Exploiting congestion information in network and higher layer protocols in multihop wireless ad hoc networks — Source link ↗

Yih-Chun Hu, David B. Johnson

Institutions: University of California, Berkeley, Rice University

Related papers:

- A performance comparison of multi-hop wireless ad hoc network routing protocols

- Dynamic Source Routing in Ad Hoc Wireless Networks

- Ad-hoc on-demand distance vector routing

- A high-throughput path metric for multi-hop wireless routing

- Ad hoc On-Demand Distance Vector (AODV) Routing

Exploiting Congestion Information in Network and Higher Layer Protocols in Multihop Wireless Ad Hoc Networks

Yih-Chun Hu
University of California, Berkeley
yihchun@cs.cmu.edu

David B. Johnson
Rice University
dbj@cs.rice.edu

Abstract

With most routing protocols for ad hoc networks, shorter paths are generally considered more desirable, making some areas of network more prone to congestion and decreasing overall network throughput. In this paper, we examine the use of congestion information to avoid these network hotspots. By locally monitoring the network interface transmission queue length and MAC layer behavior at each node, a node can establish an approximation of the degree to which the wireless medium around it is busy; this measurement reflects not only the behavior of the node itself, but also the behavior of other nearby nodes sharing the wireless medium. We suggest a number of uses of such congestion information in an ad hoc network, in the network, transport, and higher layers, and we evaluate a set of such uses through simulation. Our results based on modifications to the Dynamic Source Routing protocol (DSR) and TCP demonstrate substantial performance improvement in terms of scalability, packet delivery, overhead, and fairness resulting from this use of congestion information.

1. Introduction

In a multihop wireless ad hoc network, mobile nodes cooperate to form a network without the use of any infrastructure such as access points or base stations. The mobile nodes, instead, forward packets for each other, allowing nodes beyond direct wireless transmission range of each other to communicate. The mobility of the nodes and the fundamentally limited capacity of the wireless medium, together with wireless transmission effects such as attenuation, multipath propagation, and interference, combine to create significant challenges for network protocols operating in an ad hoc network.

In most ad hoc networks, some areas of the network have higher packet forwarding loads than other areas. For example, in a network where the nodes are uniformly distributed

in space, the nodes near the center of the network will tend to carry a higher load when the network routing protocol prefers shortest-path routes; this preference can make certain areas prone to congestion and can decrease the overall network throughput. Although some Quality-of-Service routing protocols have been proposed that can often route around areas of congestion, these protocols are more complex than traditional routing protocols. In addition, many of these protocols only find routes for flows requiring specific QoS parameters. In this paper, we design some lightweight mechanisms for detecting network congestion and for exploiting this information to improve protocol performance and behavior for all types of traffic at the network, transport, and higher protocol layers.

The parameters for measuring local network congestion around a node depend largely on the MAC layer, and many different wireless MAC layers, based on methods such as random access, TDMA, and polling, have been proposed and implemented. In this paper, we focus on the IEEE 802.11 Distributed Coordination Function (DCF) MAC protocol [12], since it has been adopted as a wireless LAN standard and is widely used in both traditional wireless systems and in multihop ad hoc networking research. Our techniques using MAC layer utilization information could also be applied easily with similar random access collision avoidance wireless MAC protocols such as MACA [20] and MACAW [2], and could be adopted with other types of MAC protocols as well.

The specific network and transport layer protocols we use in this paper are the Dynamic Source Routing protocol (DSR) [16–18] and the TCP transport protocol [24]. We make use of wireless medium congestion information to improve routing decisions in two areas: first, we modify DSR's Route Discovery to prevent the discovery of routes over which it is undesirable to carry additional traffic since the wireless medium over those hops is already quite busy, and second, we use this congestion information to control the use of certain routing protocol optimizations in DSR such as packet salvaging. Finally, we also use our congestion information at each node to influence the setting of the Explicit Congestion Notification (ECN) bits [25] in the IP header of packets carried through portions of the network where the wireless medium is particularly busy; this use

of ECN allows higher layer protocols such as TCP to also make use of this congestion information.

In Section 2 of this paper, we define our measurement of congestion at a node, and we suggest a number of uses of this information in the network, transport, and higher layer protocols within the ad hoc network. Section 3 provides an overview of the operation of the DSR protocol, which we use in our evaluation of exploiting congestion information in routing and higher layers. In Section 4, we define the example modifications to DSR and TCP that we simulated in order to quantify the effectiveness of these techniques. Section 5 describes the methodology of our simulation evaluation, and Section 6 presents our simulation results. Finally, in Section 7, we present conclusions.

2. Congestion Information

In this section, we describe in detail one method for obtaining wireless *congestion information* each node. We also discuss a range of general possible uses of congestion estimates in routing and higher layer protocols within a node; Section 4 later describes three specific uses of such measurements within the context of the Dynamic Source Routing protocol (DSR) and TCP, and Section 5, evaluates these three techniques in detail through simulation.

2.1. Measuring Congestion

We use two metrics to measure the level of congestion at a node. The first metric is the average *MAC layer utilization* level at a node, which indicates the degree to which the wireless medium around that node is busy or idle. We define the average MAC layer utilization as measured by a node to be the fraction of time during which that node either (1) has one or more packets to transmit in its transmission queue for that network interface, or (2) if that node had attempted to transmit, it would not have been able to do so then, according to the rules of the MAC layer at that node. Since the instantaneous MAC layer utilization at a node is either 0 or 1, we average this value over a period (10 seconds, in our simulations) to obtain an indication of the use of the wireless medium around that node.

