Performance Evaluation of Concurrent Multipath Transfer Using SCTP Multihoming in Multihop Wireless Networks

Ilknur Aydin Chien-Chung Shen
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716, USA
E-mail: {aydin,cshen}@cis.udel.edu

Abstract—**A transport layer protocol supporting multihoming allows an application to transmit data via multiple paths simultaneously (termed concurrent multipath transfer, or CMT for short). SCTP is an IETF-supported transport layer protocol with built-in multihoming capability. Earlier work by Iyengar et. al. proposed a CMT protocol using SCTP multihoming for improving application throughput, which studied the performance of CMT over wired networks using ns-2 simulations. Given recent advances in multiple radio nodes, multichannel radios, and multipath routing we believe that we will see more multihomed nodes in the wireless networks context. This motivates us to study the SCTP-based CMT over wireless networks. In this paper, we investigate the throughput of the applications using the SCTP-based CMT over IEEE 802.11 static multihop wireless networks with QualNet simulations.**

I. Introduction

Today, more and more devices (laptops, PDAs, mobile phones, etc.) come equipped with multiple wireless interfaces to connect to the networks. For instance, it has been a norm for PCs and laptops to have one Ethernet interface and one Wi-Fi interface. Mobile phones and network interface cards that integrate multiple wireless technologies (such as 3G/UMTS/GPRS and Wi-Fi) have been widespread. In addition, nodes with multiple radios and radios operating over multiple channels are being developed [1], [2]. A host is *multihomed* if the host has multiple network addresses [3]. All these advances and technological enablers motivate the use of *multihoming,* especially in the context of wireless networks.

A *multihome-capable transport layer protocol* allows each endpoint of a single transport layer connection to have multiple network addresses. When each network address is associated with a different network interface card which is connected to a different network, multiple (physical) paths between the multihomed hosts become feasible. Therefore, a multihome-capable transport layer protocol can support the (simultaneously) transfer of application data through *multiple paths* between multihomed hosts within a single transport layer connection. That is, a multihome-capable transport protocol can technically support *concurrent multipath transfer (CMT).* CMT can be used for different purposes such as increasing application throughput, fault-tolerance, bandwidth aggregation, load balancing, etc.

The current transport layer workhorses of the Internet, TCP and UDP, do not support multihoming. However, the emerging Stream Control Transmission Protocol (SCTP) [4], [5] has built-in multihoming support.

SCTP is originally designed to transport telephony signaling messages over IP networks. Later on, SCTP is supported by the IETF and found useful as a general purpose, reliable transport protocol for the Internet. SCTP provides services similar to TCP's (such as connection-oriented, reliable data transfer, ordered data delivery, window-based and TCP-friendly congestion control, flow control, etc.) and UDP's (such as unordered delivery, message-oriented, etc.). In addition, SCTP provides other services neither TCP nor UDP offers (such as multihoming, multistreaming, protection against SYN flooding attacks, etc.) [6]. In the SCTP jargon, a transport layer connection is called an *association*. Each SCTP packet, or generally called *SCTP protocol data unit (SCTP-PDU)*, contains an SCTP common header and multiple data or control *chunks*.

One of the most prominent features of SCTP is its built-in multihoming where an association can be established between a *set* of local and *a set* of remote IP addresses as opposed to a *single* local and a *single* remote IP address as in a TCP connection. In an SCTP association, each SCTP endpoint chooses *a single port*. Although multiple IP addresses are possible to reach one SCTP endpoint, only one of the IP addresses is specified as the *primary IP address* to transmit data to the endpoint (destination). Therefore, although standard SCTP is multihome-capable, the standard SCTP in reality do not support CMT. Iyengar et. al. [7], [8] introduced CMT to the standard SCTP to achieve increased application throughput and studied the performance of this SCTP-based CMT in various wired scenarios.

Our objective is to study the performance of SCTP-based CMT[1] over multihop wireless networks (MWNs). In this

This work is supported in part by the National Science Foundation under grant CNS-0347460.

[1]From here on, any mention of SCTP-based CMT or CMT refers to the CMT protocol as in [7].

paper, we consider a specific type of MWN, where (i) all the nodes are stationary, (ii) there is no connection to a wired network or the Internet, and (ii) the medium access is orchestrated by the IEEE 802.11 DCF MAC protocol [9].

The organization of this paper is as follow, Section II gives the background about the IEEE 802.11 DCF protocol and the SCTP-based CMT. Section III and IV present our experimental set up, results, and analysis. Section V presents related work. Finally, section VI concludes the paper.

II. Background

We briefly mention the details of the IEEE 802.11 protocol and the SCTP-based CMT in the following subsections.

A. IEEE 802.11 DCF

Spectrum is a shared and scarce resource that requires controlled access in wireless networks. The IEEE 802.11 Distributed Coordination Function (DCF) MAC protocol [9] is the *de facto* medium access standard used in multihop wireless networks.

IEEE 802.11 DCF is basically a carrier sense multiple access (CSMA) scheme with collision avoidance (CA) and positive acknowledgments (M-ACKs[2]). A node who wants to transmit data (M-DATA[3]) first senses the medium (*physical carrier sensing*). If the medium is not being used by the transmissions of the other nodes, then the sender transmits M-DATA. The receiver responds with an M-ACK after receiving the M-DATA. IEEE 802.11 DCF also uses *virtual carrier sensing* for collision avoidance. Basically, each IEEE 802.11 DCF PDU contains a duration field indicating how long it will take the sender node to transmit the M-PDU. Other nodes hearing the transmission of the M-PDU then look at the duration field and determine the minimum time that they need to defer their transmissions (maintained in the network allocation vector (NAV) of each node). IEEE 802.11 DCF also includes an optional RTS/CTS mechanism to reserve the channel before any M-DATA transmission. The sender node sends an RTS (Request-To-Send) message up to a number of times[4] to reserve the channel for the data transmission. If sender can not get a CTS (Clear-To-Send) after the tries, the sender drops the M-DATA and reports link failure to the upper layer. After getting a CTS, the sender then transmits M-DATA up to a number of times[5] until the sender gets an M-ACK from the receiver. Again, if the sender does not get an M-ACK after so many tries, then the M-DATA is dropped and an error is reported to the upper layer. RTS and CTS messages also include the duration of the entire transmission. Therefore, any other node hearing the RTS/CTS exchange, update their NAV accordingly to defer their transmissions.

