

2003

# Self-Adjusting Congestion Avoidance Routing Protocol for Ad Hoc Networks

Yi Lu

Bharat Bhargava  
*Purdue University, bb@cs.purdue.edu*

Report Number:  
03-018

---

Lu, Yi and Bhargava, Bharat, "Self-Adjusting Congestion Avoidance Routing Protocol for Ad Hoc Networks" (2003). *Computer Science Technical Reports*. Paper 1567.  
<http://docs.lib.purdue.edu/cstech/1567>

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact [epubs@purdue.edu](mailto:epubs@purdue.edu) for additional information.

**SELF-ADJUSTING CONGESTION AVOIDANCE  
ROUTING PROTOCOL FOR AD HOC NETWORKS**

**Yi Lu  
Bharat Bhargava**

**Department of Computer Sciences  
Purdue University  
West Lafayette, IN 47907**

**CSD TR #03-018  
June 2003**

# Self-Adjusting Congestion Avoidance Routing Protocol for Ad Hoc Networks \*

Yi Lu, Bharat Bhargava  
Center for Education and Research in Information Assurance and Security  
and  
Department of Computer Sciences  
Purdue University, West Lafayette, IN, 47907, USA.  
{yilu,bb}@cs.purdue.edu

## Abstract

In an ad hoc network, the wireless media is shared by multiple nodes. The contention among neighbors for the access to the shared media is the major cause for network congestion. As wireless links usually have low capacity, congestion in ad hoc networks is a more severe problem than in wired networks. The main thrust of our work is to avoid congestion at IP layer by minimizing contentions for channel access. The intermediate delay, which characterizes the impacts of channel contention, traffic load, and the length of a route, is developed as a new routing metric. Two approaches are proposed to locally estimate delay using statistic and probability methods respectively. An ad hoc routing protocol that uses intermediate delay as metric is developed, namely self-adjusting congestion avoidance (SACA) protocol.

The performance of SACA is compared against that of AODV and DSR using three types of traffic. SACA is able to deliver more than 80% data packets even under heavy traffic load. It is 50% - 60% more efficient than DSR in terms of delivery ratio. It introduces 50% less protocol overhead than DSR does. SACA delivers 40% - 400% more packets than AODV depending on the type of traffic.

## 1 Introduction

A mobile ad hoc network (MONET) is a collection of mobile nodes that are deployed as a multi-hop wireless network without the aid of any preexisting infrastructure or centralized administration. It relies on nodes cooperation to maintain network connectivity and functionality. The salient characteristics of ad hoc networks, including highly dynamic topologies, low bandwidth, energy-constrained operations, and limited computation capability, make the design of routing protocols a challenging problem. The protocols must be capable of keeping up with the drastically and unpredictably changing network topology, with minimized message exchanges, in a fully distributed way.

Wireless links have significantly lower capacity than their hardwired counterparts (e.g., 54Mbps for 802.11g VS. 9.952Gbps for OC192). The real throughput, which is affected by multiple access, fading, noise, and interference conditions, is often much less than a channel's maximum transmission rate. Congestion is typically the norm rather than the exception in ad hoc networks[8], that is, the aggregated traffic demand will frequently approach or exceed the link capacity. Traditional congestion control mechanisms such as TCP, are implemented at higher network layers. They reduce traffic sending rate upon occurrence of congestion. In ad hoc networks, the existence of multiple routes between two nodes makes it possible for the routing protocol to select an appropriate route, so that network congestion can be minimized without sacrificing traffic rate.

---

\*This research is supported by Center for Education and Research in Information Assurance and Security (CERIAS), NSF grants CCR-0001788 and ANI-0219110, and CISCO URP grant.

Many routing protocols are proposed for ad hoc networks, such as destination-sequenced distance vector (DSDV) [19], ad-hoc on-demand distance vector (AODV) [18], and dynamic source routing (DSR) [13]. Most of them adopt the content of routing information from the Internet protocols and use hop count as the metric to make routing decisions. Hop count does not provide enough information for congestion control or avoidance.

Routing with load balancing has been explored in [22][6]. The idea is to provide some additional information, such as the secondary metric based on the current load on each node, to help distribute and balance traffic load. It prevents a single node from being overwhelmed. The experimental study in [16] shows that exceeding the capacity of the channel is the major reason for network congestion. In an ad hoc network, the wireless media is shared by multiple contending nodes. The access to the shared media is complicated due to the hidden terminal problem [4] (e.g., a node will contend for the wireless channel not only for sending but also for receiving packets). The contention among multiple nodes leads to congestion. If the contention is already tense among a node's neighbors, it should not be chosen to forward packets even if there is no load on itself.

The main thrust of our work is to reduce network congestion at the IP layer by minimizing channel contentions. The essence is to avoid *hot* spots where multiple nodes are contending with each other. The global coupling effects of wireless channel access in ad hoc networks poses a great challenge to the evaluation of the degree of contentions with local information. In addition, traffic load on a node must be considered, as the store-and-forward process may also cause congestion when the capacity of a node is exceeded. The shorter routes are preferred because longer routes means higher possibility of potential congestion.

Our methodology to solve this problem is as follows: (1) We use a single server queueing system to model nodes in ad hoc networks. The impact of channel contention is quantified using the service time. The routing cost at each node is computed as the estimated delay, which reflects the effects of channel contention, current load, and expected load in the future. (2) A new routing metric, namely intermediate delay, is developed, which measures the amount of communication delay introduced by the nodes in between the source and destination. The route with the least intermediate delay will likely involve in the least channel contention. (3) Two approaches are designed to estimate the delay at a node. The first one applies statistical methods to evaluate the mean service time in case there is active traffic. The second one uses probability methods to compute the expectation of the service time according to the underlying MAC protocol when no active traffic exists. (4) A new proactive ad hoc routing protocol, self-adjusting congestion avoidance (SACA), is developed. It uses intermediate delay as the metric to avoid network congestion. Experimental studies are conducted to evaluate the performance of SACA and compare it with AODV and DSR protocols.

Our work is conducted in the framework of CSMA/CA (carrier sense multiple access with collision avoidance) paradigm, which is adopted by the widely used IEEE 802.11 standard [1]. The ideas and proposed solutions are also applicable to other contention-based media access protocols.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 introduces contention-based access to shared media, channel spatial reuse, and the idea of ad hoc routing based on intermediate delay to avoid congestion. Two approaches are proposed in section 4 to locally estimate delay. Section 5 presents the detail of self-adjust congestion avoidance routing protocol. The performance of the proposed protocol is compared against AODV and DSR in section 6. Section 7 concludes the paper.

## 2 Related Work

According to the way in which mobile nodes exchange routing information, ad hoc routing protocols may be categorized as *proactive* and *on-demand*. The proactive protocols periodically disseminate routing information among all the nodes in the network, so that every node has the up-to-date information for all possible routes. On-demand routing protocols operate on a need basis, discover and maintain only active routes that are currently used for delivering data packets.

C.E. Perkins and P. Bhagwat introduced the destination-sequenced distance-vector (DSDV) routing in [19]. DSDV extends the basic Bellman-Ford mechanism by attaching a sequence number that is originated by the

destination to each distance. It requires every node to periodically advertise its own routing table to its neighbors. Ad hoc on-demand distance vector (AODV) routing protocol is proposed by C.E. Perkins et al. AODV is also based upon distance vector, and uses destination sequence numbers to determine the freshness of routes. It operates in the on-demand fashion, as opposed to the proactive way of the DSDV protocol. D.B. Johnson and D.A. Maltz presented another on-demand protocol dynamic source routing (DSR) in [13]. Instead of using distance vector, the sender of a packet determines the complete sequence of nodes through which to forward the packet and explicitly lists the route in the packet's header. In the meantime, several research efforts are integrating the proactive and on-demand techniques. Z.J. Haas et al. introduce the zone routing protocol (ZRP) [10] for ad hoc networks, in which each node contains an *r-zone* (i.e., all nodes it can reach within  $r$  hops). Proactive routing is used within a zone while the on-demand technique is used for inter-zone communication. R.V. Boppana and S.P. Konduru present another way of combining proactive and on-demand techniques in [5]. They propose the adaptive distance vector (ADV) protocol. ADV uses routing updates to maintain routes like DSDV. It also shows on-demand characteristics by varying the frequency and the size of routing updates according to the network conditions. J.J. Garcia-Luna-Aceves and M. Spohn develop a link-state based ad hoc routing protocol, source-tree adaptive routing (STAR) [9]. To conserve transmission bandwidth and energy, nodes maintain a partial topology map of the network and only transmit changes to the source routing tree when necessary. Associativity-based routing (ABR) [22] is the first protocol that considers load as a part of the metric. The load is primitively measured as the number of routes a node is involved and used as the secondary metric. The protocol does not take into account various traffic loads on different routes. A few load balancing routing protocols are proposed thereafter, using the same idea as ABR but different methods to compute load, such as dynamic load-aware routing presented in [14].

