Cross-Layer Flow Control in Lightly-Loaded Multi-Hop Ad Hoc Networks

Alex Varshavsky
Department of Computer Science
University of Toronto
walex@cs.toronto.edu

Baochun Li
Department of Electrical and Computer Engineering
University of Toronto
bli@eecg.toronto.edu

Eyal de Lara
Department of Computer Science
University of Toronto
delara@cs.toronto.edu

Abstract

The throughput in multi-hop ad hoc networks (MANETs) is highly dependent on the sending rate and the route length from the source node to the destination. Sending packets at the optimal rate for a given route length maximizes throughput in the network, whereas slightly increasing the sending rate over the optimal value may decrease throughput by up to 55%.

This paper presents a novel cross-layer technique for flow control in lightly-loaded MANETs. The technique allows applications to send packets at the rate that maximizes throughput for a given route length. To achieve this, the routing layer notifies interested applications about routing changes, and the applications adaptively modify their sending rates based on the new route length to the destination. In static and mobile networks, this technique outperforms UDP-based flows with a fixed sending rate and doubles the throughput of TCP for networks with up to 2 concurrent flows.

1 Introduction

A multi-hop mobile ad hoc network (MANET) is a collection of mobile nodes that communicate with one another over wireless links without requiring support of a fixed infrastructure. Instead, nodes agree to relay one another's packets toward their destinations, acting as routers and automatically organizing into a cooperative network. MANETs have been proposed for use in military, disaster relief and emergency operations.

Nodes in MANET typically communicate over the same wireless channel, which prevents closely positioned nodes from transmitting simultaneously. A media access protocol (MAC) is used to arbitrate access to the channel and recover from transmission errors. The most common distributed MAC protocol used in MANETs today is the IEEE 802.11 Distributed Coordination Function (DCF) [9].

Because most traffic in MANETs traverses more than one hop, the fact that two closely positioned nodes cannot transmit simultaneously reduces the achievable throughput significantly. Li *et al.* [10], investigated the problem of achievable throughput in chain topologies and showed that the throughput is dependent on the path length. Moreover, they showed that because of the hidden terminal problem [10] the throughput degrades from its optimal value as the sending rate increases beyond a certain point.

This paper presents a novel cross-layer technique that improves throughput and delivery ratio in lightly-loaded MANETs. We first experimentally obtain the sending rates that achieve the maximum throughput for chains of nodes of different lengths. We then implement a cross-layer flow control architecture, in which the routing layer notifies applications of changes in the routing information, and the applications set their sending rate according to the precomputed value for the current route length.

Cross-layer flow control is extremely simple, but powerful. We have evaluated cross-layer flow control in static and mobile environments. Experimental results show that the cross-layer flow control architecture outperforms UDP-based protocols with a fixed sending rate. Moreover, when there is only a single flow, cross-layer flow control doubles the throughput of TCP. However, as the number of flows increases, cross-layer flow control does less well than TCP because it does not take into consideration cross-traffic incurred by multiple flows.

The rest of this paper is organized as follows. Section 2 describes the IEEE 802.11 DCF and the hidden terminal problem. Section 3 first presents throughput measurements of chain topologies of different lengths, and then describes the proposed cross-layer flow control architecture. Section 4 presents experimental results. Finally, Section 5 describes related work, and Section 6 concludes the paper and discusses opportunities for future research.

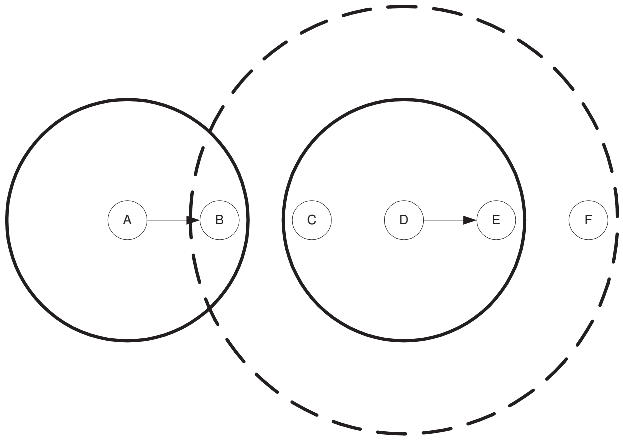

Figure 1. Interference among chain of nodes. Nodes A and D cannot transmit simultaneously.

