COPAS: Dynamic Contention-Balancing to Enhance the Performance of TCP over Multi-hop Wireless Networks

Carlos de M. Cordeiro, Samir R. Das, and Dharma P. Agrawal
Department of ECECS, University of Cincinnati
Cincinnati, OH 45221-0030 USA
{cordeicm, sdas, dpa}@ececs.uc.edu

Abstract – **Most studies on TCP over multi-hop wireless ad hoc networks have only addressed the issue of performance degradation due to temporarily broken routes, which results in TCP inability to distinguish between losses due to link failures or congestion. This problem tends to become more serious as network mobility increases. In this work, we tackle the equally important capture problem to which there has been little or no solution, and is present mostly in static and low mobility multi-hop wireless networks. This is a result of the interplay between the MAC layer and TCP backoff policies, which causes nodes to unfairly capture the wireless shared medium, hence preventing neighboring nodes to access the channel. This has been shown to have major negative effects on TCP performance comparable to the impact of mobility. We propose a novel algorithm, called COPAS (COntention-based PAth Selection), which incorporates two mechanisms to enhance TCP performance by avoiding capture conditions. First, it uses disjoint forward (sender to receiver for TCP data) and reverse (receiver to sender for TCP ACKs) paths in order to minimize the conflicts of TCP data and ACK packets. Second, COPAS employs a dynamic contention-balancing scheme where it continuously monitors and changes forward and reverse paths according to the level of MAC layer contention, hence minimizing the likelihood of capture. Through extensive simulation, COPAS is shown to improve TCP throughput by up to 90% while keeping routing overhead low.**

I. INTRODUCTION

A mobile ad hoc network (MANET) is a network in which a group of mobile computing devices communicate among themselves wirelessly, without the aid of any fixed infrastructure. Here, nodes within radio range of each other are capable of talking directly via wireless links, while those that are far apart use other nodes as relays in a multi-hop fashion.

The majority of applications in the Internet today make use of the TCP (Transmission Control Protocol) for reliable communication. TCP has been designed for and fine-tuned to wired environments; recent studies have shown that current TCP control mechanisms are inadequate for wireless networks, such as traditional one-hop wireless cellular system [1, 2]. TCP in multi-hop wireless networks has been shown to perform even worse [3, 4], since, in addition to all links being wireless and thus causing a high packet error rate, mobility causes frequent link breakages and this leads TCP to normally invoke its congestion control mechanisms even though congestion may not occur. This problem has been the objective of recent research [3, 5, 6, 7, 10], where highly mobile networks have been considered.

An equally important problem is the severe unfairness and capture conditions due to the interplay of the MAC (Medium Access Control) layer and TCP backoff policies [4, 8, 9] which result in a single node in its radio range being able to access the medium at all times, while others in its neighborhood starve. To the same extent as mobility related effects over TCP, the capture problem has been shown to drastically affect TCP performance [4, 8, 9] and is stressed in wireless MAC protocols that employ exponential backoff schemes such as IEEE 802.11 [14] and FAMA (Floor Acquisition Multiple Access) [15] as their backoff mechanisms always favor the last successful station [8, 9].

Contrary to mobility related TCP issues, the capture problem is mostly present when network nodes are static or possess small mobility since nodes stay longer within radio range of each other, while in high mobility networks nodes are often moving out of range of each other and, as a result, nodes rarely have the chance to capture or be captured. As we shall see later, the capture problem is severe enough that when nodes cannot access the medium for some amount of time they generate route error packets even though the network is completely static. For TCP traffic, this causes retransmission timers to go off and throughput to drastically degrade. Capture conditions are very likely in current generation routing protocols as the same route is used for forward and reverse traffic given a <source, destination> pair [11, 12, 13]. For TCP, this implies that data packets in the forwarding direction and ACK packets in the reverse direction compete to access the same shared medium, frequently causing ACK packets to be unable to reach the source and, thus, TCP executing its congestion control algorithms. It has been shown that TCP data packets often capture the medium preventing ACK packets from reaching their destination [8, 9]. As we show later, this problem is worsened by the presence of multiple TCP flows.

