A System and Traffic Dependent Adaptive Routing Algorithm for Ad Hoc Networks[1]

Piyush Gupta[2] and P. R. Kumar[3]

Abstract

An *ad hoc* network consists of a number of mobile hosts who communicate with each other over a wireless channel without any centralized control. The basic problem is to obtain a distributed routing scheme so that under the *network connectivity* assumption any mobile host can transmit/receive data from any other host in the network. In this paper we propose a new routing algorithm for ad hoc networks. The proposed algorithm uses a more appropriate distance measure given by the expected delay along a path, instead of the number of hops used in most of the existing algorithms. This metric allows the algorithm to adapt to changes not only in the topology of the network, but also in the traffic intensity. The algorithm uses a novel technique for estimating the path delays without requiring the links to be bidirectional or the clocks at the nodes in the network to be synchronized. The proposed algorithm is able to perform both reliable and good routing with low communication overhead and computational requirements.

1 Introduction

Reduction in computer size and increase in battery life have resulted in a scenario where fairly sophisticated computing can be performed on the move, or in other words, *mobile computing*. One can further envisage a scenario in which a group of people with mobile computers desire to share information over a wireless channel without any centralized control. For instance, mobile users strewn across a campus may want to interact with each other, or members of a rescue team may need to communicate in order to coordinate their actions. Such networks are usually referred as *ad hoc* networks.

[1]The research reported here has been supported by the National Science Foundation under Grant Nos. ECS-94-03571, the U.S. Army Research Office under Contract No. DAAL-03-91-G-0182, and the Joint Service Electronics Program under Contract No. N00014-90-J-1270.

[2]Department of Electrical and Computer Engineering, and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, 1308 West Main Street, Urbana, IL 61801. piyush@decision.csl.uiuc.edu

[3]Department of Electrical and Computer Engineering, and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, 1308 West Main Street, Urbana, IL 61801. prkumar@decision.csl.uiuc.edu

The most important issue in an ad hoc network is how a mobile host can communicate with another mobile host which is not in its direct transmission range. The common approach is to consider shortest-path routing based on the Distributed Bellman-Ford algorithm (DBF) [1, 8, 9]. In DBF, every host in the network maintains the length of the shortest path from each of its neighbor hosts to every destination in the network. With this information, a host sends data packets to a neighbor which leads to a shortest path to the destination. In order to maintain up-to-date distance information in a dynamic environment, every host monitors its outgoing links and periodically broadcasts to neighboring hosts its current estimate of the shortest distance to every network destination. The most commonly used measure of distance is the number of hops in the path. Even though this measure is easy to compute, it ignores differences in available bandwidth and queueing delays. As a result, an algorithm based on such a distance measure routes packets only over a few (shortest-distance) paths, leading to congestion.

There are two alternative algorithms suggested in the literature: Lightweight Mobile Routing (LMR) [3, 4] and Dynamic source Routing (DSR) [6]. In the former, a set of paths is determined for each source-destination pair by flooding the network with query packets. The link failures are taken care of by the *link-reversal* method [4]. As routing is done over multiple paths, traffic congestion is avoided. However, routing is not optimal as the routes are chosen without any cost consideration. Also, LMR requires the links to be bidirectional.

In the Dynamic Source Routing (DSR) algorithm, the source of a data packet determines the complete sequence of hosts through which the packet is routed before it reaches the destination. Since the network connectivity is changing with time, the source routes are dynamically constructed using a *route-discovery* protocol, i.e., whenever a host needs a route to another host and it does not have one in its cache, it dynamically determines one by flooding the network with *route-discovery* packets [6]. With this protocol, DSR does not require the links to be bi-directional. However, DSR has its own problems: Even in moderately dynamic networks DSR may result in a large communication overhead as it may have to invoke the *route-*

discovery protocol often. Moreover, the algorithm is not delay optimal—the cached routes continued to be used in spite of host movements and are removed only when a failure occurs.