The intuition behind this definition of MAC layer utilization is that the instantaneous value of this metric should be 0 only when the wireless medium around the node is available for the node to begin transmission of a *new* packet not already in that node's network interface transmission queue. Measuring this value requires the node to monitor the state of its own MAC layer. Although many current wireless network interface products such as commonly available IEEE 802.11 wireless LAN cards do not provide a MAC layer interface to support this monitoring by the operating system software in the node, it is supported by some interfaces such as the DARPA GloMo Radio API [1], and additional future wireless products may provide such an interface if it is proven useful.

As an example of measuring MAC layer utilization, we simulate it in this paper based on a detailed model of the IEEE 802.11 DCF MAC protocol [12]. We consider instantaneous MAC layer utilization level at a node to be 1 at any time that the MAC layer at that node either detects physical carrier to be present or is deferring due to virtual carrier sensing, interframe spacing, or backoff. Instantaneous MAC layer utilization at the node is also 1 at any time that the node has at least one packet in the transmission queue for its wireless network interface.

The other metric we use is the instantaneous transmission queue length. In certain cases, a node may not be experiencing much MAC layer congestion, but instead may have many packets backlogged. If that node is then chosen to forward other packets, it will increase packet latency, and may even drop packets due to a limit on the queue length. In our simulation, we use a combination of the MAC layer utilization and instantaneous queue length metrics to determine the congestion level at each node.

2.2. Uses within the Network Layer

Within the network layer, one use of congestion measurements is to allow the routing protocol to attempt to affect the routes chosen, such as to avoid choosing routes through congested portions of the network. Routing protocols for ad hoc networks can be grouped into two types: *proactive* (or periodic) protocols, and *reactive* (or on-demand) protocols. In a proactive routing protocol, nodes exchange routing information with each other (e.g., periodically) such that each node attempts to always know a current route to all possible destinations (e.g., [19, 22]). In contract, nodes using a reactive routing protocol do not exchange routing information until necessary, and instead attempt to discover all routes on-demand by an active search within the network (e.g., [17, 23]). *Hybrid* routing protocols that combine these two approaches are also possible (e.g., [10]).

In a proactive routing protocol, nodes can affect the routes chosen by the protocol by using the congestion information to alter the *metric* for certain routing table entries that it exchanges with other nodes. For example, in a *distance vector* routing protocol, the node could include an expression of its congestion level in each of its own routing advertisements; neighbor nodes receiving such advertisements could use this value to treat the link from this node as having a metric that is a function of the advertised congestion level, rather than as is common, treating each such link as having a metric of 1. For a *link state* routing protocol, a node could use its local measurement of congestion level similarly to increase the metric that it includes for its neighboring links in its own routing updates to other nodes.

In a reactive routing protocol, nodes can affect the routes chosen by the protocol through changes in the operation of the dynamic route discovery process. We describe this approach in the context of DSR in Section 4.1. In a hybrid

routing protocol, a node may naturally use a combination of mechanisms using congestion level measurements, based on either the proactive or reactive portions of the hybrid protocol, to affect the routes chosen.

Another use of congestion level measurements within the routing protocol at a node is to modify in general the behavior of the routing protocol itself, based on the level to which the wireless medium around the node is busy. For example, depending on the congestion measured by a node, optional features or optimizations within the routing protocol can be enabled or disabled, if their effectiveness might depend on whether or not the wireless medium around the node is particularly busy. We describe a specific example of this type of optimization in Section 4.2 in the context of DSR.

Another example of such protocol modification would be an adaptive distance vector routing protocol, in which a node modifies the periodic transmission of its own routing advertisement packets. If the wireless medium around the node is particularly busy, the node could reduce the frequency of its own advertisements, and could reduce the number of routing table entries included in each advertisement to include only the most important or most recently changed entries. The ADV ad hoc network routing protocol [3] performs similar adaptive optimizations, but the adaptation in ADV is based on "trigger meter" values that use network layer information, not information adapted from both the MAC and routing layers as we propose here.

As a final example in this section, the AODV routing protocol [23] could be modified such that a node does not attempt local route repair when the wireless medium around the node is particularly congested. If a node attempts and is successful at local repair, then the route to that destination will continue to pass through that node. If instead, in such cases, the node simply treated the link as broken as normal, the new route discovered for that destination could be made to route around that area of the network, when combined with modifications to Route Discovery similar to those we define in Section 4.1.

2.3. Uses within the Transport Layer

Within the transport layer, a number of different uses of congestion level measurements at a node are possible. Section 4.3 describes one approach in detail for TCP, based on setting the Explicit Congestion Notification (ECN) bits in a packet's IP header [25]; this same approach would also be applicable for any other transport layer that supported use of the ECN bits [7]. Below we suggest some other possible uses within the transport layer for congestion measurements.

Beyond setting ECN bits to improve TCP performance, it may be possible to use information about the congestion levels at nodes along a multihop ad hoc network route to allow TCP to gain additional information about network conditions. Such information about the degree to which the wireless medium around nodes is busy might enable TCP to react better by helping to differentiate conditions of wireless packet loss, congestion packet loss, or simple wireless medium contention-based packet delay.

As a final example of use at the transport layer, the IETF Reliable Multicast Transport (RMT) Working Group, in attempting to standardize reliable multicast for the Internet, is considering solutions based on negative acknowledgements and on positive acknowledgements [14]. Each of these approaches represents a tradeoff of factors such as overhead and latency; with additional information such as congestion level measurements, it might be possible to adaptively balance between these two approaches.

2.4. Uses within Other Higher Layer Protocols

Above the transport layer, information on the congestion at a node or along a path, as suggested in Section 2.3, can be used to adapt some traditional functions of of the presentation layer, such as data compression. If the congestion level indicates that the wireless medium is particularly busy, a sending node could decide to compress the data before transmission. Such use of compression represents a tradeoff between the bandwidth used for transmission versus the CPU time consumed for compression and decompression and the latency in time taken for these functions. Based on the measured congestion level, a sending node could more productively make such tradeoff decisions.