Hidden Terminals and Spatial Channel Reuse: Even though the IEEE 802.11 DCF employs an RTS/CTS mechanism, it is still prone to the *hidden terminal problem*, which occurs due to the interference caused by another transmission in the neighborhood of a receiver node. In Fig. 1, each node in the chain is equipped with an IEEE 802.11 wireless interface with transmission range of 250 meters and carrier sensing (and interference) range of 550 meters[6]. Nodes are 200 meters apart (each node can communicate only with its direct neighbors). Let's assume that there are two data transmissions in the network, one from node 1 to node 2 and the other from node 4 to node 5. Before starting the data transmission, node 1 sends an RTS to node 2, then node 2 responds with a CTS. Note that, node 4 can not hear (decode) the RTS and CTS messages because node 4 is outside the transmission range of node 1 and 2. Therefore, node 4 does not defer its transmission to node 5, while node 1 is transmitting to node 2. Thus, transmission at node 4 interferes with the reception of node 2 (since node 2 is within the interference range of node 4). Node 4 becomes a hidden node for the transmission from node 1 to node 2 and causes the loss of data (*contention-induced loss*).

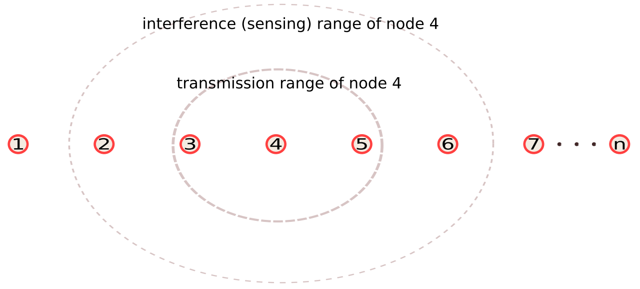

interference (sensing) range of node 4
transmission range of node 4
1
2
3
4
5
6
7 · · · n

Fig. 1. *A multihop wireless chain topology.* Node 4 is a hidden terminal for the transmission from node 1 to node 2.

The interference relationship among the nodes due to the hidden terminals is the main bottleneck of IEEE 802.11 based multihop wireless networks. In particular, the use of the channel by two different transmissions is possible only if the two transmissions are not interfering with each other (*spatial channel reuse*). For instance, in Fig. 1, transmissions between nodes 1-2 and between nodes 5-6 may occur simultaneously, but transmissions between nodes 1-2 and between nodes 4-5 can not happen at the same time.

B. CMT using SCTP Multihoming

Naively[7] transmitting data to multiple destination addresses (over different paths) within an SCTP association causes out-of-order arrivals at a multihomed SCTP receiver. Out-of-order arrivals have negative effects on the SCTP throughput due to spurious fast retransmissions and prevent congestion window growth even though when ACKs continue arriving at the sender. CMT [7] incorporates the following three algorithms to mitigate the effects of reordering at the receiver.

- *Split Fast Retransmit (SFR) algorithm.* Out-of-order arrivals at the receiver cause the receiver to send duplicate

[2]We use M-ACK to differentiate an ACK sent at the link (MAC) layer.

[3]We use M-DATA to differentiate M-PDUs carrying data from the upper layer protocol.

[4]*SHORT RETRY LIMIT*

[5]*LONG RETRY LIMIT*

[6]In a real wireless network, typically the transmission range is smaller than the interference (and sensing) range [10].

[7]That is, simply using the standard SCTP without any modifications.

SACKs or SACKs with gap ack blocks[8] which in turn cause spurious fast retransmissions at the sender. SFR addresses this issue by introducing a virtual queue per destination and deducing the missing reports per TSN (Transmission Sequence Number) correctly to prevent the unnecessary fast retransmissions.
- *Cwnd Update for CMT (CUC) algorithm.* The cwnd evolution algorithm of the standard SCTP allows the cwnd of a path to be updated only when a new cumulative ACK arrives at the sender. Therefore, with CMT, when an ACK packet with an unchanged cumulative ACK (caused by the reordering due to the use of simultaneous paths) arrives at the sender, the cwnd values of the paths are not updated. CUC addresses this issue by tracking the latest TSN received in-order per destination and hence avoids unnecessary reduction in the congestion window updates.
- *Delayed ACK for CMT (DAC) algorithm.* The standard SCTP employs a built-in delayed SACK algorithm to reduce the ACK traffic. When reordering is observed at the receiver, the delayed ack algorithm of the standard SCTP states that the receiver should immediately send an ACK without waiting any further. However, with CMT, there is frequent reordering which will then cause the ACK traffic not to be delayed. DAC attacks this issue by forcing the receiver to send delayed ACKs even when reordering is observed at the receiver to help reducing the ACK traffic.

The availability of multiple destination addresses in an SCTP association allows an SCTP sender to select one destination address for the retransmissions. However, in standard SCTP since only the primary destination address is used to send new data, there is no sufficient information about the condition of all of the other paths. On the other hand, since CMT simultaneously uses all the paths, a CMT sender maintains accurate information regarding the condition of all the paths. Therefore, a CMT sender can better select a path to send the retransmissions. CMT includes several retransmission policies such as the following.

