Concurrent Multipath Transfer using SCTP: Modelling and Congestion Window Management

T. Daniel Wallace and Abdallah Shami

Abstract—Concurrent multipath transfer (CMT) using the stream control transmission protocol (SCTP) can exploit multihomed devices to enhance data communications. While SCTP is a new transport layer protocol supporting multihomed end-points, CMT provides a framework so that transport layer resources are used efficiently and effectively when sending to the same destination with multiple IP addresses. In this paper, we present two techniques for modelling the expected throughput of a CMT session; while one is based on renewal theory, the other uses a Markov chain. As far as we know, ours is the first paper to model CMT whilst considering practical transport layer resources like a shared receive buffer (RBUF). A comparison of the models showed the Markov chain to be more accurate, but suffered from scalability issues. Alternatively, the renewal model was more cost effective, but also less accurate. We also applied our models to a new problem called *congestion window management*, where the size of each congestion window is reconfigured for optimal performance. Again, we compared two approaches: a dynamic method that makes decisions based on instantaneous throughput, and a static method that uses an integer linear program (ILP) to generate a global solution. Results showed the static method outperforming the dynamic approach by as much as 12%.

Index Terms—Multihoming, concurrent multipath transfer, stream control transmission protocol.

1 Introdcution

Adding more than one network interface to a computing device, like a *smartphone*, is called multihoming. Devices that are multihomed are advantageous as they can provide a layer of redundancy in case of network failure (e.g., service disruption). Furthermore, one network may outperform another depending on location. For instance, when a smartphone user is at home, he may choose to connect to the Internet through a WLAN for higher data rates but lower fees. Alternatively, the cellular network is best while users are in transit. The larger coverage area of a cellular network can offer guarantees on connectivity, but at a much higher price and with lower bandwidth potential.

Assuming either network's (i.e, WLAN or cellular) data rates are independent, so that one does not affect the other; why then do we not use both network interfaces simultaneously? Certainly, if two men could do the work of one in half the time, the same could be said about interfaces and download times. Unfortunately, due to the architectural constraints of standard transport layer protocols like the transmission control protocol (TCP), an Internet application (e.g., file transfer) can only use one access network at a time.

Due to recent developments, however, concurrent multipath transfer (CMT) using the stream control transmission protocol (SCTP) can exploit multihomed devices to enhance data communications. While SCTP is a new transport layer protocol that supports end-points with multiple IP addresses (i.e., a multihomed device), CMT provides a framework so that transport layer resources are used efficiently and effectively when sending to the same destination with multiple IP addresses. TCP/SCTP congestion control relies on packet loss to detect network congestion. Current wireless networks, e.g., 3G/4G networks, are over-buffered (termed as bufferbloat), which void the TCP/SCTP congestion control algorithm [1]. Hence, current smartphone devices set the maximum receive buffer size relatively small when compared to intermediate networking devices (e.g., routers found within these over-buffered networks). While others [2], [3] have directed their work toward desktop implementations (i.e., those with a surplus of computing resources), we believe CMT will have the most success with mobile devices, such as smartphones. Therefore, given limited resources and the need to set the maximum receive buffer small (i.e., to mitigate the problem of bufferbloat), the computing resources of a mobile device must be optimized in order to accommodate multiple transport layer sessions. Furthermore, since a fixed size receive buffer (RBUF) is allocated to each transport layer session, it is imperative to include the RBUF in a model of CMT. In this paper, we present two techniques for modelling the expected throughput of a CMT session; while one is based on renewal theory, the other uses a Markov chain. As far as we know, ours is the first paper to model CMT whilst considering practical transport layer resources like a shared RBUF.

We then apply our models to a new problem we call *congestion window management*. In this problem, packets are transmitted to multiple destination addresses, but always arrive at the same RBUF, where they are re-

• *T. D. Wallace and A. Shami are with the Department of Electrical and Computer Engineering, Western University, London, Ontario, Canada, N6A 5B9.*
E-mail: ashami2@uwo.ca

ordered before being passed to the application layer. Furthermore, the RBUF is limited, so flow control prevents packet transmission when the sum of all congestion windows (CWNDs) is greater or equal to the size of the RBUF. Consequently, optimal performance can be linked to the size of each CWND. As a result, we propose an integer linear program (ILP) to solve the problem of congestion window management, from a global standpoint.

The rest of the paper is organized as follows: Section 2 provides a brief literature review on transport layer modelling and concurrent multipath transfer; Section 3 outlines the network scenario and describes the system of study; Section 4 presents the respective analytic models; Section 5 introduces static congestion window management; Section 6 provides numerical results; finally, Section 7 gives conclusions.

2 Literature Review

2.1 Transport Layer Modelling

Even though our work is geared toward SCTP, the models in this paper are closely related to TCP; so much so, a comprehensive review of TCP literature is warranted[1].

Arguably, renewal theory is the most popular technique for modelling TCP throughput. The seminal ideas for renewal theory as a TCP throughput model were first discussed by Mathis et al. in [5]. The authors developed a simple model for long-lived TCP connections with periodic independent packet losses. In that work, only linear CWND growth and fast recovery were considered. Moreover, it was assumed timeouts were avoided and the congestion window was only ever halved following a loss. Although loss events were assumed independent, a train of packets following the first loss were also assumed to be lost. Thus, packet losses were considered to be correlated using the implicit assumption of drop-tail queues along the path of a TCP connection. The work in [5] was later cultivated by Padhye et al. [6] to fit a more realistic implementation of TCP Reno. This model assumed both triple duplicate losses as well as timeout events, with exponential back-off periods following a timeout. Other authors again revisited this model to address some inaccuracies and incorporate the slow-start process after timeout events [7], [8]. This work is now known as the PFTK-model (PFTK being an acronym for the last name initials of all four authors in [6]). More recently, the PFTK model was transformed to use available bandwidth instead of loss rate as a modelling parameter [9]. The authors argue bandwidth is a more efficient way to characterize the network in order to quickly approximate throughput potential. More research based on renewal theory can be found in [10]–[13].

Next, using Markov chains and fixed-point methods, a relatively newer source of TCP modelling has evolved [14]–[16]. Starting from an arbitrary arrival rate, the solution to an M/M/1/K queue provides a Markov chain with average delay and probability of packet loss. Feeding these parameters back into the Markov chain creates a new arrival rate for the M/M/1/K queue. The model converges by iterating through arrival rates until some minimum error, between input and output, is satisfied. Using this same methodology, Fu et al. [2] developed a model to capture some of SCTP's multihoming features. Later, the same Markov chain was updated to include the number of losses in a previous round [17]. Although tracking the number of losses increases state space, the model can now estimate transmissions in the interim round, i.e., between congestion avoidance and fast recovery; results showed improved accuracy for single source transfers. Other TCP models based on Markov chains can be found in [3], [18].

1. Only renewal theory and Markovian models will be evaluated. The interested reader, looking for additional transport layer models, should consult a more thorough treatise on the matter, such as [4].

2.2 Concurrent Multipath Transfer

Multihomed devices can increase application throughput by taking advantage of additional resources available through multiple network interfaces [19]. Currently, the research community refers to this process as concurrent multipath transfer (CMT), but other terms have been used like bandwidth aggregation, resource pooling, inverse multiplexing, load sharing, and even striping in similar contexts. Although CMT is still in its infancy, some of the better known problems include: unnecessary fast retransmission, crippled congestion window growth, and receive buffer blocking. While concrete solutions have been found for the first two problems [20]–[22], receive buffer blocking still remains an open issue.

Receive buffer blocking was first shown in [23]. The authors demonstrated poor transfer times using a shared RBUF between high and low bandwidth paths. In fact, in some circumstances using only the higher bandwidth path gave better results. Effectively, receive buffer blocking is caused when the sender pauses transmissions to one destination while packets with cumulative sequence numbers travel slowly along a different route. Initially, little progress was made on the receive buffer blocking problem as researchers tried various retransmission policies [24], [25]; only when scheduling became a topic of interest, could CMT capitalize on the additional network resources of multihomed devices.

Proposed in [26], a mechanism called the bandwidth aware scheduler (BAS) attempts to minimize receive buffer blocking through intelligent packet scheduling. Using bandwidth and delay measurements, BAS assigns new packets to the destination with the earliest estimated delivery time; assigned packets are then placed into a virtual send queue prior to transmission. Another scheduler, also based on BAS, assigns packets to a destination using a new metric called the *reception index* [27]. The reception index of a destination is calculated by dividing the size of the cumulative packet, any outstanding packets, and any buffered packets by a destination's

bandwidth estimate. As it turns out, however, BAS can be ineffective, especially when the difference between destination RTTs becomes significantly large. To circumvent these shortcomings, the on-demand scheduler (ODS) was developed in [28]. Unlike BAS, ODS waits for a transmission opportunity before assigning packets to a destination address. When congestion and flow control allow transmission to a destination, ODS searches the send buffer in a recursive manner, looking for a packet that cannot be delivered to another destination sooner.

More recently, modelling techniques were applied to CMT in an attempt to avoid receive buffer blocking altogether. In [29], Yang et al. modify the well known PFTK model to calculate a minimum receive buffer size, such that receive buffer blocking is nonexistent. The same authors, moreover, used a similar model in [30], to choose a configuration of destination addresses that will maximize throughput for a given receive buffer. We should point out that neither of these studies considered congestion window management; in other words, the sum of CWNDs was allowed to go beyond the size of the RBUF. Unfortunately, unless the sum of CWNDs is less than or equal to the size of the RBUF, unnecessary loss will still persist.