If an application programming interface (API) is available to pass the congestion information to user-level programs, these measurements could also, for example, be used to aid middleware application adaptation systems such as Odyssey [21] and Puppeteer [6].

3. Overview of the DSR Protocol

This section provides an overview of the Dynamic Source Routing protocol (DSR) [16–18], which we use in our evaluation of exploiting congestion information. DSR is one of a number of routing protocols proposed within the Mobile Ad Hoc Networks (MANET) Working Group of the Internet Engineering Task Force (IETF) [13]. We use DSR in our study, since the protocol has been shown to perform well in earlier simulation studies [5, 15]; DSR is an *on-demand* (or *reactive*) ad hoc network routing protocol. As suggested in Section 2.2, similar techniques using congestion information could be applied to other ad hoc network routing protocols.

The operation of DSR is based on *source routing*, in that the sender of a data packet determines the complete sequence of hops to be used as the route for that packet to its destination. In the basic version of DSR, the source route for a packet is represented in the header of the packet, although a enhancement to DSR uses *implicit source routing* to avoid this overhead in the header of each packet [11]. Instead, after the first packet containing a full source route

has been sent along the route to the destination, subsequent packets need only contain a flow identifier to represent the route, and nodes along the route maintain *flow state* to remember the next hop to be used along this route based on the address of the sender and the flow identifier; one flow identifier can designate the *default flow* for this source and destination, in which case even the flow identifier is not represented in a packet.

DSR divides the routing problem in two parts: *Route Discovery* and *Route Maintenance*, both of which operate *entirely* on-demand. In Route Discovery, a node actively searches through the network to find a route to an intended destination node. While using a route to send packets to the destination, Route Maintenance is the process by which the sending node determines if the route has broken, for example because two nodes along the route have moved out of wireless transmission range of each other.

A node that has a packet to send to a destination searches its *Route Cache* for a route to that destination. If no cached route is found, the sending node initiates Route Discovery by broadcasting a ROUTE REQUEST packet containing the destination node address (known as the *target* of the Route Discovery), a list (initially empty) of nodes traversed by this REQUEST, and a *request identifier* from this source node. The request identifier, the address of this source node (known as the *initiator* of the Route Discovery), and the target address together uniquely identify this Route Discovery.

A node receiving a ROUTE REQUEST checks to see if it has previously forwarded a REQUEST from this Discovery by examining the IP Source Address, target address, and request identifier, If it has recently seen this identifier, or if its own address is already present in the list in REQUEST of nodes traversed by this REQUEST, the node silently drops the packet. Otherwise, it appends its address to the node list and forwards the REQUEST. When a REQUEST reaches the target node or a node with a route to the target in its Route Cache, this node returns a ROUTE REPLY to the initiator of the ROUTE REQUEST. The REPLY contains a copy of the node list from the REQUEST, and can be delivered to the initiator node by reversing the node list, by using a route back to the initiator from its own Route Cache, or "piggybacking" the REPLY on a new ROUTE REQUEST targeting the original initiator. When the initiator of the request receives the ROUTE REPLY, it adds the newly acquired route to its Route Cache for future use.

In Route Maintenance, a node forwarding a packet for a source attempts to verify that the packet successfully reached the next hop in the route. A node can make this confirmation using a link-layer acknowledgement (such as is provided in IEEE 802.11 [12]), a passive acknowledgement [19], or by means of a network-layer acknowledgement. A packet is possibly retransmitted if it is sent over an unreliable MAC, although it should not be retransmitted if retransmission has already been attempted at the MAC layer. If a packet is not acknowledged, the forwarding node assumes that the next-hop destination is unreachable over this link, and sends a ROUTE ERROR to the source of the packet, indicating the broken link. A node receiving a ROUTE ERROR removes that link from its Route Cache.

A number of optimizations to the basic DSR protocol have been proposed [18]. In this paper, we describe only those optimizations that are affected by the changes we make to the protocol. One example of such an optimization is *packet salvaging*. When a node forwarding a packet fails to receive acknowledgement from the next-hop destination, as described above, in addition to sending a ROUTE ERROR back to the source of the packet, the node may attempt to use an alternate route to the destination, if it knows of one. Specifically, the node searches its Route Cache for a route to the destination; if it finds one, then it *salvages* the packet by replacing the existing source route for the packet with the new route from its Route Cache. To prevent the possibility of infinite looping of a packet, each source route includes a *salvage count*, indicating how many times the packet has been salvaged in this way. Packets with salvage count larger than some predetermined value cannot be salvaged again.

4. Evaluation within DSR and TCP

This section describes the specific uses of congestion measurements that we examined to illustrate the effectiveness of reacting to congestion information within DSR and TCP in an ad hoc network. We modified the protocol behavior based on a combination of two metrics, the measured level of *MAC layer utilization* at a node and the *network interface transmission queue length* at that node; the interface queue length is the number of packets waiting buffered at that node for transmission over its wireless network interface. The first of these metrics provides the node with a view of the current condition of the shared wireless medium around the node; the second indicates a prediction of the future load that this node will place on the wireless medium. We invoked each protocol optimization when either of these levels exceeded the threshold chosen for that particular optimization.

4.1. Modifications to DSR Route Discovery

In Route Discovery in DSR, a node performs a controlled flood of the network with ROUTE REQUEST packets, searching for a route to the target destination node. When one of the ROUTE REQUEST packets from this Route Discovery reaches the destination node or reaches another node with a route to the destination cached, this node returns a ROUTE REPLY packet to the originator of the Route Discovery.