- RTX-SAME: All of the retransmissions of a data chunk are always sent to the same destination address that the original transmission of the data chunk is sent to.
- RTX-CWND: A retransmission is sent to the active destination address with the highest cwnd value.
- RTX-SSTHRESH: A retransmission is sent to the active destination address with the highest ssthresh value.

III. Simulation Setup

Simulation is a widely used methodology in wireless networking research. It is crucial that we use proper wireless physical and link layer models in wireless simulation studies to be able to correctly evaluate the performance and behavior of higher layer protocols [11]. We implemented SCTP CMT in QualNet [12]. Before running any CMT over wireless networks simulations, we validated the correctness of our SCTP CMT QualNet module with the SCTP CMT ns-2 simulation module developed by A. Caro and J. Iyengar [13]. In this validation study, we repeated a subset of the CMT over wired networks ns-2 simulation experiments from [7] in our SCTP CMT QualNet simulation runs. The results confirmed that our SCTP CMT QualNet implementation is compatible to the CMT implementation in ns-2[9].

We then evaluated the performance of CMT in multihop wireless networks context using our SCTP QualNet module[10]. We used a chain topology as depicted in Fig. 2. The nodes in the first chain carries backlogged data traffic via the CMT or SCTP associations. The second chain is for background traffic.

First node in the first chain is the data source and the last node in the first chain is the data sink. We vary the number of hops in the chain. Each node in the first chain is equipped with *two* IEEE 802.11b wireless interfaces operating at *different* frequencies ($freq_1$ and $freq_2$) to ensure *two* independent (non-interfering) multihop wireless paths between the source and destination nodes on the first chain. Each node on the chains is located 300 meters away from each other. The transmission range is around 370 meters, interference range is around 812 meters, and carrier sensing range is around 520 meters for both of the wireless interfaces using the default values in QualNet version 4.5.1[11]. The data rate for IEEE 802.11b is 2 Mbps and RTS/CTS mechanism is on. Each SCTP data chunk carries 1000 bytes of application data.

The second chain is 450 meters away from the first chain. Each node on the second chain has only *one* wireless interface operating at $freq_2$, with the same wireless properties as the second wireless interface of the nodes in the first chain. The number of nodes in the second chain is the same as the number of nodes in the first chain for each simulation. When we want to create a background traffic (i.e., interference) for the CMT subflow running on path 2 of the first chain, we run a CBR (Constant Bit Rate) traffic on the second chain for the entire simulation time. The size of each CBR data packet is 1000 bytes.

We used static routing in the simulations to eliminate the complications regarding the effects of the routing protocol on the performance of the transport layer. The size of IP queue is set to be big enough to hold 50 SCTP data packets. Simulation time is 420 seconds. We measured the steady-state throughput at the receiving application between the 60th and 360th seconds of the simulations. Each data point on the graphs is an average of 30 runs with a 95% confidence interval. In the simulations, SCTP flows and CMT with DAC algorithm use delayed SACKs [3] with the delayed ACK factor of 2 (i.e.,

[8]"Each Gap Ack Block acknowledges a subsequence of TSNs received following a break in the sequence of received TSNs." [4]

[9]Comparison of CMT QualNet vs. ns-2 modules over wired networks scenarios are available as a technical report [14].

[10]The simulations in this paper, is conducted using svn revision 1 of the CMT implementation in QualNet 4.5.1.

[11]Note that, with these settings each node in the chains can communicate only with its direct neighbors but can interfere with the nodes up to 2 hops away.

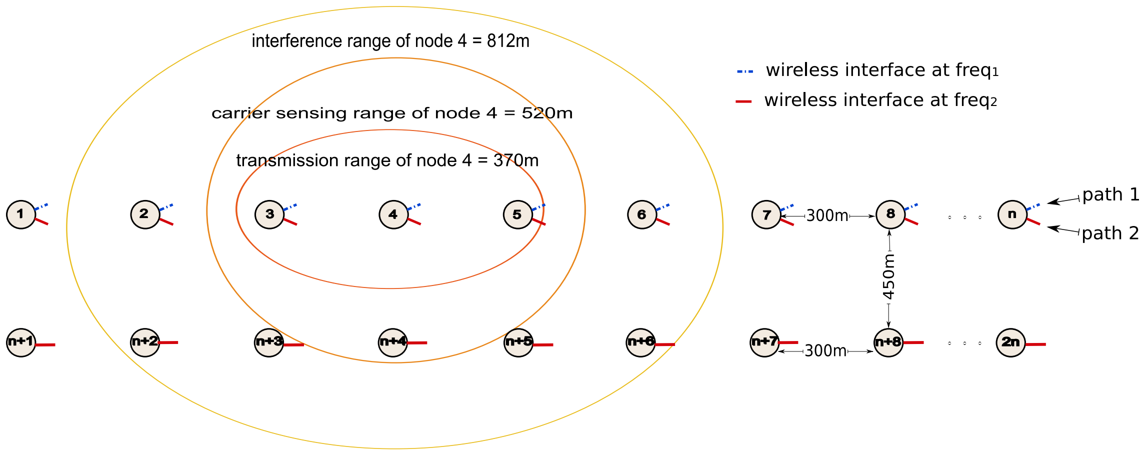

interference range of node 4 = 812m
carrier sensing range of node 4 = 520m
transmission range of node 4 = 370m
wireless interface at freq1
wireless interface at freq2
1
2
3
4
5
6
7
300m
8
n
path 1
path 2
450m
n+1
n+2
n+3
n+4
n+5
n+6
n+7
300m
n+8
2n

Fig. 2. Simulation Topology

one T-SACK packet per SCTP data packet) and the maximum T-ACK delay of 200 milliseconds.

