# Experimental Evaluation of TCP Performance in Multi-hop Wireless Ad Hoc Networks

Abhinav Gupta Ian Wormsbecker Carey Williamson
Department of Computer Science, University of Calgary
{gupta,ianw,carey}@cpsc.ucalgary.ca

## Abstract

*This paper presents experimental measurements of TCP bulk data transfer performance in a multi-hop wireless ad hoc network environment. The first part of the paper studies how TCP throughput is affected by AODV routing, user mobility, and the number of hops traversed in the network. The second part of the paper studies the effectiveness of rate-based pacing (RBP) of TCP packets in improving TCP throughput. Contrary to prior simulation results in the networking literature, our measurement results show no performance advantages for RBP TCP in our experimental scenarios.*

## 1. Introduction

Multi-hop wireless ad hoc networks present many challenges for TCP (Transmission Control Protocol). End-to-end reliable delivery of user-level data requires each TCP packet (segment) to traverse one or more intermediate hops en route to the destination. In addition to the unreliable wireless transmission at each hop, contention problems from "hidden nodes" and "exposed nodes" in the wireless network limit the number of TCP data packets that can be in flight concurrently from source to destination. These physical-layer properties constrain the TCP throughput achievable over a multi-hop path [5].

To further complicate matters, TCP acknowledgement (ACK) packets must travel upstream to the TCP source, against the downstream flow of TCP data packets. Correlated arrivals of TCP data and ACK packets lead to contention for the wireless channel, which can cause excessive collisions [10] and even packet losses [5]. These problems arise even for a *single* TCP flow on a multi-hop wireless ad hoc network. As a result, throughput degrades as the length of the multi-hop path increases.

Several approaches have been proposed in the literature to tackle these challenges. For example, Fu *et al.* [5] propose three techniques: (1) constraining the growth of the TCP congestion window size so that it does not exceed a pre-calculated optimal size; (2) implementing a form of RED (Random Early Detection) [4] at the Medium Access Control (MAC) layer to reduce contention; and (3) using adaptive pacing of TCP packets. Kuang *et al.* [9] take a different approach. They propose a multi-channel MAC protocol, with bi-directional channel reservations, to solve this problem. In both of these papers, network simulation is used as one means to demonstrate the effectiveness of the proposed solutions.

The purpose of our paper is to study the practical performance of TCP in a multi-hop wireless ad hoc network environment. There are three main contributions in this paper. First, we present empirical measurements of TCP throughput in a multi-hop wireless ad hoc network environment, using an experimental signal-strength-aware version of the *Ad-hoc On-demand Distance-Vector* (AODV) routing protocol. We demonstrate the functionality of this routing protocol, and quantify the impacts of user mobility and AODV overhead on TCP performance. Second, we demonstrate experimentally that TCP throughput degrades with the number of hops traversed (consistent with prior known results). Third, we evaluate the effectiveness of TCP rate-based pacing (RBP) in improving end-to-end throughput. Contrary to prior simulation results in the networking literature, our measurement results show no performance advantages for RBP TCP in our experimental scenarios. We attribute this outcome to wireless channel contention issues.

The rest of this paper is organized as follows. Section 2 briefly summarizes prior work on TCP performance in wireless ad hoc networks. Section 3 describes the experimental implementation of AODV with signal-strength-aware routing, and Section 4 the experimental environment. Section 5 presents the mea-

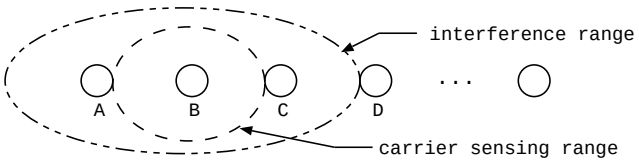


**Figure 1. Multi-hop Wireless Ad Hoc Network**

surement results for TCP throughput, establishing a baseline for TCP performance and its sensitivity to AODV routing dynamics. Section 6 studies the effectiveness of RBP TCP. Section 7 concludes the paper.

## 2. Background and Related Work

This section provides background information on wireless ad hoc networks, and on TCP performance problems in such environments.

### 2.1. Multi-hop Wireless Ad Hoc Networks

Multi-hop wireless ad hoc networks offer communications capability to mobile hosts without requiring a fixed infrastructure. In such a network, packets can traverse multiple intermediate nodes en route from the source to the destination.