Allowing this flood of ROUTE REQUEST packets from a Route Discovery to traverse an area of the network in which the wireless medium is already particularly busy creates several risks. First, the additional broadcast packets

from the Route Discovery flood further increases the use of the wireless medium in those areas. Second, the route discovered by a Route Discovery is the sequence of hops through which the ROUTE REQUEST packet was forwarded that generated the ROUTE REPLY in response, and thus, any route discovered by forwarding a ROUTE REQUEST through an area of the network in which the wireless medium is already particularly busy can only result in a discovered route through this same area; such routes are less desirable than other routes. Finally, the additional traffic resulting from a new flow of data packets using a route through such an area can cause the wireless medium in this area to be used even more heavily, possibly leading to performance degradation for all users.

To alleviate these problems, we explored the effect of modifying DSR so that nodes do not process or forward a ROUTE REQUEST packet if the node determines that the wireless medium around itself is too busy; however, if the node is the target of the ROUTE REQUEST, it processes it and returns a ROUTE REPLY as usual.

This optimization is simple to implement, although in this form, it has two limitations. First, it may cause a node to be unable to discover a route to some destination, even when a route actually exists, if the only existing routes go through busy areas of the network. Second, by forcing the Route Discovery to route around busy areas, it may cause a node to discover a route that is longer than the minimum number of hops that could have been discovered; in saving overhead within busy areas of the network, this optimization may create additional overhead totaled across other areas of the network.

A modification to this optimization that could be made to address these limitations, is to add a flag to each ROUTE REQUEST, indicating whether or not to use this optimization. A node that has a packet to send to a destination would first check its Route Cache, and if it did not have a route, would initiate a Route Discovery with the flag off; that is, such that nodes in busy areas would not forward ROUTE REQUEST packets from that Route Discovery. If the source node does not receive a ROUTE REPLY from that Discovery, it would initiate another Discovery, this time turning the flag on, allowing all nodes to forward REQUESTS belonging to this Discovery. This modification is somewhat similar to an expanding ring search, although the search here expands into busy areas rather than simply into areas at a greater hop count from the source. In our simulation, we did not implement this modification to the Route Discovery optimization, since the performance of ordinary Discovery was sufficient.

4.2. Modifications to DSR Packet Salvaging

In DSR, packet salvaging is a mechanism used by an intermediate node to avoid dropping a packet when it detects that the next hop for the packet along its original route is broken. The intermediate node opportunistically checks its own Route Cache for a route to the packet's destination, contributing its own cache information to enhance the probability of successful delivery of the packet.

However, the route that this intermediate node may select from its own Route Cache for salvaging may not be a valid route to the destination, since the Cache is not actively maintained and some nodes may have moved since this route was cached. Packet salving usually is beneficial, though, because the nodes involved may not have moved extensively recently and because nodes update their Route Cache with routing information in forwarded and overheard packets, but in some cases, the extra overhead caused by forwarding the packet along the new route may not be worth the chance that the packet will be delivered correctly rather than just being dropped.

We explored the effect of modifying packet salvaging to not salvage a packet at an intermediate node (and to drop the packet instead when the next hop on the original route has broken) if the wireless medium around the node is particularly busy. This condition is an indication that attempting to salvage the packet may create more harm than good, since sending the packet along the new route will add more overhead to the wireless medium in the area.

This modification to packet salvaging also addresses two related potential problems with salvaging in this situation. First, at a node where the wireless medium has been particularly busy, packets which could otherwise have been overheard may have a higher chance of loss, due to factors such as collision and increased noise floor. As a result, such a node will have been able to overhear less routing information from other packets and may thus have lower-quality routes in its Route Cache for any routes for which it is not directly involved in forwarding, making salvaging in this case even less desirable. Second, when a node attempts to transmit a packet to a next-hop node that is no longer a neighbor, an RTS packet is repeated several times (when using a MAC protocol like IEEE 802.11); each of RTS packets causes this node's neighbors to sense virtual carrier for the intended duration of the intended data packet. If several RTS attempts are made before determining that the link to the next hop has broken, possibly further increasing congestion around those nodes.

4.3. Use within TCP

Ramakrishnan and Jain [26] proposed Explicit Congestion Notification (ECN) as a mechanism for signaling congestion, in packets traversing congested nodes or links. Floyd [8] presented a mechanism to use the ECN mechanism to improve the performance of TCP.

We made use of ECN as a mechanism for an intermediate node to signal to the TCP sender that the wireless medium around the node is particularly busy. Using ECN for TCP provides two benefits: first, it may prevent the loss

Table 1: Congestion Metric Thresholds for Triggering Protocol Optimizations

Optimization	MAC Layer Utilization	Interface Queue Length
Suppressing ROUTE REQUEST forwarding	15%	10
Suppressing salvaging	5%	20
Setting IP header ECN bits	1%	30

of packets along that flow due to queue overflow, and second, it may allow better fairness for other flows also traversing this node.

In typical use of ECN, routers use active queue management [4, 9] to set the Congestion Experienced (CE) codepoint [25] in a packet's IP header when the average queue length at that node exceeds some threshold. Instead, we set the CE codepoint in a packet based on our combined congestion metric. When a TCP sender receives a packet with the CE codepoint set in its IP header, the TCP sender responds using its congestion control algorithm as it would to a packet drop [25]. Since our MAC layer utilization measurement represents an average of the recent level to which the wireless medium around the node is busy, the setting of the CE codepoint in a packet by an intermediate node indicates a sustained congestion condition needing action from the TCP sender.