To our knowledge, SACA is the first routing protocol that avoids congestion by reducing contentions.

### 3 Shared Media Access and Congestion Avoidance

#### 3.1 Contention-based access to shared media

In an ad hoc network, any transmitted packet is locally broadcast to the wireless channel. All and only nodes within the transmission range of the sender (i.e., the neighbors of the sender) can receive the packet. If one neighbor is sending or receiving a packet at the same time period, collision occurs. CSMA/CA requires the sender and receiver to exchange request-to-send/clear-to-send (RTS/CTS) frames prior to the actual data frame to avoid collision. The RTS/CTS exchange is a control handshake that distributes the media reservation information to neighbors to ensure no collision in data transmission phase.

We define a *single-hop flow* as a stream of packets directly transmitted from one node to another (called a *flow* in this paper for simplicity). Two flows will contend with each other if either the sender or receiver of one flow is within the transmission range of the sender or receiver of the other flow. This is called *local contention*. Figure 1 shows local contentions among flows. The example is a six-node ad hoc network. A line between two nodes denotes that they are neighbors.  $F_1$ ,  $F_2$ , and  $F_3$  are three flows.  $F_2$  contends with  $F_1$  because the receivers B and D are within each other's transmission range. At any time, only one flow is allowed to use the channel. We denote  $R_i$  by the rate of  $F_i$  and  $C$  by the capacity of the wireless channel. If  $R_1 + R_2 > C$ , congestion occurs and packets will be dropped. The locality of contentions enables the so called *channel spatial reuse* [17], e.g.,  $F_1$  and  $F_3$  can transmit to the same physical channel (in terms of frequency) simultaneously as they are not local to each other.

Channel spatial reuse together with the multi-hop nature of ad hoc routing provides a way to reduce contentions. For instance, if C wants to establish a connection session with F, selecting the route  $C \rightarrow E \rightarrow F$  instead of  $C \rightarrow D \rightarrow F$  will avoid contention between nodes B and D.

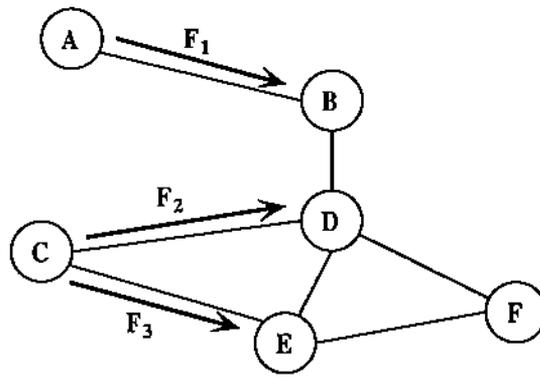


Figure 1: Network topology and flows

### 3.2 A new routing metric

A new metric is needed to realize the idea of contention reduction. The requirements for this metric includes: (1) This metric must comprehensively characterize the impacts of channel contention, traffic load, and the length of a route. (2) The cost at a mobile node  $H$ , denoted as  $C(H)$ , is computed using only local information.

We develop a new routing metric, namely *InterMediate Delay* (IMD). IMD measures the amount of communication delay introduced by the nodes in between the source and the destination (we do not consider the computation delay, which is trivial compared with the communication delay). Every node independently computes the delay it will introduce. The delay is determined by the contention with neighbors, current traffic load, and expected traffic load in near future. If the source and destination of a route are neighbors, they do not need any intermediate node, thus the IMD of this route is 0. If they are not neighbors and have no intermediate node, the IMD is  $\infty$ , which means packets can not be sent from the source to the destination (the delay is infinity).

### 3.3 Ad hoc routing based on intermediate delay

The following examples demonstrate how to apply the intermediate delay to ad hoc routing. For the demonstration purpose, a simple approach is used for delay computation

1. If the capacity of the wireless channel is  $C$ , the size of a packet is  $P$ , the delay for sending a packet is  $P/C$  (the MAC layer control messages are ignored).
2. If  $n$  nodes are contending for a channel, each one can get a share of capacity  $C/n$ . In this case, the delay for sending a packet is  $nP/C$ .

Figure 2, 3, and 4 demonstrate a ten-node ad hoc network. The line with an arrow head represents a connection session. In all examples, a connection between nodes F and G is to be established.

Figure 2 illustrates how to choose a route with the presence of other connection sessions. As shown in figure 2a, there is an active connection session between A and C when F wants to establish a connection with G. D realizes the contention with A and computes the delay to be  $2P/C$ . E will contend with C. His computation of the delay is also  $2P/C$ . Because there is no other active traffic, the delay computed by nodes H, I, and J is  $P/C$ . The IMD of the route  $F \rightarrow D \rightarrow E \rightarrow G$  is  $4P/C$ , while that of the route  $F \rightarrow H \rightarrow I \rightarrow J \rightarrow G$  is  $3P/C$ . The later one is chosen for the connection between F and G (figure 2b). This route is better in terms of channel reuse and congestion avoidance. It also introduces less end-to-end delay.

Figure 3 shows the adaption to traffic changes. At the beginning, there is no traffic in the network. Every node can make full use of the channel and introduce a communication delay of  $P/C$ , as shown in figure 3a. The shortest route (in terms of hop count) is chosen to establish the connection (figure 3b), since it introduces

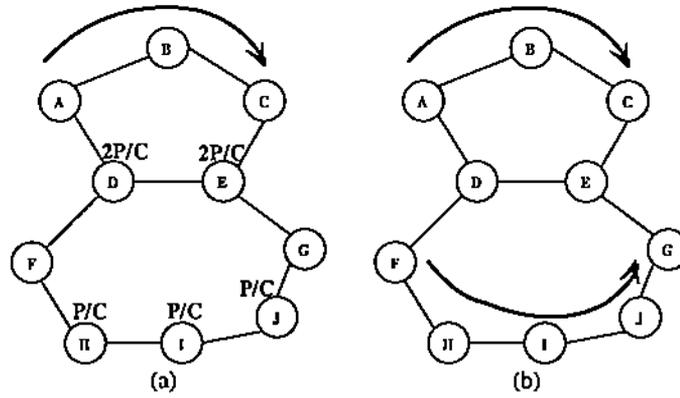


Figure 2: Choose a route with presence of other connections

least intermediate delay. Sometime after the establishment of the connection, a new connection session from A to C is introduced. This connection will follow its best route  $A \rightarrow B \rightarrow C$ . The new connection causes channel contention between A and D, and C and E. The two nodes will recompute the delay. The new delay is  $2P/C$  and the IMD of the route is  $4P/C$ . The re-computation at nodes H, I, and J still returns  $P/C$ . The route  $F \rightarrow H \rightarrow I \rightarrow J \rightarrow G$  becomes a better one, as shown in figure 3c. Node F re-establishes the connection via the new route. Figure 3d shows the final result after the adaption to the new connection session. It is the same as the result shown in figure 2b.

Figure 4 illustrates the adaption to network topology changes. The first two steps are the same as the example shown in figure 2. The longer route is chosen to avoid congestion. Suppose nodes A and C are moving and are no longer contending with D and E after sometime. D and E recompute the delay and get  $P/C$  as the result. F will find out that the route  $F \rightarrow D \rightarrow E \rightarrow G$  becomes a better one as its IMD is  $2P/C$ . The connection will be re-established as shown in figure 4d.

The above examples demonstrate the essential idea of congestion avoidance by using intermediate delay. The simple delay computation approach is not suitable for the design of a practical routing protocol: (1) At the time a node computes the delay, it may not know the number of neighbors who are contending with it. (2) Due to the locality of contention, access to a wireless channel creates global coupling effects in the entire network [17]. Even if the number of contending nodes is known, the share of capacity can not be predetermined. (3) The traffic changes, topology changes, and routing decisions all have great impacts on the IMD of a route and make it highly dynamic.