3 System Description

3.1 System Comparison: Singlehomed vs. Multihomed

Ordinarily, the transport layer is modelled as a dumbbell topology, that is, one bottleneck link dividing a set of senders from their respective receivers. More specifically, however, we typically study only one sender/receiver pair, while another m make up background traffic sources. This model is true for singlehomed end-points only (i.e., one network interface). The dumbbell topology for a singlehomed connection is shown in Fig. 1.

When an end-point is multihomed (i.e., more than one interface), the singlehomed dumbbell topology is no longer valid, since more than one network path exists between sender and receiver. For multihomed end-points, rather, the network is modelled as a series of dumbbells, where the function $m(n)$ returns the maximum number of background sources sharing the n^{th} bottleneck link. Transmission and buffering capacity, moreover, are assumed to be independent of bottleneck link, so that loss rate and end-to-end delay may differ from path to path. The network topology for a mulithomed system is given in Fig. 2.

3.2 Stream Control Transmission Protocol

We now make the following assumptions regarding the stream control transmission protocol (SCTP) [31].

1) The amount of DATA the sender is allowed to transmit to address a at time t is controlled by the address's CWND, its number of outstanding bytes (OUT) (i.e., any unacknowledged data), and the RWND. At any given time, this amount is the lesser of the CWND minus OUT and RWND. Thus

$$\text{DATA}_{a,t} = \min(\text{CWND}_{a,t} - \text{OUT}_{a,t}, \text{RWND}_t). \quad (1)$$

2) Depending on mode, SCTP increases a CWND in two ways:
   a) An address is in slow-start (SS) mode if its CWND is less than the slow-start threshold (SSTHRSH). Every time a CUMACK is received, the CWND is increased by at most, the lesser of 1) the number of outstanding bytes being newly acknowledged (NEWACK) (i.e., the number of packets removed from the send buffer), and 2) the maximum transmission unit (MTU). The new CWND is thus

$$\text{CWND}_{a,t+1} = \min(\text{NEWACK}_{a,t}, \text{MTU}). \quad (2)$$

   b) An address is in congestion avoidance (CA) mode if its CWND is greater than or equal to its SSTHRSH. In this mode the CWND is incremented by one MTU every time a sender receives an acknowledgement that advances the CUMACK, and the CWND is less than or equal to the partially acknowledged bytes (PBA) and OUT. This is accomplished by reducing the number of PBA by the previous size of the CWND (i.e., the size of the CWND before the increase). The new CWND is calculated by

$$\text{CWND}_{a,t+1} = \text{CWND}_{a,t} + \text{MTU}. \quad (3)$$

3) Every time the sender receives a SACK, NEWACK is subtracted from OUT. In other words, the CWND is opened so that new packets may be sent without increasing its size.
4) When gaps are found in a SACK, the sender increments the missing count for any packets contained within a gap. If the SACK indicates that a packet has been missing four times, the packet is considered lost. In response, the sender halves the CWND of the address to where the packet was sent, then retransmits any lost packets. Moreover, the address's SSTHRSH is also set to the size of the halved CWND, in case of a timeout event.
5) Each time new data is sent to an address, a timer called the retransmission timeout (RTO) is set. If this timer expires before an acknowledgement arrives for the transmitted data, the CWND is dropped to one and RTO is doubled. This process continues for every consecutive packet loss until some maximum RTO is reached.
6) RTO is calculated from two variables: the smoothed RTT, sRTT; and the average variation in RTT, vRTT. Every time there is a CWND update new values for sRTT and vRTT are computed using exponential weighted moving averages. While the variation in

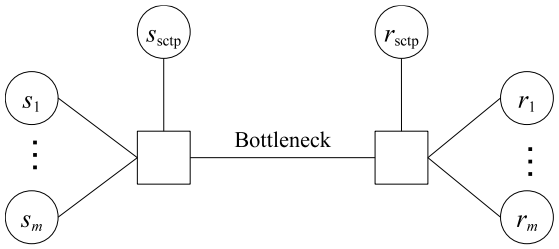

Fig. 1. Singlehomed network topology.

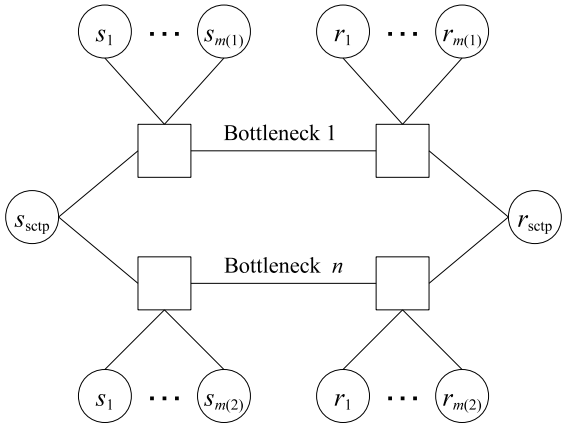

Fig. 2. Multihomed network topology.

RTT is given by

$$\text{vRTT} = (1-\beta)\text{vRTT} + \beta|\text{sRTT} - \text{RTT}|, \quad (4)$$

the smoothed RTT will be

$$\text{sRTT} = (1-\alpha)\text{sRTT} + \alpha\text{RTT}, \quad (5)$$

where $\alpha = \frac{1}{8}$, and $\beta = \frac{1}{4}$.

4 Modelling CMT

The objective of the model is to approximate the throughput of a reliable transport layer protocol employing CMT. Strictly speaking, we aim to model the performance of an SCTP session, transferring a large file from a singlehomed sender to a multihomed receiver. Our work, however, should be viewed as supplemental work to traditional transport layer modelling (e.g., a TCP connection with two singlehomed end-points), since great effort has been taken to exploit the well-known techniques of past research ventures.

4.1 Model Discrepancies

Before presenting our models, we need to highlight the differences between our implementations and those of past research efforts.

4.1.1 Markov Model

First, our model is for a single source only, therefore we do not use any queueing theory (e.g., M/M/1/k) nor implement a fixed-point method. The reason behind a single source model lies in our requirements for CMT. Since CMT specifies a shared RBUF, we assume at any given time the sum of CWNDs must be less than or equal to the size of the RBUF. Unfortunately, the solution to a queue model with multiple sources cannot offer loss rate or delay information unless every source (sharing the network path) is identical. Moreover, if we want CMT to manage CWNDs on a destination basis, we cannot expect every source to have the same statistical properties (e.g., average CWND); therefore, we must use a single source model in this endeavour.

Another difference is an increase in state-space. In previous models, the Markov chain predicts fast recovery (FR) and timeout (TO) events during congestion avoidance rounds that have yet to experience packet loss. In this way, the model will predict a FR event immediately following a round with losses. While this is not impossible, it is unlikely, and if a correlated loss model is used (e.g., if a packet is lost all remaining packets transmitted in the same round are also lost), this is most certainly untrue. Furthermore, if the model jumps from a round without losses to a round that has already invoked FR, the accuracy of the model suffers [17].

4.1.2 Renewal Model

For the most part, our renewal model is derived using a method similar to past TCP implementations, with the exception of TO probability and the exponential backoff period. Since SCTP requires 4 SACKs to trigger a TO, compared to TCP's 3 duplicate acknowledgements, the formula for the probability of a TO event needs to be modified. Previous TCP models, moreover, limit the doubling of RTO to 7 iterations, without specifying a maximum. We, on the other hand, assume $\text{RTO}_{\max}$ is given; so RTO can double any number of times.

4.2 Basic Modelling Assumptions

Given a multihomed network topology, we assume a set of network paths connect an SCTP sender/receiver pair, so that the sender may transmit over any path and all successful packets will arrive at the receiver. Each network path, moreover, is defined by its bandwidth (i.e., maximum transmission rate), probability of packet loss, and additional packet delay (made up from other delay sources like propagation, processing, and queueing effects). What is more, we assume a path to be independent of all other paths as well as its characteristics to remain constant.

Similar to past research, we also model the behaviour of our system in terms of discrete rounds. A round starts with the transmission of packets, and ends with

the reception of a SACK. Depending on the state of the system, the length of time between rounds may take as little as one RTT, one RTO, or up to the maximum allowable retransmission timeout (RTO_{max}). During a round, moreover, packets can be lost, where packet losses are characterized by assuming one of two loss models: 1) independent, or 2) correlated. An independent loss model, for example, assumes each packet transmitted in a round will have the same probability of being lost, regardless of when it is transmitted during the round. An independent loss model can be used when links are lossy (e.g., over a wireless channel). The probability of losing γ packets when ξ are transmitted under an independent loss model is given by

$$P(\gamma, \xi) = \binom{\xi}{\gamma} p^{\gamma}(1-p)^{\xi-\gamma}, \tag{6}$$

where p is the probability of packet loss on the network path.

