



SMART: Statistically Multiplexed Adaptive Routing Technique for Ad Hoc Networks *

SARVESH S. KULKARNI

Department of Electrical and Computer Engineering, Villanova University, Villanova, PA 19085, USA

G.R. DATTATREYA **

Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083, USA

Abstract. This paper develops a common solution to the problems of discovery, maintenance, and use of multiple routes in ad hoc networks. The performance criterion is the average time taken by a packet to reach its destination through multiple hops. A source node considers each of its neighbors (reachable by direct wireless transmission) as a next-hop for every possible destination. The effect of delay at a next-hop and beyond, until the packet reaches its destination, is approximately modeled as an equivalent M/M/1 queuing system. Available neighbors at every node provide multiple routes. Multiple routes are statistically multiplexed to distribute the load as well as to deal with changes in data rates and network configuration. The potential of each next-hop neighbor of a node in providing a viable route is estimated on-line and the proportions of traffic routed through these multiple neighbors are also updated adaptively.

We study this approach and conduct extensive experiments over a network with two extreme cases of simulated traffic patterns, the Poisson, and the self-similar types. Even when the network topology is static, our algorithm responds to bursts in the traffic pattern and reduces buffer losses through the use of alternative, less congested routes. We also present simulation experiments and results to demonstrate the effectiveness of our algorithm in the presence of mobility, using self-similar traffic. Mobility is simulated by means of the *random waypoint* model in which nodes move with varying speeds. Results show that our simple unified approach handles the problems of mobility as well as network congestion very well.

Keywords: ad hoc networks, equivalent queuing systems, multihop routing, self-similar traffic, statistical multiplexing, adaptive routing

1. Introduction

1.1. Background

Ad hoc networks are rapidly deployable packet radio networks. The network consists of many transmitting and receiving stations (nodes), each of which is a portable device with its own processing power. All nodes are functionally similar. Each node also acts as a store-and-forward station for routing packets. Packets carrying data as well as control information are transmitted over wireless links. A packet may originate at any node and can be destined for any other node in the network. Nodes are expected to co-operate in delivering packets to the correct destinations regardless of the source. There

are no fixed base stations and there is no central authority to help individual nodes in making routing decisions. Indeed, the nodes do not have a global view of the network topology and must make routing decisions based on local knowledge. All nodes have the same, limited transmission range. Nodes move around and may come within and go out of transmission range of one another. Physical obstructions due to the terrain and weather conditions may result in (intermittent) link failures. Furthermore, since the nodes are expected to be powered by portable batteries, they may fail when their batteries get completely discharged. The network caters to the need of possibly a temporary application in environments with very little control. Numerous applications of such networks are cited in [13].

* Portions of this paper were presented and appear in conference proceedings as follows:

1. S.S. Kulkarni and G.R. Dattatreya, Statistically multiplexed adaptive operation of ad hoc networks with self-similar traffic, in: *Proceedings of the 1999 IEEE Emerging Technologies Symposium on Wireless Communications and Systems*, Session 8, Paper 2, Richardson, TX (April 1999) pp. 1–5.
2. G.R. Dattatreya and S.S. Kulkarni, Simulation of adaptive statistically multiplexed routing in ad hoc networks, in: *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '99)*, Vol. 2, New Orleans, LA (September 1999) pp. 931–935.

** Corresponding author.
E-mail: datta@utdallas.edu

1.2. Current status of routing and mobility management in ad hoc networks

Ramanathan and Steenstrup [27] conduct a survey of routing techniques for mobile communication networks. Das et al. [5] conduct a comparative performance study of routing protocols for mobile ad hoc networks. In general, routing protocols are either table driven or source initiated. Within the table driven category, protocols vary in the amount of routing information stored and how updates are propagated in the network. Destination sequenced distance vector routing (DSDV, [24]) maintains versions of tables containing the number of hops to each destination. Two types of updates are used. A full dump

carries all the routing information. Incremental changes are transmitted through smaller packets. Other table driven techniques use selected nodes (variously known as cluster head, home location register, etc.) to maintain routing information. Due to mobility, cluster heads need to be updated periodically.

In source initiated routing, a source node needing to transmit to a destination starts a route discovery process. The process is completed once a route is found or when all possible route permutations are exhausted. Once a route is found, it is maintained by some form of a route maintenance procedure until either the destination becomes inaccessible or the route is no longer required. Ad hoc on demand distance vector routing (AODV) builds on the earlier mentioned DSDV by minimizing the number of required broadcasts, using them only when required. Johnson and Maltz [12] propose Dynamic Source Routing through the use of flooding, if a destination station is out of the source's transmission range. Temporally Ordered Routing Algorithm (TORA, see [22] for a comparative study) is source initiated, localizes control messages to a very small set of nodes, and provides alternative, multiple routes. It uses externally synchronized clocks. Sharony [28] proposes a two layer addressing protocol. Murthy and Garcia-Luna-Aceves [20] propose a loop-free protocol. Haas [9] and Haas and Pearlman [10] propose and study the performance of the Zone Routing Protocol (ZRP). ZRP is partly proactive and partly reactive. Prakash and Singhal [25], [26] propose the use of dynamic hashing and quorum for efficient location management. Akyildiz, et al. [1], Corson and Ephremides [3], Das et al. [4], Friesleben and Jansen [8], Johnson [11], Lauer [17], Vaidya and co-authors [14,15], Park and Corson [21], Toh [30], [31] are other representative publications in the area of routing and mobility management.

1.3. Motivation for the study

Routing and location management directly affect the functioning of an ad hoc network. In addition, routing affects performance. All other aspects being invariant, efficient routing enhances the overall performance of the network considerably. Routing is also a challenging and difficult task since node mobility necessitates frequent reconfiguration of routes.

Maintenance and use of multiple routes between source-destination pairs is advantageous for two reasons. One, alternative routes are readily available if one route fails. Two, load-balancing can be achieved by distributing the traffic through these multiple routes. Performance enhancement is possible by using a good statistical mix of the available multiple routes. Many researchers concentrate on finding one route between a source node and a destination node. Some approaches do find multiple routes to destinations but not all routes are used at the same time.

The Multipath Distance Vector Algorithm (MDVA, [32]) attempts to perform load-balancing through the use of multiple routing paths. However, it was designed for use in wired networks, not in ad hoc networks. Furthermore, in addition to short term updates, MDVA requires major periodic recomputation of routes. After such a recomputation, traffic must

undergo a major redistribution. Finally, MDVA moves traffic incrementally from the links exhibiting large marginal delays to those with lower marginal delays, to balance the load. In a highly dynamic environment such as that encountered by an ad hoc network, adaptation may be slow. A better approach would be to *compute* the optimal proportions of traffic that must be routed over available links to minimize the end-to-end delay.

The Ad hoc On-demand Multipath Distance Vector algorithm (AOMDV, [19]) maintains multiple routes for efficient fault-tolerance to recover from route breakages. AOMDV utilizes link-disjoint paths. Therefore, a bad link on a path results in avoidance of the entire path, which may have other good links too. Thus, AOMDV may underutilize the available network capacity. In contrast, we would like a routing algorithm to utilize paths that may overlap over the good sections while bypassing the bad sections, resulting in higher utilization. In addition, AOMDV floods control packets, and will therefore have relatively high communication overheads. Last, though AOMDV does find multiple routes, it does not utilize all of them simultaneously. Thus, AOMDV does not attempt to do any sort of load-balancing.

There are some load-sensitive algorithms for ad hoc networks such as the Load-Sensitive Routing protocol (LSR, [34]) and the Dynamic Load-Aware Routing protocol (DLAR, [18]). These consider load-balancing as the primary criterion in route selection. However, they do not use multiple routes at the same time. These considerations provide the motivation for the work presented in this paper.

1.4. Objectives and summary of the approach

In this paper, we propose and study an adaptive stochastic routing algorithm for purposes of route discovery, maintenance, and multiplexed use. The statistical model of traffic, switching and performance are coarse-grained to ensure simplicity. Deviations from our model cause corresponding deviations from optimal operation. However, such deviations are not expected to cause instability or operational problems in the network. For example, in general, two packets generated for a common destination by a source may be routed through different paths. If the packets are part of a stream requiring a virtual circuit such as a TCP connection, the operational part of the network layer itself may insist that they follow the same path. The coarse-grained statistical performance model absorbs these variations and presents some approximately equivalent parameters for the adaptive control algorithm.

First, we develop a common problem for load-balancing and maintenance of efficient multiple routes between source-destination pairs. The next idea is to maintain and update multiple routing information in a local fashion. Information for routing includes estimates of performance figures for various local routing decisions, in addition to estimates of effective control parameters to achieve good performance. The term "effective" is used to denote that at a source (or store-and-forward) node, the actions subsequent to forwarding to

a neighbor are not exactly known, but their overall statistical effect is being estimated and used.