An example of a multi-hop wireless network is shown in Figure 1. This example shows four nodes (labelled A, B, C, D) in a simple "chain" network topology. Many other topologies are possible, though we consider only the simple chain network in this paper. The topology is called multi-hop because (for example) a packet destined from A to D must use B and C as intermediate routers. Each forwarding step is called a hop.

The overall performance achieved within a multi-hop wireless network depends on the Medium Access Control (MAC) protocol and transport protocol used. For mobile nodes, performance also depends on mobility patterns and the ad hoc routing protocol used.

A popular MAC protocol for wireless networks is the IEEE 802.11 MAC [2]. This protocol requires each node to sense the channel before sending a frame. The protocol is called CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance). To address the hidden node problem, the 802.11 MAC uses a Request-To-Send (RTS) and Clear-To-Send (CTS) handshake. A node sends an RTS control frame to indicate that it has a frame to send. Upon receiving the RTS, the intended receiver returns a CTS control frame if it is okay to receive the data frame. In this fashion, a packet traverses one hop at a time toward the destination.

### 2.2. TCP Performance

Many previous studies show that TCP performance in multi-hop wireless networks is poor [5, 12, 16]. TCP throughput often decreases dramatically with the number of hops traversed by a flow, regardless of the MAC protocol used [5, 12]. The primary reason is link-layer packet losses caused by contention between data packets traveling in the same direction, and collisions between data packets and TCP ACK packets traveling in opposite directions.

Several methods have been proposed to remedy these problems. Fu *et al.* [5] propose a link-layer version of RED [4] to signal the TCP sender about impending congestion, and an *adaptive pacing* algorithm to distribute TCP data packets evenly across a multi-hop chain. Combined, these algorithms improve throughput by 5%-30%. As another example, Cordeiro *et al.* [3] propose disjoint routes for forward TCP packets and backward TCP ACKs so that contention is reduced. They report throughput improvements of 90%.

Several *multi-channel* MAC protocols have been proposed to improve the overall ad hoc network capacity [6, 7, 11, 13, 15]. We do not consider multi-channel approaches in this paper, but we have studied them in prior work [9].

## 3. Signal-Strength-Aware AODV

This section describes the design and implementation of our signal-strength-aware version of AODV routing, which provides the basis for our experimental evaluation of TCP performance.

### 3.1. AODV Overview

In an ad hoc network, mobile nodes must communicate with each other to determine appropriate routes to use. One approach is the Ad-hoc On-demand Distance-Vector (AODV) routing protocol. In AODV, mobile nodes advertise their presence in the network by broadcasting HELLO beacons periodically (e.g., once per second) to their neighbours.

AODV uses three types of control packets for managing network routes. A *Route Request* (RREQ) packet is initiated by a sender that has no known route to a desired destination. A *Route Reply* (RREP) packet is returned by a node with a known route to the destination indicated in an RREQ. Each RREQ carries a (monotonically-increasing) sequence number so that the matching RREP can be determined. When an RREP is received in response to an RREQ, the sender

records the route received, and uses it for subsequent data packets sent to that destination. A *Route Error* (RERR) packet is returned by a node along a (formerly working) route that is no longer valid (perhaps because of node movement).

AODV is designed to maintain fresh routes. A node updates its AODV routing table whenever it receives a control packet (RREQ, RREP, or RERR) with a higher sequence number than it has recorded in its routing table for a given destination.

### 3.2. Modifying AODV

Our experimental version of AODV is designed to choose *stable* routes, rather than choosing the freshest route or the shortest route. The primary rationale for this choice is that frequent route breakages are undesirable. Route breakages trigger RERR packets and a renewed route discovery process, temporarily stalling the TCP data transfer for that sender. A secondary rationale is that route flapping (i.e., rapidly changing back and forth between several candidate routing paths) is also undesirable, since it can lead to out-of-order packet delivery, and reduced TCP efficiency.

In our AODV protocol, determining the next hop to use for a "stable" route is based on received signal strength. In particular, the signal strength from HELLO beacons and control packets (RREQ, RREP, RERR) are used to ascertain the proximity of neighbour. The assumption is that all nodes transmit at the same power, and that the received signal strength from a node is inversely related to the distance from the receiver. To avoid choosing nodes at the periphery of the coverage range, only HELLO messages with a signal strength above a specified HELLO_STRENGTH_THRESHOLD are considered for processing.