## 4 Delay Estimate

One of the major challenges to implement IMD in a routing protocol is to estimate delay using only local information. In this section, two approaches are presented to accomplish the computation for a node with or without active traffic respectively.

### 4.1 The model

In an ad hoc network, when a packet arrives to a node, a routing decision must be made which determines the next hop to send the packet. Once the decision is made, the packet is placed at the tail of a queue where other packets are waiting to be transmitted out over the wireless channel. In most cases, the time spent on making a routing decision is ignorable compared with the time a packet waiting in the queue. The dynamism and

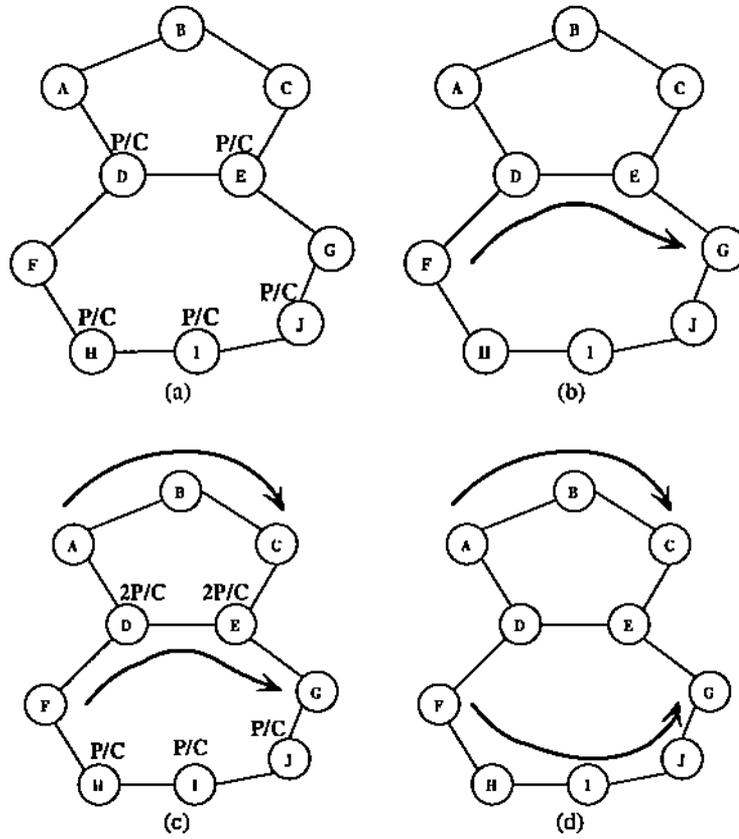


Figure 3: Adapt to traffic changes

complexity of the network make it impractical to analytically evaluate the accurate value of delay. In addition, IMD is computed based on the average delay in a short period of time. Hence, an approximation is sufficient.

We make the following assumptions so that a mobile node can be modelled as a single server queueing system [21].

1. The incoming traffic is localized with respect to time, i.e., in a short period of time, it obeys approximately the same distribution.
2. The channel access is localized with respect to both time and location (e.g., a node finds that the channel is busy, so does its neighbors).
3. A node has an infinite queue. This assumption is made because the aggregated delay along a route makes sense only when a packet can be sent to the destination.
4. The incoming traffic rate and outgoing traffic rate are independent. This assumption is made because (a) the complexity of channel contentions washes out the dependency to some extent; (b) only approximate evaluation is required.

The following notations are used to describe the system.

- $\lambda$ : The arriving rate of incoming packets. Packets may come from other mobile nodes or upper layer applications.
- $\mu$ : The service rate, i.e., the number of packets sending out over the wireless channel per second. It is determined by the capacity of the channel and the contention with neighbors.

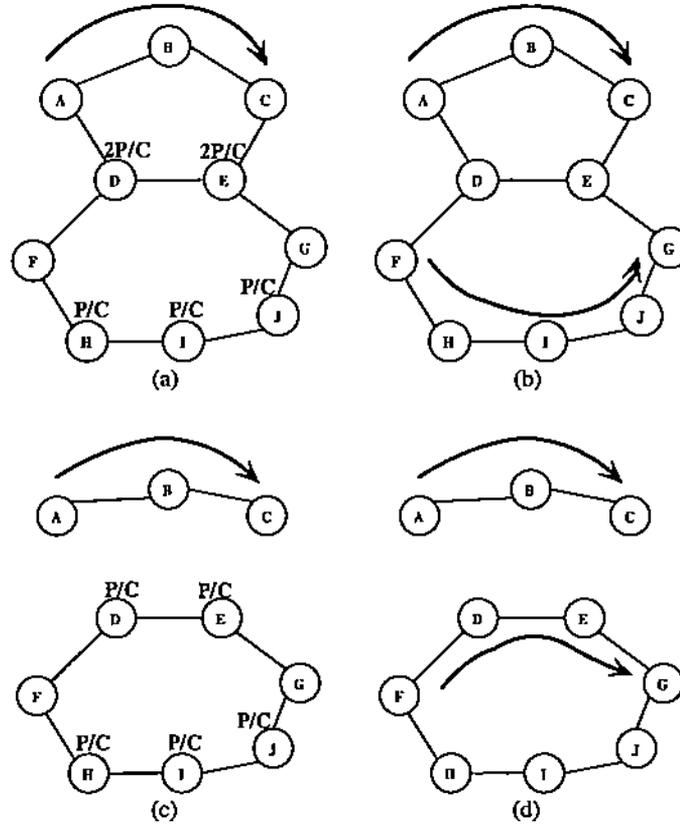


Figure 4: Adapt to topology changes

$T_Q$ : The delay in the queue.

$T_S$ : The average service time for a packet to be transmitted out ( $T_S = \frac{1}{\mu}$ ).

$T_D$ : The delay in the system ( $T_D = T_Q + T_S$ ). It is what we want to estimate.

$L$ : The current queue length.

The in-queue delay  $T_Q$  can be evaluated using equation 1 by the application of Little's law [21].

$$T_Q = \frac{\lambda}{\mu(\mu - \lambda)} + T_S L \quad (1)$$

In a time period  $\Delta t$ ,  $\lambda$  is estimated using the following equation, where  $N_A$  is the number of packets arrived within this period.

$$\lambda = \frac{N_A}{\Delta t} \quad (2)$$

From equations 1 and 2, the delay at node is evaluated as follows.

$$\begin{aligned} T_D &= T_Q + T_S \\ &= \frac{\frac{N_A}{\Delta t}}{\frac{1}{T_S}(\frac{1}{T_S} - \frac{N_A}{\Delta t})} + T_S(L + 1) \end{aligned}$$

$$= \frac{T_S(L+1) - \frac{N_A}{\Delta t}(T_S)^2 L}{1 - \frac{N_A}{\Delta t} T_S} \quad (3)$$

So the problem is deduced to compute the service time  $T_S$ . Two cases are studied: a node with active traffic (i.e., it sends out packets over the wireless channel recently) and a node without active traffic.

## 4.2 With Active Traffic

If a node transmits out packets recently, the mean value of the service time can be obtained using the statistical method. Let  $N_S$  be the number of sent packets and  $T_B$  be the time that the node spent on sending packets ( $T_B$  is less than or equal to  $\Delta t$  because the node may be idle).

$$T_S = \frac{T_B}{N_S} \quad (4)$$

If  $\frac{N_S}{T_B} < \frac{N_A}{\Delta t}$ , which means the packet sending rate is less than the packet arriving rate, we assign  $\infty$  to  $T_D$  (i.e., the delay in queue may be arbitrarily large). Otherwise, it is computed from equations 3 and 4 as follows.

$$\begin{aligned} T_D &= \frac{(L+1)\frac{T_B}{N_S} - \frac{N_A}{\Delta t} L \left(\frac{T_B}{N_S}\right)^2}{1 - \frac{N_A}{\Delta t} \frac{T_B}{N_S}} \\ &= \frac{(L+1) - L \frac{N_A}{\Delta t} \frac{T_B}{N_S}}{\frac{N_S}{T_B} - \frac{N_A}{\Delta t}} \end{aligned} \quad (5)$$

To estimate delay, we only need to count the incoming and outgoing packets, current queue length, and the time when the node is sending packets.