2 Preliminaries

In this section, we first briefly describe the IEEE 802.11 Distributed Coordination Function (DCF) [9]. We then describe the hidden terminal problem.

2.1 Distributed Coordination Function

In DCF, when a node wishes to transmit a packet, it first defers its transmission for a randomly chosen interval, and then senses the transmission medium. If the medium is idle, the node transmits a Request-To-Send (RTS) packet. On receiving the RTS, the destination node replies with a Clear-To-Send (CTS) packet. On receiving the CTS packet, the source node starts the data transmission. Once the DATA packet was successfully received, the destination node sends an acknowledged (ACK) to the source node. The RTS-CTS exchange is needed to let the neighbor nodes of both the source and the destination know that the data transmission is about to begin. Clearly, a node does not transmit neither RTS nor CTS if it is aware of an ongoing transmission in its neighborhood.

Unfortunately, RTS-CTS exchange prior to the data transmission does not guarantee successful data packet delivery. Both the RTS-CTS exchange and the DATA-ACK exchange can fail either because of channel errors or the hidden terminal problem [10] described below. DCF drops a data packet if it fails to perform RTS-CTS exchange for 7 consecutive times or receive an ACK after 4 consecutive data packet retransmissions.

2.2 The Hidden Terminal Problem

A hidden terminal is a node that is unaware of a transmission in its vicinity and whose attempt to transmit data will corrupt the ongoing transmission. We refer to the existence of hidden terminals in a network as the *hidden terminal problem*. We further refer to the range at which nodes can successfully receive packets as the *transmission range*, and to the range at which the current transmission will corrupt other ongoing transmissions as the *interference range*.

Consider a chain of nodes depicted in Figure 1. Assume that node A is sending packets to node B. The solid-line circles show the maximum transmission range of nodes A and D and the dotted-line circle show the interference range of node D. Note that node D is unaware of the ongoing transmission because it received neither RTS nor CTS packets. A transmission of a packet by node D will therefore corrupt packets received by node B.

As we will show in the next section, the hidden terminal problem greatly affects the maximal achievable throughput in MANETs.

3 Cross-Layer Flow Control

In this section, we first present our measurements of throughput in chain topologies for both unidirectional and bidirectional flows. We then describe our proposed cross-layer flow control algorithm.

3.1 Chain Throughput

We have measured throughput in chain topologies, varying the chain length and the sending rate. For our simulations we used the `ns-2` simulator [5] with CMU wireless extension [11]. The physical radio characteristics were chosen to approximate the Lucent WaveLAN direct sequence spread spectrum radio with a 250m nominal transmission range, 550m interference range and a raw capacity of 2Mb/s. All experiments use distributed coordination function (DCF) of IEEE 802.11 as the MAC protocol.

Nodes where positioned 200 meters apart, while the first node in the chain transmitted 1000 byte packets toward the last node in the chain for 100 seconds. Every point in the graphs has been averaged over 50 runs. We have experimented with two traffic patterns: (i) CBR - the source sends packets using a constant bit rate; (ii) CBR-ACK - the source sends packets with a constant bit rate, and the destination upon receipt of the data packet sends a 100B ACK back to the source. This traffic pattern may be used by UDP-based applications that require reliable packet delivery [3].