### 3.3. Implementation Overview

Our implementation of signal-strength-aware AODV is based on the public-domain AODV implementation provided by Uppsala University (UU). We modified AODV-UU to incorporate signal strength functionality, choosing stable routes based on the signal strength received from the neighbouring nodes.

The AODV-UU implementation contains several modules (see Figure 2). The three kernel modules are `kaodv`, `k_route`, and `libipq`. The `koadv` module registers packet handling functions for three Netfilter hooks: `NF_IP_LOCAL_OUT` for handling locally generated packets; `NF_IP_PRE_ROUTING` for handling incoming packets prior to routing; and `NF_IP_POST_ROUTING` for re-routing packets prior to sending them. Packets arriving via `NF_IP_PRE_ROUTING` or `NF_IP_LOCAL_OUT` are queued in user space for AODV to process them, while those arriving via `NF_IP_POST_ROUTING` (i.e., packets to be sent out by the system) are re-routed using the latest information in the kernel routing table. The `libipq` module provides user-space queueing of IP packets. It uses a netlink socket to communicate with Netfilter, so that it can decide whether to drop (`NF_DROP`) or accept (`NF_ACCEPT`) packets arriving from user space. The `k_route` module modifies the kernel routing table. The user space modules handle the packets, depending on their type, as described in the next section.

### 3.4. Packet Handling

Data packets and AODV control packets are handled differently. Initial packet processing is performed by the `packet_input` module, which checks to see if a packet is a control packet or a data packet.

If the packet is a control packet, an accept verdict is returned to `libipq`, and the packet is copied to the proper socket for handling. Control packets are handled using a UDP socket on port 654, the well-known port assigned for AODV operation. The control packets are passed to the `aodv_socket` module. In our AODV variation, the `link_strength` module first checks the signal strength of the received control packet. If the signal strength is below the `HELLO_STRENGTH_THRESHOLD`, the control packet is not processed further. Otherwise, the type field is checked and the appropriate AODV module is called to handle the packet.

If the packet is a data packet, then it is analyzed further. If the destination IP address is the receiving host, then the packet is accepted and handled as usual by Linux. The same applies for broadcast packets. For all other cases, the AODV routing table is checked for a valid route to the indicated destination. If a route is found, then the next hop of the packet is set, and the packet is forwarded. If no route is found, then two possibilities arise. If the packet was generated locally (determined from the source IP address), then the packet is queued temporarily in user space (i.e., `libipq` is called by `packet_queue`) until AODV route discovery (RREQ) completes. If the packet was not generated locally, then the packet is discarded, after sending an RERR to the source of the packet.

### 3.5. Signal-Strength Awareness

The signal strength associated with a control packet is obtained using an `ioctl` call on the socket accepting

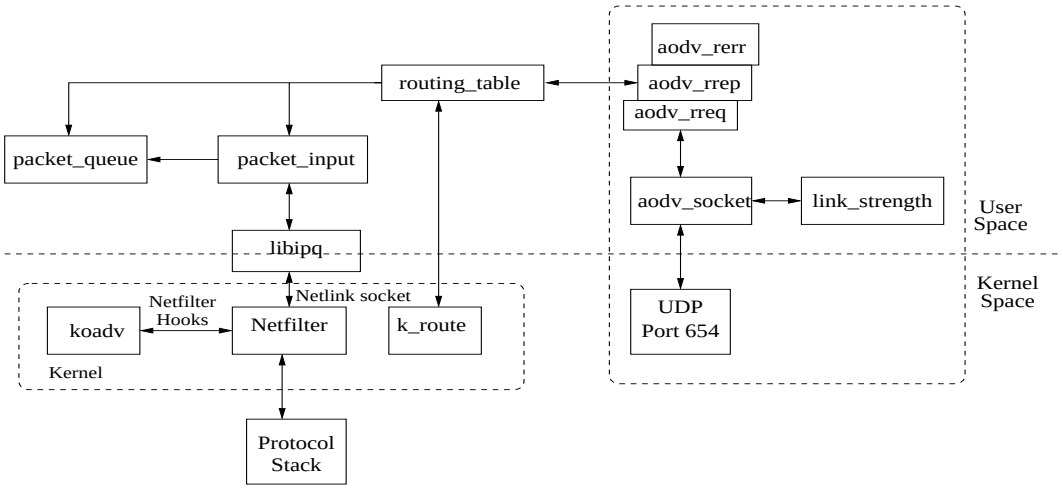


**Figure 2. Packet Handling and Route Update Modules in Signal-Strength-Aware AODV**

the control packets. The request code is `SIOCGIWSPY`, and the signal strength information is returned using an `iwreq` structure. The link quality data is copied to `iw_qual` to determine signal strength in dBm.