Alternatively, a correlated loss model assumes a packet is lost with probability p as long as no other packet has been lost in the same round. Assuming a packet has already been lost, all remaining packet transmissions are lost with probability 1. This loss model represents a drop-tail queue where packets are lost when a bottleneck queue reaches capacity, thereby dropping any packets arriving after capacity has been reached. Under the correlated loss model, the probability of losing γ packets when ξ are transmitted is calculated by

$$P(\gamma, \xi) = \begin{cases} (1-p)^{\xi}, & \gamma = 0 \\ p(1-p)^{\xi-\gamma}, & 0 < \gamma \leq \xi \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Packet RTTs are calculated using the following equation

$$\text{RTT} = \begin{cases} d + \frac{1}{b}, & \text{if } b \cdot d \geq \text{CWND} \\ \frac{\text{CWND}}{b}, & \text{otherwise,} \end{cases} \tag{8}$$

where b is the bandwidth of a bottleneck link (in packets/second), and d is a constant delay experienced along the path. By assuming delay to be constant, including queueing delay from background traffic sources, at least the first case of Eq. (8) should be considered adequate. With that said, queueing delay from the same source can vary; for example, when the number of transmissions exceeds the BDP, packets will be backlogged as the path becomes congested. This creates a queueing effect and increases RTT. If the number of packets in the queue, q, were known ahead of time, we would have

$$\text{RTT} = d + \frac{q}{b}. \tag{9}$$

Even if q is unavailable, we know that acknowledgements should start arriving at regular intervals when the CWND is greater than the BDP. When this happens, a packet leaves the path every $1/b$ seconds. Assuming a packet leaves the moment a new round begins, another CWND/b iterations will occur before the packet can leave the path and end the round.

Finally, we assume the variation in RTT to be RTT/2. Therefore the initial RTO will always be $3 \cdot \text{RTT}$ since

$$\text{RTO} = \text{RTT} + 4 \cdot \text{Var[RTT]}. \tag{10}$$

4.3 Markov Model

Our first model uses a discrete-time Markov chain (DTMC) to model the behaviour of a multihomed network path; the goal of which is to generate a state-transition probability matrix $\mathbf{Q}$, and solve the steady-state probability distribution $\boldsymbol{\pi}$ for the DTMC.

We first define a state to be a round where the CWND is ω packets, ξ packets will be transmitted, and τ is the current SSTHRSH. At the end of a round, the system transitions from state i to i', or (ξ, γ, τ) to (ξ', γ', τ'), where the transition probability is defined by element $Q(i, i')$ and the steady-state probability distribution is computed as

$$\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{Q}. \tag{11}$$

Next, and for the sake of convenience, we will classify and group each state into one of the following descriptive subsets: Congestion Avoidance (CA), Slow-start (SS), and Exponential Back-off (EB). Finally, we formulate a throughput expression based on the steady-state probability.

4.3.1 Congestion Avoidance

We begin our rendering of CA mode by defining the set of CA states as

$$\mathcal{C} = \{(\omega, \xi, 1) : 2 \leq \omega \leq \omega_{\max}, 1 \leq \xi \leq \omega, \omega \in \mathbb{Z}, \xi \in \mathbb{Z}\}, \tag{12}$$

where $\xi_{\max}$ is the size of the RBUF in packets[2].

During CA mode, transition probabilities are dependent on ω, ξ, and γ. If $\omega = \xi$, however, we need only to concern ourselves with ω and γ since it means that no packets were lost in the previous round; allowing us to ignore FR events. Assuming that $\omega = \xi$ and $\gamma < \omega$, the system will stay in CA, and after one RTT will transition from $(\omega, \xi, 1)$ to $(\omega', \xi', 1)$ with probability $P(\gamma, \omega)$, where $\omega' = \omega$ and

$$\xi' = \begin{cases} \omega - \gamma, & \text{if } \gamma \geq 1 \\ \omega + 1, & \text{if } \gamma = 0, \omega < \omega_{\max} \\ \omega, & \text{otherwise.} \end{cases} \tag{13}$$

Alternatively, if $\gamma = \omega$, no packets will be acknowledged and a TO event will send the system into EB mode. In this case, the system will transition to $(1, 1, 2^{\lfloor \log_2 \omega/2 \rfloor})$ with probability $P(\omega, \omega)$ in one RTO.

If $\omega > \xi$, then packets must have been lost in the previous round. Whether or not the system stays in CA is now conditional on ω and γ as well as ξ. Given that $\omega > \xi$, the system will stay in CA mode only if a FR

2. State (1,1,1) is reserved for EB.

event is triggered. To invoke a FR event, the sender must receive four out-of-order SACKs in the next round. Regardless of γ and ω, as long as $\xi < 4$, a TO is inevitable and the system will transition into state $(1, 1, 2^{\lfloor \log_2 \omega/2 \rfloor})$ with a probability of 1 after one RTO.

Even if $\xi \geq 4$, however, there remain two conditions for which a TO event can still occur. The first is intuitive, if less than 4 packets are acknowledged, so that $\xi - \gamma < 4$, a TO event occurs simply because a FR did not. The second condition, on the other hand, happens when too many packets are lost in the current round, so that the number of outstanding packets is greater or equal to the size of the CWND after FR. Following a FR event, the CWND is divided by 2, so the number of packets transmitted in the next round will be $\omega/2 - \gamma$. Clearly, if $\gamma \geq \omega/2$, then no packets will be transmitted in the next round and a resulting TO event will occur. Assuming that $\omega > \xi$ and $\xi \geq 4$, after the duration of one RTT the system will transition to $(\omega/2, \omega/2 - \gamma, 1)$ with probability $P(\xi, \gamma)$ as long as $\xi - \gamma > 3$ and $\omega/2 - \gamma > 0$; otherwise, after one RTO it will transition to $(1, 1, 2^{\lfloor \log_2 \omega/2 \rfloor})$ with probability

$$\sum_{i=1}^{\xi} P(\xi, i), \quad \text{if } i \geq \xi - 3 \text{ or } i \geq \omega/2. \tag{14}$$

4.3.2 Slow-start (SS)

A SS state exists when the CWND is less than its SSTHRSH (i.e., $\omega < \tau$). During a SS round, moreover, every CUMACK increases the CWND by one MTU; so if no packets are lost, the CWND is doubled between SS rounds. The set of SS states for our Markov model is defined as

$$\mathcal{S} = \{(\omega, \xi, \tau) : \omega = \xi = 2^i, 1 \leq i \leq \log_2(\tau), \\ \tau \in \mathcal{T}, \tau > 1, i \in \mathbb{Z}\}, \tag{15}$$

where $\mathcal{T}$ is the set of slow-start thresholds, given by

$$\mathcal{T} = \{2^i : 0 \leq i \leq \lfloor \log_2 (\xi_{\max}/2) \rfloor, i \in \mathbb{Z}\}. \tag{16}$$

Assuming that $\omega < \tau$ and $\gamma = 0$, the system will either remain in SS or move into CA, depending on the values of ω and τ. If $\tau > 2\omega$, the system will stay in SS and transition to $(2\omega, 2\xi, \tau)$; otherwise, the system will start CA mode and transition into $(2\omega, 2\xi, 1)$. In either case, the transition probability will always be $P(0, \omega)$, and the duration of time between transitions will be one RTT.

On the other hand, if $\gamma > 0$, the system can either remain in SS until a FR, or TO and move into EB. In an attempt to reduce state space, when $1 \leq \gamma < \omega$ we move the system into CA by letting $\tau' = 1$. Although this simplifies the model, we believe this should have little if no impact on throughput since a pending FR (in the following rounds) would move the system into CA anyways. Therefore, assuming $1 \leq \gamma < \omega$, the system will transition to $(2\omega - \gamma, 2(\omega - \gamma), 1)$ with probability $P(\gamma, \omega)$ after one RTT.

Finally, if $\gamma = \omega$, the system will TO and move into EB by transitioning to $(1, 1, 2^{\lfloor \log_2 \omega/2 \rfloor})$ with probability $P(\omega, \omega)$ after one RTO.

4.3.3 Exponential back-off

EB mode starts immediately following a TO event and continues until a packet is successfully delivered to the receiver. We define the set of EB states by

$$\mathcal{E} = \{(0, \xi, \tau) : (2 \leq \xi \leq M, \tau = 1) \cup (\xi = 1, \tau \in \mathcal{T})\} \tag{17}$$

where M is the maximum number of consecutive TO events, calculated as

$$M = \max\left(1, \left\lfloor \log_2 \frac{\text{RTO}_{\max}}{\text{RTO}} \right\rfloor\right). \tag{18}$$

To understand our definition of $\mathcal{E}$, we actually redefine ξ to mean the number of consecutive TO events instead of the number of packets that are sent in a round. Although this is unorthodox, we should mention that the number of packets transmitted during an EB round will always equal the size of the CWND (i.e., $\omega = 1$). What changes, rather, is the time between consecutive EB rounds. Following every TO event, the RTO is doubled until RTO reaches some maximum value (i.e., $\text{RTO}_{\max}$). Later on, it will become necessary for us to generate an approximation for the duration of time between rounds; and since the time between EB rounds doubles after every consecutive TO event, we will need to know the probability of being in any particular EB round.

Since we only need to consider the transmission of one packet, the transition probability out of an EB state will always be $P(0, 1)$, with a round time of one RTT. In fact, the system will always transition to $(2, 2, \tau)$ after a successful packet delivery[3]. Furthermore, as long as $M > \xi + 1$, a failed transmission attempt will always transition the system into $(1, \xi + 1, 1)$ with probability $P(1, 1)$ after 2^{ξ}RTO. But if $M = \xi + 1$ or $M = \xi$, another TO event will keep the system in the same EB state, $(1, M, 1)$, every $\text{RTO}_{\max}$.