## 4.3 Without Active Traffic

No active traffic on a node does not mean a packet can be sent out with the shortest delay, because the neighbors may be using the channel. The expectation of the service time can be determined by using probability methods to evaluate the process of packet transmission. We take the IEEE 802.11 distributed coordination function (DCF) as an example to show how it can be done. The method is also applicable to other MAC protocols with slight modifications.

Figure 5 illustrates the process of transmitting a unicast data packet using RTS/CTS. We briefly review it for the purpose of evaluating the expectation of transmission time. The detailed description of the process is available in [1].

When a data packet is ready to transmit, the sender picks up a random backoff time  $b \times T_{slot}$  after observing an idle channel for  $T_{DIFS}$  time, where  $b$  is a random number uniformly distributed over  $[0, CW]$  and  $T_{slot}$  is a value specified by the physical layer. The sender starts to transmit the RTS frame when the backoff time reaches zero. The receiver transmits a CTS frame after  $T_{SIFS}$  time upon receiving the RTS frame if the media is idle. The neighbors of the sender and receiver set the network allocation vector (NAV) correspondingly to indicate that the media is reserved. The sender waits for  $T_{SIFS}$  time and transmits the data after receiving the CTS frame. The receiver waits for  $T_{SIFS}$  time replies with an acknowledge (ACK) frame after receiving the data. The expectation of time cost for a successful attempt to transmit is

$$E[T_s] = T_{DATA} + T_{RTS} + T_{CTS} + T_{ACK} + T_{DIFS} + 3T_{SIFS} + E[T_{backoff}] \quad (6)$$

where  $T_{DATA}$ ,  $T_{RTS}$ ,  $T_{CTS}$  and  $T_{ACK}$  are, respectively, time periods for transmitting a data packet, RTS frame, CTS frame, and ACK frame,  $T_{DIFS}$  and  $T_{SIFS}$  are DCF interframe and short interframe time periods, and  $T_{backoff}$  is the time spent on the backoff procedure.

Sender

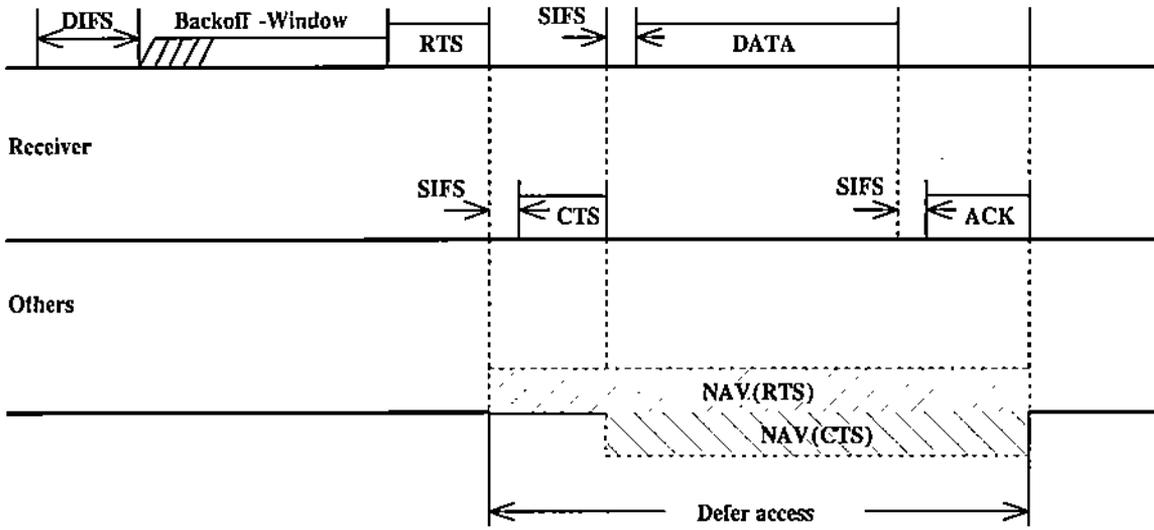


Figure 5: Transmission of a unicast data packet using RTS/CTS of the IEEE 802.11 standard

The attempt to transmit fails if a CTS frame has not been received at the end of  $T_{timeout}$  time following the transmission of the RTS frame. The sender then restarts this process. The expectation of time cost for a failed attempt is

$$E[T_f] = T_{RTS} + T_{CTS} + T_{timeout} + E[T_{backoff}] \quad (7)$$

During the backoff procedure, if no channel activity is indicated for the duration of a particular backoff slot, the backoff time is decremented by  $T_{slot}$ . Otherwise, the procedure is suspended without decrementing the backoff time. It resumes following observing an idle channel for  $T_{DIFS}$  time. The time spent on the backoff procedure is

$$T_{backoff}(b) = N_b T_{slot} + N_{DIFS} T_{DIFS} \quad (8)$$

where  $N_b$  is the number of slots in the procedure ( $b$  of which are indicated as idle) and  $N_{DIFS}$  is the number of suspensions. We assume the probability that a channel is busy will not change during the transmission period and denote it as  $p$ .  $N_b$  is a Geometric distribution. For a given random number  $b$ , the expectation of time spent on backoff procedure is

$$\begin{aligned} E[T_{backoff}(b)] &= E[N_b]T_{slot} + E[N_{DIFS}]T_{DIFS} \\ &= \frac{b}{1-p}T_{slot} + bpT_{DIFS} \end{aligned} \quad (9)$$

The expectation of time spent on backoff for an attempt is

$$\begin{aligned} E[T_{backoff}] &= E[E[T_{backoff}(b)]] \\ &= E\left[\frac{b}{1-p}T_{slot} + bpT_{DIFS}\right] \\ &= \frac{CW}{2(1-p)}T_{slot} + \frac{CW}{2}pT_{DIFS} \end{aligned} \quad (10)$$

The contention window (CW) parameter takes an initial value  $aCW_{min}$  at the first attempt. It takes the next

value in the series every time an attempt to transmit fails until the CW reaches the value  $aCW_{max}$ . Once it reaches  $aCW_{max}$ , the CW will remain at that value until it is reset by a successful attempt. The set of CW values are sequentially ascending integer powers of 2 minus 1, from  $aCW_{min}$  to  $aCW_{max}$ , which are specific to the physical layer. For example, direct sequence spread spectrum (DSSS) physical layer management information base (MIB) defines  $aCW_{min}$  to be 31 and  $aCW_{max}$  to be 1023. In the rest part of the section, we assume DSSS is used as the physical layer. Let  $CW^n$  be the CW of the  $n$ -th attempt to transmit, then

$$CW^n = \begin{cases} 2^{n+4} - 1, & 1 \leq n \leq 6; \\ 2^{10} - 1, & n > 6. \end{cases}$$

Let  $T_s^n$  and  $T_f^n$  be the time cost of a successful transmission and a failed transmission for the  $n$ -th attempt, respectively. From equations 6, 7 and 10, we have

$$E[T_s^n] = T_{DATA} + T_{RTS} + T_{CTS} + T_{ACK} + T_{DIFS} + 3T_{SIFS} + \frac{CW^n}{2(1-p)}T_{slot} + \frac{CW^n}{2}pT_{DIFS} \quad (11)$$

$$E[T_f^n] = T_{RTS} + T_{CTS} + T_{timeout} + \frac{CW^n}{2(1-p)}T_{slot} + \frac{CW^n}{2}pT_{DIFS} \quad (12)$$

The receiver of the RTS frame will transmit a CTS frame after  $T_{SIFS}$  time if the NAV indicates that the channel is idle. Otherwise, the receiver will not respond to the RTS frame. Since channel access has locality characteristic and the busy channel is the reason for failure. The possibility of a successful attempt  $P_s$  is approximately  $1 - p$ .