In this paper, we propose an algorithm called COPAS (*COntention-based PAth Selection*) to address TCP performance drop due to the capture problem. COPAS implements two novel routing techniques in order to contention-balance the network, namely, the use of disjoint forward (for TCP data) and reverse (for TCP ACK) paths to reduce the conflicts between TCP packets traveling in opposite directions, as well as a dynamic contention-balancing technique that continuously monitors network contention and selects routes with minimum contention to avoid capture conditions. Unlike recent research that either evaluates a single TCP session [3, 5, 6, 10], or when multiple TCP sessions are considered the network is fully mobile [7], or the connections mostly cover one hop employing unrealistic topologies such as ring and string [4, 8, 9], in this work we have performed extensive simulations where we consider multiple TCP connections under several scenarios, and where the network is comprised of only static hosts. This is the worst case scenario where capture conditions are mostly severe since nodes remain within radio range of each other continuously, and where multiple TCP flows compete to have access to the shared medium. The algorithm we propose here could cooperate with any of the other proposed schemes in [3, 5, 6, 7, 10], since we tackle TCP degradation in a static to low mobility network while they cope up with mobility related issues.

The rest of this paper is organized as follows. Section II precisely defines and gives the motivation for the problems attacked in this paper as well as points out the related work, while section III elaborates on our novel COPAS algorithm. Next, sections IV and V present the simulation model and extensive results of our scheme. Finally, this paper is concluded in section VI.

II. Problem Definition, Motivation and Related Work

By far, TCP is the most widely used transport protocol over the Internet today, and every application which requires reliability eventually runs on top of TCP. Therefore, its use over MANETs is a certainty. We can identify two main problems to the large performance detriment of TCP over wireless multi-hop ad hoc networks, where these problems reveal themselves in opposite mobility patterns. The first is the so-called link breakage problem which becomes increasingly worse as nodes become highly mobile. The second is called capture problem and is mostly severe when the network is static or with little mobility.

While current research has dedicated most of its effort to tackle the link breakage problem [3, 5, 6, 7, 10], no effective solution has been proposed to tackle the capture problem. The capture problem is due to the interplay of the MAC layer and TCP backoff policies, which causes nodes to unfairly capture the wireless shared medium and always favor the last successful station [4, 8, 9]. Also behind this problem is the fact that proposed routing protocols for mobile ad hoc networks often employ the very same route for forward and reverse traffic, and as these protocol evaluations have been made using of UDP traffic only, capture conditions have passed unnoticed. Only a partial MAC layer solution to the problem has been suggested in [16], where a yield time scheme is proposed to address the unfairness problem in the specific case of IEEE 802.11. However, this causes the aggregate throughput to degrade even more, and conflicts of TCP packets flowing in opposite directions still remain. In [23, 24] the issue of fairness is addressed, but they to not cope up with the problems we discuss here. Nevertheless, we see our proposed solution and the ones in [16, 23, 24] to be complementary rather than competing since we employ network layer strategies while they suggest MAC layer techniques.

As an example of the conflict of TCP data and ACK packets traveling in opposite directions, consider Figure 1 where a TCP connection exists between nodes S and D. Now let us assume that at the time node S is sending a TCP data packet to node A destined to D, node D itself senses the medium in order to forward an ACK packet to node B destined to S. Surprisingly, node D will detect node's S transmission to node A and will backoff. This is because in carrier sense wireless networks, the interfering range – and sensing range – is typically larger than the communication range [18]. As a matter of fact, WaveLAN wireless systems [22] are engineered in such a way that the interfering and sensing range are more than two times the size of the communication range [8]. Therefore, larger sensing and interfering ranges will severely degrade the network performance in wireless multi-hop networks [8, 18]. The larger interfering range makes the hidden terminal problem worse, while the larger sensing range intensifies the exposed terminal problem. In particular for TCP which relies on traffic in both forward and reverse direction, this is a serious detriment to performance. This problem can be aggravated in such a way that, in Figure 1, some nodes along the forward traffic completely capture the shared medium and totally prevent TCP ACK packets from flowing back and reaching the source node, hence resulting in TCP timeouts and consequent performance degradation. This is exaggerated when multiple TCP connections use nearby nodes as they tend to capture each other, hence badly degrading the overall throughput.

Our approach, elaborated in the next section, tackles this problem by finding two disjoint paths with minimum "contention" during route construction phase, and monitoring these paths while they are still valid. We accomplish to find minimum contention paths by monitoring MAC layer behavior. TCP data and ACK packets travel through the two disjoint paths. Since the traffic pattern may change in the network, we also dynamically monitor the network contention to alter routes if needed.

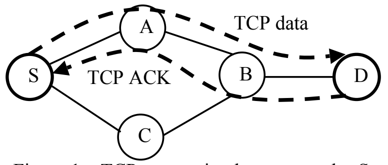


Figure 1 – TCP connection between nodes S and D

Note that here, we distinguish between load and contention. Load is indicated by the number of data packets flowing through a node, while contention is indicated by radio interference seen by the MAC layer of a node. A node with low load can still experience high contention. Our goal here is to focus on minimum contention path rather than minimum load path, as contention is seriously detrimental to TCP performance [4, 8, 9]. We also note here that routing over minimally loaded paths was previously reported in [17], but this work was done with UDP traffic.