5.   Evaluation Methodology

We based our evaluation of the use of congestion level measurements in DSR and TCP on the version of DSR that uses implicit source routing and flow state, as described in Section 3, since this version shows the best reported performance for the DSR protocol [11]. Using the *ns-2* network simulator, we simulated this version of the DSR protocol, both with and without the specific modifications for use of congestion information within DSR and TCP described in Section 4. All behavior of TCP in our simulations was created by *ns* without modification, based on our setting of the ECN bits in the IP header of packets as appropriate. The version of the *ns* simulator that we used provides a physical and MAC layer model including proper modeling of backoff, contention, collisions, capture, and propagation; it models the IEEE 802.11 Distributed Coordination Function (DCF) MAC [12] over a 2 Mbps wireless network with a a nominal maximum transmission range of 250 m.

Due to the varying affect that contention in the wireless medium and congestion has on each of our several optimizations, we chose different levels at which to enable each. Table 1 shows the measured MAC layer utilization and interface queue lengths at which we enabled each optimization. We chose these values by intuition and have not yet undertaken any attempt to tune them for performance.

We evaluated the performance of these modifications over a wide range of scenarios, with nodes moving according to the *Random Waypoint* mobility model. Each node independently chooses a random starting point and waits there for a duration called the *pause time*. It then randomly chooses a destination, and moves there at a velocity chosen uniformly between 0 and a maximum velocity of $v_{\max}$. When the node arrives at this destination, again waits for the pause time, and then begins moving at a new randomly chosen velocity to a new randomly chosen destination; each node independently repeats this movement pattern throughout the simulation.

All of our simulations use a pause time in the Random Waypoint model of 0 seconds and a maximum node movement velocity ($v_{\max}$) of 20 m/s. This value of pause time represents a network in which all nodes are in continuous motion, with each node turning and moving toward a new destination as soon as it reaches its current destination.

The data traffic in our simulations was based both on Constant Bit Rate (CBR) sources and TCP sources. We performed three sets of experiments.

The first two sets of experiments used CBR traffic and evaluated the effect of our protocol modifications using congestion measurements in DSR. One of these sets of experiments was performed using 50 mobile nodes in a simulation area of 1500 m × 300 m modeling 900 seconds of simulated time for each run, and the other set was performed using 100 mobile nodes in an area of 1000 m × 1000 m modeling 1000 seconds of simulated time for each run; in both of these sets of experiments, we simulated a number of CBR traffic sources, varying from 2 to 30 CBR sources per run, with each source sending 4 512-byte packets per second.

Our final set of simulation experiments evaluated the effect of our protocol modifications in DSR and TCP on a set of TCP flows; these experiments were performed using 100 mobile nodes in a simulation area of 1000 m × 1000 m modeling 1000 seconds of simulated time for each run. In these experiments, we simulated 20 TCP streams per run, with each TCP source sending data continuously during the execution of the simulation.

In the first two sets of experiments, we measured the following four metrics in our simulations:

- *Packet Delivery Ratio* is the fraction of data packets originated by the application layer that are successfully received at their destination

- *Path Optimality* shows the degree to which the protocol is able to discover and use the shortest available
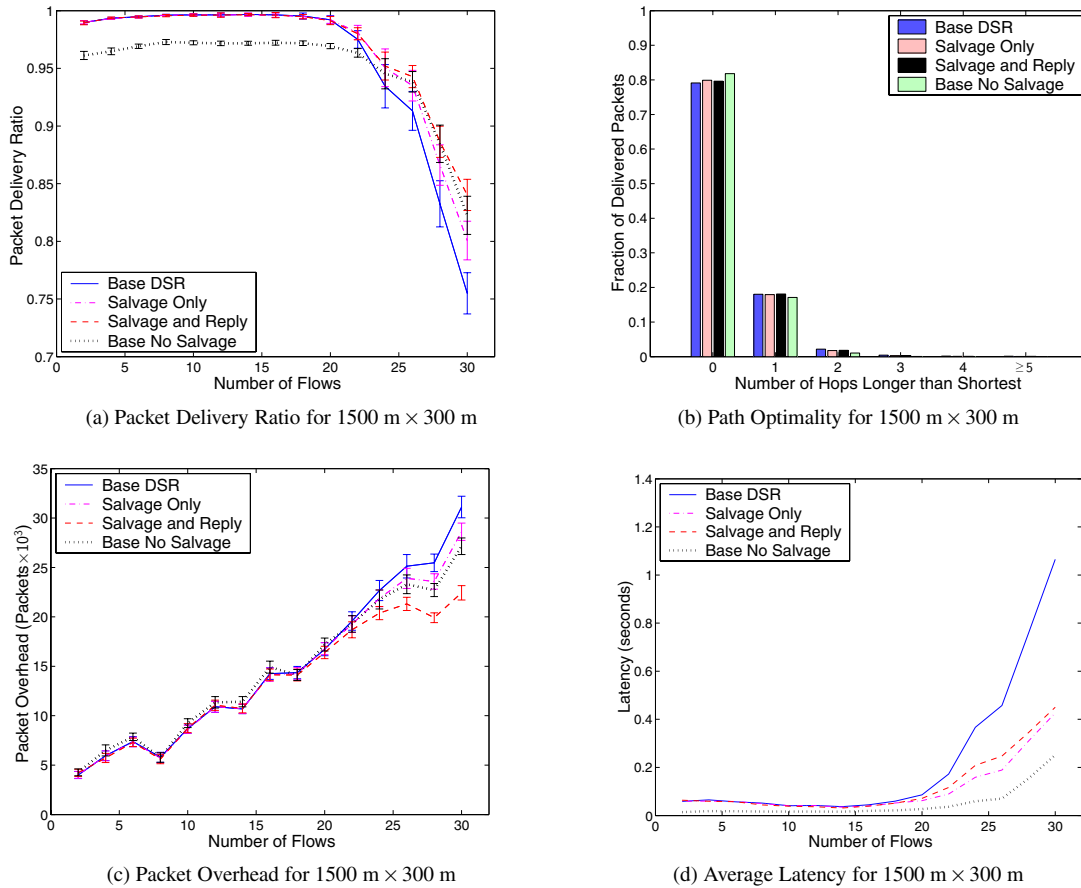
(a) Packet Delivery Ratio for 1500 m × 300 m