The expected transmission time makes sense only to successfully delivered data packets. We assume that there is no limit on retry (i.e., the sender will keep trying until the packet is delivered). The expected transmission time is.

$$E[T_{trans}] = P_s E[T_s^1] + \sum_{i=1}^{\infty} ((1 - P_s)^i P_s (E[T_s^{i+1}] + \sum_{j=1}^i E[T_f^j])) \quad (13)$$

Because the  $CW^n$  is fixed when  $n > 6$ ,  $E[T_s^n]$  and  $E[T_f^n]$  are fixed too when  $n > 6$ . Let  $E[T_s^{n>6}]$ ,  $E[T_f^{n>6}]$ , and  $T_{trans}^{n>6}$  be the transmission time after  $CW^n$  is fixed.

$$\begin{aligned} E[T_{trans}^{n>6}] &= P_s E[T_s^{n>6}] + (1 - P_s)(E[T_f^{n>6}] + E[T_{trans}^{n>6}]) \\ \Rightarrow E[T_{trans}^{n>6}] &= \frac{P_s E[T_s^{n>6}] + (1 - P_s) E[T_f^{n>6}]}{P_s} \end{aligned} \quad (14)$$

Notice  $E[T_s^{n>6}] = E[T_s^6]$  and  $E[T_f^{n>6}] = E[T_f^6]$ . We get

$$E[T_{trans}^{n>6}] = \frac{P_s E[T_s^6] + (1 - P_s) E[T_f^6]}{P_s} \quad (15)$$

From equations 13 and 15, the expected transmission time (i.e., the service time) can be computed using the following equation.

$$\begin{aligned} E[T_{trans}] &= P_s E[T_s^1] + \sum_{i=1}^5 ((1 - P_s)^i P_s (E[T_s^{i+1}] + \sum_{j=1}^i E[T_f^j])) \\ &\quad + (1 - P_s)^6 \left( \frac{P_s E[T_s^6] + (1 - P_s) E[T_f^6]}{P_s} + \sum_{j=1}^6 E[T_f^j] \right) \end{aligned}$$

$$\begin{aligned}
&= (1-p)E[T_s^1] + \sum_{i=1}^5 (p^i(1-p)(E[T_s^{i+1}] + \sum_{j=1}^i E[T_f^j])) \\
&\quad + p^6 \left( \frac{(1-p)E[T_s^6] + pE[T_f^6]}{(1-p)} + \sum_{j=1}^6 E[T_f^j] \right) \tag{16}
\end{aligned}$$

Once the physical layer parameters are determined, the delay of transmitting a packet can be estimated by using equation 3, 16, and the possibility that the channel is busy. This computation is done in  $O(1)$  time.

## 5 Self-Adjusting Congestion Avoidance Routing Protocol

### 5.1 Overview

The self-adjusting congestion avoidance (SACA) routing protocol is designed based on the ideas discussed in section 3.3. SACA is a distance vector routing protocol. One of the major differences between SACA and other distance vector based routing protocols is that SACA uses IMD instead of hop count as the distance.

To send packets to others, Each node maintains a routing table that contains entries to all known nodes. The data structure of a routing entry is shown in table 1a. The field *next\_hop* is the next node on the route towards the destination *dst*, *imd* is the intermediate delay of the route, and *seqnum* is the sequence number representing the “freshness” of the route. The sequence number is maintained by the destination. Routes with more recent sequence numbers are always preferred for making routing decisions. For routes with the equal sequence number, the one with the smaller IMD is chosen.

```

class REntry {
  REntry();
  addr_t dst;
  addr_t next_hop;
  float imd;
  uint seqnum;
  float min_advertised_delay;
  time advertise_ok_at;
  bool need_advertise;
  uint MAC_callback_cnt;
  Event* trigger_event;
  Event* timeout_event;
  PacketQueue* q;
}

```

variable	meaning	value
MIN_INTERVAL	minimum time between two advertisements	1 seconds
MAC_CALLBACK	how many callbacks indicate a broken link	2
STARTUP_ADVERTISE	how many advertisements are sent during startup	5
PERIOD_ADVERTISE	time between two full advertisements	15 seconds
DEFAULT_TTL	default value of TTL	30

(a) Data structure of a routing entry

(b) Major constants

Table 1: Data structure and constants

SACA is a proactive protocol, it requires every node to periodically advertise the routing table to its neighbors. Significant new routing information such as a new route or a broken route may trigger advertisement as well. When a node makes the advertisement, it includes the estimated delay that this node may introduce in the advertisement packet. A broken or unavailable route is assigned a delay of  $\infty$  (i.e., a value greater than the maximum allowed end-to-end delay). A route with  $\infty$  delay is considered as invalid and is usually not included in advertisements.

Table 1b shows the major constants and their values that are used in SACA.

## 5.2 Basic operations

### Handle received packets

Figure 6 illustrates the procedure of handling a packet received from either upper layer applications or MAC layer (i.e., from other nodes). This procedure consists of two phases: IP header validating and updating that is shown in the left part of the flowchart and packet handling that is shown in the right part.

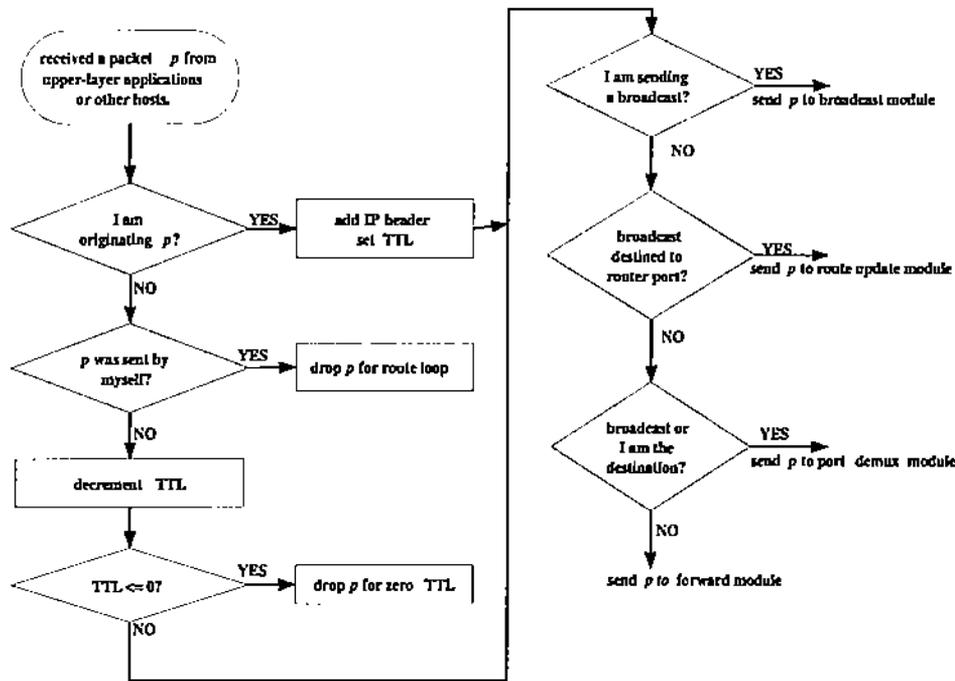


Figure 6: Flowchart illustrating how SACA agent handles a received packet

Route loop is checked in the first phase by examining the source address of the packet and the value of TTL (as the hop count of a route is much less than the value of TTL, TTL reaching zero usually indicates a loop occurs in the route). The IP header checksum is computed and validated outside this procedure. Based on the destination address and port, the packet is sent to the route update module, the upper layer applications, or MAC layer to broadcast or unicast.

### Advertise routes

SACA uses route advertisements to disseminate information throughout the network. Two types of advertisements are defined in SACA protocol. One, called “full advertisement”, carries all the available routing information. The other, called “partial advertisement”, carries only information changed since the last advertisement. Full advertisements are generated relatively infrequently. If a partial advertisement contains most of routing entries, it is upgraded to a full advertisement so that the next partial advertisement will be smaller.

Two events will trigger an advertisement. The first one triggers a full advertisement, which is scheduled PERIOD\_ADVERTISE seconds after the previous full advertisement. In the bootstrapping phase, a node may schedule full advertisements more frequently. The other event triggers a partial advertisement upon receiving significant new routing information, including: (1) a more recent sequence number, which helps SACA to adapt in circumstances similar to the example shown in figure 3; (2) a broken link, as discussed in [15], propagating bad news quickly will improve system performance.

```

MakeAdv(periodic) {
  for each routing entry rte
    entry_count++;
    if rte.need_advertise == TRUE
      count++;
    if rte.advertise_ok_at > now
      unadvertiseable++;
    if count >= entry_count*2/3
      periodic = TRUE;
  make an advertisement packet p;
  add estimated delay to p;
  if periodic == TRUE
    increment sequence number by 2;
    add sequence number to p;
    add entry_count - unadvertiseable to p;
    for each routing entry rte
      if rte.advertise_ok_at <= now
        add rte to p;
        rte.need_advertise = FALSE;
  else
    add sequence number to p;
    add count to p;
    for each routing entry rte
      if rte.need_advertise == TRUE
        add rte to p;
        rte.need_advertise = FALSE;
}

```

**Alg. 1** Make an advertisement packet

Each routing entry is associated with two flags: *need\_advertise* and *advertise\_ok\_at*. A partial advertisement only contains those entries whose *need\_advertise* is set. A full advertisement includes all entries whose *advertise\_ok\_at* is earlier than the current time. In both cases, the estimated delay the node may introduce and the sequence number are included in the advertisement packets. Before each full advertisement, the sequence number is incremented by 2 so that the sequence number maintained by the destination is always an even integer.