III. The COPAS Algorithm

We propose a novel algorithm called COPAS (*COntention-based PAth Selection*) for building exactly two least contented routes to be used as forward and reverse paths between each source and destination pair. These routes are to be rebuilt as the contention along the path changes. COPAS builds routes on-demand and can be implemented in any on-demand routing protocol such as AODV [11] and DSR [12].

A. Route Establishment

In on-demand protocols, a route discovery process is initiated when a route to a destination is needed and none is available. The source floods the network with a ROUTE REQUEST (RREQ) packet to discover a route to the destination. When the destination receives the RREQ, it responds with a unicast ROUTE REPLY (RREP) packet back to the source. The details of how these procedures are done are protocol specific. For example, DSR uses source routing and stores routing information in the form of route caches, whereas AODV employs hop-by-hop routing and stores routing information in route table entries.

In COPAS, upon receipt of a non-duplicate RREQ packet, to this packet nodes append a weighted average ($\overline{\mu}_{BACKOFF}$) of the number of times it has backed off in the last $T_{BACKOFF}$ seconds due to activity in the medium. The RREQ packet is then re-broadcast. By keeping track of the recent average number of times a node has backed off, we are actually determining how busy the wireless shared medium is in the neighborhood of a node. More times a node backs off, means that more busy is the medium around it. This gives us precise information on the contention experienced along the paths traveled by a RREQ. In order to calculate $\overline{\mu}_{BACKOFF}$, every node determines how many times it has backed off ($N_{BACKOFF}$) at each $T_{BACKOFF}$ seconds and computes the following:

$$\overline{\mu}_{BACKOFF} = (1-\alpha)\times\overline{\mu}_{BACKOFF} + \alpha\times N_{BACKOFF} \qquad (1)$$

where α is a constant less than 1. We anticipate that contention will be better modeled by emphasizing more recent information. Thus, we recommend $\alpha > 0.5$.

After receiving the first RREQ packet, the destination waits for an appropriate amount of time ($T_{COLLECT}$) to learn all possible routes. In order to learn all the routes and their contention information, the destination node accepts duplicate RREQ received from different previous nodes. When the RREQ collection timer expires, COPAS employs two selection criteria in order to choose exactly two routes: path disjointness, and least contented routes.

Disjoint path routing has been explored before in connection with both DSR [20] and AODV [19] routing protocols. COPAS uses similar techniques to choose all possible node-disjoint routes (between source and destination) at the destination and selects the two least contented routes based on the information collected by the arriving RREQ packets. Least contented routes are computed by evaluating the sum of the contentions experienced by the RREQ packets on each node-disjoint route and then minimizing the sum over all disjoint routes available. Ties are resolved by favoring lower route lengths (in hops) and then by the arrival order of the RREQ packets. In the absence disjoint paths, COPAS behaves similarly to existing routing protocols with the difference that it can take advantage of network contention information.

The destination responds with two RREPs along the chosen paths. Along with the RREP, the destination also sets a `direction` flag in the packet header to indicate to the source node which path is to be used as forward (for TCP data packets) and reverse (for TCP ACK packets) traffic. This `direction` information is also kept in a node's routing table. To illustrate this, consider the scenario of Figure 2(a) wherein the source node S sends a RREQ packet towards the destination node D. In this case and with the contention values as depicted in the Figure 2(a), the destination first applies the disjointness path rule and finds out routes i = <S-B-E-H-D>, j = < S-C-J-D >, and k = < S-G-I-F-D> to be disjoint. Next, it applies the minimum contention sum rule and ends up selecting routes i and k to be used as reverse (for TCP ACK) and forward (for TCP data) paths respectively, as showed in Figure 2(b).

Employing disjoint forward and reverse paths is also desirable for robustness reasons. Capture conditions can be so severe that links appear to be broken even when there is no mobility. Therefore, to guarantee continuous operation even in link breakage situations we make use of previously established forward and reverse routes. In a capture scenario, it is usually the MAC layer which reports to the network layer the link breakage (the IEEE 802.11 standard has this capability [14]) since it is in this layer where the capture problem is rooted. When a route is disconnected, the immediate upstream node of the broken link sends a ROUTE ERROR (RERR) message to the source of the route notifying the route invalidation. Nodes along the path to the source remove the route entry upon receiving this message and relay it to the source.