We compared CMT against three other schemes.

- **AwareApp**: an application that always picks the SCTP association that uses the *better* path to send data (i.e., one single-homed SCTP association over path 1 of the first chain in Fig. 2).
- **UnawareApp**: an application that always picks the SCTP association that uses the *worst* path to send data (i.e., one single-homed SCTP association over path 2 of the first chain in Fig. 2).
- **AppStripe**: an "ideal application" that has the best possible performance expected by an application that stripes data perfectly over multiple paths. Essentially, AppStripe represents the aggregated performance of multiple independent SCTP associations running over different paths. Note that, in our simulations, the throughput of AppStripe is the aggregated throughput of AwareApp and UnawareApp.

We investigated the performance of CMT in two settings: (i) with unconstrained receiver buffer (rBuf) at the transport layer (Section IV-A) and (ii) with constrained receiver buffer (Section IV-B). Our goal is to shed light into the following questions.

- How does CMT perform in MWNs as compared to AppStripe, AwareApp, and UnawareApp? How is CMT's performance in MWNs different or similar compared to the CMT performance over wired networks and why (Section IV-A)?
- How influential the receiver buffer (rBuf) blocking problem is on the CMT performance over MWNs? Does rBuf blocking still have a big impact on the CMT performance over MWNs, as it does in the wired case (Section IV-B)?
- How well do the RTX policies of CMT perform in MWNs especially under the constrained rBuf (Section IV-B)?

IV. Results and Analysis

In the following sub-sections, we present the simulation results for CMT, first with an unconstrained rBuf and then with a constrained rBuf.

A. CMT with Unconstrained rBuf

Initially, we did not have a constraint on the size of the receiver buffer (rBuf)[12] of the transport connections.

CMT's initial design goal was to obtain an application throughput as good as the throughput of AppStripe (i.e., one CMT association is performing as good as the aggregated performance of the multiple independent SCTP associations) [7]. However, studies of CMT over wired networks showed that, when the receiver buffer is unconstrained, one CMT flow performs better than the "ideal" data striping application AppStripe [7]. One of the main reasons for the surprisingly better performance of CMT in the *wired* networks as compared to AppStripe is that a CMT flow *shares a single sequence space*[13] across all of the CMT subflows. Therefore, CMT T-ACKs[14] returning from any of the paths to the CMT sender can simultaneously acknowledge data in all of the CMT subflows running over different paths (i.e., one T-ACK can increase the cwnd of all the CMT subflows simultaneously). Therefore, CMT is more resilient to ACK losses on the reverse path.

While investigating the performance of CMT over MWNs, our initial hypothesis was that sharing the sequence space might not bring a clear advantage to CMT over AppStripe. We believe that over MWNs, CMT and AppStripe will have *different* spatial channel reuse because of the following two reasons (see Fig. 3 by focusing on the transport layer mechanics, i.e., the shaded rectangle area in the figure).

- $(Hypo-1)$ *Reduced interference between T-DATA and T-ACK packets:* First of all, in CMT, T-ACKs dynamically return from *any* of the paths to the CMT sender. While T-ACKs are returning to the CMT sender from a path,

[12]We used 2^{32-1} bytes as the size of the receiver buffer. This is the biggest possible value for an rBuf, as allowed by SCTP or CMT. Note that, with this large rBuf size, rBuf is not a performance bottleneck for CMT associations in the configurations we have.

[13]TSN (Transmission Sequence Number) space used for congestion control and reliability.

[14]T-ACKs stands for the ACK packets at the transport layer.

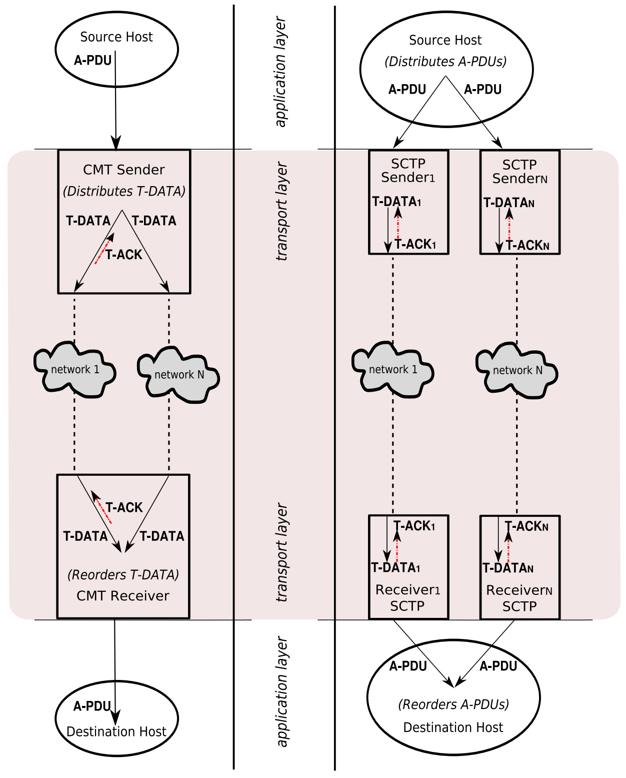


Fig. 3. **CMT on the left**: T-ACKs are sent to the CMT sender via *any* of the return paths and acknowledge the data in *all* of the CMT subflows (i.e., one T-ACK can increase the cwnd of all of the CMT subflows simultaneously). **AppStripe on the right**: T-ACKs are *per* SCTP association and acknowledges only the data of the corresponding SCTP association.