4.3.4 Throughput

In this model we express throughput as the number of packets received per unit time, denoted by η. The average throughput of a source will then be

$$\eta = \frac{E[\xi] - E[\gamma]}{E[\delta]}, \tag{19}$$

where $E[\xi]$ is the expected number of packets transmitted per round, $E[\gamma]$ is the expected number of packets lost per round, and $E[\delta]$ is the expected duration of time between rounds.

The expected number of packets transmitted per round is given by

$$E[\xi] = \sum_{i \in \{\mathcal{C}, \mathcal{S}\}} \pi(i)\xi(i) + \sum_{i \in \mathcal{E}} \pi(i), \tag{20}$$

3. In this case we are assuming $\xi_{\max} > 1$.

where $\pi(i)$ returns the probability of being in state i and $\xi(i)$ is the number of packets transmitted in state i.

Using either Eq. (6) or (7), the expected number of packets lost per round can be expressed by

$$E[\gamma] = \sum_{i \in \{\mathcal{C},\mathcal{S}\}} \pi(i) \sum_{j=1}^{\xi(i)} jP(j,\xi(j)) + \sum_{i \in \mathcal{E}} \pi(i) P(1,1). \quad (21)$$

Before we provide our expression for $E[\delta]$, we first define $\mathcal{D}$ to be a matrix of round times (i.e., the time between state transitions), and $\mathcal{A}$ to be the set of all states (i.e., $\mathcal{A} = \{\mathcal{C}, \mathcal{S}, \mathcal{E}\}$). Finally, the expected duration of time between rounds is calculated by

$$E[\delta] = \sum_{i \in \mathcal{A}} \sum_{i' \in \mathcal{A}} \pi(i) D(i,i'), \quad (22)$$

where $D(i,i')$ is an element of $\mathcal{D}$ representing the time it takes to transition from state i to i'.

4.4 Renewal Model

Our second model uses renewal theory: an assumption that a stochastic process continually restarts at regular intervals. When we consider the additive increase/multiplicative decrease (AIMD) algorithm used for congestion control, a continuous sawtooth pattern begins to form. Our goal, therefore, is to represent this pattern as an average interval in order to formulate a closed-form expression for the throughput of a single network path.

By letting S_t and L_t be the number of packets transmitted and lost in the time interval $[0,t] : t > 0$, we can express the average throughput of an SCTP session by

$$\eta = \lim_{t \to \infty} \frac{S_t - L_t}{t}, \quad (23)$$

or

$$\eta = \frac{E[S] - E[L]}{E[T]}, \quad (24)$$

where $E[S]$, $E[L]$, and $E[T]$ are the expected packets sent, the expected packets lost, and the expected time of an interval, respectively.

Similar to the Markov model, the renewal model is also broken into three descriptive modes of transmission: congestion avoidance (CA), exponential back-off (EB), and slow-start (SS).

4.4.1 Congestion Avoidance Mode

We begin our discussion of CA mode by focusing our attention on Fig. 3; a depiction of a continuous series of CA periods. Each CA period consists of R transmission rounds where S packets are sent and L packets lost. Using renewal theory, our goal is to approximate the throughput of the i^{th} CA period. During CA_i, new packets are sent every RTT, where the number of packets transmitted equals the current CWND, W. If no packets are lost, W increases until some maximum window is reached. Following a round of losses, however, W remains unchanged but in the next round the number of packets transmitted equals $W - L$. Typically, a CA period ends with FR and the process begins all over again with a CWND of $W/2$ packets.

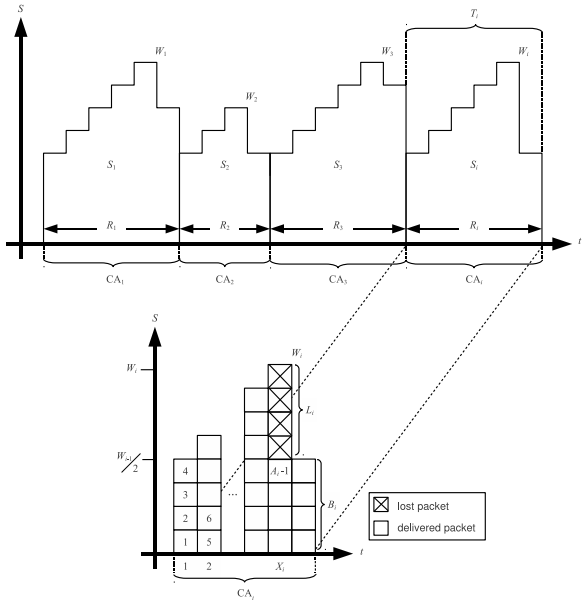

Fig. 3. A continuous series of congestion avoidance periods.

We observe that CA_i will always begin with $W_i = W_{i-1}/2$, increasing by one packet every round thereafter. The number of packets transmitted during CA_i can now be expressed by

$$\begin{aligned} S_i^{\text{CA}} &= \sum_{k=0}^{X_i - 1} \left(\frac{W_{i-1}}{2} + k \right) + B_i \\ &= \frac{X_i}{2}(W_{i-1} + X_i - 1) + B_i, \end{aligned} \quad (25)$$

where B_i is the number of packets sent in a round following a loss.

Now if we let A_i be the send count when the first loss occurs and assume a correlated loss model, we can also write

$$S_i^{\text{CA}} = A_i + W_i - 1. \quad (26)$$

By assuming a correlated loss model, packets are (successfully) received according to a geometric distribution. Therefore, the probability that $A_i = k$ packet transmissions can easily be calculated by $p(1-p)^{k-1}$. So the expected send count after the first loss will simply be

$$E[A] = \sum_{k=1}^{\infty} pk\,(1-p)^{k-1} = \frac{1}{p}. \quad (27)$$

From (26) and (27) it follows that

$$E[S^{\text{CA}}] = \frac{1-p}{p} + E[W]. \quad (28)$$

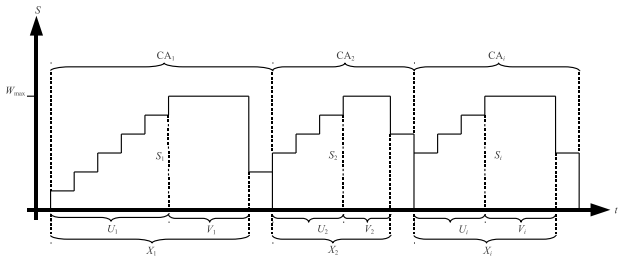

Fig. 4. Evolution of the CWND constrained by $W_{\max}$.

Since the CWND at the end of CA_i can be written as

$$W_i = \frac{W_{i-1}}{2} + X_i - 1, \tag{29}$$

when using (29) in (25), we get

$$S_i^{\text{CA}} = \frac{X_i}{2}\left(\frac{W_{i-1}}{2} + W_i\right) + B_i. \tag{30}$$

By assuming $\{X_i\}$ and $\{W_i\}$ to be mutually independent sequences with i.i.d. random variables, equating (28) and (30) yields

$$\frac{1-p}{p} + E[W] = \frac{E[X]}{2}\left(\frac{E[W]}{2} + E[W] - 1\right) + B_i. \tag{31}$$

Applying this assumption to (29) also gives

$$\begin{aligned} E[W] &= \frac{E[W]}{2} + E[X] - 1 \\ &= 2E[X] + 1. \end{aligned} \tag{32}$$

Furthermore, if we assume B_i to be uniformly distributed between 1 and W_{i-1}, $E[B]$ simplifies to $E[W]/2$. Likewise, since $L_i = W_i - B_i$, then $E[L]$ will also be $E[W]/2$. Finally, solving (31) for E[X], then substituting into (32) yields

$$E[W] = \sqrt{\frac{8(1-p)}{3p} + \frac{1}{9}} - \frac{1}{3}. \tag{33}$$

Similarly, we will also have

$$E[X] = \sqrt{\frac{2(1-p)}{3p} + \frac{1}{36}} + \frac{5}{6}. \tag{34}$$

Finally, since the number of transmission rounds in CA_i will always be

$$E[R] = E[X] + 1, \tag{35}$$

the expected duration of CA_i is just

$$E[T] = E[R] \cdot \text{RTT}. \tag{36}$$

Before moving on, however, we need to address the possibility of a constrained RBUF. During connection setup, the receiver advertises a maximum buffer size, which ultimately determines the sender's maximum CWND, $W_{\max}$; so when $W = W_{\max}$, the send count (per round) stays the same. With respect to Fig. 4, we will now describe how to model this effect. During the first CA period, the CWND grows linearly from 1 to $W_{\max}$ for U_1 rounds, then remains constant for V_1 rounds until a FR event. Finally, the CWND drops to $\frac{W_{\max}}{2}$, the sender recovers, and the process repeats, only this time starting from $\frac{W_{\max}}{2}$, so

$$W_{\max} = \frac{W_{\max}}{2} + U_i - 1. \tag{37}$$

The calculation for the number of packets sent during CA_i will now have the form

$$S_i^{\text{CA}} = \frac{U_i}{2}\left(\frac{W_{\max}}{2} + W_{\max}\right) + V_i W_{\max} + B_i. \tag{38}$$

Substituting (37) into (38), moreover, gives

$$E[S^{\text{CA}}] = \frac{3}{8}W_{\max}^2 + W_{\max}E[V] + \frac{1}{4}W_{\max}. \tag{39}$$

Then using (28) and solving for $E[V]$ yields

$$E[V] = \frac{1-p}{pW_{\max}} - \frac{3}{8}W_{\max} + \frac{3}{4}. \tag{40}$$

From Fig. 4, we see that $X_i = U_i + V_i$, so

$$E[X] = \frac{1-p}{pW_{\max}} + \frac{W_{\max}}{8} + \frac{3}{4}. \tag{41}$$

We now have two throughput expressions for CA mode: (1) when $E[W] < W_{\max}$, and (2) when $E[W] \geq W_{\max}$[4].