Figure 3 shows a graphical illustration of the signal strength variation for a mobile node, first moving away from and then back toward a static node in the ad hoc network. The raw signal strength value fluctuates at short time scales because of the wireless radio propagation characteristics. To dampen these fluctuations and better track user mobility, we use an exponentially weighted moving average for the signal strength. Preliminary experiments showed that a weight of $\alpha = 0.25$ for the most recent signal strength sample provided good results. The smoothed value of signal strength is used as the decision variable for control packet processing, as mentioned above.

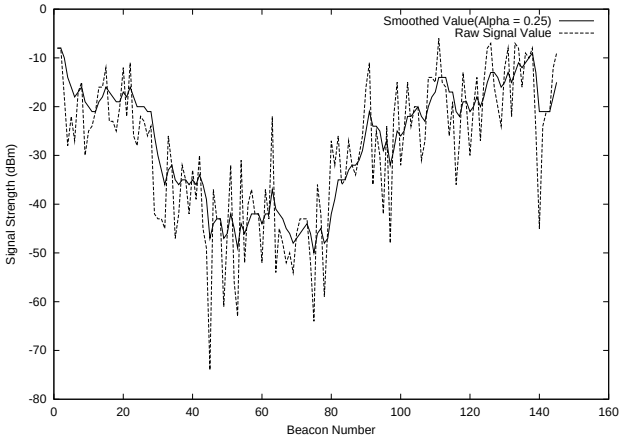


**Figure 3. Example of Signal Strength versus Time for a Mobile Node**

## 4. Experimental Methodology

The experimental environment for our work consists of 5 laptop computers (3 IBM Thinkpads, 2 Compaq notebooks) in a wireless ad hoc network in the ICT building at the University of Calgary. All of the laptops were running RedHat Linux 8.0 (kernel version 2.4.18-14). Each laptop was equipped with an IEEE 802.11b Cisco Aironet 350 PCMCIA wireless network card. The transmit rate for the cards was configured for automatic rate selection, using channel 6. The transmission power was set to the minimum value allowed, so that a 4-hop ad hoc network could be established in 50 meters of hallway on one floor of the ICT building.

We used a chain topology for our ad hoc network with Node 1 at one end running `netserver`, and Node 5 (the only mobile node) at the other end running `netperf`, as shown in Figure 4. The laptops were arranged so that only adjacent nodes were within the transmission range of each other. We ran the signal-strength-aware version of AODV on all the laptops for these tests.

TCP traffic was generated using `netperf TCP_STREAM` tests, with each test lasting 120 seconds. Unfortunately, we were unable to use `tcpdump` to capture TCP packet traces: running `tcpdump` froze the wireless cards when the interface was put into promiscuous mode. To remedy this problem, we added our own kernel instrumentation to record TCP packet-level statistics.