(b) Path Optimality for 1500 m × 300 m

(c) Packet Overhead for 1500 m × 300 m

(d) Average Latency for 1500 m × 300 m

Figure 1: Simulation results for CBR traffic in 1500 m × 300 m scenarios with 50 mobile nodes, with each source node sending 4 512-byte CBR packets per second; results are averaged over 40 simulation runs, with the error bars representing the 95% confidence interval of the mean.

routes. The simulator is able to determine at all times the length of the shortest theoretically available route, assuming a transmission range of 250 m per hop; we measured the fraction of delivered data packets that were routed by the protocol over routes of various lengths relative to this shortest optimal route.

- *Packet Overhead* is the number of individual transmissions of routing packets. For example, if a ROUTE ERROR traverses 4 hops, it contributes 4 to the Packet Overhead.

- *Latency* is the elapsed time from the origination of a packet at the source application to its first receipt at the destination application.

In the third set of experiments, we measured the goodput and fairness of the set of TCP connections. We define the goodput of each TCP stream here as the number of bytes of the TCP data stream correctly delivered to the receiver, such that that byte and all previous bytes of the stream were delivered with no missing TCP segments.

6. Results

The results from the first two sets of simulation experiments described in Section 5 are shown in Figures 1 and 2. We defer the presentation of the results from our third set of experiments until Section 6.3, where we discuss those results.

Figure 1 shows the four metrics defined in Section 5 for simulation runs of 50 nodes in an area of 1500 m × 300 m, and Figure 2 shows the corresponding set of results for simulation runs of 100 nodes in an area of 1000 m × 1000 m. In these graphs, the error bars shown represent the 95% confidence interval of the mean. We discuss these results below in Sections 6.1 and 6.2.

6.1. Suppressing Salvaging

When salvaging was disabled in congested areas, as in Section 4.2, performance was identical with lower load, but packet delivery ratio, overhead, and latency all showed substantial improvements at higher load. For example, in the 1000 m × 1000 m scenarios, at the high load of 26 flows,
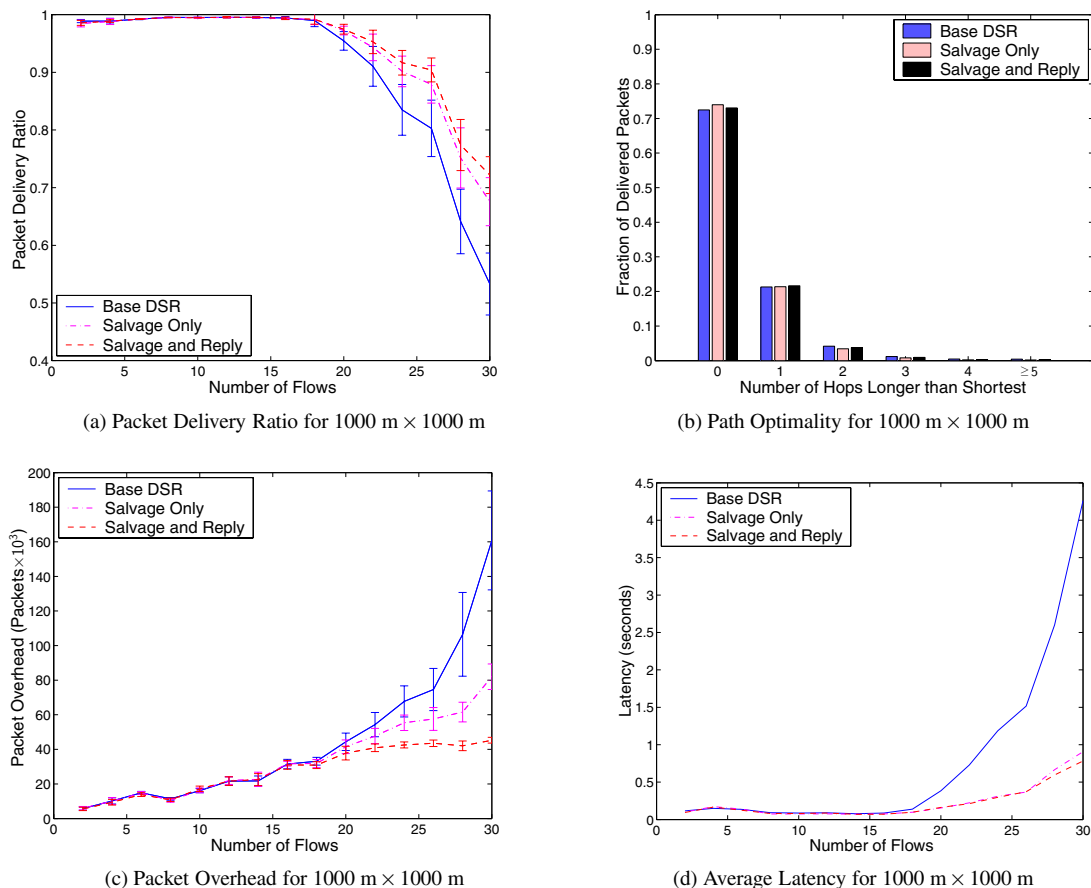
(a) Packet Delivery Ratio for 1000 m × 1000 m

(b) Path Optimality for 1000 m × 1000 m

(c) Packet Overhead for 1000 m × 1000 m

(d) Average Latency for 1000 m × 1000 m

Figure 2: Simulation results for CBR traffic in 1000 m × 1000 m scenarios with 100 mobile nodes, with each source node sending 4 512-byte CBR packets per second; results are averaged over 10 simulation runs, with the error bars representing the 95% confidence interval of the mean.

representing a data rate of 426 kbps, the unoptimized version of DSR delivers just 80% of its packets, while the optimized version of DSR delivers almost 88% of its packets. At the same load, packet overhead decreased by over 25%, and average latency dropped by more than a factor of 4.