T-ACKs contend for the channel only with the T-DATA packets of the CMT subflow in that path. Whereas, in the AppStripe case, each SCTP receiver returns T-ACKs to its corresponding SCTP sender. That is, there is always contention between the T-ACKs and T-DATAs of an independent SCTP flow. Therefore, when we consider the channel contention between the T-DATA and the T-ACK packets, CMT has *a better spatial channel reuse* across all the paths compared to the aggregated spatial channel reuse of AppStripe subflows running across different paths. This is a clear advantage for CMT.

- $(Hypo-2)$ *Increased self-interference between T-DATA packets:* However, since CMT T-ACKs simultaneously acknowledge multiple CMT subflows, the cwnd of each CMT subflow can grow *more and faster* compared to the cwnd growth of each independent SCTP flow. Cwnd growth reduces the spatial channel reuse (because as cwnd grows, more T-DATA packets are injected into the network and hence more T-DATA packets compete for the channel along the data forwarding path). Cwnd growth can cause performance degradation in TCP when the cwnd of TCP grows beyond the optimal value [15][15]. Therefore, extra increase in cwnd of each CMT subflow might hurt the throughput of each CMT subflow and hence might hurt the overall throughput of CMT compared to AppStripe.

[15]A single-homed SCTP also shows *similar* symptoms [16] since SCTP's congestion control mechanics is "similar" to TCP's.

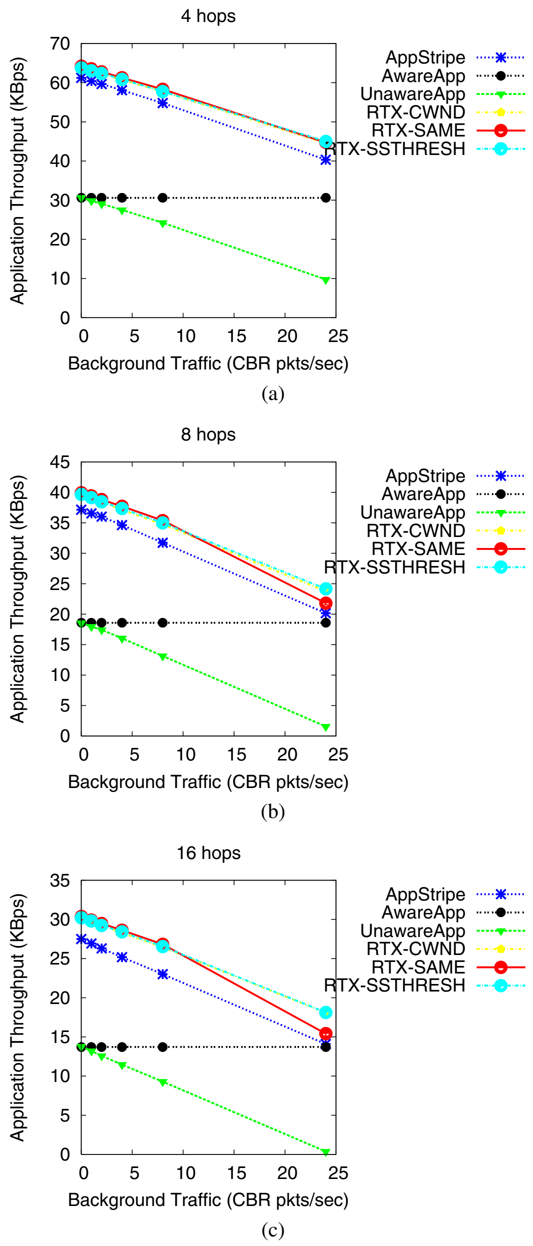


Fig. 4. CMT vs. AppStripe, AwareApp, and UnawareApp with unconstrained rBuf

Therefore, *sequence space sharing* can either increase (as explained in $Hypo-1$ above) or decrease (as explained in $Hypo-2$ above) the spatial channel reuse of CMT compared to AppStripe. Hence, the throughput of an application over CMT could be higher or lower compared to the throughput of the ideal AppStripe.

We have evaluated the performance of CMT with RTX-SAME, RTX-CWND, and RTX-SSTHRESH retransmission policies. Simulation results with unconstrained rBuf size for 4-, 8-, and 16-hop topologies are presented in Fig. 4. We have

TABLE I
CMT (WITH RTX-CWND) VS. APPSTRIPE FOR UNCONSTRAINED RBUF

	0 CBR pkts/sec	8 CBR pkts/sec	24 CBR pkts/sec
4-hop	4.20 %	5.31 %	10.55 %
8-hop	6.55 %	9.48 %	17.88 %
16-hop	9.61 %	14.91 %	28.44 %

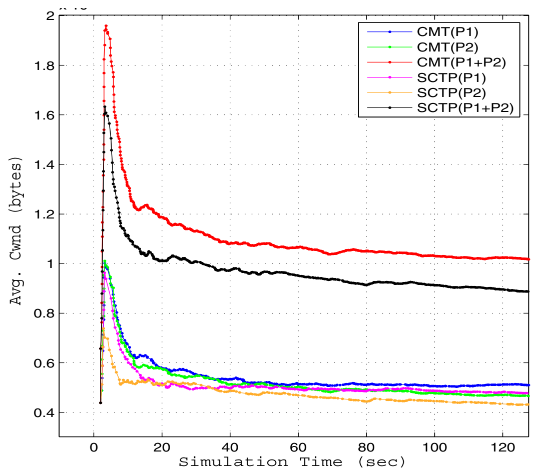


Fig. 5. Average cwnd growths for the flows

the following observations.