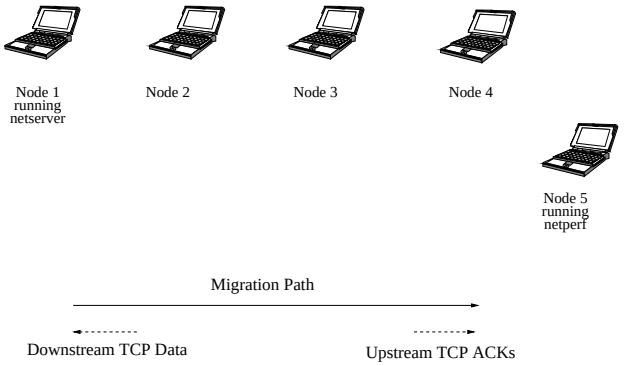


**Figure 4. Wireless Ad Hoc Network Topology**

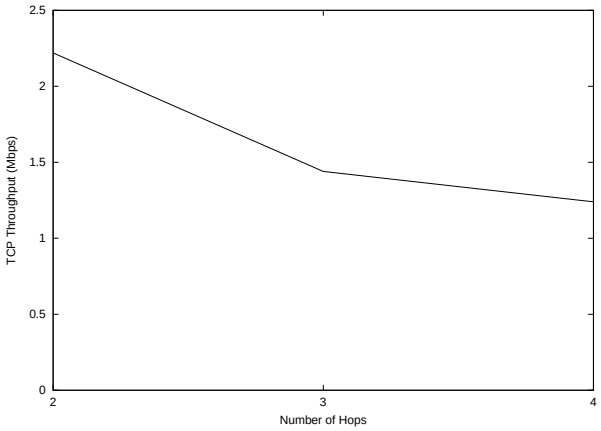


**Figure 5. TCP Throughput versus Number of Hops for Stationary Client**

## 5. TCP Throughput Results

This section presents the results from our TCP throughput experiments using the signal-strength-aware version of AODV. The experiments focus on TCP throughput, and the impacts of network size, user mobility, and AODV routing overhead.

### 5.1. Throughput Measurements

The first experiment studies the sensitivity of TCP throughput to the path length in a multi-hop wireless ad hoc network. The laptops were arranged in a static chain topology. The `netperf` software was run on client Node 5 to send TCP data packets to Node 1 running `netserver`. The TCP throughput was determined from a 2-minute test. Since throughput varied from one run to the next, the median throughput from ten runs was used as a robust estimate.

Figure 5 shows the results from the first experiment. The results (as expected) show that throughput decreases with the number of hops traversed. As a point of reference, the maximum achievable TCP throughput in a single-hop (direct) WLAN configuration with our equipment is 5.1 Mbps [10]. Here, we only consider multi-hop configurations. For two hops (1 intermediate node), the throughput is 2.22 Mbps. For three hops, the throughput is 1.44 Mbps, and for four hops, the throughput is 1.24 Mbps.

The lower throughput in the multi-hop configurations occurs because of hidden node and exposed node problems. Because adjacent nodes are within the carrier sensing range of each other, they cannot transmit at the same time. This property reduces the throughput when the number of hops increases.

The second experiment measured the TCP throughput for a mobile client. The topology and traffic generation for this experiment was the same as before. The only difference is that the mobile client with the laptop walks from one end of the network to the other, beginning near the server (i.e., direct connectivity), and ending just past (but within the range of) the last hop. We consider three different walking speeds. The slow-walking client moves at approximately 0.33 m/sec (1.2 km/hr). The fast-walking client moves at approximately 1.0 m/sec (3.6 km/hr). The medium-walking client moves at a rate in between these two values.

**Table 1. TCP Throughput Results (Mbps)**

| Hops | Static | Slow | Medium | Fast |
|---|---|---|---|---|
| 2 | 2.22 | 2.40 | 2.26 | 1.96 |
| 3 | 1.44 | 2.17 | 2.05 | 1.66 |
| 4 | 1.24 | 1.59 | 1.30 | 1.18 |

Table 1 shows the throughput results for the mobile client. For convenience, the throughput results for the static client are also shown. Note that the measured TCP throughput for the Slow/Medium mobile client is actually higher than that measured for the stationary client from the first experiment. This result occurs because the mobile client begins with direct connectivity to the server (i.e., high throughput) and then experiences worse and worse throughput as they walk further and further away from the server. The throughput achieved early in the transfer skews the measurement results, creating the illusion of higher throughput for the mobile client. Despite this idiosyncracy, the number of hops still clearly influences the overall throughput achieved.

Table 1 also shows the impact of the speed of the mobile client on the TCP throughput. The fast-walking client experiences worse TCP throughput than the

**Table 2. Route Discovery Time (msec)**

| Hops | Min | Median | Max | Mean | StdDev |
|---|---|---|---|---|---|
| 2 | 3 | 7 | 965 | 50.8 | 173.6 |
| 3 | 6 | 10 | 3,212 | 292.8 | 633.9 |
| 4 | 9 | 331 | 5,183 | 613.3 | 999.6 |

slow-walking client. The primary reason is that the fast-walking client spends less time in direct contact with the server, and relatively more time in the multi-hop part of the network. The throughput disadvantage for the fast-walking client is more pronounced for the 3-hop (23%) and 4-hop (26%) networks than for the 2-hop network (18%), since the fast-walking client reaches the lower throughput locations sooner. Also important are the route changes that occur during the walk, and the non-zero route discovery time required for each of these events. For the fast-walking client, the outages induced by route discovery have a larger relative impact on the TCP throughput.