The third idea is to generate and send control packets at regular time intervals, at rates reflecting small proportions of the actual traffic. The advantage of these control packets is that they can explore alternative paths and collect cumulative data for estimation of control parameters for good performance. Control packets are not flooded. Control packets can be lost. Packet losses can be due to a node being out of range, temporarily nonfunctional, or congested. Indeed, all forms of non-receipt of a packet are treated by the same statistical means. The use of such a general approach to deal with packet losses and delays enables us to preserve network functionality and good performance in the presence of mobility. Routing is performed over a flat network without topological hierarchies.

1.5. Applications and limitations of the approach

The routing protocol proposed in this paper targets applications requiring 30–50 nodes in an ad hoc network configuration. Some uses are: (a) disaster relief, (b) search and rescue missions, and (c) localized police and military operations. Such operations require transmission of voice and video as well as large image files (maps). These files get fragmented into smaller packets and result in self-similar traffic [23]. A flurry of activity is typical in such applications and this results in heavy loads. In addition, erratic movement with high mobility of nodes relative to one another is characteristic of such applications. The techniques developed in this paper are applicable in such cases. The scalability of the approach developed here (to networks of size larger than 50 nodes) is not addressed in this paper.

1.6. Organization of the paper

The rest of the paper is organized as follows. Section 2 discusses the assumptions on the structure of the network and the communication organization. Our routing strategy based on local decisions and the load-balancing problem are also introduced. Section 3 works out the solution that combines routing and performance optimization. Techniques to estimate the performance and to update the control parameters are presented. In section 4, details of simulation experiments including traffic generation, initial set up, various experiments and results are described. Section 5 discusses a possible extension to our routing algorithm to implement in-order delivery of data packets. Section 6 lists some general nonnumerical observations drawn from our experience. Section 7 concludes the paper.

2. Routing and load-balancing problem

2.1. The network model

A set of M nodes S_1, \dots, S_M , each with a unique identifier (ID) is spread over a geographical area. Each node S_i is configured for direct (one hop) wireless communication with a

subset of nodes denoted by $N(S_i)$. However, S_i may not be able to transmit to all $S_m \in N(S_i)$ due to possible mobility, congestion, and down time of S_m or even of S_i itself. The set $N(S_i)$ is potentially the set of all nodes in the system, barring S_i itself. Therefore, at a given time, S_i may be capable of directly transmitting to only a small subset of $N(S_i)$. We call this subset $A(S_i)$, the set of active neighbors. The elements of $A(S_i)$ vary with time. Each S_i develops and maintains a list of current $A(S_i)$. This list is updated frequently to keep the information current.

Data packets of varying lengths arrive at (or are generated at) each node for possible relay transmission to specified destinations. Each node is a possible source, destination, as well as a store-and-forward relay station. Packets arrive at source S_i for destination S_j with a mean λ_{ij} . Each node S_i has a mean service rate of μ_i . The arrival process and the service time distributions at each node are discussed in detail in section 4.1. Buffers are provided at each node to accommodate waiting traffic.

Each node S_i is an M/M/1 queuing system. Arrivals from different sources (either generated at S_i itself or forwarded from nodes in $N(S_i)$) merge into one single queue at S_i . The total delay at a node is the sum of queuing, channel access and service delays including the post-collision wait and retransmission times. Any physical link delay is negligible and is included in the service delay. At the source, a generated packet enters the queue. At the destination, the packet enters the queue and is “processed” or “consumed” when it gets into the service mode, and is not transmitted. The overall system is a feed-forward network of single queues. A path from source S_i to destination S_j is a sequence of nodes (with no repetitions) through which relay transmission is feasible, as dictated by existing interconnections.

2.2. The routing strategy

In deterministic routing, each station maintains a routing list of neighboring nodes as a function of possible destination nodes. The size of the list at each node will be $M - 1$ for an M node network. We would like the path for a packet from source S_i to destination S_j to be nondeterministic to balance the flow of traffic among many paths. In general, the number of possible paths between a source–destination pair would be quite large even for a sparsely connected network. In our model of probabilistic routing, let S_n be a node with a packet to be transmitted for an eventual destination S_j . Node S_n picks a node $S_m \in A(S_n)$ with a probability $q_n(jm)$. Thus, each node maintains a probability matrix $Q_n = [q_n(jm)]$ with $M - 1$ rows and k_n columns where k_n is the number of possible neighbors of S_n .

A straightforward probabilistic routing approach described above has the limitation of a packet being routed through a loop. To avoid that, we assume that each node stamps the packets with the ID of the visited nodes, in the order it visits them. An intermediate node forwards a packet to one of its neighbors *not yet* visited by the packet. The probabilities of picking neighbors are modified under the above condition as

follows. Let a packet from source S_i meant for destination S_j reach an intermediate node S_n after visiting the set of nodes denoted by Ω . The probability with which S_n should forward the packet to its neighbor S_m is modified to

$$\frac{q_n(jm)}{1 - \sum_{S_l \in \Omega} q_n(jl)}. \quad (1)$$

The probability values play a crucial role in the routing strategy. At S_n , even though the number of active neighbors of S_n may be several, we expect that only a few of $q_n(jm)$ are reasonable probability values and the rest are negligible, if not actually 0, for a given destination S_j . In this strategy, since a node is visited at most once, it is possible for a packet to get routed in a wrong direction to the point that it gets “cornered” in a wrong node, having visited all the neighbors of such a corner. If a packet gets cornered, its “visited node list” may be reset so it can extricate itself by revisiting a node it had once visited. Some limit on the number of such resets will ensure that packets do not wander around for ever.

The above probabilistic routing attempts to avoid congestion by distributing the traffic and sharing the burden of queuing and forwarding over alternative paths. The use of a simple probability matrix at every node simplifies the task of picking from alternative paths right at the source. Routing decisions become local. Given the set of Q_n , $n = 1, \dots, M$, the actual probability of every path for every source–destination pair can be easily computed. Clearly, λ_{ij} , μ_i , and Q_n determine the overall performance measures, such as the average cumulative delay of a packet, the delay of a packet given its source–destination pair, the average queue lengths at nodes, etc. We assume that for the operating parameters λ_{ij} and μ_i , there exists a solution for the load-balancing problem. This simply means that we have enough resources to handle the overall communication requirements.

2.3. The load-balancing model

The performance optimization problem with the above routing model is the determination of optimal probability matrices at every node. Simple special cases of such problem and corresponding solutions appear in literature on load-balancing in networks of loosely coupled computers [6], [2], [29], and [16]. In the simple example illustrated in figure 1, a traffic stream of unit rate has two alternative paths to a destination and each path is an M/M/1 queue with service rates of 2.25 and 1. Routing 90% of traffic through the faster server and 10% through the slower is optimal.

In a general multihop network, we would need to identify all possible paths for all possible source–destination pairs, use probability variables for different paths for every traffic type (source–destination pair), write a global performance index of “expected delay of a packet”, and determine the optimal probability values for all paths. We would then have to translate these path probabilities into link probabilities, that is, the entries in the Q_n matrices. Clearly, this is a very complicated nonlinear optimization problem with no simple solution algorithm for the general case. The number of parameters to

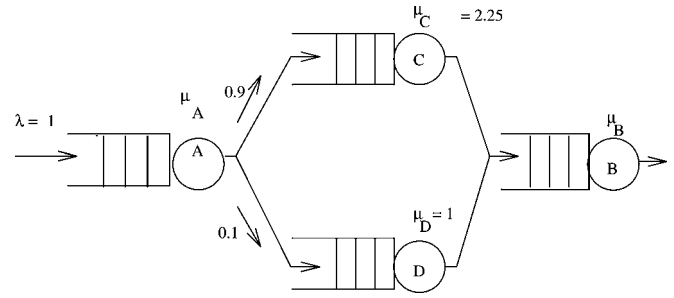


Figure 1. An example of optimization using multiple routes.

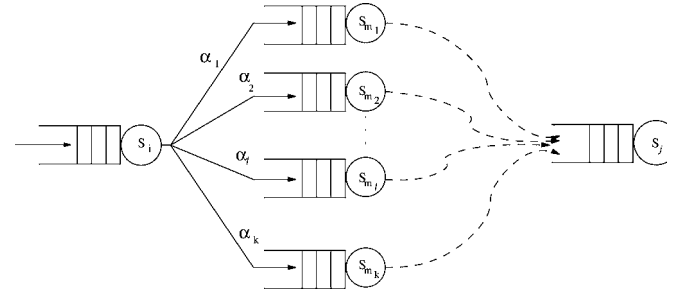


Figure 2. Routing possibilities from S_i to S_j through next hop neighbors.

be identified and solved for is unrealistically large. Besides, such a solution is useful provided the network configuration, all the traffic densities, and the service rates are known in advance and remain constant. In the next section we develop a practical approach and an adaptive operational algorithm that the individual nodes can run in a distributed fashion.