In traditional on-demand routing protocols, the source reconstructs a new route by flooding a RREQ when informed of a route disconnection. In COPAS – in addition to flooding a RREQ to reconstruct the broken route – we redirect TCP packets using the second alternate path when available, hence providing uninterrupted communication. It is up to TCP to recover from potential lost packets due to link breakage, while we try to minimize the route disruption by rerouting data packets. In this case, COPAS behaves similar to existing approaches.

B. Dynamic Contention-Balancing and Route Reconstruction

Traffic pattern across the network changes a lot with time and space. Therefore, routes that were optimal during the initial route construction process may not be good paths anymore as contention might have increased with the new traffic pattern. Therefore, we implement a scheme to dynamically monitor and change routes between any <source, destination> pair that have their contention noticeably increased.

While the data communication is active, intermediate nodes continuously piggyback their contention information, as given by equation (1), on packets flowing through the forward and reverse paths. Source and destination nodes can thus monitor the status of the reverse and forward routes respectively. If either route starts experiencing a contention exceeding a threshold ($\text{BACKOFF}_{\text{THRES}}$), a new and less contented route is selected to replace the high contented path. Routes are hence reconstructed dynamically in order to keep balanced the overall contention level in the network or, in other words, COPAS seeks to continuously contention-balance the network. The process of building new routes is similar to the initial route discovery process except that the source or destination floods a RREQ packet to each other, depending on which path is to be replaced (forward or reverse). The receiving end, upon receiving RREQ packets, proceeds in the same way as already described in order to select a disjoint and least contented route. However, in this case, we make two additional modifications. First, we try to find a new route with equal or similar path length as the previous one so that the round-trip time perceived by TCP does not change much. Second, the receiving end does not need to send back a RREP packet and simply starts using the newly discovered route.

C. Replies from Intermediate Nodes

On-demand routing protocols often make use of prior routing information (e.g., a route cache) in order to quickly establish paths between nodes [11, 12]. In case route caches are used, this can have a negative impact on the performance of COPAS since it would not be able to acquire up-to-date information of path contention. Therefore, in COPAS, intermediate nodes cannot send a RREP back to the source even when they have route information to the destination. To utilize the most up-to-date contention information when selecting routes and to minimize overlapped routes which might be suffering from capture problems, we strictly prohibit intermediate nodes from replying to RREQs. Intermediate nodes replying to RREQs has an advantage of reducing the propagation of flooded packets, but may result in increased contention and a RREP storm (as in the case of DSR), that is, too many nodes sending RREPs at the same time raising the levels of both contention and collisions.

IV. Simulation Environment and Methodology

We compare COPAS (implemented as a modified version of DSR) and the base DSR [12] routing protocol which uses shortest path, and compare their performance under TCP. The results of this paper are based on simulations using the *ns-2* (Network Simulator) [21] which has support for simulating multi-hop wireless networks with features including physical and MAC protocols. Our network model consists of two simulation scenarios, for 50 and 100 nodes, in a 1000m x 1000m flat rectangular area. To evaluate the worst case capture scenario, the nodes in the network are static for an entire simulation run that lasts for a total of 600 seconds. All nodes communicate with identical, half-duplex wireless radios that are modeled after the commercially available IEEE 802.11-based WaveLAN [22] radio interface, which possesses a bandwidth of 2 Mbps and nominal transmission range of 250m.

As for the traffic simulation, we have used the TCP-Reno implementation (also employed in [3, 8]) over which FTP file transfers have been conducted. TCP segment size is of 512 bytes, and the maximum window is of 32 packets. Additionally, we vary the number of TCP connections from 1, 3, 5, 10, and 15, where connections start randomly in the first 100 seconds and last for the rest of the simulation period. Unless otherwise noted, all of our simulation results are averaged over 25 scenarios. Each scenario, generated randomly, designates the initial placement of nodes in the considered area.

We use the following performance metrics to compare COPAS and base DSR protocol under the different number of TCP connections:

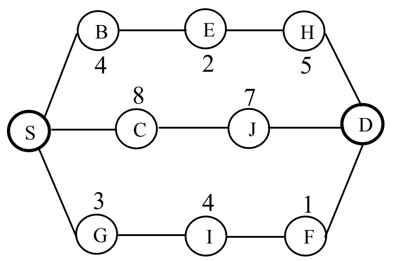


Figure 2(a) – Network contention perceived at node D

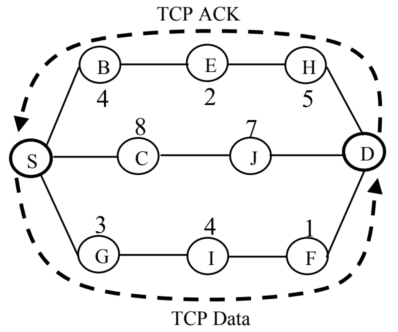


Figure 2(b) – Routes selected by node D

- Throughput – This is the amount of data received by a TCP receiver per unit time;
- Average number of backoffs per second – This is the number of MAC layer backoffs a node experiences per unit time;
- Normalized routing overhead – This is a cost/benefit metric. It is the ratio of the number of bytes of routing packets transmitted per number of bytes of data packets delivered at the TCP receiver;
- Average end-to-end delay of data packets – This includes all possible delays such as buffering during route discovery, queuing, retransmission at the MAC layer, propagation and transfer times.

Table 1 illustrates the parameter configuration used in our COPAS implementation available in *ns-2*.

TABLE 1 – COPAS parameter setup

Parameter	Value
$T_{BACKOFF}$	1 second
$T_{COLLECT}$	1.5 second
α	0.75
$BACKOFF_{THRES}$	6 backoffs per node

V. Simulation Results

A. Throughput

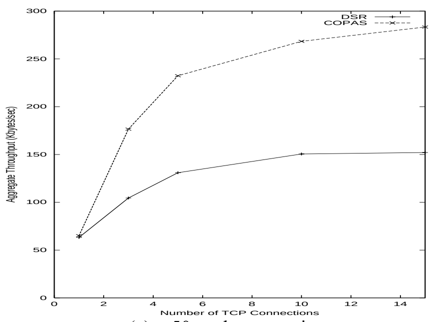
Figure 3(a) compares the throughput achieved by COPAS and DSR in the 50 nodes configuration, and for an increasing number of TCP connections. As we can see from this figure, the aggregate throughput achieved by COPAS is much higher than that of DSR, and as we increase the number of TCP connections COPAS becomes even more effective at finding alternate least-contented paths, while DSR does not succeed in finding better routes. We observe that the aggregate throughput does not increase at the same pace with the number of connections, even on a static network, for two main reasons. Firstly, a given connection C1 going though, say, nodes A, B, C and D, might be capturing the medium around node A but might have been captured in the neighborhood of node C. That is, capturing is something random that TCP connections suffer along the path, hence degrading the overall throughput. The second reason is that capture changes with time. This is to say that a given node A could have been captured by another node B in time t_1, while at time t_2 it is node B which might have been captured by node A. Therefore, while a node is being captured by some other node the TCP retransmission timer goes off and it enters slow start. Eventually, some nodes carrying TCP traffic are captured and force TCP to enter slow start. As we increase the number of TCP connections, the probability of nodes being captured increases resulting in a sensible drop of TCP per connection throughput (Figures 4 and 5) and only a small increase in aggregate throughput (Figure 3). COPAS achieves a drastically higher throughput than DSR (an average of 90% improvement) by choosing distinct forward and reverse paths based on current network contention.

Similar to Figure 3(a), Figure 3(b) also shows the aggregate throughput but now for the 100 nodes scenario. Although COPAS still achieves higher throughput, the relative performance differential is lower. There are two main reasons. First, since now the network has many more nodes and hence previously emptied spots are now occupied, TCP data packets are sort of distributed in the network and TCP connections do not interfere as much as in the 50 nodes configuration. The second reason is that with a larger number of nodes, path lengths between a given source and destination gets longer, hence enabling TCP data packets to effectively achieve pipelining among nodes along the same route but out of radio range of each other. A similar pipelining effect has also been shown in [3, 4]. However, COPAS still achieves higher throughput as it avoids conflict by separating TCP data from ACKs, and for an increased number of TCP connections capture conditions eventually occur irrespective of how large the network is, revealing COPAS to have superior performance.

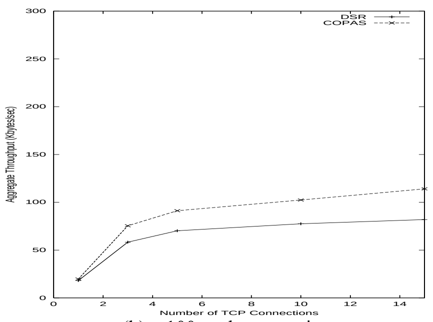
Besides analyzing the average throughput, it is also interesting to study the dynamic behavior of the throughput perceived by a given network node. Figures 4(a), 4(b), 4(c) and 4(d) depict the instantaneous TCP throughput experienced by a pre-selected node during 220 seconds of the connection lifetime, for DSR and COPAS with 1 and 5 TCP connections respectively, in the 50-nodes scenario. As we can see, DSR and COPAS throughputs are similar when there is only one connection in the network, with COPAS still having slightly higher performance due to its distribution of TCP data and ACK in different paths. However, for 5 TCP connections COPAS demonstrates a more stable performance than DSR since it continuously reevaluates routes and reconstructs them according to the shared medium contention. As we can see from Figure 4(c), the throughput of TCP over DSR remains zero or close to zero most of the time, while COPAS throughput bounces back from zero as COPAS reacts to capture conditions by finding alternate routes. Figures 5(a), 5(b), 5(c), and 5(d) show similar results, but now for the 100-nodes scenario.