The results for CBR-based experiments are presented in Figure 2. The higher lines represent chains of shorter length. For chains of 2,3 and 4 nodes the throughput is

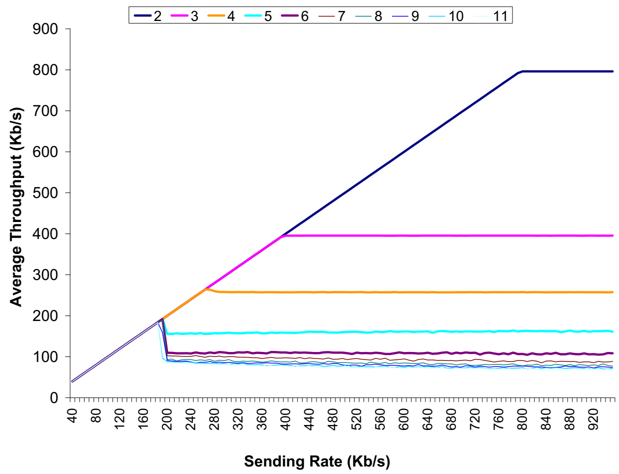

Figure 2. CBR throughput as a function of sending rate. Lines represent chains of different lengths (higher lines represent chains of shorter length).

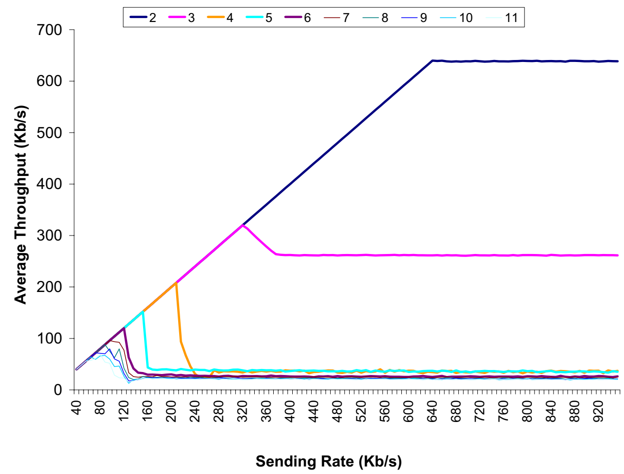

Figure 3. CBR+ACK throughput as a function of sending rate. Lines represent chains of different lengths (higher lines represent chains of shorter length).

gradually increasing up to a certain point and then stays flat. This is an expected result since with up to 4 nodes, no hidden terminals are present in the network. Nodes overhear one another's transmissions and almost no collisions occur. The extra packets produced by higher sending rates overflow the sender's buffer and are being discarded. However, for the chains of more than 4 nodes, DCF fails to optimally schedule packets. As a result, a decrease of up to 55% of the optimal throughput may be experienced!

The results for CBR-ACK are plotted in Figure 3. Interestingly, the throughput slides from the optimal level even for the chain of 3 nodes. Although in this case no hidden terminals are present, after a certain point, the sending rate becomes large enough to incur contention between the data packets and the acknowledgments. Consequentially, some data and ack packets overflow data link buffers and are being dropped. For chains of 4 nodes, in contrast to the CBR traffic, the CBR-ACK traffic suffers from the notorious hidden terminal problem. This happens because both the source node and the destination node may try to transmit packets simultaneously and collide with each other. Because of the hidden terminal problem, the throughput slope for chains of 4 and more nodes is sharper than for the chain of 3 nodes.

3.2 Cross-Layer Flow Control

We next describe a novel cross-layer flow control technique that leverages the observation that the optimal network throughput in multi-hop MANETs is dependent on the route length from the source to the destination and the application's sending rate.

The main idea behind our cross-layer flow control is simple. The application that is interested to achieve the optimal throughput should adaptively modify its sending rate according with the current route length to the destination node. Fortunately, route length information is usually maintained by the underlying routing layer, and therefore can be shared with interested applications.

Applications that are interested in adapting their sending rate register a callback function at the underlying routing layer. The callback function is invoked whenever the route length to the destination node changes, which may occur, for example, as a result of a new route discovery or a current route break.

On accepting the notification, the application sets the new sending rate based on the precomputed sending rate values obtained in the previous section. We are currently working on an analytical algorithm that will allow applications to compute the optimal sending rate, thus reducing the need to store precomputed values. Note that applications are not aware of the specific route that the underlying routing layer uses to route packets, but only about an availability of such a route and its length . If no route to the destination exists, application may either stop transmitting at all, or periodically transmit packets to encourage farther route discoveries.