4.4.2 Exponential Back-off

Every time a new round begins, the sender starts a timer and waits up to one RTO to receive an acknowledgement. If this timer expires before receiving an acknowledgement, the sender triggers a TO event. Following a TO, the sender enters exponential back-off (EB) mode, dropping the CWND to one and doubling RTO. This process is continued for every subsequent TO until reaching $\text{RTO}_{\max}$, where it remains constant until a packet is finally acknowledged. Based on this situation, throughput is calculated by

$$\eta = \frac{E[S^{\text{CA}}] - E[L^{\text{CA}}]}{E[T^{\text{CA}}] + P^{\text{TO}}E[T^{\text{EB}}]}. \tag{42}$$

In (42), $E[S^{\text{CA}}]$, $E[L^{\text{CA}}]$, and $E[T^{\text{CA}}]$ are the expected number of packets transmitted, the expected number of packets lost and the length of time during CA mode, respectively; $E[T^{\text{EB}}]$ is the duration of an EB period; and P^{TO} is the probability of a TO event.

There are now two possible outcomes at the end of a CA period: FR or TO. To invoke a FR event, the sender must receive at least four acknowledgements no more than rounds after the first loss round; otherwise a TO will occur. Given the certainty of at least one loss, we can express the probability that the first j packets are acknowledged after W were sent as

$$P(j, W) = \frac{p(1-p)^j}{1-(1-p)^W}. \tag{43}$$

4. The informed reader will notice this is the same result as the PFTK model [6].

If we now apply (43) to all scenarios that would end a CA period, we can calculate the probability of a TO event by

$$P^{\text{TO}} = \sum_{j=0}^{3} P(j,W) + \sum_{j=4}^{W-1} P(j,W) \sum_{k=0}^{3} p(1-p)^k = \frac{1-(1-p)^8-(1-(1-p)^4)(1-p)^W}{1-(1-p)^W}. \quad (44)$$

We will now calculate the average duration of an EB period. Recalling that RTO is doubled for each unsuccessful retransmission until some maximum RTO is reached, the duration of j consecutive TO events is expressed by

$$T_j^{\text{TO}} = \begin{cases} (2^{j-1})\text{RTO}, & j \leq M \\ (j-M)\text{RTO}_{\max}, & j > M, \end{cases} \quad (45)$$

where M is solved using Eq. (18). In this case, however, we are constrained by $\text{RTO}_{\max} \geq 2\text{RTO}$. The expected duration of an EB period would then be

$$E[T^{\text{EB}}] = \sum_{j=1}^{\infty} T_j^{\text{TO}}(1-p)p^{j-1} = \frac{(1-p)(1-2^M p^M)}{1-2p}\text{RTO} + \frac{p^M}{1-p}\text{RTO}_{\max}. \quad (46)$$

Following EB, the sender will increase its CWND according to slow-start (SS) mode, i.e., the number of packets transmitted per round will double every RTT. The number of packets transmitted in SS period i is calculated by

$$S_i^{\text{SS}} = 1+2+2^2+\cdots+2^{H_i-1} = \sum_{k=1}^{H_i} 2^{k-1} = 2^{H_i}-1, \quad (47)$$

where H_i is the number of SS rounds before CA resumes. H_i is easily solved by rearranging (47) so that

$$H_i = \log_2\left(S_i^{\text{SS}}+1\right). \quad (48)$$

Assuming the last term in (47) should be $W_i/2$, we can write

$$W_i = 2^{H_i}. \quad (49)$$

Then substituting (49) into (47) gives

$$S_i^{ss} = W_i - 1, \quad (50)$$

and (50) into (48) yields

$$H_i = \log_2 W_i. \quad (51)$$

Since there must be at least one SS round, (51) is rewritten as

$$H_i = \max(1, \log_2 W_i). \quad (52)$$

From (50) we can write the expected number of packets transmitted during slow-start as

$$E[S^{\text{SS}}] = E[W] - 1. \quad (53)$$

Finally, the expected duration of SS mode will be

$$E[T^{\text{SS}}] = \max(1, \log_2 E[W]) \cdot \text{RTT}. \quad (54)$$

Combining each of SCTP's transmission modes, the final throughput equation using renewal theory will be

$$\eta = \frac{E[S^{\text{CA}}] - E[L^{\text{CA}}] + P^{\text{TO}}E[S^{\text{SS}}]}{E[T^{\text{CA}}] + P^{\text{TO}}(E[T^{\text{EB}}] + E[T^{\text{SS}}])}. \quad (55)$$

5 Congestion Window Management

In this section we present two ways of tackling the congestion window management problem. While one method is greedy, taking only instantaneous throughput into account, the other uses average throughput to place limits on the CWND of each destination address.

5.1 Dynamic Optimization

Presented in [28], the first method tries to maximize throughput in a greedy fashion by adjusting CWNDs based on instantaneous throughput; from here on out we will refer to this method as dynamic congestion window management. For example, if the sum of CWNDs is equal to the size of the RBUF, but the destination with the highest available bandwidth can improve throughput by increasing its CWND; intuitively, another destination address will first have to lower its CWND before another can be raised.

Dynamic congestion window management seeks to improve instantaneous throughput by simultaneously lowering one CWND while raising another when the sum of CWNDs is equal to the size of the RBUF. This is accomplished by ranking each destination; a destination with a higher rank can reduce the CWND of one that is ranked lower. Moreover, destination rank is decided by available bandwidth, with delay breaking a tie.

5.2 Static Optimization

What if a higher ranked destination also had a higher loss rate, so its CWND grew rapidly but was also cut back on a more frequent basis? In that case, lowering one CWND to raise another would still improve instantaneous throughput, but over the long run, average throughput could be suffering. To address this issues, we suggest a more static approach to congestion window management where a limit is placed on the CWND of a destination address. In this way ranks are eliminated, and an optimization technique is used to find the CWND limit of each destination address that will maximize throughput. In this subsection, we present an integer linear program (ILP) that calls either of the models from Section 4 to search for the best configuration of CWND limits; we call this static congestion window management.

Additionally, even though all CWNDs will have a limit, they are not always bound to this limit. For example, a destination can grow its CWND above its limit

if the following two conditions are true: 1) the sum of the CWNDs is less than the RBUF, and 2) the CWND is less than its BDP. Limits are still the rule, however, so if another destination tries to increase its CWND and finds the sum of CWNDs equal to the RBUF, a CWND operating above its limit will be reduced.

An ILP will now be formulated to maximize the throughput of CMT. First, we let the function $\eta(b_i, d_i, p_i, t_i^{\max}, c_i^{\max})$ return the expected throughput[5] of a single path transfer; provided that $i \in N = \{1, \ldots, n\}$ and b_i, d_i, p_i, $t_i^{\max}$, and $c_i^{\max}$ are respectively the bandwidth (in packets/second), delay (in seconds), loss rate, $\text{RTO}^{\max}$ (in seconds), and CWND limit (in packets) of destination i. Assuming, furthermore, the size of the RBUF (in packets) is given by r, our ILP for static congestion window management is expressed in just five equations:

$$\textbf{maximize} \sum_{i}^{n} \eta(b_i, d_i, p_i, c_i^{\max}, t_i^{\max}) \tag{56}$$

$$\textbf{s.t.} \sum_{i}^{n} c_i^{\max} \leq r, \tag{57}$$

$$1 \leq c_i^{\max} \leq \lceil b_i \cdot d_i + 1 \rceil, \tag{58}$$

$$c_i^{\max} \in \mathbb{Z}, \tag{59}$$

$$\forall i \in N. \tag{60}$$

5.3 Complexity

Using brute force, our ILP can only be solved in exponential time. For example, assuming $\lceil b_i \cdot d_i + 1 \rceil \geq r, \forall i \in N$, then solving the static congestion window management problem will have a computational complexity of $\mathcal{O}(r^n)$. Alternatively, the problem could be solved in polynomial time if we assumed n to be constant (e.g., $n = 2$ for a smartphone with 802.11 and GSM interfaces). Even still, if r is large enough, a real-time system could find an exhaustive search intolerable.

In a real-time system, deadlines are critical; so process runtimes are always deterministic. If it so happens that the runtime to solve our ILP is intolerable (i.e., the real-time system cannot wait for a solution), finding an optimal solution may not be possible. As a compromise, however, a heuristic can usually find one or two satisfactory solutions – albeit suboptimal – in a reasonable amount of time. Next, we will describe a simple heuristic to solve the static congestion window management problem when finding an optimal solution proves too costly.