### 5.2. Route Discovery Time

We conducted separate experiments to measure the route discovery time for our signal-strength-aware version of AODV. The route discovery time is the elapsed time between sending an RREQ and receiving the corresponding RREP. This time depends on the distance to the destination, the number of nodes in the network, and the characteristics of the wireless channel.

We observed high variability in the route discovery times. Table 2 shows the results from 40 runs. The median route discovery time was about 10 milliseconds (msec) for the 2-hop network and the 3-hop network, but it increased to 331 msec for the 4-hop network. This is because multiple RREQs had to be sent before a RREP was received. We suspect that RREQ packets were dropped at the link layer due to interference from Node 2. While Node 2 was not within the transmission range of Node 5, it was within the interference range, leading to collisions during RTS/CTS handshakes. For most of the cases, an RREP was received after the second RREQ attempt. The time interval before repeating the RREQ was 320 msec. Assuming that the RREP was received from the adjacent node in another 10 msec, this explains the 331 msec route discovery time for the 4-hop network.

The worst cases observed required over 5 seconds to establish a route. These unpredictable route discovery times can adversely affect TCP throughput.

**Table 3. AODV Routing Overhead**

| Num Hops | Data Packets | Control Packets | Overhead Ratio |
|---|---|---|---|
| 2 | 24,369 | 347 | 0.014 |
| 3 | 17,369 | 405 | 0.023 |
| 4 | 13,249 | 499 | 0.037 |

### 5.3. Routing Overhead

Our experiments measured the AODV control packet overhead for route discovery and route maintainence in our simple network topology. The overhead packets considered are RREQ, RREP, RERR, and HELLO (a special type of RREP packet). These control packets are UDP packets arriving on port 654. A WildPackets Airopeek Sniffer located at the center of the network was used to capture all the control packets. We calculated the *overhead ratio* as the number of control packets divided by the number of data packets.

Table 3 shows the results from selected experiments. The routing overhead ratio clearly increases with the number of hops, though it is always below 4%. The overhead ratio for the 4-hop network is more than double that for the 2-hop network. This increase comes from the additional nodes in the network, each sending HELLO messages to maintain route and connectivity information. Another reason is that fewer TCP data packets are delivered per unit time when the network path length is larger.

### 5.4. End to End Delay

A final experiment was conducted to measure the round-trip time (RTT) in the multi-hop ad hoc network. This experiment was conducted for different packet sizes and different hop counts on the chain topology. RTT was measured using adaptive `ping` so that there was never more than one unanswered ping request in the network at a time.

Table 4 shows the observed RTT values in milliseconds. The results show the expected trends: the RTT tends to increase with packet size, and with the number of hops traversed.

The RTT behaviour is important because it influences TCP throughput. The larger the RTT is, the lower the TCP throughput tends to be. Estimating RTT is also important for TCP rate-based pacing (RBP) [8].

**Table 4. Measured Round Trip Times from Client to Server (msec)**

| Num Hops | Ping Size (bytes) | Min | Median | Max | Mean | StdDev |
|---|---|---|---|---|---|---|
| 2 | 64 | 3.00 | 3.99 | 7.46 | 4.21 | 0.98 |
| | 1024 | 10.20 | 10.85 | 17.80 | 11.36 | 1.41 |
| 3 | 64 | 4.53 | 5.91 | 61.40 | 6.78 | 5.70 |
| | 1024 | 15.70 | 17.50 | 67.10 | 18.57 | 5.32 |
| 4 | 64 | 6.01 | 7.70 | 12.80 | 7.94 | 1.17 |
| | 1024 | 20.80 | 22.30 | 26.60 | 22.59 | 1.28 |

## 6. Results for TCP Pacing

Several papers in the literature have proposed rate-based pacing (RBP) to "spread out" TCP packets in time and improve TCP performance [1, 5, 8]. Early work proposed this technique for Web TCP traffic [1, 8], while more recent work by Fu *et al.* [5] has proposed this explicitly for multi-hop wireless ad hoc networks. In particular, they propose link-layer pacing of packets to coordinate the movement of packets several hops apart in a multi-hop network. Their pacing mechanism operates in conjunction with a link-layer version of RED to handle wireless contention.