To evaluate the effectiveness in using congestion information in making decisions about whether or not to salvage, we also compared our scheme to a version of DSR that never salvages. We ran these simulations only for the 1500 m × 300 m scenarios. When compared to a version of DSR that never salvages, the salvaging optimization based on congestion information shows significantly better performance at lower loads. For example, with 20 flows, representing a data rate of 327 kbps, the version of DSR using congestion information delivered 99.21% of offered packets, where on the same scenarios, the version of DSR that never salvaged delivered just 96.93% of packets. At the same load, packet overhead is also slightly lower without salvaging, due to the positive effects of spreading cache information through source routes. At higher loads, salvaging

actually decreases packet delivery ratio relative to the base version of DSR; choosing whether or not to salvage based on congestion level retains much of the benefit to packet delivery ratio of not salvaging when the utilization is low, without sacrificing the ability to salvage when congestion is not a problem.

A possible improvement to this scheme would be to not forward salvaged packets at congested nodes; a node could examine the "salvage count" field in the DSR header of each packet that it forwards, and make forwarding decisions based on salvage count and local measured congestion level. This technique would provide even more of the benefits of never salvaging, but only at nodes where not salvaging is beneficial. Preliminary results show that such an approach could split the difference between never salvaging and using congestion information at higher levels of congestion, while maintaining the higher performance of using congestion levels at lower congestion. This approach cannot fully achieve the benefits of not salvaging in congested areas because a congested node may have a neigh-

bor that is not congested; if that neighbor salvages a packet and sends it to the node, the node would not forward it, so the initial transmission was wasted. It may also be possible to "push" congestion information one hop farther, allowing neighbors to see congestion levels of neighboring nodes, either by piggybacking the information on existing data and routing packets, or by including it as part of an RTS/CTS handshake, but such pushed information may be stale.

The version of DSR that never salvages always has better latency and path optimality, since no packets are rerouted in-flight; however, at lower traffic loads, this is at the cost of some packets not being successfully delivered.

Using congestion levels to influence salvaging decisions provides, to a large extent, the advantages of both choices.

6.2. Suppressing Route Discovery

When ROUTE REQUEST propagation was determined based on congestion level, our simulations showed a slight but statistically significant increase in packet delivery ratio in the $1500\,\mathrm{m} \times 300\,\mathrm{m}$ runs, as well as a more substantial improvement in packet overhead for both sets of scenarios. For example, in the $1500\,\mathrm{m} \times 300\,\mathrm{m}$ scenarios, with an offered load of 26 flows, representing a data rate of 426 kbps, the packet delivery ratio with both salvaging and Route Discovery optimizations enabled was 94.29%, and was only 93.50% with just the salvaging optimizations; enabling Route Discovery optimizations also reduced overhead by 12%. At the same load in the $1000\,\mathrm{m} \times 1000\,\mathrm{m}$ scenarios, enabling Route Discovery suppression based on congestion information increased packet delivery ratio from 87.91% to 90.41%, while decreasing overhead by 32%.

By using measured congestion levels to avoid congested areas in discovered routes, DSR can more evenly spread the offered load across different forwarding paths in the network.

6.3. TCP Fairness

Figure 3 shows the results of our third set of experiments, described in Section 5, evaluating the effect on a set of TCP flows when using our protocol modifications using congestion measurements. These experiments used all protocol modifications to DSR and TCP described in Section 4. This graph shows the number of bytes of goodput delivered for each TCP flow over 10 simulation runs with 20 TCP flows per run, or 200 total TCP flows; the y-axis scale on this graph is logarithmic, in order to show the detail in the curves plotted.

In these simulations, we caused each TCP sender to react using ECN, as described in Section 4.3, when an area of the network through which that flow was routed experienced congestion in terms of high levels of usage of the wireless medium in that area or long queue length at an intermediate forwarding node on the route. In addition, since in an ad hoc network, routes can change frequently, to help
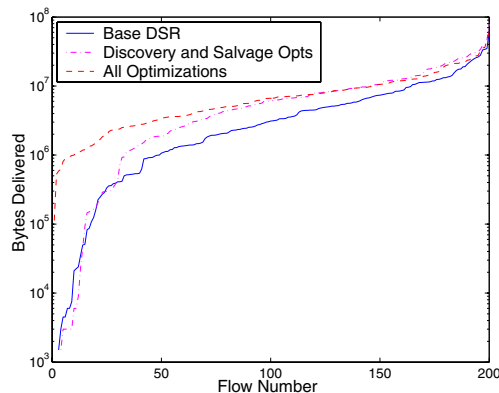

Figure 3: Number of bytes delivered per TCP flow in $1000\,\mathrm{m} \times 1000\,\mathrm{m}$ scenarios with 100 mobile nodes and 20 TCP flows in each run; each TCP flow over 10 simulation runs is shown separately, sorted by the number of bytes of goodput delivered to the receiver for that flow.

ensure fairness, we also cause a TCP sender to begin slow-start as soon as that node receives a new ROUTE REPLY, indicating a change in the route for that TCP connection.