Hence none of the existing algorithms that we are aware of is able to handle all the aspects of routing in ad hoc networks. In this paper we propose a highly-adaptive algorithm which comes closer to satisfying all the requirements of an ideal routing algorithm. The proposed algorithm uses a more appropriate distance measure given by the expected delay along a path. Using this measure, the algorithm is able to route taking into consideration the link capacities, queueing delays and the number of ongoing connections. The proposed algorithm uses a novel technique to estimate path delays, and is thus able to route, without requiring the links to be bi-directional. In spite of all these improvements, the algorithm is simple as well as having low communication overhead and storage requirements.

The rest of the paper is organized as follows. Section 2 describes the model used for ad hoc networks. It also states the assumptions made about the network. Section 3 describes in detail various aspects of the proposed routing algorithm. Some preliminary simulation results are given in section 4. Section 5 gives some directions for future work.

2 Network Model and Assumptions

We model an ad hoc network as a directed graph $\mathcal{G} \equiv (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the vertex set of the graph and $\mathcal{E}$ is the set of all directed edges in the graph. Vertex i in the graph corresponds to host i in the network, while a directed edge between two vertices i and j, denoted by (i, j), represents that node i in the network can transmit messages to node j (Hereafter the terms *nodes* and *hosts* will be used interchangeably). At time instant t, a set of nodes called the *receiving neighborhood* of i, and denoted by $N_i^R(t)$, is associated with each node i, which indicates the set of all nodes to which i can send a message in a single hop. The *transmitting neighborhood* of i, denoted by $N_i^T(t)$, is similarly defined. Note that in general $N_i^T(t) \neq N_i^R(t)$ as we do not assume that the links are bidirectional. Finally, we define $D_{sk}^d(t)$, for $k \in N_i^R(t)$ and at time t, as the expected delay of a data packet routed from s to d in the following way: s transmits the packet to k, and k routes the packet to d over a minimum-delay path.

We make the following assumptions:

- **A1:** The network topology and traffic rates are such that queueing delays are dominant over propagation delays.
- **A2:** Each node has an accurate clock, even though clocks at different nodes need not be synchronized.
- **A3:** There exists a mechanism to prioritize packets.

3 System and Traffic Dependent Adaptive Routing Algorithm

The System and Traffic Dependent Adaptive Routing Algorithm (STARA) that we propose is based on shortest-path routing. Unlike most existing shortest-path routing algorithms (cf. Section 1), STARA employs mean delay as a distance measure. As a result, STARA is able to route data packets taking into consideration the link capacities and queueing delays.

The basic idea of STARA is as follows: The main objective of the algorithm at a node s is to determine the "best" next hop for a data packet to be sent to destination d, i.e., to determine a node $k^* \in N_s^R(t)$ which lies on a minimum-delay path from s to d. However, if a pure delay minimization policy is used, different paths from s to d will have different delays. This will result in a high packet re-sequencing memory requirement. Hence, STARA instead minimizes the mean delay such that all utilized paths have equal delays. For this purpose, STARA adaptively estimates $D_{sk}^d(t)$ for every source-destination pair (s, d) and for each $k^* \in N_s^R(t)$ (cf. Section 2). With this information, s allocates its traffic among its neighbors $N_s^R(t)$ such that all *utilized* nodes in $N_s^R(t)$ have an equal mean delay to d. In case of a link failure, which may be caused due to node mobility or otherwise, the nodes at the two ends of the failed link declare themselves unavailable for routing to their *transmitting* neighbors. The neighbors then stop sending data packets to the nodes on the failed link, but continue to *probe* these nodes for alternate paths or for recovery of the failed link.

We now describe the various aspects of the algorithm in detail.

3.1 Initialization

The first step in the algorithm is that every node in the network determines its receiving and transmitting neighborhood sets. For this purpose, each node s broadcasts a *probing* packet. All the nodes who hear the broadcast include node s in their transmitting neighborhood set, as well as *constrained-flood* an acknowledgment (ACK) to node s. By constrained-flooding of a packet we mean that any node hearing the packet *re*broadcasts it, unless the packet has already gone through M_h number of hops (M_h is a parameter of the algorithm). Now node s forms N_s^R as the set of all nodes from which it receives an ACK to

its broadcast.

Having obtained the neighborhood sets, each node s in the network initializes $D^d_{sk}(t) = 0, k \in N^R_s$. As a result, STARA first *explores* all the paths from s to d before it chooses any particular path or a set of paths for routing the data packets.