As a starting point for our study, we simulated TCP pacing in a multi-hop network topology, using the ns-2 network simulator [14]. The TCP Reno model was modified to implement rate-based pacing, using [8]:

$$InterPacketDelay = \frac{RTT}{CurrentWindow + V}$$

where $RTT$ is the round trip time and $CurrentWindow$ is the current congestion window size (cwnd) in segments. The variable $V$ controls the aggressiveness of the pacing by artificially increasing or decreasing the denominator [8]. For simplicity, $V$ is fixed at 1 in all experiments.

Experiments were performed comparing Reno TCP and RBP TCP, with throughput as the primary performance metric. A simple chain topology as depicted in Figure 6 was used for all experiments. The two endpoints of the chain are used as the TCP sender and receiver. The intermediate nodes in the chain forward packets, but do not generate any traffic of their own. In the simulations, chain length ranges from 2 to 16 nodes. The 2 node scenario has direct connectivity between the sender and the receiver, while the 16-node scenario has 14 intermediate nodes to traverse. Each link is 11 Mbps. RTS/CTS is used, and TCP Delayed-ACKs are disabled.

Figure 7 shows the simulation results. Both RBP TCP and Reno TCP perform similarly (as expected)

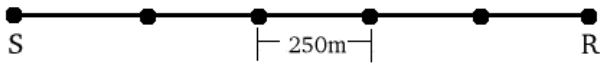


**Figure 6. Example Chain Network Topology**

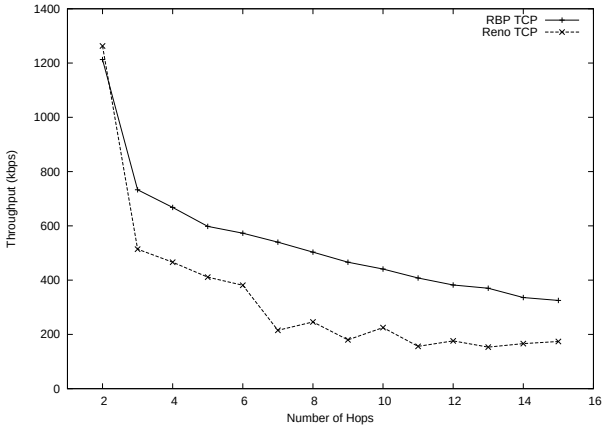


**Figure 7. Simulation Results for Reno TCP and RBP TCP**

when there are only 1 or 2 hops in the network. However, once packets traverse 3 or more hops to the receiver, RBP TCP shows a throughput advantage. The advantage ranges from 10% to 50%, depending on the number of hops. Our results are qualitatively similar to those reported by Fu *et al.* [5], indicating that our simulation model captures the essence of TCP pacing.

Next, we conducted RBP TCP experiments using our experimental ad hoc network. We implemented adaptive pacing of TCP data packets in the Linux kernel of the mobile client (Node 5), and verified its correct operation using network packet traces collected from the network. We then repeated several of the TCP throughput experiments from Section 5.1. We conducted these experiments using the 11 Mbps configuration of IEEE 802.11b.

**Table 5. Throughput Results for TCP Pacing (Mbps)**

| Num Hops | Reno TCP | | RBP TCP | |
|---|---|---|---|---|
| | Mean | SDev | Mean | SDev |
| 1 | 4.69 | 0.02 | 4.21 | 0.22 |
| 2 | 2.22 | 0.06 | 2.13 | 0.03 |
| 3 | 1.44 | 0.04 | 1.31 | 0.11 |
| 4 | 1.24 | 0.01 | 1.08 | 0.03 |

The results from these experiments are shown in Table 5. Our results show *no* performance advantage for RBP TCP in our multi-hop wireless ad hoc network.

There are several reasons for these disappointing results. First, the simulation results have an idealized model of wireless channel contention. Each node has precise, finite values for transmission range and interference range, and the nodes are carefully spaced to provide transmission connectivity while minimizing interference overlap. In our experimental scenario, even though we carefully placed our nodes and controlled the transmit power, there are many wireless propagation characteristics that differ from the idealized model in the simulation. In particular, inter-node interference may be much greater than that assumed in the simulation. Second, the simulation model does not consider AODV routing dynamics. Our experimental measurements show that AODV introduces many artifacts: HELLO beacons, routing packet overhead, non-deterministic forwarding delays, and dynamically varying network round trip times. Third, our pacing implementation was done at the TCP layer rather than the link layer, and operates in isolation. In particular, we do not use a link-layer version of RED.