Our heuristic simply reduces the number of searches needed to find a solution by using a subset of values available for $c_i^{\max}$. For example, if n = 2, $r = 100$ packets, $b_1 = b_2 = 1000$ p/s, and $d_1 = d_2 = 100$ ms, both $c_1^{\max}$ and $c_2^{\max}$ could take on any value in the set $\mathbb{C} = \{x : x \in \mathbb{Z} : 1 \leq x \leq r\}$. An exhaustive search, therefore, would need a total of 10000 iterations to generate an optimal solution. Alternatively, if we used the set $\{2x : x \in \mathbb{C}, x \leq \frac{|\mathbb{C}|}{2}\}$, the total number of iterations drops to 2500. Moreover, if we used the set $\{10x : x \in \mathbb{C}, x \leq \frac{|\mathbb{C}|}{10}\}$, just 100 iterations are needed. A general formula for our heuristic (i.e., a subset of CWND limits for destination i), is given by

$$\{kx : 1 \leq x \leq \left\lfloor \frac{a}{k} \right\rfloor, a = \min(r, \lceil b_i \cdot d_i + 1 \rceil), k \leq a\}. \tag{61}$$

6 Numerical Results

6.1 Model Comparison

We will now compare the accuracy of either model with simulated results[6]. Previously, we established a framework allowing each network path to be modelled independently, therefore we only need to use the single-homed network topology when evaluating accuracy.

6.1.1 Simulation and Model Parameters

We varied the following four parameters in order to generate a range of scenarios: probability of loss event (p), bandwidth (b), delay (d), and receive buffer size (r). With that said, simulation parameters were focused around $p = 0.1\%$, b = 21 Mbps, d = 40 ms, and r = 128 KB. We chose these parameters for the following reasons:

1) Measurement studies have produced a wide range of statistics, but there is little consensus on average loss rates [32]–[34]. In our work, we assume lower loss rates for ideal conditions.
2) Even though 4G LTE is currently being rolled out, the best available service for most of North America is still only evolved high speed packet access (HSPA+). Under ideal conditions, HSPA+ can offer speeds as high as 21 Mbps.
3) We compared packet RTTs from the University of Western Ontario to a number of other institutions (e.g., the University of Toronto, Princeton University, the University of California, Berkeley). While RTTs varied anywhere between 10 and 100 ms, we decided on a delay of 40 ms because this was the average among institutions within our timezone.
4) A minimum RBUF of 128 KB for an SCTP session employing CMT was recommended in [24].

Furthermore, each packet was 1500 bytes long, the maximum RTO was 60 seconds, and the send buffer was assumed to be infinitely large. In terms of network topology, we used the dumbbell topology from Fig. 1, where a source and destination are separated by a bottleneck link.

6.1.2 Results and Discussion

The results of our comparison are shown in Figs. 5 - 8. While in all scenarios the Markov model proved most accurate at estimated throughput, renewal theory did a reasonable job despite some areas of deviation. For example, in Fig. 6 the renewal model overestimated

5. Approximated using either the Markov or renewal model (see 4.3 and 4.3).

6. Simulated results were generated using *ns*-2.

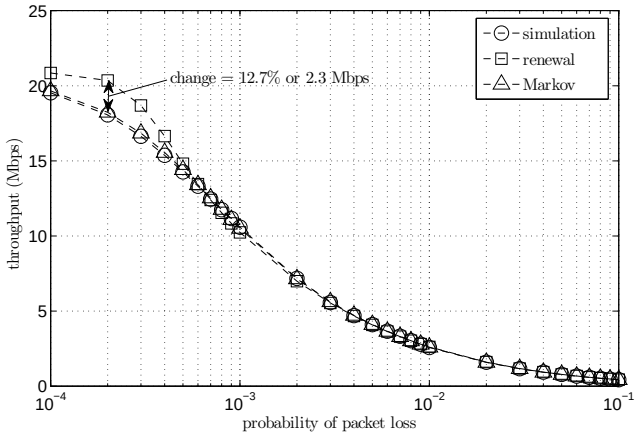

Fig. 5. Model comparison: b = 21 Mbps, d = 40 ms, RBUF = 128 KB.

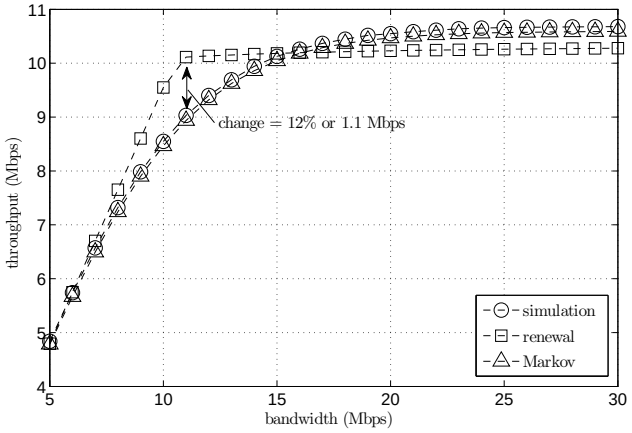

Fig. 6. Model comparison: $p = 10^{-3}$, d = 40 ms, RBUF = 128 KB.

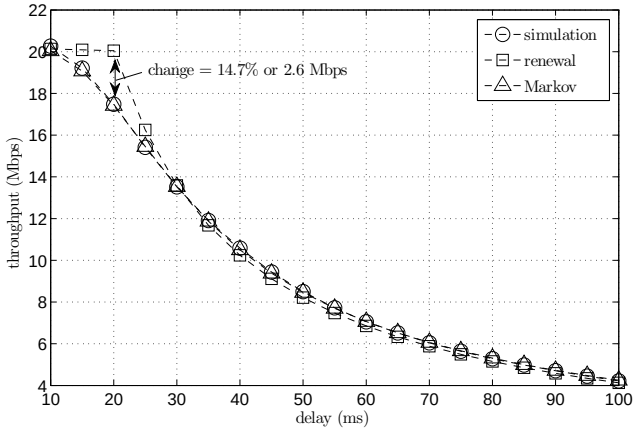

Fig. 7. Model comparison: $p = 10^{-3}$, b = 21 Mbps, RBUF = 128 KB.

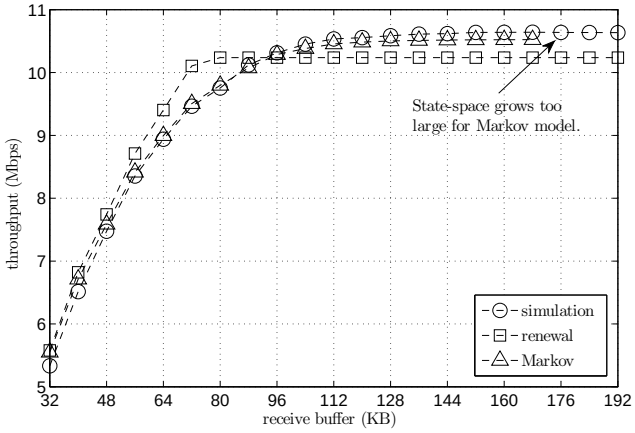

Fig. 8. Model comparison: $p = 10^{-3}$, b = 21 Mbps, d = 40 ms.

throughput by a margin of at most 14.7% (or 2.6 Mbps), but percent difference never exceeded 1.1% (or 0.23 Mbps) when using the Markov model.

Since the Markov model takes all considerations into account, its pinpoint accuracy should come as no surprise. Therefore, if model parameters were to be sampled in real-time, the Markov model should guarantee a good estimate on throughput potential. Unfortunately, since the Markov model needs to solve a system of linear equations, it suffers from scalability issues. For instance, Fig. 8 shows the limitations of the Markov model; when the RBUF is 176 KB, the state-space becomes so large that we can no longer solve for the stationary probability distribution[7].

Even with more memory, however, the Markov model will still face longer processing delays as the number of states increase. To put this into context, $\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{Q}$ is typically solved using Gaussian Elimination (GE), where multiplication and subtraction operations are considered units of computation. Assuming there are n equations, the total number of operations on the left side of the system of equations require $\frac{1}{3}(n^3 - n)$, while the right side needs n^2 [35]; giving GE a computational complexity of $\mathcal{O}(n^3)$. In the Markov model, n represents the total number of valid states, i.e., all CA, SS, and EB states. Since the number of CA and SS states are dependant on $\omega_{\max}$, n increases with $\omega_{\max}$[8]. In addition, the number of EB states is controlled by the difference in RTO and $\text{RTO}_{\max}$, so n also increases with $\text{abs}(\text{RTO} - \text{RTO}_{\max})$. Therefore, if memory and processing power are in short supply, the Markov model may prove too costly to employ in real-time.

Alternatively, the renewal model has minimal computational requirements, making it advantageous when processing power and memory are limited. Still, the renewal model is less accurate; making results questionable. On the contrary, we believe a more favourable result is possible if L (i.e., the number of packets lost at the end of a CA period), had better characterization.

7. The Markov model was implemented using MATLAB R2009b 32-bit on a Linux desktop computer with 3 GB of RAM.

8. In these experiments $\omega_{\max}$ is a function of the packet size and the RBUF.

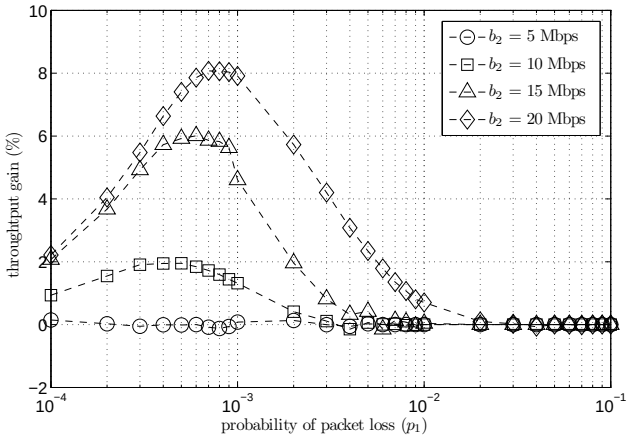

Fig. 9. Static vs. Dynamic: d_2 = 40 ms, $p_2 = 10^{-4}$, r = 128 KB.