- The throughput of CMT over MWNs is $\geq$ the throughput of AppStripe (i.e., aggregated throughput of AwareApp + UnawareApp). This is similar to the wired case as reported in [8]. Table I shows how much CMT performs better than AppStripe, as the number of hops and the loss (interference) in path 2 increases. The values in the table was calculated for CMT with the RTX-CWND policy, using the formula $\frac{(CMT\ throughput - AppStripe\ throughput)*100}{AppStripe\ throughput}$.
- Since we observed that CMT's throughput is $\geq$ AppStripe's in MWNs, we wanted to check $Hypo-2$ further by looking into several traces to understand how the cwnd's of CMT, AppStripe flows and their subflows grow. Fig. 5 shows a progression of the average cwnd's under moderate background traffic (8 CBR pkts/sec) for a 16-hop configuration. What we see in this figure is that cwnd of the CMT subflow 1 grows slightly more (less than one data packet size) than cwnd of the SCTP flow on path 1. In the same way, cwnd of CMT subflow 2 grows slightly more than the cwnd of the SCTP flow on path 2. As we stated in $Hypo - 2$, cwnd of the CMT subflows grow more and faster compared to cwnd of the corresponding AppStripe subflows. However, for our simulation configurations, cwnd growth is not wild enough to hurt the throughput of individual CMT subflows. Hence, the overall cwnd growth of the CMT flow becomes more (almost one data packet size) than the cwnd growth of AppStripe, which leads to higher throughput for CMT.
- As the number of hops increases, the throughput of CMT, AppStripe, AwareApp, and UnawareApp all get smaller. We speculate that the main reason for throughput reduction is that the throughput of an SCTP association[16] decreases as RTT and loss rate of the path increase. Each hop increases the RTT. In addition, as the number of hops increases, the simultaneous transmitters on the chain increase, and hence the contention for the channel (loss rate of the path) increases.

B. CMT with Constrained rBuf

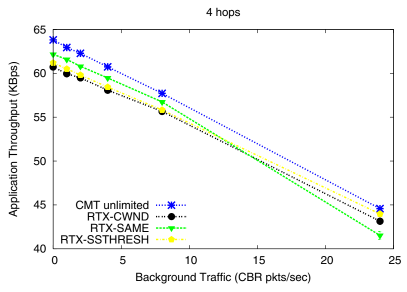
Secondly, we looked into the performance of CMT over MWNs with a limited rBuf size. Smaller rBuf sizes can be a performance bottleneck for CMT due to the *rBuf blocking problem*. The rBuf blocking problem of CMT is explained in [8]. CMT data receiver maintains a single receiver buffer which is shared between the CMT subflows. The receiver uses the rBuf (i) to store the application data arriving out-of-order and (ii) to store the data that the receiving application is not ready to consume. To help flow control, a data receiver sends information about available rBuf space to the data sender, using the $arwnd$ (advertised receive window) field in the SACK chunks. Data sender then calculates $peerRwnd$ value of the association using (i) the $awrnd$ value in the SACKs and (ii) the data that is sent but not acked yet. Data sender uses the $peerRwnd$ to determine how much more data the rBuf at the receiver can hold. The sending rate of path$_i$ (destination address$_i$) at the data sender is then set to $min(cwnd_i, peerRwnd)$[17].

As the receiver keeps data arriving out-of-order from different paths at the rBuf, the available rBuf space shrinks. While the receiver is waiting for the the missing data to come, out-of-order data can not be delivered to the receiving application. In the meantime, the CMT sender calculates the $peerRwnd$ to be very small or zero. This means the sending rate of *any* CMT subflow becomes very small or zero. Therefore, the data sending rate of the *entire* CMT association is blocked, preventing CMT from sending data via any of the paths.

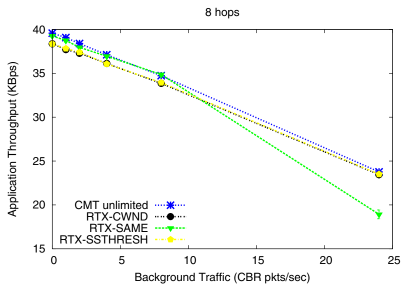
The rBuf blocking problem is unavoidable for CMT, especially if the rBuf size is small and delay, loss, or bandwidth characteristics of the paths CMT subflows run through differ greatly. We looked into the CMT performance for 128, 64, 32, and 16 KB rBuf sizes under light to heavy background traffic on path 2. The results for 64 KB, 32 KB, and 16 KB rBuf are depicted in Figs. 6, 7, and 8 respectively. We observe the following.

- As the rBuf size gets smaller, rBuf becomes a bigger limiting factor in the overall CMT performance (comparing 64 KB vs. 16 KB configurations). In addition, it seems CMT is especially sensitive to the smaller hop (RTTs) configurations (comparing 4 hop vs. 16 hop configurations). We did not see any deterioration in CMT throughput for the 128 KB rBuf except for the RTX-SAME under heavy background traffic (results are not

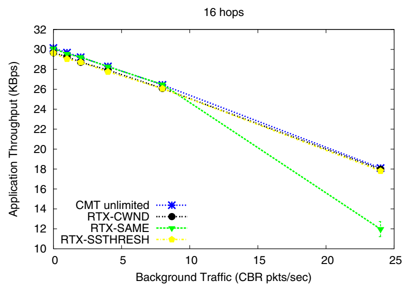
[16]is similar to the throughput of a TCP connection [17].
[17]SCTP CMT sender maintains cwnd *per* destination address.