The pseudo code of making an advertisement packet is shown in alg. 1.

### Maintain routes

Route maintenance updates the routing entries upon receiving an advertisement and determines whether to trigger a partial advertisement. Each time a node  $i$  receives from one of its neighbors  $j$ , an advertisement of a route to a node  $x$ , with sequence number  $seq_j^x$  and intermediate delay  $imd_j^x$  (including the delay introduced by  $j$ ), it changes the next hop if and only if one of the following two is true.

1. The new route contains a newer (valid) sequence number (i.e.,  $seq_j^x > seq_i^x$ ) and  $imd_j^x < \infty$ .
2.  $seq_j^x = seq_i^x$  and the new route introduces less intermediate delay (i.e.,  $imd_j^x < imd_i^x$ ).

These two constraints guarantee that SACA will not introduce loops in routes. In the first case, as proved in [19], a loop cannot be created if nodes use newer sequence number to pick routes. The loop-free property holds in the second case due to the theorem proved in [11], which states that distance vector algorithms always maintain loop-free paths in presence of static or decreasing link weights.

Alg. 2 demonstrates the procedure of route maintenance. An extra functionality of route maintenance is shown in function *ProcessAdvEntry*. The reception of an advertisement entry with an older sequence number will trigger a partial advertisement to help the neighbor to obtain the up-to-date route.

```

ProcessAdv(pkt) {
  sender = source address of pkt;
  delay = estimate delay in pkt;
  seqnum = sequence number in pkt;
  rte = routing entry to sender;
  if rte does not exist
    add a routing entry rte;
    rte.dst = rte.next_hop = sender;
    rte.imd = 0;
    rte.seqnum = seqnum;
    rte.need_advertise = TRUE;
    trigger advertisement for rte;
  else
    rte.next_hop = sender;
    rte.imd = 0;
    if (rte.seqnum < seqnum)
      rte.seqnum = seqnum;
      rte.need_advertise = TRUE;
      trigger advertisement for rte;
  for each advertisement entry adv
    if adv.dst == my_address
      if adv.imd != 0
        schedule a full advertisement;
    else
      adv.imd = adv.imd + delay;
      adv.next_hop = sender;
      ProcessAdvEntry(adv);
}

ProcessAdvEntry(adv) {
  rte = routing entry to adv.dst;
  if rte does not exist
    add a routing entry rte;
    rte.dst = adv.dst;
    rte.next_hop = adv.next_hop;
    rte.imd = adv.imd;
    rte.seqnum = adv.seqnum;
    rte.need_advertise = TRUE;
    trigger advertisement for rte;
  else if rte.seqnum == adv.seqnum
    if rte.imd > adv.imd
      UpdateRoute(rte, adv);
  else if rte.seqnum < adv.seqnum
    if adv.imd < INFINITY or rte.next_hop == adv.next_hop
      UpdateRoute(rte, adv);
    rte.need_advertise = TRUE;
    trigger advertisement for rte;
  else if rte.seqnum > adv.seqnum
    if rte.imd < INFINITY and adv.imd == INFINITY
      rte.need_advertise = TRUE;
    trigger advertisement for rte;
}

```

Alg. 2 Route maintenance

### Handle broken links

A broken link to a neighbor may be inferred if no advertisement is received from it for a while. Each neighbor is associated with a *timeout\_event*, which will be triggered after  $2 \times PERIOD\_ADVERTISE$  seconds. This event indicates that the link to the neighbor has broken. It will be reset whenever an advertisement is received from the neighbor.

The MAC callback is another mechanism to detect broken links, since CSMA/CA will report an error when it fails to transmit a packet. The failure may be caused by a broken link, or channel contention. Continuous occurrence of the failure indicates that either the neighbor is not available or the contention is too tense. We do not want to pick this neighbor as the next hop in both cases. If the number of continuous callbacks exceeds the preset threshold `MAC_CALLBACK`, the *timeout\_event* is triggered.

```

MACCallback(pkt)
{
  next_hop = next hop of pkt;
  drop pkt for MAC callback;
  rte = routing entry to next_hop;
  rte.MAC_callback_cnt++;
  if rte.MAC_callback_cnt > MAC_CALLBACK
    if rte.timeout_event exists
      cancel rte.timeout_event;
  HandleTimeout(rte);
}

HandleTimeout(rte)
{
  for each routing entry rte2
    if rte2.next_hop == rte.dst and rte2.imd < INFINITY
      rte2.imd = INFINITY;
      rte2.seqnum++;
      rte2.need_advertise = TRUE;
      trigger advertisement for rte2;
}

```

Alg. 3 Handle broken links

Alg. 3 shows how MAC callback triggers the event and how SACA handles a broken link. When a link

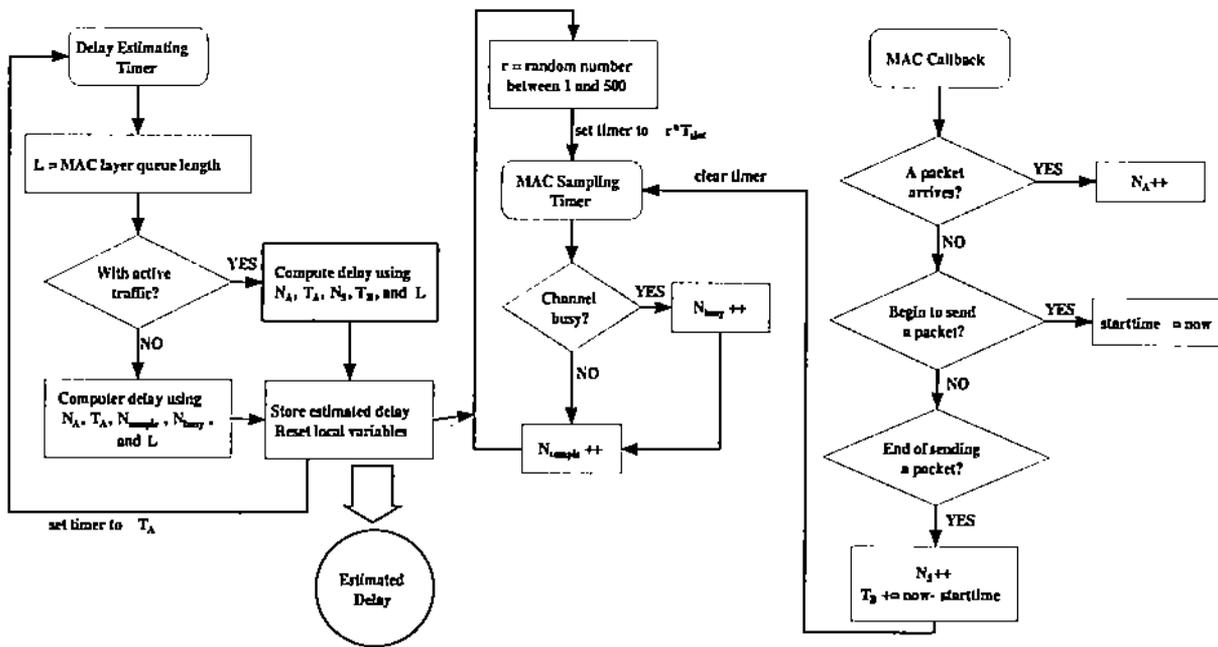


Figure 7: Delay estimate

to a neighbor is indicated broken, any route through that neighbor is immediately assigned  $\infty$  to IMD and the sequence number is incremented by 1. Thus a broken route is always associated with an odd sequence number while the valid one is associated with an even sequence number.