When ECN bits are set in congested areas of the network, flows traversing many hops, and other flows traversing few hops, are evenly penalized, improving TCP fairness. In our simulations, this ECN behavior substantially increased total throughput for more than half the total flows, relative to the results when this ECN modification is not used but all other protocol modifications are still present. Though setting ECN bits slightly decreases the overall throughput, more flows receive a reasonable level of service. This result is expected in any system designed to increase fairness: a multi-hop TCP flow will require more aggregate wireless bandwidth for the same amount of end-to-end delivered bandwidth, so increasing the throughput for connections traversing more hops will have an adverse effect on TCP connections traversing fewer hops.

7. Conclusion

In this paper, we have explored mechanisms for reacting to congestion information in multihop wireless ad hoc networks. Whereas most previous proposals for such optimizations have been to ensure acceptable service to flows requiring certain Quality-of-Service parameters, in this paper, we developed some lightweight mechanisms to allow all flows, including best-effort traffic, to benefit from the avoidance of congested areas. By monitoring the length of its network interface transmission queue and the behavior of the MAC layer on its own wireless network interface, a node can establish an approximation of the degree to which the wireless medium in its area is busy. This measurement reflects not only the behavior of the node itself, but also the

behavior of other nodes around it sharing the same wireless medium.

We have suggested a number of uses of such measurements of congestion in an ad hoc network, at the network, transport, and higher layers, and we simulated a set of such uses in the Dynamic Source Routing protocol (DSR) and TCP. Our simulations demonstrated substantial improvement to DSR and TCP in terms of scalability, packet delivery, overhead, and fairness resulting from this use of congestion information. Although we applied our changes to some areas of DSR to quantitatively demonstrate the usefulness of these optimizations, similar techniques could be applied to other ad hoc network routing protocols and a number of other optimizations are possible as well. For example, a node using a distance vector routing protocol such as DSDV [22] or ADV [3] could increase the time between advertisements during periods in which the wireless medium around the node is particularly busy, and a node using AODV [23] could choose to not attempt local repair during such locally congested periods.

References

[1] Dave Beyer et al. Radio Device API. DARPA Global Mobile Information Systems (GloMo) Program, Defense Advanced Research Projects Agency, Arlington, VA.

[2] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. MACAW: A Media Access Protocol for Wireless LAN's. In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 212–225, August 1994.

[3] Rajendra Boppana and Satyadeva P. Konduru. An Adaptive Distance Vector Routing Algorithm for Mobile, Ad Hoc Networks. In *Proceedings of IEEE INFOCOM 2001*, pages 1753–1762, March 2001.

[4] Bob Braden et al. Recommendations on Queue Management and Congestion Avoidance in the Internet. RFC 2309, April 1998.

[5] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking* (MobiCom'98), pages 85–97, October 1998.

[6] Eyal de Lara, Dan S. Wallach, and Willy Zwaenepoel. Puppeteer: Component-based Adaptation for Mobile Computing. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems*, March 2001.

[7] A. Dracinschi and S. Fdida. Efficient Congestion Avoidance Mechanism. In *Proceedings of the 25th Annual IEEE Conference on Local Computer Networks (LCN'00)*, November 2000.

[8] Sally Floyd. TCP and Explicit Congestion Notification. *ACM Computer Communication Review*, 24(5):10–23, 1994.

[9] Sally Floyd and Van Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[10] Zygmunt J. Haas and Marc R. Pearlman. The Performance of Query Control Schemes for the Zone Routing Protocol. In *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 167–177, June 1998.

[11] Yih-Chun Hu and David B. Johnson. Implicit Source Routing in On-Demand Ad Hoc Network Routing. In *Proceedings of the Second Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, pages 1–10, October 2001.

[12] IEEE Computer Society LAN MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11-1997. The Institute of Electrical and Electronics Engineers, New York, New York, 1997.

[13] IETF MANET Working Group. Mobile Ad Hoc Networks (MANET) Charter. Available at http://www.ietf.org/html.charters/manet-charter.html.

[14] IETF RMT Working Group. Reliable Multicast Transport (RMT) Charter. Available at http://www.ietf.org/html.charters/rmt-charter.html.

[15] Per Johansson, Tony Larsson, Nicklas Hedman, Bartosz Mielczarek, and Mikael Degermark. Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks. In *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networking (MobiCom 1999)*, pages 195–206, August 1999.

[16] David B. Johnson. Routing in Ad Hoc Networks of Mobile Hosts. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications* (WMCSA'94), pages 158–163, December 1994.

[17] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[18] David B. Johnson, David A. Maltz, and Josh Broch. The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks. In *Ad Hoc Networking*, edited by Charles E. Perkins, chapter 5, pages 139–172. Addison-Wesley, 2001.

[19] John Jubin and Janet D. Tornow. The DARPA Packet Radio Network Protocols. *Proceedings of the IEEE*, 75(1):21–32, January 1987.

[20] Phil Karn. MACA — A New Channel Access Method for Packet Radio. In *Proceedings of the 9th Computer Networking Conference*, pages 134–140, September 1990.

[21] Brian D. Noble, M. Satyanarayanan, Dushyanth Narayanan, James Eric Tilton, Jason Flinn, and Kevin R. Walker. Agile Application-Aware Adaptation for Mobility. In *Proceedings of the 16th ACM Symposium on Operating System Principles*, October 1997.

[22] Charles E. Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, August 1994.

[23] Charles E. Perkins and Elizabeth M. Royer. Ad-Hoc On-Demand Distance Vector Routing. In *Second IEEE Workshop on Mobile Computing Systems and Applications* (WMCSA'99), pages 90–100, February 1999.

[24] Jon Postel. Transmission Control Protocol: DARPA Internet Program Protocol Specification. RFC 793, September 1981.

[25] K. K. Ramakrishnan, Sally Floyd, and David L. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168, September 1991.

[26] K. K. Ramakrishnan and Raj Jain. A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer. In *SIGCOMM 98 Symposium Proceedings on Communications Architectures and Protocols*, pages 303–313, 1988.