3. Development of solution

3.1. Adaptive estimation of routing probabilities

Let the packets originating at node S_p meant for destination S_j represent a class of traffic D_{pj} . Consider the case wherein a packet in this class D_{pj} is queued at node S_i and must be forwarded to $S_j \notin A(S_i)$ through a neighbor $S_m \in A(S_i)$. That is, S_i must then select an appropriate neighbor S_m through which S_j can be reached, over possibly multiple hops. Figure 2 illustrates the topology as seen by node S_i . Each of the curved lines in figure 2 may have several store-and-forward queues before reaching S_j . Thus, a path from a source node to a destination node is a succession of single queues, in general.

As a simple approximation, we model the effect of the succession of all the queuing systems in a path as some equivalent M/M/1 queuing system with some equivalent, unknown parameters. Thus, if a packet from a traffic stream with the original source node S_n and final destination S_j is at an intermediate node S_i , then S_i considers routing it to one of its active neighbors in $A(S_i)$, to one that has not already been visited by the packet. If all other activities in the network are statistically stationary, these neighbors in $A(S_i)$ offer different M/M/1 choices for the traffic stream, as a crude approximation. By routing appropriate proportions of the traffic to different active neighbors, the load can be balanced to reduce

(minimize, under ideal conditions) the expected delay of such traffic.

Buzen and Chen [2] first studied such an optimization problem. Dattatreya and Venkatesh [6] study the same problem and develop adaptive estimators for the optimum proportions under conditions of unknown traffic and server parameters. Consider the total traffic at S_i with the destination S_j . A portion of this traffic may have originated at other nodes. For the moment, ignore the nodes already visited by this traffic so that we can consider all the neighbors of S_i to route the traffic. Let the current estimate of the arrival rate of such traffic be $\hat{\lambda}$. Let node S_i have $G_l \in A(S_i)$, $l = 1, \dots, k_i$, as potential next-hop neighbors to which it (that is, the node S_i) can forward traffic meant for destination S_j . Initially, $A(S_i) = \{S_1, \dots, S_m\}$, the set of all nodes with which S_i can attempt direct transmission. Control packets with destination S_j sent out by S_i provide an estimate of the delay. This delay estimate is not merely the queuing delay. Rather, it is the aggregate of the queuing delay, the service, the channel access time and the propagation delay at every hop on that path. The advantage of this coarse-grained approach is as follows. As network conditions vary, the queuing delay or the channel access delay at any node may vary tremendously. The effects of these individual variations are incorporated in the aggregate estimated delay along a path. Therefore, the number of operating parameters is kept to a minimum. In our experiments, we do not explicitly simulate the channel access times. However the effect of this delay is accounted for in our simulations as explained in section 3.4.

Let R_l be the estimated delay of control packets routed through G_l . Let the current routing probabilities used by the control packets (as well as by the data packets) be $\hat{\alpha}_l$, $l = 1, \dots, k_i$. Equation (1) describes how to deal with packets that have already visited some nodes. The estimate of the equivalent service rate of the next-hop neighbor G_l evaluates to

$$\frac{1}{\hat{R}_l} + \hat{\alpha}_l \hat{\lambda}. \quad (2)$$

We consider k_i of the neighbors of S_i with the fastest estimated equivalent service rates for routing the traffic. Define

$$\hat{d}_l = \frac{1}{\hat{R}_l \hat{\lambda}}. \quad (3)$$

Then,

$$\hat{\alpha}_l = \frac{1 + \sum_{m=1}^{k_i} \hat{d}_m \hat{d}_l^2}{\sum_{m=1}^{k_i} \hat{d}_m^2} \hat{d}_l^2 - \hat{d}_l \quad (4)$$

is the updated estimate of routing probabilities for each $l = 1, \dots, k_i$, for load-balancing. See [6] for details. However, we stress some important differences to the approach in [6] that we make in this paper. The estimation equation (4) as proposed in [6] was developed for the single hop case. In this paper, by approximating a sequence of queues to a single M/M/1 queue with *some equivalent parameters*, we extend the above load-balancing approach to the multihop scenario. The model in [6] is meant for a single scheduler/dispatcher

to optimize the average delay for a single class of traffic. Our current approach deals with multiple implementations of the algorithm developed here, at each node, and for multiple classes of traffic. Last, the model in [6] was based on Poisson traffic whereas our load-balancing approach handles self-similar traffic too.

The overall approach we propose is for each node to adaptively estimate routing proportions separately for each stream of traffic (based on final destinations of packets).

3.2. The role of control packets

The estimation of parameters required to compute the routing probabilities is accomplished through the gathering of information by control packets. Each node generates and routes control packets for various destinations. The idea is to let the control packet traffic represent a small percentage of actual data traffic generated at a node. However, unlike data traffic that follows a Poisson or self-similar distribution, control traffic is generated periodically. The rate of control packets can be a predetermined percentage of the data traffic rate, but subject to a minimum rate. For a 30 node network, a minimum control packet generation rate of 1 packet per second per generating node is suggested. The statistics of traffic generated at a node can be estimated at the node itself. The required parameters are the rates of traffic and the average delay from the current source to a destination through a next-hop neighbor. The parameters can be estimated using cumulative averages, running averages over a sliding window or exponentially forgetting cumulative averages [16] over the observed values. In our approach, cumulative delays are updated in an exponentially forgetting form. In our experiments, we use a weight of 0.75 for the most recent delay observation and a weight of 0.25 for combined previous estimates. A control packet gets stamped with the ID of every node it visits, in order. Since this information is its only payload in addition to the timestamp inserted by the source, the control packet is usually small in size. When the control packet reaches its intended destination, it reverses its path to reach the source. The source thus obtains the round trip delay experienced by the control packet and uses half the value as the recent delay observation from source to destination. After a round trip from the source to destination and then back to the source, the control packet gets extinct. If an attempted multihop transmission of a control packet from node S_i to destination S_j does not result in S_i receiving the control packet back within a specified time-out period, the control packet is considered lost. In such a case, the source to destination delay value of the lost control packet is set to some predetermined fictitious and high value. We use a value equal to thrice the value of the timeout delay. This is useful for average delay estimation.

3.3. Network initialization and operation

When the network first starts operation or when a node is powered up, only control packets are generated at some predetermined high rates. Control packets are forwarded to all

possible neighbors with equal probability. In other words, when a node starts operating, the route probability matrix at that node has identical entries in every cell, all normalized to 1, for any given destination. As the network operates, possibly with only control packets at the beginning, the effects of various neighbors for various destinations are estimated by a node. Routing probabilities are estimated from such information and updated periodically, using equation (4).

The route and delay information, returned by control packets is used to update the routing tables. The delay information is collected and the route probability matrix is computed and updated every 15 or 30 seconds. After about 4 updates after the network starts operating, the routing tables are ready for use, and data packet traffic starts. Thereafter, at every node, control packets are routed with approximately the same routing policy as that used over packets of real data. Approximately 20% of the control packets generated by a node are routed to its existing neighbors to monitor their status. The remaining 80% are routed to all other nodes in the network. This is done to determine the end-to-end delays to those nodes and to detect their change of status possibly to *active neighbors* later.

3.4. Modeling the MAC effect

In any scheme that relies on channel contention, several nodes compete for channel access. Collisions are inevitable and necessitate retransmissions of packets. Hence the actual throughput of the channel is far below the available capacity. In addition, the channel access times (and throughput) may vary considerably depending on factors such as the number of nodes competing for channel access in a neighborhood and the traffic load at that moment in that neighborhood. Further, variances in load can take the form of bursty arrivals, possibly with large differences in the actual time for successful transmission.

To account for these effects, we model the service time for a packet as a Pareto random variable. The heavy tailed nature of the Pareto distribution ensures that the service times vary considerably from the mean. This allows us a way to model large variations in channel access times and packet sizes.