3.2 Delay estimation

Estimating expected delay along different paths is the most important aspect of our routing algorithm. Ideally, we would like to have an exact estimate of delay that a data packet would face along a path, so that the routing problem then becomes one of choosing a path with the minimum delay. Routing in ad hoc networks becomes complicated because of the following reasons:

- **P1:** Path delays are dependent on events which happen at different physical locations.
- **P2:** Path delays are dependent on the topology of the network as well as on the distribution of traffic in the network, both of which are time-varying.
- **P3:** The algorithm actually needs an estimate of $D^d_{sk}(t)$, which, in general, has a nonzero variance.

In order to tackle the first problem STARA uses a modified version of the *end-to-end acknowledgment* protocol. According to this protocol, the destination is required to send an acknowledgment (ACK) to the source on receiving a data packet. However, the packet and its ACK will, in general, travel over a different set of nodes. Even if they do travel along the same (undirected) path, they will face different delays due to different traffic conditions in the two directions. Hence an accurate estimate of the delay from source to destination cannot be obtained from the round-trip delay of the data packet and its ACK combined. A solution could be to require that the clocks at different nodes are synchronized. In such a case, the source can time-stamp the data packet before sending it to the destination; the latter on receiving the packet can send back an ACK containing the packet delay (obtained as the difference between the time the destination received the packet and the time-stamp on the packet). However, the assumption that the clocks located at physically-separated nodes in the network are synchronized is hard to achieve. Nevertheless, we now indicate how the above procedure can still be used even if the clocks at different nodes are not synchronized, but are only *accurate* (Assumption **A2**). This is so, because, as indicated above and described in detail later, STARA only needs to have accurate estimates of the *differences* between delays along different neighbors for a source-destination pair, i.e., $(D^d_{s\,k_1} - D^d_{s\,k_2})$, for $k_1, k_2 \in N^R_s(t)$. These differences will be correct even if the clocks at s and d are not synchronized, but have a fixed offset.

Now if acknowledgments were accorded similar *priority* as data packets at the nodes, ACKs would face large delays (of the same order as delays of data packets). This may make the path delay information in ACKs obsolete by the time they reach the sources. Hence, in order to reduce the queueing delay for ACKs, and therefore the delay in obtaining the delay information, STARA gives priority to an ACK over a data packet (cf. Section 2: Assumption **A3**).

Problems **P2** and **P3** listed above are standard problems associated with the estimation of any time-varying parameter. The standard practice is to use either *fixed-size window* averaging or *exponential-forgetting* [7]. In STARA, we use the second scheme for estimating $D^d_{sk}(t)$, i.e.,

$$D^d_{sk}(t) = (1-\lambda)\sum_{l=0}^{\infty} \lambda^l \cdot D^d_{sk}(t-l), \qquad (1)$$

where $\lambda \in [0,1)$ is the *forgetting* factor. In other words, the estimate is obtained as an exponentially weighted average over all past and present values of the parameter, giving more weightage to more recent values. By changing λ, the weightage given to past information can be changed, which will be useful in adapting to changes in network topology (cf. Section 3.4).

Note that STARA will, in general, route data packets only over a small subset of the set of all available paths between any source-destination pair. The remaining paths are not used because of large delays associated with these paths. However, path delays are time varying and it may so happen that a path which was bad becomes good at a later time. However, if the source does not send any traffic along this path, it cannot learn of any such profitable changes. In order to avoid such a situation, STARA sends dummy packets, called *probing* packets, along paths for which a large time has elapsed since the last update of their delay estimates. A destination on receiving such a probing packet sends back an ACK similar to the one it would have sent for a data packet. The probing packet is given the same priority as a data packet so that the obtained delay estimate reflects the delay a data packet would have faced on this path.

3.3 Routing

Using the procedure outlined above, every node s has an estimate of the expected delay it would incur if it routes a packet for destination d to its neighbor $k \in N^R_s(t)$, i.e., for $D^d_{sk}(t)$. In a pure delay minimization policy, s will route packets to its neighbor k^*, where k^*

is such that

$$D^d_{sk^*}(t) = \min_{k \in N^R_s(t)} D^d_{sk}(t). \quad (2)$$

However, in such a policy, different paths will have different delays (as delay minimization requires equalization of the first derivatives of delays with respect to traffic over different paths [1], which, in general, gives rise to different delays over different paths). Thus, such a policy will have a high packet re-sequencing memory requirement.