B. Average Number of Backoffs

Here, we analyze the average number of times the 802.11 MAC layer backs off for increasing number of TCP connections. Figures 6(a) and 6(b) depict such curves, comparing DSR and COPAS for the 50 and 100 nodes scenarios respectively. Nodes using COPAS experience much less contention than those using DSR, especially in the 50 nodes scenario where many nodes are involved in TCP packet forwarding and hence COPAS can take advantage by finding routes other than always the shortest as in DSR. A direct consequence of this fact is that nodes running COPAS have a higher probability of successfully accessing the medium.

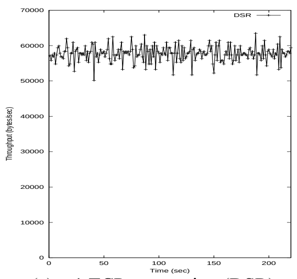


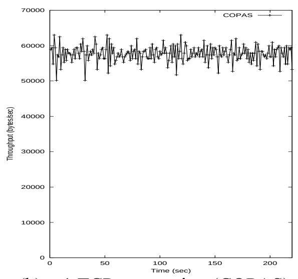
(a) – 50-nodes scenario

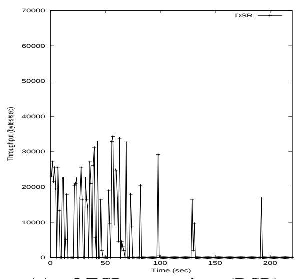


(b) – 100-nodes scenario

Figure 3 – Average Aggregate throughput


(a) – 1 TCP connection (DSR)


(b) – 1 TCP connection (COPAS)


(c) – 5 TCP connections (DSR)

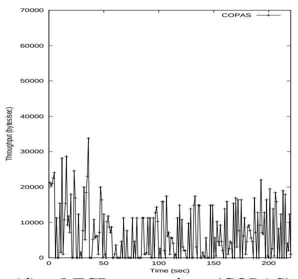
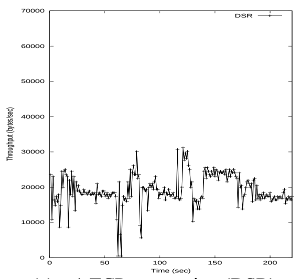
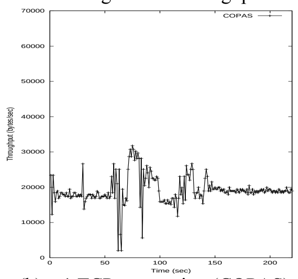
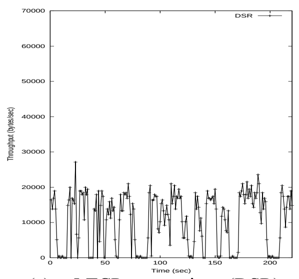

(d) – 5 TCP connections (COPAS)

Figure 4 – Throughput oscillation in the 50-nodes scenario


(a) – 1 TCP connection (DSR)


(b) – 1 TCP connection (COPAS)


(c) – 5 TCP connections (DSR)

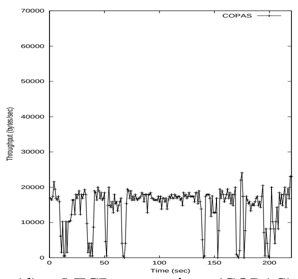

(d) – 5 TCP connections (COPAS)

Figure 5 – Throughput oscillation in the 100-nodes scenario

C. Normalized Routing Overhead

This is an important metric as it relates how much effort in terms of routing packets is necessary in order to obtain the increased TCP throughput. The normalized routing overhead for the 50 and 100 nodes scenarios are shown in Figures 7(b) and 7(c). As we can see, although COPAS makes use of a new RREQ each time a route has to be rebuilt due to increased contention, it is still able to incur much less overhead per unit bandwidth. There are two reasons for that. Firstly, since COPAS balances the network according to contention, situations like the one described in section V.A where capture conditions trigger RERRs become infrequent and, as a result, much less RERR packets are generated in COPAS. The second reason comes directly from the implementation of DSR [21], where destination nodes respond to each and every RREQ they receive with a RREP so that nodes can learn all routes. On the other hand, COPAS replies to exactly two RREQs, hence incurring less overhead at the expense of less information.