4. Simulation experiments

4.1. Traffic models for simulation

The Poisson traffic model is traditionally used to represent traffic in studies on conventional telecommunications networks. However, data traffic patterns in present day computer based data networks show high degrees of burstiness. Real life data traffic in packet data networks has been observed to be bursty over *several* time scales [33], [7]. This is in stark contradiction with the Poisson traffic in which the average number of arrivals tends to be more and more predictable as we increase the time interval for averaging. Such bursty traffic is said to exhibit self-similarity. The Poisson and the

self-similar traffic models are, in some sense, two extreme cases of burstiness. We examine the behavior of our adaptive routing approach under these two extreme cases. Two reasons justify the use of the same approach for both cases. One, a single approach that handles a variety of traffic such as pure Poisson, self-similar, MMPP (Markov modulated Poisson process), etc. is clearly advantageous due to its flexibility. Two, studies of queues with self-similar traffic (see [7]) show that the performance curves (delay and buffer occupancy) of queues with self-similar traffic saturate at much smaller values of the normalized load, in comparison with the results for queues with Poisson traffic. However, the shapes of the curves are similar; the curves are concave upwards and rise steeply.

The Poisson traffic model is well understood. The interarrival times as well as the service times are independent and identically distributed exponential random variables. The self-similar traffic model is based on a Pareto random variable [7], [33]. The data (for possibly multihop transmission) at a station is considered to be the result of multiplexing numerous data sub-sources. Each such data sub-source has intervals of *ON* and *OFF* time periods. A packet of data (contiguous sequence of bits) flows into the multiplexer during an *ON* period. The *ON* as well the *OFF* time periods are modeled as independent identically distributed (iid) Pareto random variables. In numerous different observations of real life data traffic, the underlying Hurst parameter of the Pareto distribution has been very close to 0.9 (see [7], [33]). Therefore, X , each of the *ON* and the *OFF* time periods has the following probability density function:

$$f_X(x) = \begin{cases} 1.2x^{-2.2}, & x > 1, \\ 0, & x < 1. \end{cases}$$

The numerical constants 1.2 and 2.2 are the consequences of the choice of 0.9 for the Hurst parameter, so as to approximate Ethernet traffic characteristics reported in [7] and [33]. The mean of X , $E[X] = 6$ and X has an unbounded variance.

Once a packet is stored and transmitted, the *ON* time of a packet at the output of the transmitter is proportional to the *ON* time of the input packet and the transmitter speed. Therefore, equivalently, a sequence of packets in a data sub-source is a sequence of arrival time instants and *equivalent* packet lengths. Equivalent packet lengths correspond to the overall transmission time which includes channel access and retransmissions. Each inter-arrival time is the sum of two iid Pareto random variables. Each packet length corresponds to one of the above two random numbers. Thus, if X_1 is the *ON* time and X_2 is the *OFF* time, the inter-arrival time is $X_1 + X_2$ and the packet length is X_1 . Inter-arrival times can be scaled with a factor a (without changing the Hurst parameter) to generate data packets of any desired rate. In our simulation experiments, we multiplexed 100 such data sub-sources to yield one external data source at every node in the network. The resulting external data input rate at a node is $100/(12a)$ packets per unit time. The packet size can also be similarly scaled without changing the Hurst parameter. Therefore, if a packet of size X_1 data units is stored (X_1 is a random variable as above) and transmitted with a speed of μ data units per unit time, the ex-

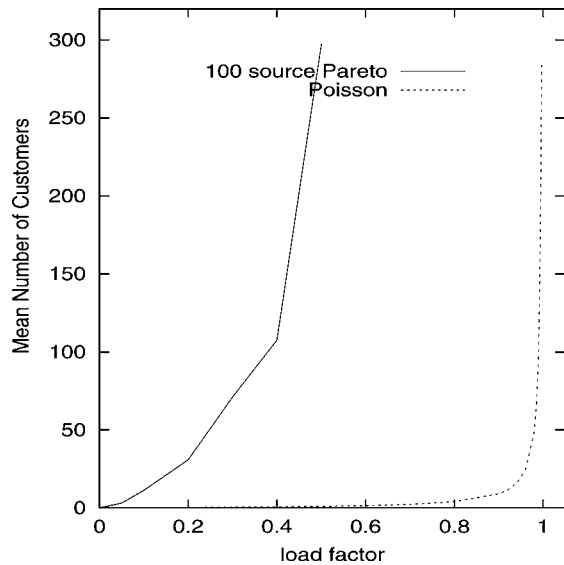


Figure 3. Buffer occupancy vs. load for a single queue.

pected packet *ON* time at the output of the transmitter will be $6/\mu$ time units. A data unit can be, for example, one kilobit.

We simulated the behavior of a single queue using Poisson as well as 100-source multiplexed Pareto (self-similar) traffic. The resulting load curves (refer to figure 3) clearly illustrate that whereas Poisson traffic saturates the queue at around 90% of the offered load, the 100-source multiplexed Pareto traffic saturates the queue at barely 50–60% of the load. In various simulation runs, we found that the steepness of the self-similar curve varied significantly so that saturation occurred at anywhere between 40–80% of the load. This behavior is a consequence of the very bursty nature of the self-similar traffic.

4.2. Network for fixed topology experiments

We chose a randomly generated 30 node graph shown in figure 4 to conduct simulation experiments with a fixed topology. This topology provides several multiple routes to demonstrate route multiplexing. It is well connected (but not strongly connected). At the start of simulation, only control packets are generated (in our experiment, with half the expected average per-node traffic) with equally likely destinations and routed equally likely to all possible neighbors. After four updates, 20% of control packets are routed to the three best neighbors and the remaining 80% are routed to other nodes in the network, to monitor possible changes in their status. At this time, the network is also open for transmission of data packets. The control packet generation rate is then modified and maintained at a predetermined fractional level of the data traffic rate. For the purpose of simulation, the distribution and mean of the control packet size were kept the same as those for data packets. However, in real life, since the control packets are relatively light-weight, the actual byte overhead will be considerably lower.

The routing probability estimates are updated regularly at intervals of 30 seconds. The updates of routing probabilities

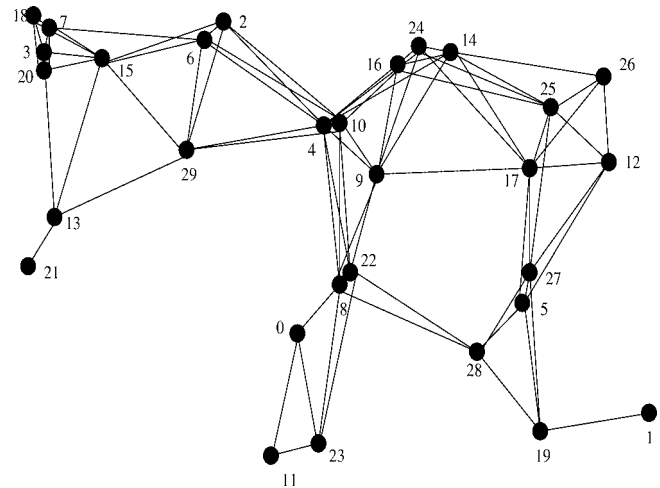


Figure 4. Experimental network for static topology.

are based on the cumulative average delay information collected by all control packets sent out by node S_i to destination $S_j, \forall j \neq i$. If a control or data packet reaches node m and has already visited all of node m 's neighbors, it is said to be cornered, as mentioned earlier. In the present experiment, such packets are counted as lost (to estimate the fraction of packets that are so lost). Both control and data packets are dropped if they encounter a full buffer. Data packets are treated as datagrams and are not acknowledged at this layer of network operation.

Each node has a buffer size of 150 packets and generates traffic for random destinations. The time-out value for control packets is set to 5 seconds. The particular network topology used in our experiments results in an average hop-count of approximately 4 for data packets. The service rate μ at each node is kept at 20 kbps. We model this service rate to take into account the raw transmission rate as well as the MAC inefficiencies due to collisions and retransmissions. This is explained in section 3.4.

4.3. Experiments with fixed topology

We first simulated the operation of the example network of figure 4 using our adaptive algorithm at different loads by varying the rate of generated traffic. In particular, we ran the network with average network-wide traffic loads of 25, 40, 50, 65, 80 and 90% with both Poisson as well as self-similar traffic. Of course, the actual measured load at an individual node varies considerably from the mean depending on that node's interconnection to the rest of the network. Additional variations in the short term mean are imposed by the traffic pattern itself. The above load is the combined effect of the data packet as well as the control packet traffic. The rate of generation of control traffic is kept at 10% of the data traffic generation rate to limit the overhead due to control traffic. However, this rate is subject to a lower bound of 1 control packet per second at each node. The lower bound ensures that enough control packets are generated so that changes in topology or traffic are detected in a timely fashion.