### Estimate delay

Every node estimate delay using the approaches presented in section 4.  $\Delta t (T_A)$  determines how frequently the delay is estimated. In our implementation, it is set to PERIOD\_ADVERTISE, the time interval between two full advertisements.  $N_A$ ,  $N_B$ , and  $T_B$  are counted using a MAC callback function. This function is invoked when a packet arrives at MAC, a packet is ready to transmit, and a packet is transmitted. The probability  $p$  is determined by randomly sampling. Each node maintains a sampling timer, which will be randomly triggered about 200 times per second. When the timer is triggered, SACA checks state of the channel. This timer is set when a new estimate process begins. It is cleared if active traffic is detected at the node. Figure 10 illustrates the procedure of delay estimate. Two timers, delay estimate timer and MAC sampling timer, are maintained. The delay estimate timer is triggered every  $T_A$  seconds to compute the new delay. The estimated delay is stored in a global variable so that other modules can access it.

### Lazy route query

SACA does not provide the dedicated route query operation as on-demand protocols do. We introduce a new technique called *lazy route query*. When a node wants to send packets to a destination but does not have a valid route, it includes this route with  $\infty$  delay in the next advertisement. Neighbors who have valid routes to the destinations will advertise those routes to help the querier. Lazy route query works very well with the proactive approach, because (1) every node periodically advertises the routing table, changes are one of the neighbors has already had a valid route; (2) multiple routes may be queried in one advertisement.

## 6 Performance Evaluation

The most recent version (2.1b9a) of the network simulator ns2 [2] is used for the simulation study. The wireless interface works like the 914 MHz Lucent WaveLAN direct-sequence spread-spectrum (DSSS) radio interface [3]. The IEEE 802.11 distributed coordination function (DCF) with CSMA/CA is used as the MAC layer protocol. The *random waypoint* model [7] is used to generate movements for mobile nodes.

Because many performance comparisons show that on-demand protocols perform better than proactive ones [7] [12], we compare SACA with two on-demand protocols, AODV and DSR, which have received wide attention in recent studies of ad hoc routing protocols [7] [12] [20]. The CMU's implementation of DSR is used in the simulation. The AODV is implemented by the AODV group. To get convincing results, all optimizations for AODV and DSR are enabled in the simulation. We implemented SACA based on the algorithms presented in the previous section. The parameter values used for SACA are given in table 5.1.

### 6.1 Traffic load

The overall goal of the experiments is to evaluate the capability of routing protocols in terms of congestion avoidance. UDP connections are used in the simulation. Each connection is specified as a randomly chosen source-destination (S-D) pair. Every connection starts at a time randomly chosen from 0 to 100 seconds. The packet sizes are fixed as 512 bytes. Three types of traffic are studied in the experiments.

- *Constant Bit Rate (CBR) traffic*: generates traffic according to a deterministic rate.
- *Exponential On/Off (EXPOO) traffic*: generates traffic according to an exponential on/off distribution. Packets are sent at a fixed rate during on periods, and no packets are sent during off periods. Both on and off periods are taken from an exponential distribution.
- *Pareto On/Off (POO) traffic*: generates traffic according to a pareto on/off distribution. This is identical to the exponential on/off distribution, except the on and off periods are taken from a pareto distribution. These sources can be used to generate aggregate traffic that exhibits long range dependency.

We have simulated 30 connections, each of which has an average traffic rate of 15 Kb/s. The aggregated traffic load is 450 Kb/s, which puts much stress on the routing protocols in the sense that congestion will likely occur.

### 6.2 Performance metrics

We compare the routing protocols using the following metrics:

- *Packet Delivery Ratio*: The ratio of the data delivered to the destinations (i.e., throughput) to the data sent out by the sources.
- *Average End-to-end Delay*: The average time it takes for a packet to reach the destination. It includes all possible delays in the source and each intermediate node, caused by routing discovery, queuing at the interface queue, transmission at the MAC layer, etc. Only successfully delivered packets are counted.
- *Normalized Protocol Overhead*: The routing load per unit data successfully delivered to the destination. The routing load is measured as the number of protocol messages transmitted hop-wise (i.e., the transmission on each hop is counted once). A unit data can be a byte or a packet.

In the simulation, the pause time is varied over {10, 20, 40, 80, 160} seconds. Five scenarios are generated for each experiment, and the average values are used for analysis. The values of parameters used in the simulation are given in table 2.

Simulation time	1000 seconds
Independent runs	5
Mobility model	random waypoint
Simulation area	1000m × 1000m
Maximum speed	3 m/s
Wireless transmission range	250m
Channel capacity	2 Mb/s
Number of mobile nodes	60
Number of connection sessions	30
Packet size	512 bytes
CBR rate	15 Kb/s
EXPOO rate	45 Kb/s
EXPOO on time	500 ms
EXPOO off time	1000 ms
POO rate	45 Kb/s
POO on time	500 ms
POO off time	1000 ms
POO shape	1.5

Table 2: Simulation parameters

### 6.3 Experiment results

#### Packet delivery ratio

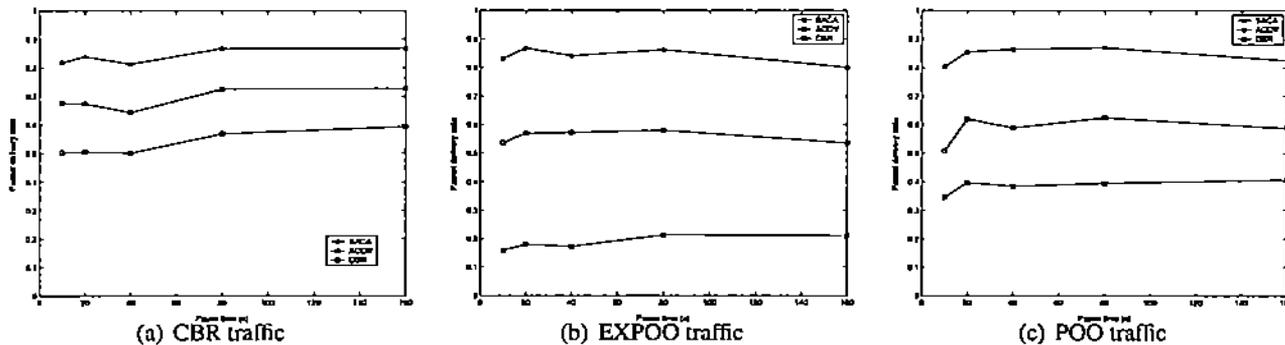


Figure 8: Packet delivery ratio

We observe from figure 8 that every protocol provides a rather stable packet delivery ratio with the increase of node mobility (decrease of the pause time). It indicates that all protocols are able to adapt to topology changes. However, the achieved delivery ratios are quite different. SACA delivers more than 80% of the data packets regardless of the traffic type. The delivery ratio of DSR does not affected much by the traffic type. It is always between 50% to 60%. AODV is sensitive to traffic type. It is able to achieve about 70% delivery ratio for CBR traffic, but only 20% and 40% for EXPOO and POO traffic, respectively. In terms of delivery ratio, SACA is 50% - 60% more efficient than DSR, and 40% - 400% more efficient than AODV depending on the traffic type.

## Protocol overhead

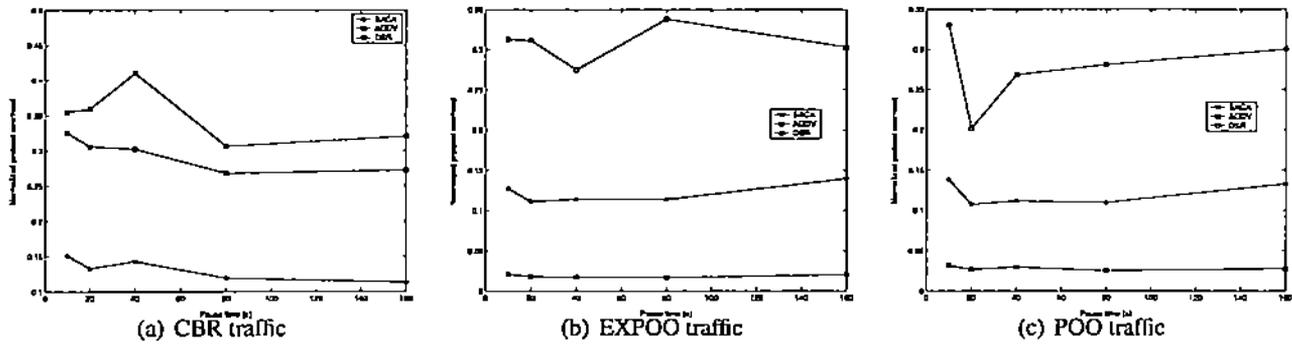


Figure 9: Normalized protocol overhead

As shown in figure 9, SACA introduces 10% - 15% normalized protocol overhead in all experiments, which is less than the half of that introduced by DSR. AODV has the highest overhead among three protocols for CBR traffic. We observe from figure 9b and 9c that AODV introduces very little protocol overhead for EXPOO and POO traffic, which indicates that AODV does not send out many route request messages in these two cases. It partially explains why AODV fails to achieve higher delivery ratio. When a packet is dropped due to congestion, AODV does not consider it as a route error (because no route request is sent for local repair). Packets are continuously sent to the congested routes and are dropped without being noticed. It results in fewer route requests and lower delivery ratio.