We have implemented and evaluated the cross-layer flow control technique in ns-2. Next section presents our evaluation results.

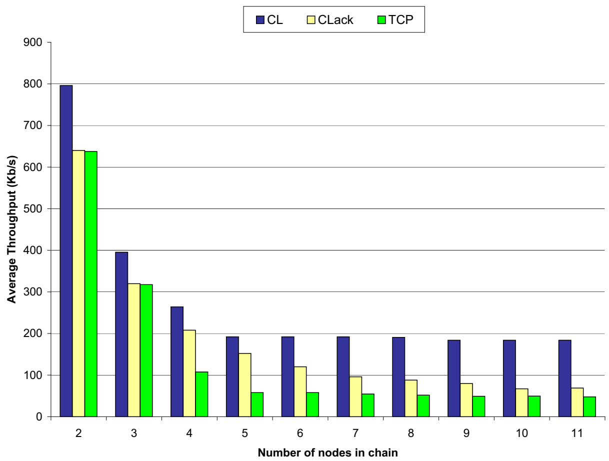

Figure 4. Comparison of achievable throughput of different flow control protocols for chains of different lengths.

4 Evaluation

In this section, we use extensive simulations to compare the performance of our cross-layer flow control technique described in Section 3 to that of TCP-Reno and other UDP-based protocols with a fixed sending rate.

We further refer to the version of our cross-layer approach that adaptively sends CBR traffic without generating acknowledgments as CL, and to the version that generates an acknowledgment for every packet received by a destination as CL_{ack}. We consider CL_{ack} protocol to quantify the advantages of the optimal pacing in the presence of acknowledgments flowing back from the destination to the source, and to show that it is possible to implement a reliable flow control protocol based on our technique that does a better job than TCP.

4.1 Static Chain Topologies

We have evaluated CL, CL_{ack} and TCP protocols on the chain topologies presented in section 3. The results are presented in Figure 4. It is not surprising that CL outperforms both CL_{ack} and TCP because it sends packets only in one direction at the optimal possible rate. Interestingly, CL_{ack} greatly outperforms TCP for chains of 4 and more nodes. This is because TCP is unaware of the maximal possible sending rate and thus periodically tries to inject more packets than the network can sustain, thus causing congestion, packet losses, retransmissions, and eventually decrease in the overall throughput. The other thing to note is that the throughput results for CL stay almost constant as the number of nodes in the chain increases, whereas for TCP and CL_{ack} the throughput decreases with the longer chains. This is because as the chain length increases, more hidden terminals are present in the network, increasing probability for packet losses.

Consider the following example. Similarly to Fu *et al.* [6], we stream 1000B packets from the first node to the last node in a chain of 6 nodes for the period of 105 seconds. In TCP, 905 packets were sent, but only 835 received. All drops were due to failure of the 802.11 MAC layer to successfully transmit RTS packets. In UDP, streaming packets at a rate of 24 packets per second, results in 2520 sent and delivered packets. However, sending at 25 packets per second results in 2565 packets sent, but only 1405 packets received. Only 122 of the drops were the 802.11 MAC layer drops. All the others were as a result of a link layer buffer overflow or a timeout at the routing layer [1]. This example motivates the importance of UDP stream pacing in MANETs.

4.2 Mobile Networks

We evaluate the performance of CL, CL_{ack} and TCP protocols in terms of throughput and delivery in mobile networks. The results are for networks of 100 nodes randomly placed on both rectangular 500m x 3000m and square 1200m x 1200m flat spaces. In both scenarios, every node has about 13 neighbors on average, enough to preserve connectivity even in mobile networks. Similarly to related research [2, 4], the rectangular shape was chosen to force the use of longer routes between communication pairs. Nodes move following the random waypoint model [2] with no pause time. In this model, a node chooses a random point within the space and starts moving toward that point at a speed randomly chosen from an interval 0-V_{max} (in our experiments V_{max} is equal to 10m/s). Upon reaching its destination, the node selects another destination and speed, repeating the process.