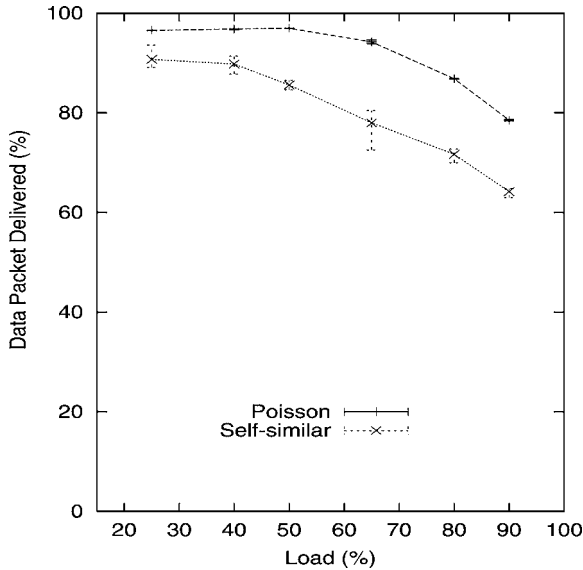


Figure 5. Percentage of data packets delivered vs. load.

For each load, we operated the network for 5 hours. The time of operation of the network is the real time period over which a real network would function. Simulation time is the computer time to run the simulation for a specified network operation time. The plotted results are the averages over 10 separate runs for each load. The variation of results over the 10 separate runs are indicated in the figure as an interval around each plotted average. These intervals are also called the error bars. The network topology was fixed throughout the duration of the simulation so that the effects of only the traffic variations could be observed. The results are illustrated in figure 5. When the network is run with Poisson traffic with the nominal load at 50% or less, the buffer losses are negligible. A small fraction of the generated data packets (about 2%) are lost due to cornering. However, at loads of 65% and above, even Poisson traffic causes the network to lose a significant fraction of data packets. Under similar load conditions, the network fares worse when the traffic is self-similar in nature. The extreme burstiness in the traffic causes the system to lose approximately 7–8% of the packets even at loads of 40% and less. At higher loads, the decline in the fraction of successfully delivered packets is even more pronounced. If we assume that a 10–15% packet loss in the buffers constitutes severe congestion, then we find that Poisson traffic brings on congestion at about 80% of the network-wide load. On the other hand, self-similar traffic induces such severe congestion when the network load is at just 50%.

We tested our adaptive algorithm to determine its effectiveness in countering congestion. In this experiment too, the network topology chosen was the same as before and it was fixed for the duration of the network operation. First, we ran the adaptive algorithm on the fixed network for 50,000 seconds (i.e. over 13 hours) of operation time. In the next experiment, we ran the adaptive algorithm with identical data for about 300 seconds (20 updates of the routing tables) so as to enable initial route discovery and ensure complete stabilization of existing routes. After 300 seconds, no more updates

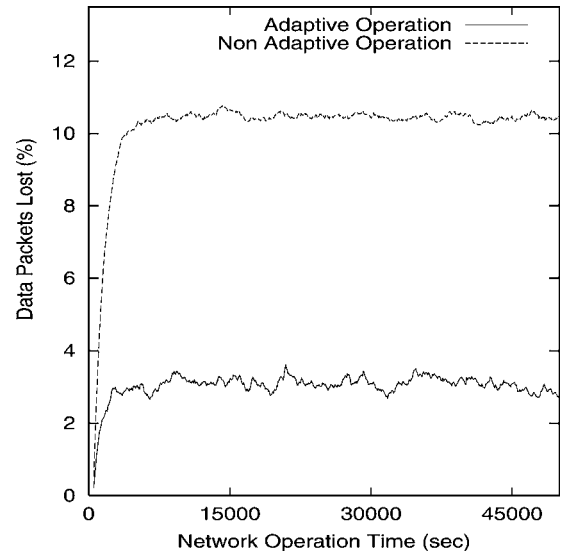


Figure 6. Percentage of data packet loss vs. time, Poisson traffic.

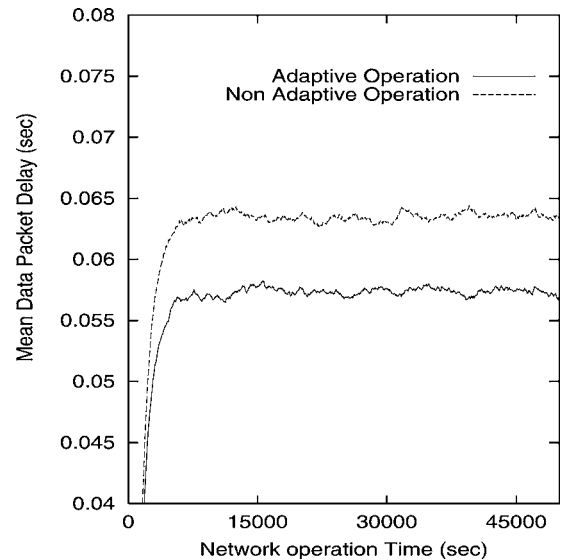


Figure 7. Mean data packet delay vs. time, Poisson traffic.

were made to the routing tables, rendering the routing *non-adaptive*. We performed these experiments with both Poisson and self-similar traffic under varying loads.

We found that the routing probabilities reach stable values within 4–6 updates even for higher than moderate congestion. For low congestion loads, the routing probabilities converge to 0 or 1 indicating that the adaptive algorithm finds and maintains more or less unique routes. For higher loads, we see very good splits in probability values indicating good statistical multiplexing. In other words, when the load increases, a greater number of available alternative routes are used actively.

We illustrate the differences in the adaptive and nonadaptive operation in figures 6–11. For these illustrations, the network load was 50%. The packet loss percentages and packet delays are shown as moving averages across the duration of the network operation.

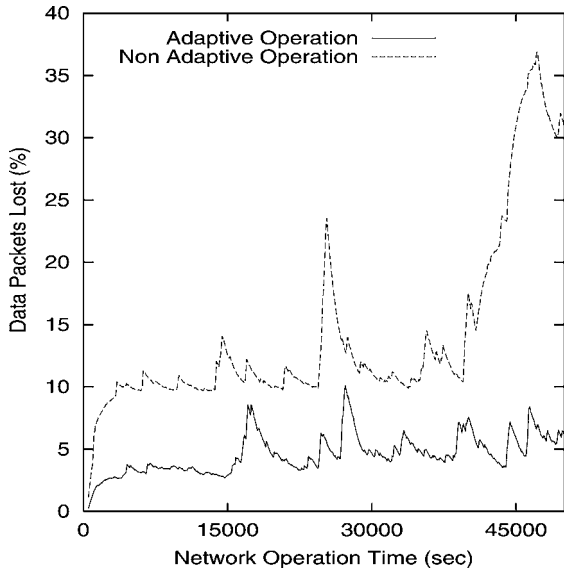


Figure 8. Percentage of data packet loss vs. time, self-similar traffic.

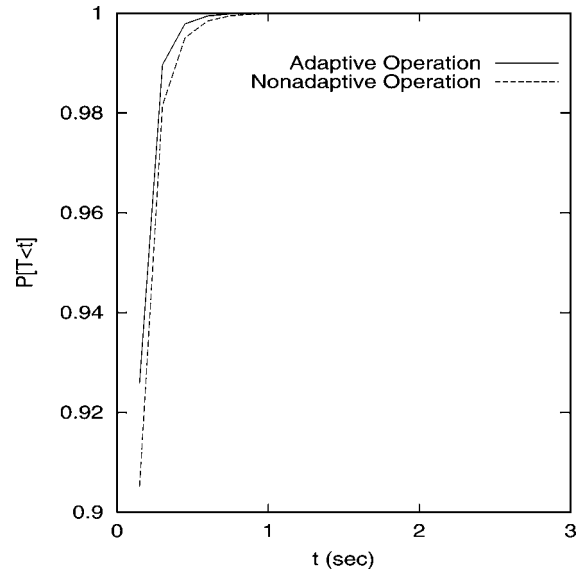


Figure 10. CDF of data packet delay, poisson traffic.

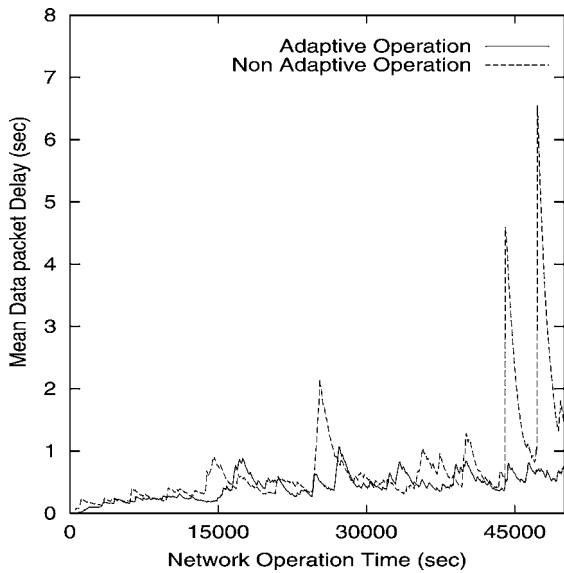


Figure 9. Mean data packet delay vs. time, self-similar traffic.