## End-to-end delay

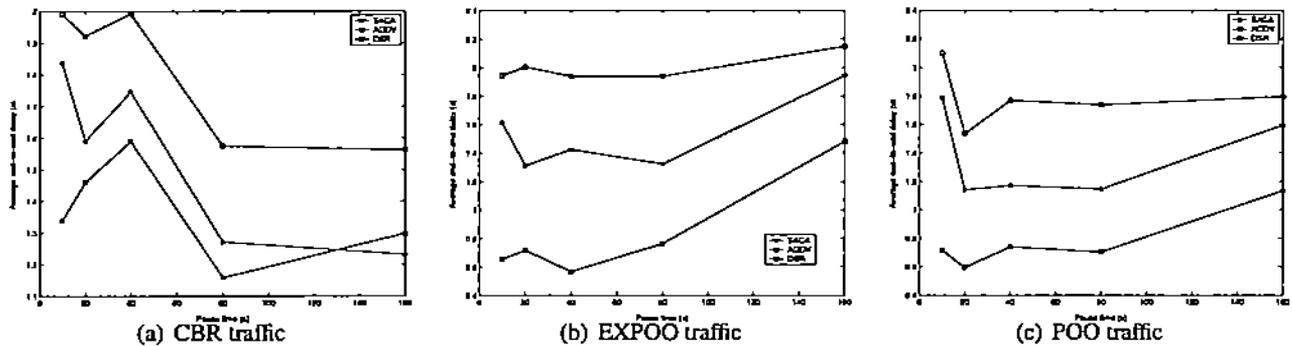


Figure 10: Average end-to-end delay

Figure 10 illustrates that the average end-to-end delay of SACA ranges from 1.2 to 1.9 seconds, which is always less than that of DSR. The reason is that SACA chooses routes based on the intermediate delay instead of hop count. AODV, which also selects routes according to hop count, introduces even lower end-to-end delay. It results from the different techniques used by SACA and AODV to handle broken routes. When a route has broken, SACA buffers the packets to that destination until a new route is established. These buffered packets usually have long delay, because SACA just passively waits for new advertisements. On the other hand, AODV tries to locally repair the route by sending out a route request immediately. If the route cannot be repaired, the packets to that destination are dropped. Packets on a broken route are sent out shortly or dropped. In both cases, they have little impact on the average end-to-end delay.

## 7 Conclusion

Congestion is typically the norm rather than the exception in ad hoc networks. It mainly results from contentions among nodes for the access to the shared wireless media. We propose self-adjusting congestion avoidance (SACA) routing protocol that takes advantage of channel spatial reuse to reduce contention. SACA is a distance vector routing protocol that uses intermediate delay instead of hop count as the distance. Estimating delay at a node using only local information is critical to the protocol. When a node has active traffic, statistical methods are used to evaluate the mean of delay. In case no active traffic exists, we analyze the underlying MAC protocol and apply probability methods to compute the expectation of delay.

Experimental studies are conducted to compare SACA with AODV and DSR using three types of traffic, constant bit rate traffic, burst traffic, and traffic that exhibits long range dependency. SACA is able to deliver more than 80% data packets even under heavy traffic load. It outperforms DSR in all measured performance metrics. SACA is 50% - 60% more efficient than DSR in terms of delivery ratio. It introduces less than half of protocol overhead than DSR does. For constant bit rate traffic, SACA delivers 40% more packets with 2/3 less protocol overhead than AODV does. For the other two types of traffic, SACA introduces more protocol overhead, but its delivery ratio is 1 - 4 times more than AODV's. These results indicate that SACA is a very attractive ad hoc routing protocol. It is especially of benefit in sensor networks where topology changes are much less frequent than traffic changes. More importantly, the approaches we propose for delay estimation can be applied in other research areas, such as quality of service (QoS) applications. The intermediate delay obtained from the routing protocol can be used by TCP to improve the accuracy of round-trip-time (RTT) estimation.

## References

- [1] ANSI/IEEE Std 802.11, 1999 Edition.
- [2] The network simulator - ns-2. <http://www.isi.edu/nsnam/ns/>.
- [3] WaveLAN/PCMCIA card user's guide, Lucent Technologies, October 1996.
- [4] A. A. Bertossi and M. A. Bonuccelli. Code assignment for hidden terminal interference avoidance in multihop packet radio networks. *IEEE/ACM Transactions on Networking (TON)*, 3(4):441-449, 1995.
- [5] R. V. Boppana and S. P. Konduru. An adaptive distance vector routing algorithm for mobile, ad hoc networks. In *Proceedings of IEEE Infocom 2001*, volume 3, pages 1753-1762, Anchorage, Alaska, April 2001. IEEE.
- [6] A. Boukerche, S. K. Das, and A. Fabbri. Message traffic control capabilities of the r-dsdv protocol in mobile ad hoc networks. In *Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 105-112. ACM Press, 2001.
- [7] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. *The fourth annual ACM/IEEE international conference on Mobile computing and networking*, pages 85 - 97, October 1998.
- [8] S. Corson and J. Macker. Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations. *RFC 2002*, January 1999.
- [9] J. J. Garcia-Luna-Aceves and M. Spohn. Transmission-efficient routing in wireless networks using link-state information. *Mobile Networks and Applications*, 6(3):223 - 238, June 2001.

- [10] Z. J. Haas, M. R. Pearlman, and P. Samar. The zone routing protocol (ZRP) for ad hoc networks. *IETF Internet Draft*. <http://www.ietf.org/internet-drafts/draft-ietf-manet-zone-zrp-02.txt>, July 2002.
- [11] J. Jaffe and F. Moss. A responsive distributed routing algorithm for computer networks. *IEEE Transactions on Communications*, 30(7):1758–1762, July 1982.
- [12] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based performance analysis of routing protocols for mobile ad-hoc networks. In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, pages 195–206, Seattle, August 1999. ACM.
- [13] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile Computing*, pages 153–181, 1996.
- [14] S.-J. Lee and M. Gerla. Dynamic load-aware routing in ad hoc networks. In *Proceedings of ICC 2001*, volume 10, pages 3206–3210, 2001.
- [15] Y. Lu, W. Wang, Y. Zhong, and B. Bhargava. Study of distance vector routing protocols for mobile ad hoc networks. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom'2003)*, Texas, March 2003. IEEE.
- [16] Y. Lu, Y. Zhong, and B. Bhargava. Packet loss in mobile ad hoc networks. Technical Report CSD-TR 03-009, Dept. of CS, Purdue University, 2003.
- [17] H. Luo, P. Medvedev, and S. Lu. A self-coordinating approach to distributed fair queueing in ad hoc wireless networks. In *Proceedings of INFOCOM 2001*, volume 3, pages 1370–1379, 2001.
- [18] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc on-demand distance vector (AODV) routing. *IETF Internet Draft*. <http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-13.txt>, February 2003.
- [19] C. E. Perkins and P. 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.
- [20] C. E. Perkins, E. M. Royer, S. R. Das, and M. K. Marine. Performance comparison of two on-demand routing protocols for ad hoc networks. *IEEE Personal Communications*, 8(1):16–28, February 2001.
- [21] S. M. Ross. *Probability Models for Computer Science*. Academic Press, 2002.
- [22] C.-K. Toh. Associativity-based routing for ad-hoc mobile networks. *Wireless Personal Communications Journal*, 4(2):103–139, March 1997.