Instead, STARA minimizes the mean delay such that all utilized (nonzero flow) paths have equal delays. For this purpose, s adapts its allocation of traffic over its neighbors, $\{p^d_{sk} : k \in N^R_s(t)\}$, as follows:

$$p^d_{sk}(t) = p^d_{sk}(t-1) + \alpha(t) \cdot \left(D^d_s(t) - D^d_{sk}(t)\right), \quad (3)$$

where $\alpha(t)$ is the adaptation step-size and

$$D^d_s(t) = \sum_{k \in N^R_s(t)} p^d_{sk}(t) \cdot D^d_{sk}(t) \quad (4)$$

is the average delay from s to d for the traffic allocation at time t.

We next show how the adaptation rule (3) achieves the desired result of equalizing delay over all paths between every source-destination pair.

Theorem 3.1 *Let $\mathcal{N}$ be a network as described in section 2. Let $\mathcal{P}$ be a traffic allocation on $\mathcal{N}$ such that for every source-destination pair, (s,d), the mean delay is minimized under the constraint that all nonzero-flow paths from s to d have equal delays. Then, all nonzero-flow paths from any intermediate node to d also have equal delays.*

Proof: If possible, let there be an intermediate node n such that two nonzero-flow paths, P_1 and P_2, from s to d via n, have different delays from n to d (illustrated in Figure 1). Let us denote the components of the path P_i from s to n and n to d by P_{i1} and P_{i2} respectively. Also, let D_{ij} denote the mean delay over P_{ij}. Then, since all nonzero-flow paths from s to d have equal delays, we have

$$D_{11} + D_{12} = D_{21} + D_{22}. \quad (5)$$

Also, by the above assumption, P_{12} and P_{22} have unequal delays. Without loss of generality, let

$$D_{22} < D_{12}. \quad (6)$$

Then, by adding D_{11} on both sides, we have

$$D_{11} + D_{22} < D_{11} + D_{12}. \quad (7)$$

Figure 1:

Also, for (5) to hold

$$D_{11} < D_{21}. \quad (8)$$

Combining (6) and (8),

$$D_{11} + D_{22} < D_{21} + D_{12}. \quad (9)$$

(5),(7) and (9) imply that the path P_{11} followed by P_{22} has the least delay of the four possible combinations. By increasing the flow on this minimum-delay path, the average delay from s to d can be decreased. But this violates the assumption that $\mathcal{P}$ is a minimum-delay traffic allocation. Hence our assumption that P_{12} and P_{22} can have unequal delays is not correct. This completes the proof. ■

Theorem 3.2 *Let $\mathcal{N}$ be a network with all links having strictly increasing traffic-delay characteristics, and such that the traffic flow in $\mathcal{N}$ is within its capacity. Then $\mathcal{N}$ has a unique traffic allocation such that all nonzero-flow paths between any pair of nodes have the same mean delay.*

Proof: (Sketch) The problem is first mapped into an electrical network problem. Traffic flow and mean delay on a link in the communication network respectively map to current flow and voltage drop on the corresponding edge in the electrical network. This can be done, as different paths between any pair of nodes are required to have the same mean delay (Kirchoff's voltage law) and traffic is conserved at each node (Kirchoff's current law). Now an application of Tellegen's theorem [2] and the strictly increasing traffic-delay characteristics of the links give the required result. For a complete proof, see [5]. ■

3.4 Adapting to changes in network topology

Since the nodes are mobile and the communication channel is wireless, the topology of the network is continuously changing. These changes result in failures of links as well as formation of new links. In the absence of a good model for interdependence among these

link failures and link formations, they are considered as independent events. With this assumption, we now describe how STARA behaves when a link fails or a new link is formed. Note that the above assumption is approximately satisfied when the data rate is much higher than the rate at which network changes occur.

Link failures: We consider a directed link failure; a bi-directional link failure can be considered as a combination of two directed-link failures.