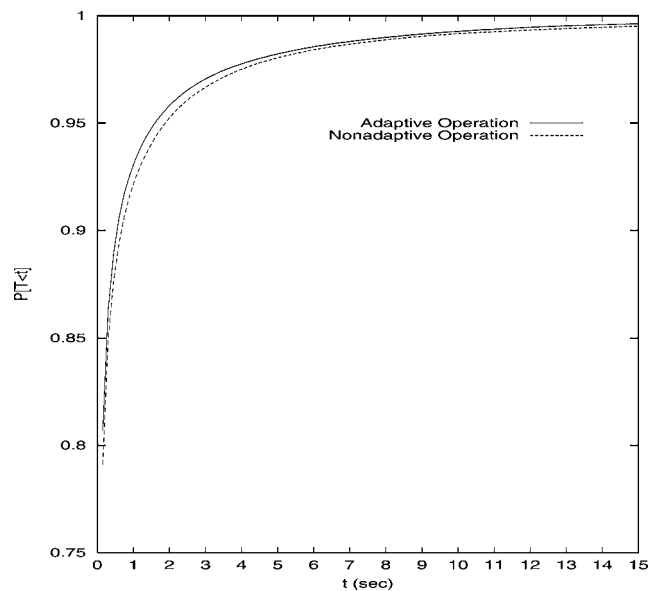


Figure 11. CDF of data packet delay, self-similar traffic.

From figures 6 and 8, it is apparent that adaptive operation results in far fewer congestive losses than nonadaptive operation regardless of the traffic pattern. In particular, in the Poisson traffic case, packet losses under adaptive operation are less than a third of those under nonadaptive operation. In the self-similar case, the difference is more pronounced. At some point of time, the packet losses go up to 35% under nonadaptive operation, whereas the adaptive operation manages to limit the losses to under 10%. A similar trend is seen in the mean delay experienced by data packets, in figures 7 and 9, with the adaptive operation fairing much better. In particular, for self-similar traffic, the adaptivity enables suppression of the spikes in delays experienced by packets when packet bursts occur. This is possible because, when a burst of packets comes along and congests routes, the algorithm routes traffic along alternative paths avoiding the congested

hot spots. This effectively curtails the packet losses and delay spikes. An examination of the probability matrices at the different nodes at various times during the network operation verifies this inference.

Figures 10 and 11 illustrate the Cumulative Distribution Function (CDF) of the packet delays for Poisson and self-similar traffic, respectively. The adaptive operation ensures that a larger proportion of packets experience lower delays under both types of traffic. However, the difference in scales is noteworthy. Whereas all successfully delivered data packets in the Poisson case are delivered in under 2 seconds, in the self-similar case, even after 15 seconds, not all packets have been delivered. Intuitively, this is a consequence of the higher buffer occupancy of self-similar queues under similar loads.

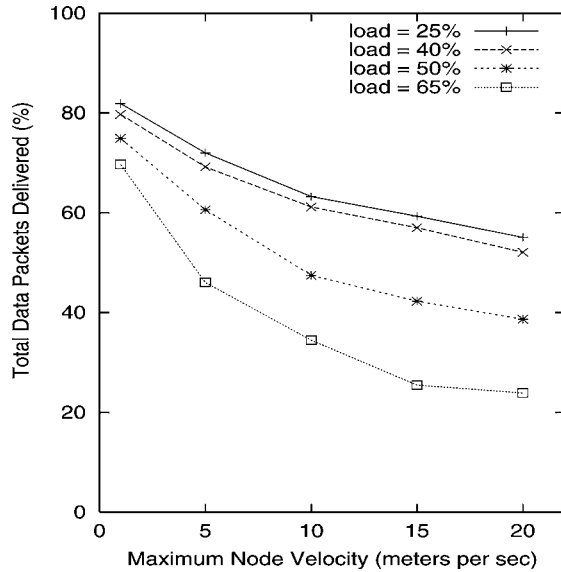


Figure 12. Percentage of data packets delivered vs. max. node velocity, self-similar traffic.

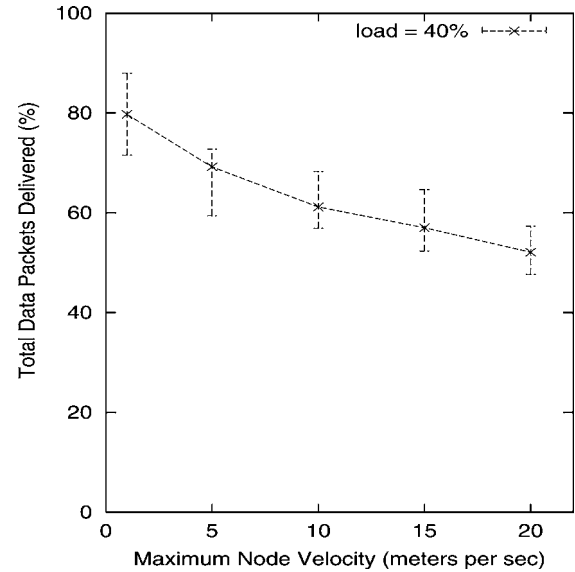


Figure 14. Error bars for results in figure 12 for 40% load.

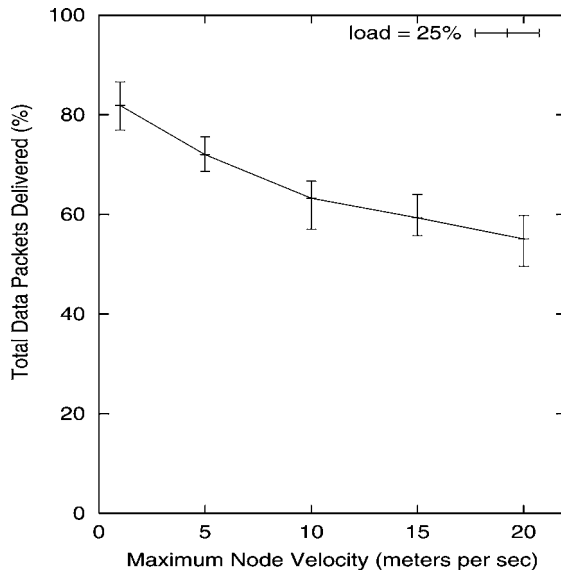


Figure 13. Error bars for results in figure 12 for 25% load.

In conclusion, in our experiments with fixed topology we see that adaptive routing is clearly beneficial.

4.4. Mobility experiments

We conducted mobility experiments to test the efficacy of our algorithm under varying degrees of mobility and load. For our experiments on mobility, we used the *random waypoint* model [12], explained as follows.

We place 30 nodes at random in a 1.5 km by 2.5 km rectangular area. After the network starts operating, each node selects a destination point within the rectangular area, at random. Next, it selects a velocity uniformly distributed between 1 m/s and some maximum value and proceeds to the destination point with that velocity. Once it reaches the destination, it repeats this process by selecting a new destination point and

velocity. Nodes do not pause once they reach their destination. Therefore, all the nodes in our mobility model are in constant motion with varying velocities.

We set the transmission range of each node to 600 meters. This ensures that the network has a reasonable chance of remaining connected. However, it is possible for the network to get partitioned and that does occur several times during a given simulation run. Whenever a packet is transmitted from node S_i to node S_j , the current positions of S_i and S_j are computed to determine whether they are within transmission range of each other at that moment. If they are within range of each other, the packet is forwarded to S_j . If not, the packet is lost in transit due to a non-existent link.

In our mobility experiments, we set the time-out value for control packets to 3 seconds. Route updates are made every 15 seconds. The transmission range, the service rate and the buffer size at each node are kept unchanged from the earlier experiments with fixed topology.

Figure 12 shows the percentage of successful data packet deliveries as a function of the maximum node velocity and load. The traffic used was self-similar. Each point on the plot represents the average of 10 simulation runs, each for 5 hours of network operation time. All operating parameters were kept the same for a set of 10 simulation runs. Figure 12 shows the plots for 4 different loads for comparison. Figures 13–16 redraw one plot each from figure 12, for a specified load, along with the corresponding error bars. The interval between the error bars around a plotted point contains the variation of results over 10 separate runs. The error bars overlap in some cases for the four different loads. Therefore, the results are shown in five different figures. The average number of hops needed by a packet to get to its destination was observed to be approximately 4, whereas the length of the longest path taken was 12 hops.