D. Average End-to-End Delay

Figure 8(a) and 8(b) show the end-to-end delay for both COPAS and DSR in the 50 and 100 nodes scenarios. As expected, COPAS experiences little higher delay than that of DSR. The reason for this fact is that COPAS ends up choosing paths that are eventually longer (in number of hops) than the shortest one in order to avoid contention. If we return back to Figure 2(b), we see that the trend of COPAS is to choose routes that somehow "go around" the shortest route because their nodes are likely to be out of radio range, and hence path contention is smaller. Therefore, end-to-end delay is slightly higher.

VI. Conclusions

The capture problem of exponential backoff-based MAC protocols (e.g., IEEE 802.11 and FAMA) has been shown to have a negative influence on TCP performance over MANETs. Since TCP is by far the most widely used transport protocol today, techniques to mitigate such effects are necessary. We have proposed a novel algorithm, called COPAS, that achieves this using two techniques — choosing disjoint forward and reverse paths for TCP data and ACK packets and contention-balancing the whole network. Contention-balancing takes into consideration the number of MAC layer backoffs the nodes have experienced recently. COPAS can be deployed on top of any on-demand routing protocol, such as DSR and AODV. Through extensive simulations, we have demonstrated that COPAS provides up to 90% improvement in TCP throughput than baseline DSR.

References

[1] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," in *Proceedings of ACM SIGCOMM'96*, August 1996.

[2] H. Balakrishnan and R. Katz, “Explicit loss notification and wireless web performance,” in *Proceedings of IEEE Globecom*, October 1998.

[3] G. Holland and N. Vaidya, “Analysis of TCP performance over mobile ad hoc networks,” in *Proceedings of ACM MobiCom’99*, August 1999.

[4] M. Gerla, K. Tang, and R. Bagrodia, “TCP Performance in Wireless Multi-hop Networks,” in *IEEE WMCSA’99*, February 1999.

[5] J. Liu and S. Singh, “ATCP: TCP for mobile ad hoc networks,” *IEEE J-SAC*, vol. 19, no. 7, pp. 1300–1315, July 2001.

[6] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, “A feedback-based scheme for improving TCP performance in ad hoc wireless networks,” *IEEE Pers. Comm. Mag.*, vol. 8, no. 1, Feb. 2001.

[7] T. Dyer and R. Boppana, “A comparison of TCP performance over three routing protocols for mobile ad hoc networks,” in *Proceedings of ACM MobiHoc’01*, October 2000.

[8] S. Xu, and T. Saadawi, “Does IEEE 802.11 MAC Protocol Work Well in Multi-hop Wireless Networks?,” in *IEEE Communications*, June 2001.

[9] S. Xu, and T. Saadawi, “Revealing the Problems with 802.11 MAC Protocol in Multi-hop Wireless Ad Hoc Networks,” *Journal of Computer Networks*, Vol. 38, No. 4, March 2002.

[10] F. Wang, and Y. Zhang, “Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response,” in *Proceedings of ACM MobiHoc’02*, June 2002.

[11] C. Perkins, E. Royer, and S. Das, “Ad Hoc On Demand Distance Vector Routing (AODV),” Internet Draft, March 2001 (work in Progress).

[12] D. Johnson, D. Maltz, Y.-C. Hu, and J. Jetcheva, “The dynamic source routing protocol for mobile ad hoc networks (DSR),” Internet Draft, Nov. 2001 (work in progress).

[13] Y.-B. Ko, and N. Vaidya, “Location-aided Routing (LAR) in mobile ad hoc networks,” in *ACM/IEEE MobiCom’98*, October 1998.

[14] IEEE Std. 802-11. “IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification,” June 1997.

[15] C. Fullmer, and J.J. Garcia-Luna-Aceves, “Floor Acquisition Multiple Access (FAMA) for packet radio networks,” Computer Communication Review, October 1995.

[16] K. Tang, and M. Gerla, “Fair Sharing of MAC under TCP in Wireless Ad Hoc Networks,” in *Proceedings of IEEE MMT'99*, October 1999.

[17] S.-Ju Lee, and M. Gerla, “Dynamic Load-Aware Routing in Ad Hoc Networks,” in *Proceedings of IEEE ICC’01,* June 2001.