For both CBR and CBR-ACK traffic patterns, we have experimented with 5 different fixed sending rates: 240, 480, 720, 960 and 1200 Kb/s. We refer to the CBR protocols that send packet at a fixed rate of X Kb/s as CBR_X, and to the CBR-ACK protocols as ACK_X. All protocols send data packets of 1000B and acknowledgments of 100B. All experiments, use DSR as an underlying routing protocol.

We have experimented with 1 and 2 simultaneous flows. Each flow is defined by a pair of randomly chosen source and destination nodes. In each experiment, the source node streams packets toward the destination node for 100 seconds. Each point in the graphs is averaged over 20 random topologies. In all graphs, the bars represent the achieved

[1] At the routing layer, when a packet does not have a route to a destination it waits in a send buffer until either the route is found or timeout occurred. On a timeout the packet is dropped

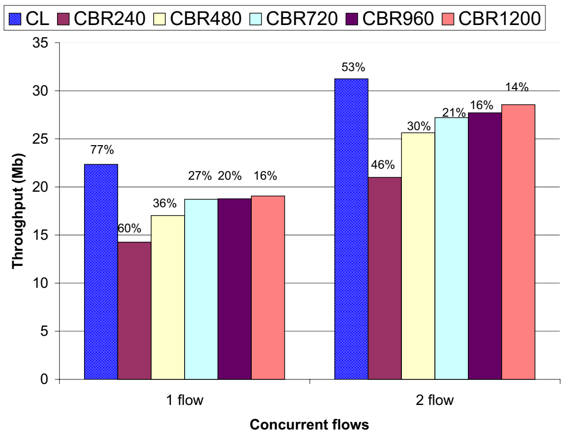

Figure 5. 500m x 3000m Throughput and delivery ratios of CBR flows.

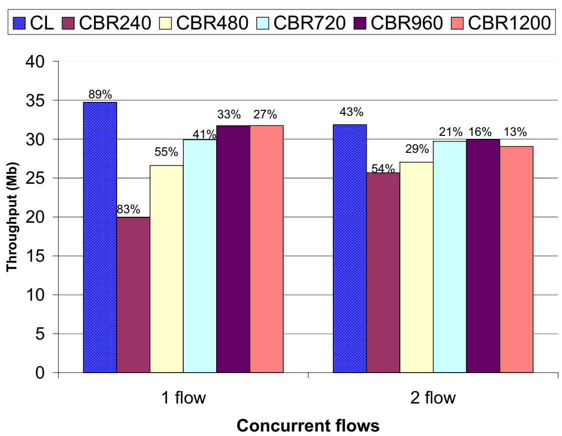

Figure 6. 1200m x 1200m Throughput and delivery ratios of CBR flows.

throughput in terms of Megabits received. The numbers on top of the bars represent delivery ratios.

To understand the results better, we encourage the reader to reference the graphs presented in Figures 2, 3 and 4.

Figures 5 and 6 plot the throughput and delivery ratio for one and two CBR flows performed in rectangular and square scenarios respectively.

CL consistently outperforms the other protocols in terms of both throughput and delivery ratio. This is not surprising since CL sends packets at the optimal possible rate. The results for the rectangular area are more impressive than for the square area. This is because the average route length for the square scenario is 4 and consequentially almost no hidden terminals are present in the network. Packets that cannot be delivered simply overflow buffers and are being discarded. In the rectangular experiment, the average route length is 8.5 and hidden terminals are common. Since sending at the optimal rate avoids the hidden terminal problem, CL achieves better results than protocols with the fixed sending rates.