(a)

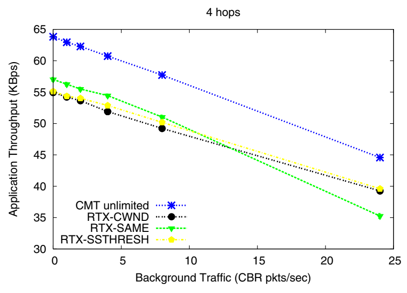


(b)

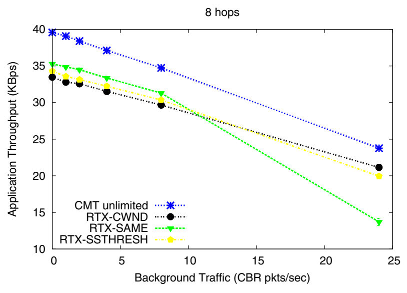


(c)

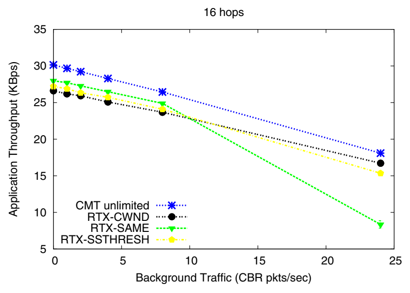
Fig. 6. CMT with 64 KB rBuf vs. CMT(RTX-CWND) with unconstrained rBuf



(a)

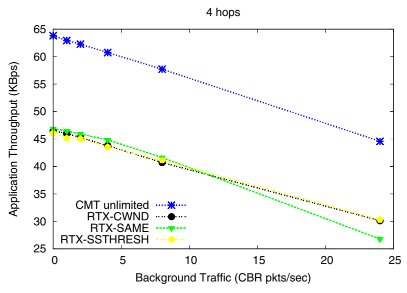


(b)

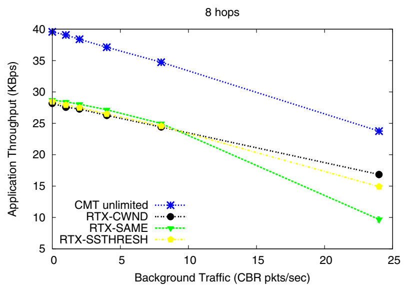


(c)

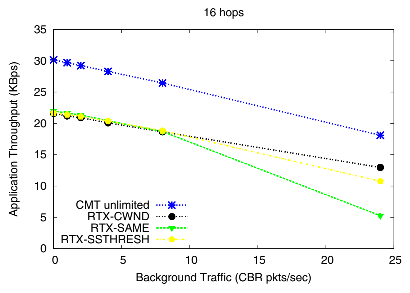
Fig. 7. CMT with 32 KB rBuf vs. CMT(RTX-CWND) with unconstrained rBuf



(a)



(b)



(c)

Fig. 8. CMT with 16 KB rBuf vs. CMT(RTX-CWND) with unconstrained rBuf

presented here). Therefore, for the configurations in this paper, 128 KB seems to be a sufficient number for a rBuf size.

- Another comment is about the performance of the RTX policies of CMT over MWNs. The selection of a RTX policy is particularly important for the constrained rBuf cases. This is because making sure that the rtx's reach to the receiver as early as possible and with minimal loss, increases data delivery rate to the receiver application, which in turn empties the rBuf at the receiver, faster. Iyengar et. al. studied the impact of RTX policies and rBuf blocking in [18], [19]. They concluded that rBuf blocking is unavoidable for CMT but rBuf blocking problem can be mitigated with the selection of a proper RTX policy. They showed that CMT benefits from *loss based RTX policies* (such as RTX-CWND and RTX-SSTHRESH) more as compared to the RTX-SAME policy. Basically, with a loss based RTX policy, retransmission of a data chunk is sent to the lowest loss path among all of the available paths. They suggested to use cwnd (and ssthresh) of a path to approximate the loss rate of the path in the wired networks. That is, with the RTX-CWND (or RTX-SSTHRESH) policy, retransmission of a data chunk is sent to the path with the highest cwnd (or ssthresh) value. We observed that for light to medium background traffic (0-8 CBR pkts/sec on path 2), RTX-CWND (or RTX-SSTHRESH) shows similar or slightly worse performance than RTX-SAME. However, under heavy background traffic (24 CBR pkts/sec), RTX-SAME is clearly worse than RTX-CWND (or RTX-SSTHRESH) especially as hop count (RTT) increases[18]. In addition, we observe that under heavy background traffic RTX-CWND is slightly better than RTX-SSTHRESH for longer hops. We speculate that this is because cwnd is a faster moving value compared to ssthresh and hence can keep up with

[18]Similar to the reports in [19] for CMT over wired networks.

the channel condition better.

V. Related Work

Understanding TCP and (single-homed) SCTP performance over MWNs is important to understand the performance of SCTP-based CMT in MWNs. The performance of TCP in IEEE 802.11 based multihop wireless networks has been studied by various researches [15], [10], [20], [21]. In their seminal paper [15], Fu et. al. talked about the location-dependent contention and spatial channel reuse. The paper shows that the main reason for the losses TCP experiences in an IEEE 802.11 DCF based multihop wireless network is due to the contention-induced losses at the link-layer rather than the buffer over flows at the routers or intermediate nodes. They study the TCP throughput with ns-2 simulations and real hardware experiments. They showed that TCP's throughput is the highest when TCP can operate at a specific window size that allows the best spatial channel reuse. However, cwnd of a TCP connection typically grows beyond this window which in turn reduces the spatial channel reuse and hence causes the TCP connection to experience a below-optimal performance. They showed that for a simple chain topology the optimal value for the cwnd of a TCP flow that achieves the highest TCP throughput is h/4 (h is the number of hops) in a string topology. Kawadia et. al. [21] studied the performance of TCP in a real indoor experimental environment with off-the-shelf IEEE 802.11b cards, and presented results similar to [15]. TCP and single-homed SCTP have "similar" congestion control and flow control characteristics. Only very few papers reported the performance of a single-homed SCTP over multihop wireless networks, such as [16].

