}$, and w_{thresh} and vary the loss probabilities, p_f and p_b from 0.01

to 0.5. We considered file of three different sizes, a small file of size 2KB, a little bit larger file of size 20KB, and a larger file of size 200KB. We ran our program iteratively, iterations stop when the difference between the previous w_{thresh} value and the current is less than 10^{-6}.

Based on the absorbing Markov chains we obtained from our MATLAB programs, we computed performance measures that include average number of visits to TR state, time to complete transmission, total number of retransmissions during an average transmission, licensed user interruptions, and spectrum sensing interruptions.

Finally, we presented some figures that showed comparison between the original TCP Reno with CRN features and our modified TCP Reno in terms of average number of visits to TR state, time to complete transmission, and total number of total number of retransmissions during an average transmission. In the presence of LU interruption and spectrum sensing, the modified TCP was able to perform better than the original TCP in almost all the cases. The modified TCP was able to reduce the total number of retransmissions in all the file sizes we considered. When loss probability reaches 0.09, the modified TCP failed to reduce the time to complete transmission. However, for most applications the acceptable loss probability is less than 0.05.

For future work, we will try to implement our proposed modifications to TCP on radio devices equipped with cognitive features. We will also try to run some experiments using the NS-2 simulator. Since we require end systems to exchange spectrum sensing information during connection set-up, we proposed that this should be added to the TCP segment header. TCP segment header has an option field called Options. The length of the Options field is variable between $0 - 320$ bits.

During connection set-up, TCP uses the Options field to specify options such as acknowledgement policy and maximum segment size. The Options field can be used to exchange spectrum sensing schedule during connection set-up. We will also try to run some simulations using the NS-2 simulator.

Bibliography

[1] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty. Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey. *Computer Networks Journal*, 50(13):2127–2159, September 2006.

[2] I. F. Akyildiz, W.-Y. Lee, and K. R. Chowdhury. CRAHNs: Cognitive radio ad hoc networks. *Ad Hoc Networks*, 7(5):810–836, July 2009.

[3] M. Allman, V. Paxson, and W. Stevens. RFC 2581 : TCP congestion control. Internet Engineering Task Force (IETF), April 1999.

[4] A. Arvidsson and A. E. Krzesinski. A model of a TCP link. In *15th Specialist Seminar on Internet Traffic Engineering and Traffic Management (ITC), Wurzburg, Germany*, 2002.

[5] R. Braden. RFC 1122 : Requirements for internet hosts – communication layers. Internet Engineering Task Force (IETF), October 1989.

[6] L. S. Brakmo and L. L. Paterson. TCP Vegas: End to end congestion avoidance on a global Internet. *IEEE Journal on Selected Areas In Communications*, 13 (8):1465–1480, October 1995.

[7] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang. TCP westwood: End to end congestion control for wired/wireless networks. *Wireless Networks*, 8(5):467–479, September 2002.

[8] K. R. Chowdhury, M. Di Felice, and I. F. Akyildiz. TP-CRAHN: A transport protocol for cognitive radio ad-hoc networks. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), Rio de Janeiro, Brazil*, pages 2482–2490, April 2009.

[9] FCC. Federal communications commission spectrum policy task force. Report of the Spectrum Efficiency Working Group, November 2002.

[10] M. D. Felice, K. R. Chowdhury, and L. Bononi. Modeling and performance evaluation of transmission control protocol over cognitive radio ad hoc networks. In *Proceedings of the 12th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), Tenerife, Canary Islands, Spain*, pages 4–12, October 2009.

[11] S. Floyd and B. Henderson. RFC 2582 : The NewReno modifications to TCP's fast recovery algorithm. Internet Engineering Task Force (IETF), April 1999.

[12] S. Ha, I. Rhee, and L. Xu. CUBIC: A new TCP-friendly high-speed TCP variant. *Operating Systems Review*, 42(5):64–74, July 2008.

[13] E. Hossain and V. Bhargava, editors. *Cognitive Wireless Communication Networks*. Springer, 2007.

[14] T. Issariyakul, L. S. Pillutla, and V. Krishnamurthy. Tuning radio resource in

an overlay cognitive radio network for TCP: Greed isn't good. *IEEE Communications Magazine*, 47(7):57–63, July 2009.

[15] V. Jacobson. Congestion avoidance and control. *Computer Communication Review*, 18(4):314–329, August 1988.

[16] V. Jacobson. Modified TCP congestion avoidance algorithm. end2end-interest mailing list, ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail, April 1990.

[17] W.-Y. Lee and I. F. Akyildiz. Optimal spectrum sensing framework for cognitive radio networks. *IEEE Transactions on Wireless Communications*, 7(10):3845–3857, October 2008.

[18] C. Luo, R. F. Yu, H. Ji, and V. C. M. Leung. Cross-layer design for TCP performance improvement in cognitive radio networks. *IEEE Transactions on Vehicular Technology*, 59(5):2485–2495, June 2010.

[19] M. Mathis, J. Mahdavi, F. S., and A. Romanow. TCP selective acknowledgement options. RFC 2018, Internet Engineering Task Force (IETF), October 1996.

[20] J. Postel. RFC 793 : Transmission control protocol. Internet Engineering Task Force (IETF), September 1981.

[21] J. Postel. RFC 879 : The TCP maximum segment size and related topics. Internet Engineering Task Force (IETF), November 1983.

[22] D. Sarkar and H. Narayan. Transport layer protocols for cognitive networks. In *IEEE Conference on Computer Communications Workshops (INFOCOM), San Diego, CA, USA*, pages 1–6, March 2010.

[23] A. M. Slingerland, P. Pawelczak, R. Prasad, A. Lo, and R. Hekmat. Performance of transport control protocol over dynamic spectrum access links. In *Proceedings of the* 2^{nd} *IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN), Dublin, Ireland*, pages 486–495, April 2007.

[24] Q. Zhao and B. M. Sadler. A survey of dynamic spectrum access. *IEEE Signal Processing Magazine*, 24(3):79–89, May 2007.