All these factors disrupt the "perfect world" assumptions for RBP TCP in the simulation. The main take-home message from our experiments is that ns-2 simulation results for TCP performance in multi-hop wireless ad hoc networks should be interpreted with caution, unless they have been validated against experimental measurements.

## 7. Summary and Conclusions

This paper studies TCP performance in a multi-hop wireless ad hoc network environment, making three main contributions. First, we present empirical measurements of TCP throughput in a multi-hop wireless ad hoc network environment running an experimental version of the AODV routing protocol. We demonstrate the functionality and performance of this protocol, and quantify the impacts of user mobility and AODV routing overhead on TCP performance. Second, we confirm prior results that TCP throughput degrades with the number of hops traversed. Third, we evaluate the effectiveness of TCP rate-based pacing. Unlike earlier simulation results in the literature, we find no performance advantage for RBP TCP. The results suggest that the performance of RBP TCP is highly sensitive to channel contention and AODV routing dynamics in a multi-hop wireless ad hoc network.

## References


[1] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the Performance of TCP Pacing", *Proceedings of IEEE INFOCOM*, Tel Aviv, Israel, March 2000.

[2] ANSI/IEEE Standard 802.11b, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz band", 1999.

[3] C. Cordeiro, S. Das, and D. Agrawal, "COPAS: Dynamic Contention-Balancing to Enhance the Performance of TCP over Multi-hop Wireless Networks", *Proceedings of IC3N'02*, Miami, FL, October 2002.

[4] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEE Transactions on Networking*, Vol. 1, No. 4, pp. 397-413, August 1993.

[5] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The Impact of Multi-hop Wireless Channel on TCP Throughput and Loss", *Proceedings of IEEE INFOCOM'03*, San Francisco, CA, April 2003.

[6] W. Hung, K. Law, and A. Leon-Garcia, "A Dynamic Multi-Channel MAC for Ad Hoc LAN", *Proceedings of 21st Biennial Symposium on Communications*, Kingston, ON, Canada, June 2002.

[7] N. Jain, S. Das, and A. Nasipuri, "A Multichannel CSMA MAC Protocol with Receiver-Based Channel Selection for Multi-hop Wireless Networks", *Proceedings of the IEEE ICCCN'01*, Phoenix, AZ, October 2001.

[8] J. Ke, "Towards a Rate-Based TCP Protocol for the Web", *Proceedings of MASCOTS'2000*, San Francisco, CA, pp. 36-45, October 2000.

[9] T. Kuang and C. Williamson, "A Bidirectional Multi-Channel MAC Protocol for Improving TCP Performance on Multi-Hop Wireless Ad Hoc Networks", submitted for publication, 2004.

[10] T. Kuang, F. Xiao, and C. Williamson, "Diagnosing Wireless TCP Performance Problems: A Case Study", *Proceedings of SCS SPECTS Conference*, Montreal, PQ, pp. 176-185, July 2003.

[11] J. So and N. Vaidya, "A Multi-channel MAC Protocol for Ad Hoc Wireless Networks", Technical Report, Dept. of Computer Science, University of Illinois at Urbana-Champaign, January 2001.

[12] K. Tan and M. Gerla, "Fair Sharing of MAC under TCP in Wireless Ad Hoc Networks", *Proceedings of IEEE MMT'99*, Venice, Italy, October 1999.

[13] A. Tzamaloukas and J. Garcia-Luna-Aceves, "A Receiver-Initiated Collision-Avoidance Protocol for Multi-Channel Networks", *Proceedings of IEEE INFOCOM'01*, Anchorage, USA, April 2001.

[14] VINT Group, "Network Simulator ns-2", available at `http://www.isi.edu/nsnam/ns`

[15] S. Wu, Y. Tseng, C. Liu, and J. Sheu, "A Multi-Channel MAC Protocol with Power Control for Multi-Hop Mobile Ad Hoc Networks", *The Computer Journal*, Vol. 45, No. 1, pp. 101-110, 2002.

[16] S. Xu and T. Saddawi, "Does the IEEE 802.11 MAC Protocol Work Well in Multi-hop Wireless Ad Hoc Networks?", *IEEE Communications Magazine*, June 2001.