From figure 12, we notice that at low to moderate congestion levels (network load of 25 and 40%, respectively), the algorithm delivers over 80% of the packets, at speeds of

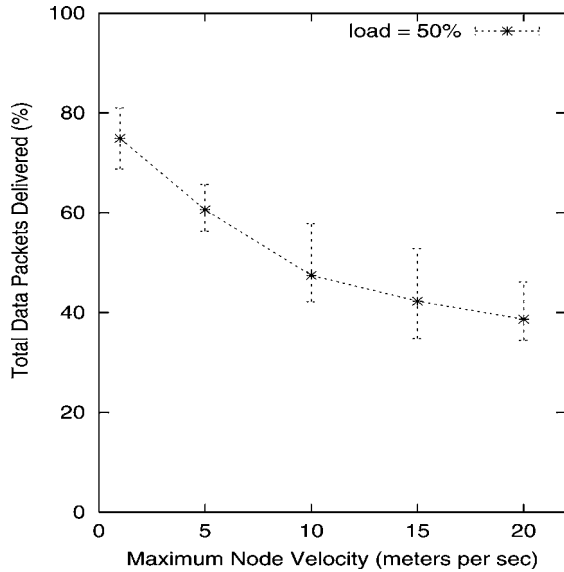


Figure 15. Error bars for results in figure 12 for 50% load.

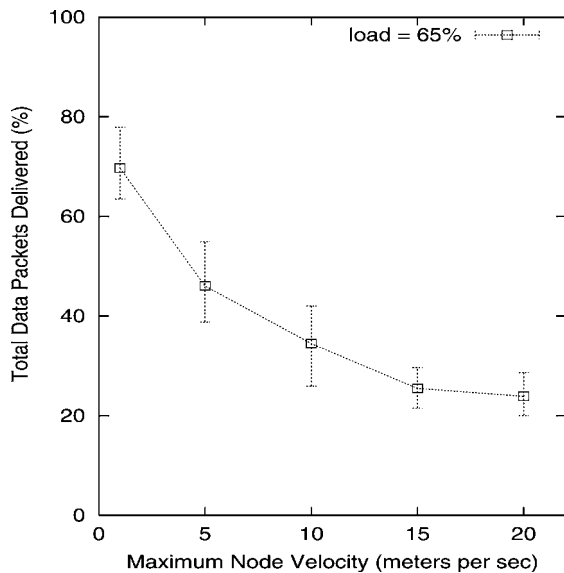


Figure 16. Error bars for results in figure 12 for 65% load.

1 m/s. At higher node velocities, the ratio of successful deliveries falls off, but the algorithm still delivers close to 60% of the packets. At higher loads (50%, 65%), the effects of route congestion are more pronounced and there is a significant drop-off in the ratio of successful deliveries of the data packets. At these loads, most of the nodes are congested to some extent and the adaptive algorithm is not as effective in alleviating congestion, since alternative, less congested routes are not available. Hence it is clear that the network load plays a crucial part in effective throughput of data packets, even when the loads are as low as 50–60%, for self-similar traffic.

Cornering losses are 2–3% at maximum speeds of 1–5 m/s. At higher speeds, they rise to 3–8%. The rise of cornering losses at higher speeds is explained as follows. At higher speeds, routes change even as the packet is in transit. Consequently, a packet may find itself at an intermediate node

with all possible routes to the destination broken. Since the packet cannot go back to the node that forwarded it to the current node, it has nowhere to go. Hence it is “cornered”, and is therefore dropped. Obviously, with higher node speeds, cornering losses due to such a phenomenon will increase.

There is one other cause in addition to cornering and full buffers that results in packet losses. Sometimes, a packet may be (incorrectly) forwarded to a node that was a neighbor but which may have moved out of transmission range. This results in additional losses that increase as the node velocities increase.

In our experiments, we noticed that between 5–10% of the data packets generated for transmission are lost due to network partitioning that occurs during any given experiment. Had partitioning not occurred, the data packet delivery ratios reported in figure 12 would have been higher by that amount.

5. In-order delivery of packets

In this paper, we have described and implemented our algorithm that routes on a per-packet basis thereby fully utilizing alternative routes to destinations wherever available. Therefore, a significant number of packets may arrive out of order. However, if in-order delivery is essential as in the case of TCP, the routing algorithm can still function effectively by maintaining very little additional information at each node as follows.

We can treat traffic along each TCP source–destination connection as a single traffic class. When a node receives a packet belonging to a new traffic class, it selects the next-hop neighbor in the usual fashion. In addition, it also makes an entry in a secondary table recording the next hop for all subsequent packets belonging to the same class. Thereafter, all packets belonging to that class are routed through the same next hop by consulting this secondary table. Assuming that there are n active TCP connections at a node, the node needs to maintain just $O(n)$ entries in its secondary table. Then, the different TCP flows to the same destination can be routed differently, even while preserving the packet ordering within each flow. Simultaneously, UDP traffic can continue to use the primary routing table on a per-packet basis. In this way, both UDP and TCP traffic can utilize multiple routes whenever available.

6. Discussion

The following are important observations and inferences drawn from the application of the adaptive routing algorithm in simulation experiments.

- In multihop networks, the effect of self-similar traffic is similar to that of traditional traffic, but with vastly different parameters. We approximate the effect of a sequence of queues in a multihop path by a single store-and-forward model to deal with different types of traffic, for the purpose of load-balancing in ad hoc networks.

- The routing algorithm (SMART) presented in this paper is proactive in its operation. Consequently, it is particularly suited for use in environments with relatively heavy and bursty traffic, and where the network nodes exhibit a high degree of mobility.
- The general shapes of the profiles of performance criteria such as the delay (as functions of the load) carry over from a single store-and-forward queue to a sequence of such queues in a network. Of course, the parameters for the performance profiles for a sequence of queues will be vastly different from the corresponding ones for a single queue. This helps to concisely represent the cumulative effects of routing packets to a remote destination, as follows. For each class of traffic incident at a node S meant for a destination D , the effects of routing the traffic through the different outgoing links from S should be stored in a table at S .
- Statistical multiplexing of traffic over multiple paths has many attractive features.
 - Alternative paths are always available to circumvent congested nodes.
 - Better utilization of resources and hence higher overall performance.
 - Ordered delivery of a stream of packets can be offered; in other words, it is possible to selectively disable statistical multiplexing if such a service is requested. The traffic pattern is not much different since it is already assumed to be extremely bursty.
 - Different classes of traffic may naturally tend to follow different paths depending on the capacities of different nodes/subnets.
- Load-balancing can be achieved by optimal statistical multiplexing of traffic over available alternative paths, at every node. Different weights (for example, based on different pricing) for the performance of different classes can be handled.
- The available options for multiplexing can get pruned in order to satisfy all the constraints of the traffic class, capacities of nodes links, etc. The adaptive algorithm balances the load over and above these constraints.

7. Concluding remarks

In ideal multihop packet switched networks, minimizing the expected delay of a packet from source to destination intrinsically solves the problems of network operation such as finding the best routes and eliminating congestion. Such a static optimization problem is very complex even when all the system parameters are known in advance. In practical ad hoc networks, very little information is available to take advantage of such an ideal approach. Each node requires information about only its currently active neighboring nodes which are in its direct transmission range. The rates of different classes of traffic incident to a node can be estimated. A node can

send out control packets with various destinations, route them through its neighbors, and expect that other nodes will function similarly. We use this principle and evaluate the potential of each neighbor for each class of traffic by measuring the delay experienced by control packets routed through different neighbors. This approach is used adaptively and the proportions of a class of traffic to be routed through different neighbors is maintained and updated frequently. This provides statistically multiplexed multiple routes for each class of traffic. The store-and-forward activity at each node is modeled as an M/M/1 queue. The effect of a statistical mix of a sequence of such M/M/1 queues is also approximated to be an M/M/1 queuing system. The advantage of such an approximation is that the average delay at and beyond a neighbor is the only statistic required to update the proportions of traffic to be routed to the different neighbors. The model is statistical and coarse grained; as such, any other operational requirement can be imposed without creating logical conflicts.

The approach is illustrated with simulation experiments. Simulation results indicate that the algorithm can be used effectively for the two extreme cases of pure Poisson and fully bursty self-similar traffic. Even when the network topology is static, the bursty nature of self-similar traffic introduces congestion points in the network. However, the algorithm reduces packet losses and mean packet delays by routing traffic over less congested routes. Hence, it can deal with bursts in data rates.

Mobility experiments clearly demonstrate that our algorithm can adapt effectively to the mobile environment, under different degrees of mobility. Furthermore, the experiments illustrate the adverse effects of the bursty nature of self-similar traffic and the extent of congestion in a mobile environment, even at relatively low network loads. Our algorithm handles the dual problems of mobility and congestion well using a unified approach.