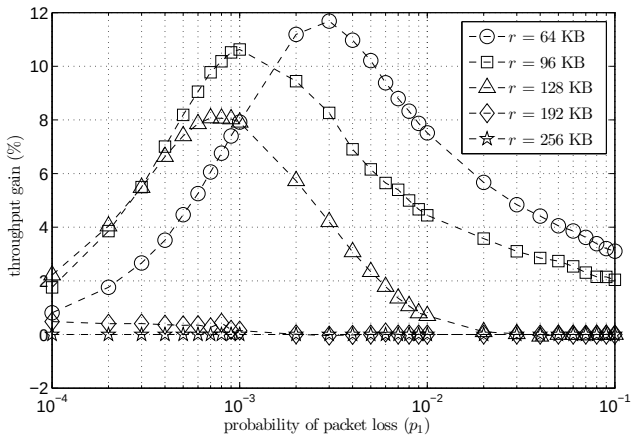

Fig. 10. Static vs. Dynamic: b_2 = 20 Mbps, d_2 = 40 ms, $p_2 = 10^{-4}$.

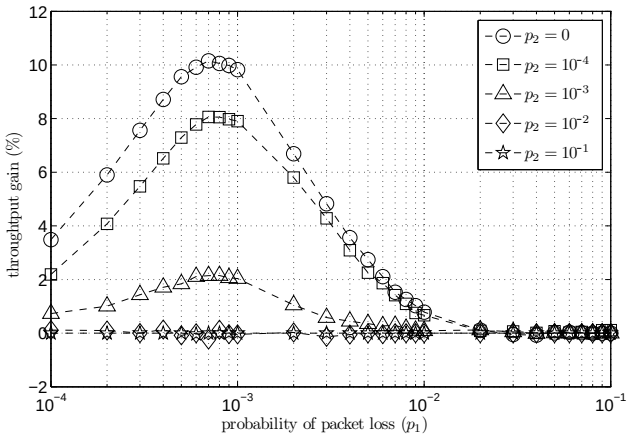

Fig. 11. Static vs. Dynamic: b_2 = 20 Mbps, d_2 = 40 ms, r = 128 KB.

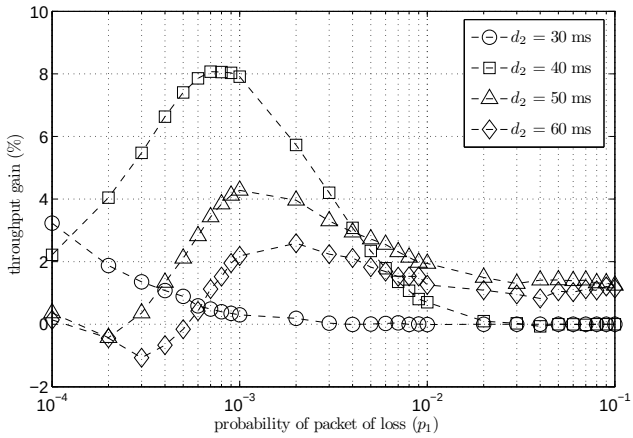

Fig. 12. Static vs. Dynamic: b_2 = 20 Mbps, $p_2 = 10^{-4}$, r = 128 KB.

Currently, L is assumed to be uniformly distributed between 1 and W, so $E[L] = W/2$; when in fact, the expected number of losses should be expressed by

$$E[L] = \sum_{i=1}^{W} ip(1-p)^{i-1} = \frac{1-(1+pW)(1-p)^W}{p}. \quad (62)$$

Unfortunately, the exponent in Eq. (62) complicates the relatively straightforward solution of W (see Eq. (31) - (33)). If (62) were used instead of $E[L] = W/2$, numerical methods might be the only way to solve for W; possibly increasing processor demand or memory requirements.

6.2 Congestion Window Management

Using the multihomed network topology from Fig. 2 with $n = 2$, we will now evaluate the performance of static vs. dynamic congestion management. For easy comparison, we simply calculated the percent gain, in terms of throughput, when using the static approach instead of the dynamic one. For brevity, moreover, we only implemented static congestion window management using the Markov model[9].

6.2.1 Simulation Parameters

In all tests the probability of packet loss to destination 1 (i.e., p_1) was varied between 10^{-4} and 10^{-1}, with the following four parameters remaining constant: 1) b_1 = 21 Mbps, 2) d_1 = 40 ms, 3) $t_1^{\max} = t_2^{\max} = 60$ seconds, and 4) packet size = 1500 bytes. Furthermore, to create a variety of scenarios, in each test we changed one of the following: r, p_2, b_2, and d_2.

6.2.2 Results and Discussion

The results of our findings are shown in Figs. 9 - 12. In most cases the static approach improved performance, even as high as 12%. Unfortunately, there were some instances where no improvement could be made, and some cases where dynamic congestion window management yielded even better results.

9. Since the Markov model can support both independent and correlated loss models, we now assume all losses are independent and p to be the probability of packet loss.

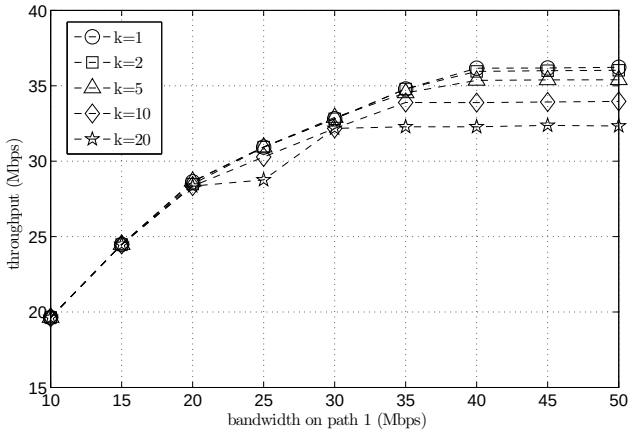

Fig. 13. Heuristic evaluation: throughput.

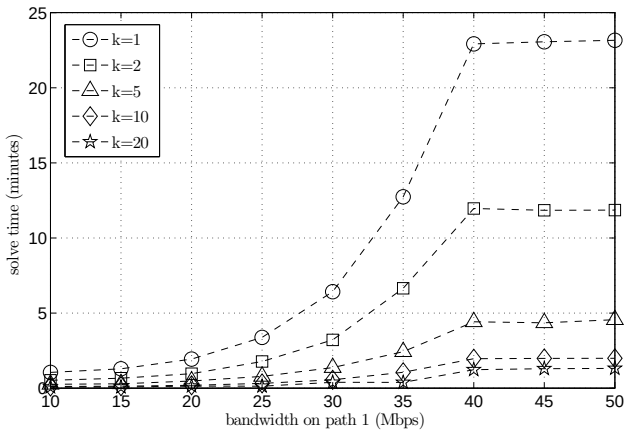

Fig. 14. Heuristic evaluation: solve time.

In the case of Fig. 9, the RBUF is 128 KB and destination 1 will use up to 108 KB; leaving just 20 KB available for destination 2. When b_2 is low, destination 2 does not need a very high CWND to maximize throughput (e.g., at 5 Mbps, $c_2^{\max}$ = 26 KB), so the static method has little impact. But when b_2 is larger, destination 2 will need a higher CWND, and therefore more of the RBUF to leverage performance. Furthermore, the static approach is always more effective when $p_1 \in [10^{-3}, 10^{-2}]$ because it makes sense to increase the CWND of a stable (yet lower ranked) destination when a higher rank becomes volatile.

A similar explanation can be made for Fig. 10. Again, destination 1 is ranked highest (i.e., $b_1 > b_2$); so when the RBUF is lower than $c_1^{\max}$ and p_1 is minimal, destination 2 will send very little. In any event, both destinations have comparable bandwidth, so the difference in choosing one over the other is marginal. As p_1 increases, however, the dynamic scheme will let destination 2 grow its CWND, but only temporarily. Every time destination 1 lowers its CWND (e.g., due to packet loss) destination 2's CWND can grow again, but it is for this reason that the static method is better; raising its CWND only to lower it again at the behest of another–more unstable destination–is an inefficient use of resources. Of course, static congestion window management becomes less significant once the RBUF becomes large enough to support both destinations' bandwidth potentials, regardless of loss rate.

In Fig. 11, improvement goes to zero as loss rate increases on path 2. The main reason, however, does not change; as loss rate increases, average CWND reduces and less of the RBUF is needed by a destination. Therefore, when the RBUF is large enough, or even underutilized, static management can do no better.

Contrary to what we have seen so far, Fig. 12 shows us that static management is not always the better choice. For the majority of the plot, however, static management is dominant; boosting performance more than 8%. But when the delay on path 2 is considerably large (compared to path 1), there are areas where dynamic management is more desirable (e.g., $p_1 < 10^{-3}$). Clearly, path 1 is most volatile when $p_1 \in [10^{-3}, 10^{-2}]$, and in Figs. 9 - 11, static management was able to capitalize.