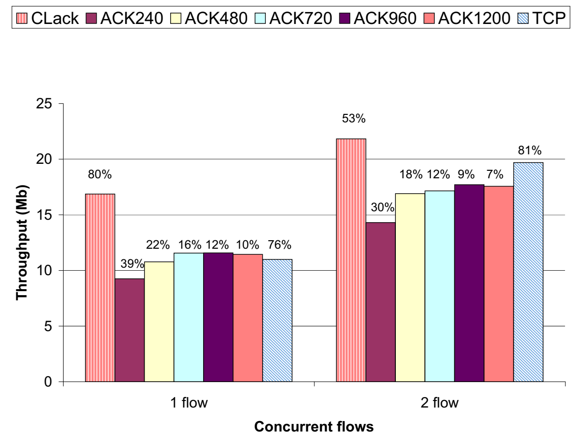

Figure 7. 500m x 3000m Throughput and delivery ratios of CBR-ACK flows.

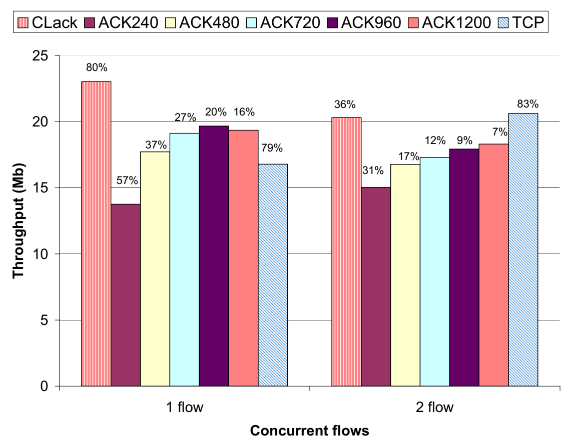

Figure 8. 1200m x 1200m Throughput and delivery ratios of CBR-ACK flows.

For the experiments with two flows, the benefits of limiting the sending rate are less impressive. When several competitive flows are present in the network, cross-traffic causes congestions, reducing the benefits of optimal pacing.

Figures 7 and 8 plot the throughput and delivery ratio for one and two CBR-ACK flows performed in rectangular and square scenarios respectively.

CL$_{ack}$ outperforms non-TCP protocols in terms of throughput and delivery ratio in all scenarios. Again, this is not a surprising result since CL$_{ack}$ sends packets at the op-

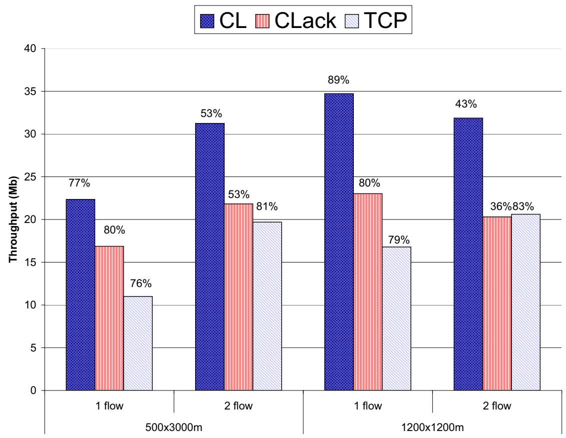

Figure 9. Comparison of CBR, CBR-ACK and TCP.

timal sending rate. For the experiments with only one flow, CL_{ack} outperforms TCP by 53% at the rectangular case and by 37% at the square case. Initially, we were puzzled to discover such a significant difference between the experiments. Closer examination of the individual runs revealed that for the square experiments larger portion of the average is contributed by scenarios with 2 to 3 hops from the source to the destination. As was shown in Figure 4, for those route lengths the performance of TCP and CBR-ACK is similar.

As more flows are introduced, TCP comes closer and even beats CL_{ack} in terms of both throughput and delivery ratio, which is expected since CL_{ack} does not take into account cross-traffic and congestion in the network.

Interestingly, the results for CL_{ack} in comparison to the protocols with the fixed sending rate are more impressive than for CL in comparison to the other protocols. This is because when both data packets and acknowledgments travel through the network, the need to pace the packet transmissions becomes more urgent. This was also shown in Figures 2 and 3, where excessive sending rate results in stabilization of the throughput at 70 Kb/s rate for the CBR traffic pattern and 20 Kb/s rate for the CBR-ACK traffic pattern.