References

- [1] I.F. Akyildiz, J.S.M. Ho and Y.-B. Lin, Movement based location update and selective paging schemes, *IEEE/ACM Trans. Networking* 4 (August 1996) 629–638.
- [2] J.P. Buzen and P.P.S. Chen, Optimal load balancing in memory hierarchies, in: *Proc. of Information Processing '74* (North-Holland, Amsterdam, 1974) pp. 271–275.
- [3] M.S. Corson and A. Ephremides, A distributed routing algorithm for mobile wireless networks, *ACM J. Wireless Networks* 1 (February 1995) 61–81.
- [4] B. Das, E. Sivakumar and V. Bhargavan, Routing in ad-hoc networks using a spine, in: *Proc. of IEEE Intl. Conf. on Computer Comm. and Networks* (1997) pp. 34–39.
- [5] S.R. Das, C. Robert, Y. Jiangtao and S. Rimli, Comparative performance evaluation of routing protocols for mobile ad hoc networks, in: *Proc. of IEEE Seventh Intl. Conf. on Computer Communications and Networks (IC3N '98)* (October 1998) pp. 153–161.
- [6] G.R. Dattatreya and R. Venkatesh, Adaptive performance optimization of loosely coupled processors, *IEEE Trans. Syst. Man Cybern.* 21 (May/June 1991) 607–619.
- [7] A. Erramilli, O. Narayan and W. Willinger, Experimental queueing analysis with long-range dependent packet traffic, *IEEE/ACM Trans. Networking* 4 (April 1996) 209–223.

- [8] B. Friesleben and R. Jansen, Analysis of routing protocols for ad hoc networks of mobile computers, in: *Proc. of 15th IASTED Intl. Conf. on Applied Informatics*, Innsbruck, Austria (February 1997) pp. 133–136.
- [9] Z.J. Haas, A new routing protocol for the reconfigurable wireless network, in: *Proc. of IEEE 6th Intl. Conf. on Universal Personal Comm. (IUPUC 1997)*, San Diego, CA (October 1997) pp. 562–566.
- [10] Z.J. Haas and M.R. Pearlman, Determining the optimal configuration for the zone routing protocol, *IEEE J. Selected Areas Commun.* 17 (August 1999) 1395–1414.
- [11] D.B. Johnson, Routing in ad hoc networks of mobile hosts, in: *Proceedings of Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA (December 1994) pp. 1–6.
- [12] D.B. Johnson and D.A. Maltz, Dynamic source routing in ad hoc wireless networks, in: *Mobile Computing*, eds. T. Imielinsky and H. Korth (Kluwer Academic, 1996) pp. 153–181.
- [13] L. Kleinrock, Nomadic computing – An opportunity, *Computer Communication Review* 25 (1995) 36–40.
- [14] Y. Ko and N.H. Vaidya, Location-Aided Routing (LAR) in mobile ad hoc networks, in: *Proc. of MOBICOM '98*, Dallas, TX (October 1998) pp. 66–75.
- [15] P. Krishna, M. Chatterjee, N.H. Vaidya and D.K. Pradhan, A cluster-based approach for routing in ad hoc networks, in: *Proc. of USENIX Symposium on Location Independent and Mobile Computing* (April 1995).
- [16] A. Kumar, Adaptive load control of the central processor in a distributed system with a star topology, *IEEE Trans. Computers* 38 (November 1989) 1502–1512.
- [17] G. Lauer, Packet radio routing, in: *Routing in Communication Networks*, ed. M. Steenstrup (Prentice-Hall, 1995) pp. 351–396.
- [18] S.-J. Lee and M. Gerla, Dynamic load-aware routing in ad hoc networks, in: *Proceedings of the IEEE International Conference on Communications (ICC)*, Helsinki, Finland (June 2001) pp. 3206–3210.
- [19] M.K. Marina and S.R. Das, On-demand multipath distance vector routing for ad hoc networks, in: *Proc. of IEEE ICNP*, Riverside, CA (November 2001) pp. 14–23.
- [20] S. Murthy and J.J. Garcia-Luna-Aceves, An efficient routing protocol for wireless networks, *Mobile Networks Appl.* 1 (1996) 183–197.
- [21] V.D. Park and M.S. Corson, A highly adaptive distributed routing algorithm for mobile wireless networks, in: *Proc. of INFOCOM 1997* (1997) pp. 1405–1413.
- [22] V.D. Park and M.S. Corson, A performance comparison of TORA and Ideal Link State routing, in: *Proc. of IEEE Symposium on Computers and Communication (ISCC '98)* (June 1998).
- [23] K. Park and W. Willinger (eds.), *Self-Similar Network Traffic and Performance Evaluation* (Wiley, New York, 2000).
- [24] C.E. Perkins and P. Bhagwat, Highly dynamic Destination Sequenced Distance-Vector routing (DSDV) for mobile computers, in: *Proc. of ACM SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications* (August 1994) pp. 234–244.
- [25] R. Prakash and M. Singhal, Dynamic hashing + quorum = efficient location management for mobile computing systems, in: *Proc. of ACM Symposium on Principles of Distributed Computing (PODC)*, Santa Barbara, CA (August 1997) p. 291.
- [26] R. Prakash and M. Singhal, Impact of unidirectional links on wireless ad-hoc networks, in: *Proc. of DIMACS Workshop on Mobile Networking and Computing*, Rutgers University (March 1999).
- [27] S. Ramanathan and M. Steenstrup, A survey of routing techniques for mobile communications networks, *Mobile Networks Appl.* 1 (1996) 89–104.
- [28] J. Sharony, A mobile radio network architecture with dynamically changing topology using virtual subnets, in: *Proc. of ICC/SUPERCOM*, Dallas, TX (June 1996) pp. 807–812.
- [29] A.N. Tantawi and D. Towsley, A general model for optimal static load balancing in star network configurations, in: *Proc. of Performance '84*, ed. E. Gelembé (North-Holland, New York, 1984).
- [30] C.-K. Toh, *Wireless ATM and Ad Hoc Networks* (Kluwer Academic, 1997).
- [31] C.-K. Toh, A novel distributed routing protocol to support ad-hoc mobile computing, in: *Proc. of IEEE 15th Annual Intl. Phoenix Conf. on Comm.* (March 1996) pp. 480–486.
- [32] S. Vutukury and J.J. Garcia-Luna-Aceves, MDVA: A distance-vector multipath routing protocol, in: *Proc. of IEEE INFOCOM 2001*, Vol. 1, Anchorage, Alaska (April 2001) pp. 557–564.
- [33] W. Willinger, M.S. Taqqu, R. Sherman and D.V. Wilson, Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level, *IEEE/ACM Trans. Networking* 5(1) (1997) 71–76.
- [34] K. Wu and J. Harms, Load-sensitive routing for mobile ad hoc networks, in: *Proceedings of the 10th IEEE International Conference on Computer Communications and Networks (ICCCN)*, Phoenix, AZ (October 2001) pp. 540–546.



Sarvesh S. Kulkarni received the Bachelor's degree in computer engineering from the University of Bombay in 1994. From 1994 to 1996, he worked as Customer Support Engineer with Datamatics Ltd., and as Software Engineer with Mahindra-British Telecom Ltd. in Bombay, India. He received the Master's degree and the Doctor of Philosophy degree in Computer Science from the University of Texas at Dallas in 1998 and 2002, respectively. From January 2000 through August 2000, he also worked as student employee at Texas Instruments Inc., Dallas, TX in their Application Specific Products – Electronic Design Automation group. He is currently an Assistant Professor in the Department of Electrical and Computer Engineering, at Villanova University, Pennsylvania. His research interests include wired and wireless networks, design of network communication protocols, network traffic modeling and performance analysis of computer networks.

E-mail: sarvesh.kulkarni@villanova.edu

G.R. Dattatreya received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Madras, M.E. in electrical communication engineering, and Ph.D. from the Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India in 1975, 1977, and 1981, respectively. During 1981–1982, he was a Senior Scientist at the Scientific Analysis Group, Delhi, India, and worked on Pattern Recognition and Speech Processing problems. During 1983–1985, he was a Visiting Assistant Professor at the Machine Intelligence and Pattern Analysis Laboratory, Department of Computer Science, University of Maryland, College Park, where he taught and conducted research in Information Processing. He is currently an Associate Professor in the Department of Computer Science, University of Texas at Dallas. During June–December 1996, he was a consultant on the Malaysia Polytechnic Project, Batu Pahat, Johor, Malaysia. During June 1999–May 2000, he was a Visiting Professor at the Center for Artificial Intelligence, ITESM, Monterrey, Mexico. His current research interests are Stochastic Modeling, Parameter Estimation, and Adaptive Optimization in Communication, Signal Processing, and Computer Network Systems.

E-mail: datta@utdallas.edu