In addition to the SCTP-based CMT studied in this paper, there are other proposals, aiming to exploit multihoming of SCTP to allow transmission of data through multiple paths simultaneously. IPCC-SCTP [22], LS-SCTP [23], SBPP-SCTP and Westwood SCTP [24], [25], WISE-SCTP [26], and cmp-SCTP [27] are some of such efforts. To our best knowledge, none of these SCTP-based proposals or any other multihome capable transport protocol is studied in the context of *multihop* wireless networks.

VI. Conclusions and Future Work

In this paper, we studied the concurrent multipath transfer using SCTP multihoming over IEEE 802.11 multihop wireless networks (MWNs). In particular, we studied the performance of the SCTP-based CMT scheme proposed in [8] over MWNs. Similar to the wired network case, we found out and explained why an application over CMT has higher throughput compared to the ideal AppStripe application or an application over a single SCTP association in the MWN context. Our future goal is to improve the performance of CMT over MWNs by adapting the CMT mechanisms to the MWN context.

References

[1] DARPA's Wireless Network After Next (WNAN) Program, http://www.darpa.mil/sto/solicitations/WNaN/.

[2] BBN's PIRANA project, http://www.ir.bbn.com/ ramanath/.

[3] R. Braden, "Requirements for Internet Hosts – Communication Layers," Internet Engineering Task Force, RFC 1122, October 1989.

[4] R. Stewart, "Stream Control Transmission Protocol," Internet Engineering Task Force, RFC 4960, September 2007.

[5] R. Stewart, I. Arias-Rodriguez, K. Poon, A. Caro, and M. Tuexen, "Stream Control Transmission Protocol (SCTP) Specification Errata and Issues ," Internet Engineering Task Force, RFC 4460, April 2006.

[6] R. Stewart and P. D. Amer, "Why is SCTP needed given TCP and UDP are widely available?" ISOC MEMBER BRIEFING 17, June 2004, http://www.isoc.org/briefings/017/index.shtml.

[7] J. Iyengar, P. Amer, and R. Stewart, "Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 951–964, October 2006.

[8] J. Iyengar, "End-to-end Concurrent Multipath Transfer Using Transport Layer Multihoming," Ph.D. dissertation, University of Delaware, Newark, DE, USA, April 2006.

[9] "IEEE Std. 802.11, Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) Specification," 1999.

[10] S. Xu and T. Saadawi, "Does the 802.11 MAC Protocol Work Well in Multihop Wireless Ad hoc Networks," *IEEE Communications Magazine*, June 2001.

[11] M. Takai, J. Martin, and R. Bagrodia, "Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks," in *ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, Rome, Italy, July 2001.

[12] DEGAS Networking Group SCTP web site, http://degas.cis.udel.edu/SCTP/.

[13] A. Caro and J. Iyengar, SCTP ns-2 Simulation Module, http://pel.cis.udel.edu/.

[14] I. Aydin and C.-C. Shen, "SCTP QualNet Simulation Module: Details and a Comparison with SCTP ns-2 Module," University of Delaware, Newark, DE, Tech. Rep. 2009/336, April 2009.

[15] Z. Fu, P. Zerfoz, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The Impact of Multihop Wireless Channel on TCP Throughput and Loss," in *IEEE INFOCOM*, 2003.

[16] G. Ye, T. Saadawi, and M. Lee, "SCTP Congestion Control Performance In Wireless Multi-Hop Networks," in *MILCOM*, 2002.

[17] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," *ACM SIGCOMM*, pp. 303–314, 1998.

[18] J. Iyengar, P. Amer, and R. Stewart, "Receive Buffer Blocking In Concurrent Multipath Transfer," in *GLOBECOM*, November 2005.

[19] R. de Oliveira and T. Braun, "Performance Implications of a Bounded Receive Buffer In Concurrent Multipath Transfer," *Computer Communications*, vol. 30, no. 4, pp. 818–829, 2007.

[20] K. Chen, Y. Xue, S. Shah, and K. Nahrstedt, "Understanding Bandwidth-Delay Product in Mobile Ad Hoc Networks," *Elsevier Computer Communications Journal*, vol. 27, no. 10, pp. 923–934, 2004.

[21] V. Kawadia and P. Kumar, "Experimantal Investigations into TCP Performance Over Wireless Multihop Networks," in *Sigcomm Workshops*, Philadelphia, USA, August 2005.

[22] G. Ye, T. Saadawi, and M. Lee, "IPCC-SCTP: an enhancement to the standard SCTP to support multi-homing efficiently," in *IEEE International Conference on Performance, Computing, and Communications*, 2004, pp. 523–530.

[23] A. A. El, T. Saadawi, and M. Lee, "LS-SCTP: a Bandwidth Aggregation Technique for Stream Control Transmission Protocol," *Computer Communications*, vol. 27, pp. 1012–1024, 2004.

[24] C. Casetti and W. Gaiotto, "Westwood SCTP: load balancing over multipaths using bandwidth-aware source scheduling," in *IEEE VTC2004-Fall*.

[25] F. Perotto, C. Casetti, and G. Galante, "SCTP-based Transport Protocols for Concurrent Multipath Transfer," in *IEEE WCNC 2007*, Hong Kong, March 2007, pp. 2969–2974.

[26] R. Fracchia, C. Casetti, C.-F. Chiasserini, and M. Meo, "WiSE Extension of SCTP for Wireless Networks," in *IEEE ICC*, Seoul, South Korea, May 2005.

[27] J. Liao, J. Wang, and X. Zhu, "cmpSCTP: An Extension of SCTP to Support Concurrent Multi-Path Transfer," in *IEEE ICC*, May 2008, pp. 5762–5766.