6.3 Heuristic

Before leaving this section we offer some results using our simple heuristic. In this experiment, we varied b_1, but kept the following parameters constant: r = 128 KB, b_2 = 10 Mbps, d_1 = 25 ms, d_2 = 50 ms, $t_1^{\max} = t_2^{\max} = 60$ seconds, and packet size = 1500 bytes. Moreover, we looked at how the heuristic faired with increasing values of k. Figs. 13 and 14 display the results of this test.

Not surprisingly, the size of k makes little difference in terms of throughput when b_1 is small. This is because the scope of the problem (or the search space) is already quite low, and lowering it any any further will have marginal impact. Nevertheless, increasing bandwidth will increase BDP which creates more CWND limits to choose from. Therefore, using a smaller value of k ensures a more refined choice during static congestion window management, and leads to higher throughput values as is shown in Fig. 13.

Unfortunately, a smaller k also means more iterations which takes more time to find an optimal solution. Finally, Fig. 14 shows how lowering the value of k can save a considerable amount of time to solve the static congestion window management problem.

7 Conclusion

In this paper, we developed a framework to model, as well as optimize, a CMT session. First, two modelling techniques were evaluated; one based on a Markov chain, and the other using renewal theory. Although either model makes a number of assumptions regarding the mechanics of CMT, approximations were shown to be consistent with simulated results. Unfortunately, the Markov model had scalability issues, limiting any real-time implementation, despite being more accurate. On the other hand, renewal theory was more cost effective in

terms of computational complexity, but approximations were not always accurate.

The second contribution of this paper includes the design of an ILP for congestion window management. Using one of our proposed models, the ILP can generate a set of CWND limits so that average throughput is maximized during a CMT session. We called this a "static" approach to congestion window management and compared it to another method which is more dynamic. Since the dynamic approach adjusts the size of CWNDs based on instantaneous throughput, the static approach typically yielded improved results by evaluating the connection over the long run.

REFERENCES

[1] L. Z. W. Y. L. K. Jiang, H. and I. Rhee, "Understanding bufferbloat in cellular networks," in *Proc. of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*, 2012, pp. 1–6.

[2] S. Fu and M. Atiquzzaman, "Performance modeling of sctp multihoming," in *Proc. IEEE Global Communication Conf.*, vol. 2, 2005, pp. 786–791.

[3] J. Padhye, V. Firoiu, and D. Towsley, "A stochastic model of tcp reno congestion avoidance and control," Department of Computer Science, University of Massachusetts, Tech. Rep. UMASS-CS-TR-1999-02, 1999.

[4] J. Olsén, "Stochastic modeling and simulation of the tcp protocol," Ph.D. dissertation, Department of Mathematics, Uppsala University, 2003.

[5] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the tcp congestion avoidance algorithm," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 3, pp. 67–82, 1997.

[6] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling tcp reno performance: a simple model and its empirical validation," *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 133–145, 2000.

[7] R. Dunaytsev, Y. Koucheryavy, and J. Harju, "The pftk-model revised," *Comput. Commun.*, vol. 29, pp. 13–14, 2006.

[8] Z. Chen, T. Bu, M. Ammar, and D. Towsley, "Comments on modeling tcp reno performance: a simple model and its empirical validation," *IEEE/ACM Trans. Netw.*, vol. 14, no. 2, pp. 451–453, 2006.

[9] J. Hwang and C. Yoo, "Formula-based tcp throughput prediction with available bandwidth," *IEEE Commun. Lett.*, vol. 14, no. 4, pp. 363–365, 2010.

[10] A. Kumar, "Comparative performance analysis of versions of tcp in a local network with a lossy link," *IEEE/ACM Trans. Netw.*, vol. 6, no. 4, pp. 485–498, 1998.

[11] V. Firoiu and M. Borden, "A study of active queue management for congestion control," in *Proc. IEEE Intl. Conf. on Computer Communications*, vol. 3, 2000, pp. 1435–1444.

[12] C. Barakat, "Tcp/ip modeling and validation," *IEEE Network*, vol. 15, no. 3, pp. 38–47, 2001.

[13] N. Parvez, A. Mahanti, and C. Williamson, "An analytic throughput model for tcp newreno," *IEEE/ACM Trans. Netw.*, vol. 18, no. 2, pp. 448–461, 2010.

[14] C. Casetti and M. Meo, "An analytical framework for the performance evaluation of tcp reno connections," *Comput. Netw.*, vol. 37, no. 5, pp. 669–682, 2001.

[15] A. Wierman, T. Osogami, and J. Olsén, "A unified framework for modeling tcp-vegas, tcp-sack, and tcp-reno," in *Proc. IEEE Intl. Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2003, pp. 269–278.

[16] M. Garetto, R. Cigno, M. Meo, and M. Marsan, "Closed queueing network models of interacting long-lived tcp flows," *IEEE/ACM Trans. Netw.*, vol. 12, no. 2, pp. 300–311, 2004.

[17] T. Wallace and A. Shami, "An analytic model for the stream control transmission protocol," in *Proc. IEEE Global Communication Conf.*, 2010, pp. 1–5.

[18] S. Fortin-Parisi and B. Sericola, "A markov model of tcp throughput, goodput and slow start," *Perform. Evaluation*, vol. 58, no. 2, pp. 89–108, 2004.

[19] T. Wallace and A. Shami, "A review of multihoming issues using the stream control transmission protocol," *IEEE Commun. Surv. Tutorials*, vol. 14, no. 2, pp. 565–578, 2012.

[20] G. Ye, T. Saadawi, and M. Lee, "Ipcc-sctp: An enhancement to the standard sctp to support multi-homing efficiently," in *Proc. IEEE Intl. Conf. on Performance, Computing, and Communications*, 2004, pp. 523–530.

[21] A. El Al, T. Saadawi, and L. M., "Ls-sctp: A bandwidth aggregation technique for stream control transmission protocol," *Comput. Commun.*, vol. 27, no. 10, pp. 1012–1024, 2004.

[22] J. Iyengar, P. Amer, and R. Stewart, "Concurrent multipath transfer using sctp multihoming over independent end-to-end paths," *IEEE/ACM Trans. Netw.*, vol. 14, no. 5, pp. 951–964, 2006.

[23] J. R. Iyengar, P. D. Amer, and R. Stewart, "Receive buffer blocking in concurrent multipath transfer," in *Proc. IEEE Global Communication Conf.*, vol. 1, 2005, pp. 121–126.

[24] ——, "Performance implications of a bounded receive buffer in concurrent multipath transfer," *Comput. Commun.*, vol. 30, no. 4, pp. 818–829, 2007.

[25] J. Liu, H. Zou, J. Dou, and Y. Gao, "Reducing receive buffer blocking in concurrent multipath transfer," in *Proc. IEEE Intl. Conf. on Circuits and Systems for Communications*, 2008, pp. 367–371.

[26] C. Casetti, W. Gaiotto, and D. di Elettronica, "Westwood sctp: Load balancing over multipaths using bandwidth-aware source scheduling," in *Proc. IEEE Vehicular Technology Conf.*, vol. 4, 2004, pp. 3025–3029.

[27] M. Fiore, C. Casetti, and G. Galante, "Concurrent multipath communication for real-time traffic," *Comput. Commun.*, vol. 30, no. 17, pp. 3307–3320, 2007.

[28] T. Wallace and A. Shami, "On-demand scheduling for concurrent multipath transfer under delay-based disparity," in *Proc. Intl. Wireless Communications and Mobile Computing Conf.*, 2012, pp. 833–837.

[29] W. Yang, H. Li, and J. Wu, "Pam: Precise receive buffer assignment method in transport protocol for concurrent multipath transfer," in *Proc. Intl. Conf. on Communications and Mobile Computing*, 2010, pp. 413–417.

[30] W. Yang, H. Li, F. Li, Q. Wu, and J. Wu, "Rps: range-based path selection method for concurrent multipath transfer," in *Proc. IEEE Intl. Wireless Communications and Mobile Computing Conf.*, 2010, pp. 944–948.

[31] R. Stewart, "Rfc 4960: stream control transmission protocol," *Request for Comments, IETF*, 2007.

[32] J. Bolot, "End-to-end packet delay and loss behavior in the internet," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 23, no. 4, pp. 289–298, 1993.

[33] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S. Diot, "Packet-level traffic measurements from the sprint ip backbone," *IEEE Network*, vol. 17, no. 6, pp. 6–16, 2003.

[34] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Measurement and classification of out-of-sequence packets in a tier-1 ip backbone," *IEEE/ACM Trans. Netw.*, vol. 15, no. 1, pp. 54–66, 2007.

[35] G. Strang, *Linear algebra and its applications*. Hartcourt Brace Jovanovich College Publishers, 1988.

Daniel Wallace received his B.E. degree in Software Engineering from Lakehead University, Thunder Bay, ON, Canada, in 2004, and his M.E.Sc. and Ph.D. from the University of Western Ontario, ON, Canada, in 2007 and 2012, respectively. His research interests include network modelling, transport layer protocols, and mobile computing.

Abdallah Shami received Ph.D. Degree in Electrical Engineering from the City University of New York in September 2002. Since July 2004, he has been with Western University, Canada where he is currently an Associate Professor in the Department of Electrical and Computer Engineering. His current research interests are in the area of wireless networking and cloud computing.