[18] J. Sobrinho, and A. Krishnakumar, “Quality-of-Service in Ad Hoc Carrier Sense Multiple Access Wireless Networks,” *IEEE J-SAC*, 17(8): 1353-1368, 1999.

[19] M. Marina and S. Das, “On-demand Multipath Distance Vector Routing in Ad Hoc Networks,” *in the Proceedings of ICNP*, November 2001.

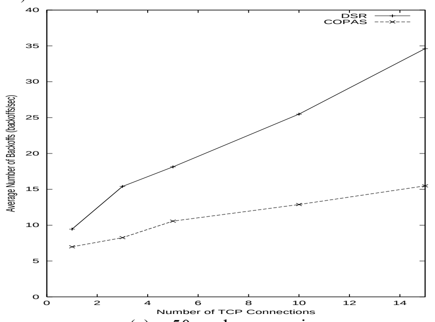
[20] A. Nasipuri, R. Casteneda, and S. Das, “Performance of Multipath Routing for On-Demand Protocols in Mobile Ad Hoc Networks,” in *ACM/Kluwer MONET Journal,* Vol. 6, No. 4, 2001, pp. 339-349.

[21] NS-2 Network Simulator, http://www.isi.edu/nsnam/ns/index.html.

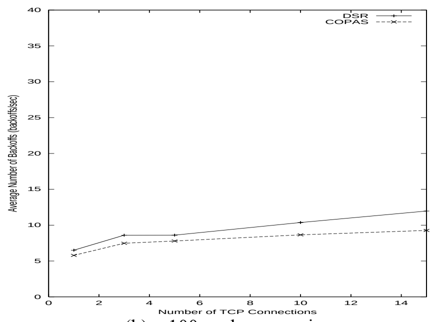
[22] B. Tuch, “Development of WaveLAN, an ISM Band Wireless LAN,” *AT&T Tech. Journal*, vol. 72, no. 4, July 1993.

[23] B. Bensaou, Y. Wang and C. C. Ko, “Fair Media Access in 802.11 based wireless ad-hoc Networks,” *Proceedings of Mobihoc’2000*, Aug. 2000.

[24] N. H. Vaidya, P. Bahl and S. Gupta, “Distributed Fair Scheduling in a Wireless LAN,” *Proceedings of Mobicom’2000*, Aug. 2000.

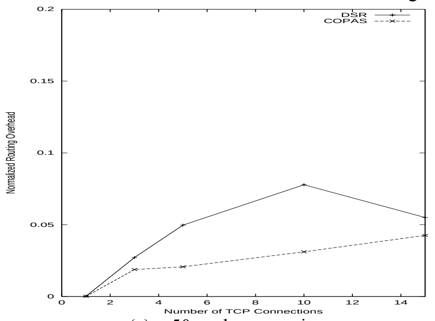


(a) – 50-nodes scenario

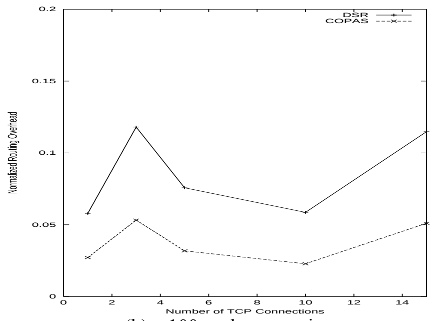


(b) – 100-nodes scenario

Figure 6 – Average number of backoffs

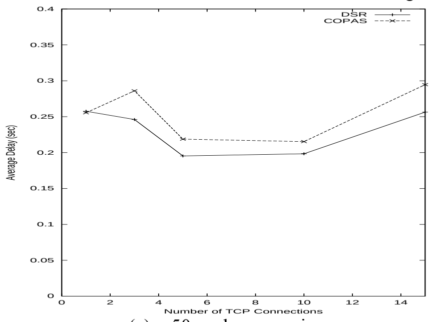


(a) – 50-nodes scenario

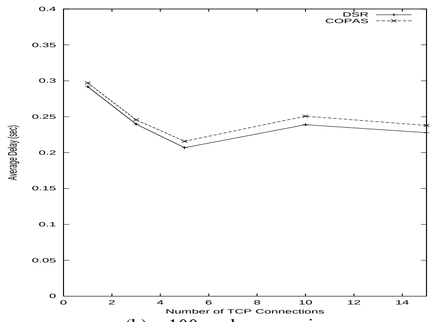


(b) – 100-nodes scenario

Figure 7 – Normalized routing overhead



(a) – 50-nodes scenario



(b) – 100-nodes scenario

Figure 8 – Average end-to-end delay