Figure 9 summarizes the performances of CL, CL_{ack} and TCP in mobile environments. Not surprisingly, CL outperforms both CL_{ack} and TCP in all scenarios, and for a single flow experiments in rectangular environments, it doubles the throughput of TCP.

4.3 Summary and Discussion

In summary, if reliable delivery is not an issue and the load on the network is low, CL is the preferred solution. If reliable delivery is desirable, but only one flow is present in the network, CL_{ack} should be used. Otherwise, if reliable delivery is an issue or multiple flows are present in the network, TCP is the preferable solution.

5 Related Work

Considerable previous research has gone into flow control algorithms for both wireline and wireless networks [1, 12]. The most common and widely used flow control protocol today is definitely TCP. Albeit its wide use in the Internet today, TCP was reported to behave poorly in wireless multi-hop networks [7, 8]. TCP treats route breaks as a sign of congestion, timeouts and consequentially resets its congestion window, reducing the throughput drastically. Holland *et al.* [8] suggest to use an explicit link failure notification to disable congestion control mechanism in TCP until a new route is found. Fu *et al.* [6] show that TCP tends to use an average congestion window that is bigger than the best possible window, and propose two link layer techniques to improve TCP throughput. Our approach differs from these efforts in that it studies the throughput characteristics of UDP-based flows instead of TCP-based flows.

Chen *et al.* [3] have proposed an end-to-end rate-based flow control scheme. In this scheme, intermediate routers along the path to the destination compute the bandwidth available for the flow, and store this information at the data packet's headers. On receiving data packets, the destination periodically sends a special packet to the source node, revealing the maximal available bandwidth. In contrast, our scheme does not require support of intermediate routers and does not consume bandwidth by sending periodic messages to the source node if not necessary.

6 Conclusions and Future Work

We have shown that the throughput in wireless networks is highly dependent on both the sending rate and the path length to the destination. We have then proposed a novel cross-layer flow control technique that uses the precomputed values to achieve an optimal throughput in mobile ad hoc networks.

In our technique, interested applications can subscribe to receive notification from the routing layer on every routing change. The applications can then adapt their sending rate according to the precomputed rate for a given path length. We showed that our technique consistently outperforms UDP-based protocols with fixed transmission rates. When only one flow is present in the network, our technique beats TCP in terms of throughput by a factor of 2.

In the future, we plan to make our technique aware of the congestion in the network, and be able to adapt its sending rate accordingly. We also plan to extend our evaluation to multiple flows.

References

[1] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. H. Katz. A comparison of mechanisms for improving tcp performance over wireless links. *IEEE/ACM Transactions on Networking*, Dec. 1997.

[2] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. of MobiCom*, 1998.

[3] K. Chen, K. Nahrstedt, and N. Vaidya. The utility of explicit rate-based flow control in mobile ad hoc networks. In *Proc. of WCNC*, 2004.

[4] S. R. Das, C. E. Perkins, and E. E. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *Proc. of INFOCOM*, pages 3–12, 2000.

[5] K. Fall and K. Varadhan. ns notes and documentation. The VINT Project, UCBerkeley, LBNL, USC/ISI, and Xerox PARC, Nov. 1997. Available at http://citeseer.nj.nec.com/fall00ns.html.

[6] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla. The impact of multihop wireless channel on TCP throughput and loss. In *Proc. of INFOCOM*, 2003.

[7] M. Gerla, K. Tang, and R. Bagrodia. TCP performance in wireless multi-hop networks. In *Proc. of WMCSA*, 1999.

[8] G. Holland and N. Vaidya. Analysis of TCP performance over mobile ad hoc networks. In *Proc. of MobiCom*, 1999.

[9] IEEE. "IEEE std 802.11 - wireless LAN medium access control (MAC) and physical layer (PHY) specifications", 1997.

[10] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *Proc. of MobiCom*, pages 61–69, 2001.

[11] The CMU Monarch project. wireless and mobility extensions to ns-2, Oct. 1999.

[12] L. Xu, K. Harfoush, and I. Rhee. Binary increase congestion control for fast long distance networks. In *Proc. of INFOCOM*, 2004.