If a directed link, say (i, j), fails, then i proportionately reallocates the traffic carried by the failed link over other nodes in $N_i^R(t)$. In case there are none, node i declares itself unavailable for routing to its transmitting neighbors $N_i^T(t)$. Such a link failure is detected when node i does not receive a single ACK for N packets sent to j (N is a parameter of the algorithm).

Link formation: If a directed link, say (i, j), is formed between i and j, node i will have new paths to a destination d. Such a link formation is detected if node j can *hear* node i, for $i \notin N_j^T(t)$. In such a case, j sends an *unsolicited* ACK to i, containing its delay estimates to network destinations. On receiving the delay information, node i updates its $R_i^d(t)$.

Formation of a bi-directional link can be similarly considered.

4 Simulation Results

In this section we present some preliminary simulation results for our routing algorithm STARA.

We consider an ad hoc network similar to one studied in [6]. The network consists of 20 mobile hosts, moving around in a square of side 9 meters. Initially, each host is randomly placed in the square. Each host then pauses at its current position for a period, called the *pause time*, and then randomly chooses a new location to move to, and a velocity between 0.3 and 0.7 meters per second at which to move there. Each host continues this behavior, alternately pausing and moving to a new location, for the duration of simulation. Each host has a radio range of 3 meters, and within this range a host can transmit to another host with probability 0.95.

At any given time, a host can transmit data to at most 3 other hosts. While a host transmits to less than three hosts, it initiates a new transmission to a randomly chosen host after an exponentially-distributed time, with mean of 5 seconds. Within a single transmission, a host sends a geometrically distributed number of packets with an mean of 1000 packets, at a rate chosen uniformly between 20 and 50 packets per second. Packet lengths are chosen such that 70% percent have 1000 bytes and the remainder have 32 bytes. If a packet does not reach its destination in the first transmission, it is retransmitted up to two more times. The bandwidth for transmitting data is 100 Kbytes per second.

Figure 2: Delay from Host 1 to Host 20 for pause time=50 seconds.

As in [6], we perform 20 simulations for different movement rates (controlled by choosing different pause times). We plot in Figure 2 the dynamics of mean delay in sending a byte from host 1 to host 20 for a fixed pause time of 50 seconds. Note that STARA is able to adapt quickly to host mobility.

Performance evaluation of STARA in other scenarios (for example, the variation of the mean delay in the network as a function of pause time) is being carried out.

5 Conclusion

In this paper we have presented a highly-adaptive routing algorithm STARA for ad hoc networks. STARA uses a more appropriate distance measure given by the expected delay along a path, instead of the number of hops used in most of the existing proposals. This metric allows our algorithm to adapt to changes not only in the topology of the network, but also in the traffic intensity. STARA uses a novel technique for estimating path delays without requiring the links to be bidirectional or the clocks at the nodes in the network to be synchronized. The algorithm is able to perform both reliable and good routing with low communication overhead and computational requirements.

Further investigation of some aspects of our routing algorithm STARA merits attention.

- In the presented version of the algorithm, node mobility is modeled as a sequence of independent link failures and new link formations. A better model for mobility can make the algorithm adapt faster to topological changes in the network.

- Our algorithm avoids traffic congestion by using multi-path routing. However, when the traffic becomes too large, flow control needs to be exercised in order to avoid congestion. Any such flow control is strongly dependent on the routing algorithm used, and ideally the two should be combined. Hence it is desirable to incorporate flow control into our algorithm.

- Finally, it is often the case that a node needs to send the same data packets to more than one destination. Routing in such cases is referred to as *multi-cast* routing. An extension of our algorithm for multi-cast routing is being worked on.

References

[1] D. Bertsekas and R. Gallager, *Data Networks*, second edition, Prentice Hall Inc., 1992.

[2] L. Chua, C. Desoer and E. Kuh, *Linear and Nonlinear Circuits*, McGraw-Hill Inc., 1987.

[3] M. S. Corson and A. Ephremides, "A distributed routing algorithm for mobile wireless networks," ACM/Baltzer J. Wireless Networks, vol. 1, no. 1, Feb 1995, pp. 61–82.

[4] S. Corson, J. Macker, and S. Batsell, "Architectural considerations for mobile mesh networking," Internet Draft RFC Version 2, May 1996.

[5] P. Gupta and P. R. Kumar, "A System and Traffic Dependent Adaptive Routing Algorithm for Ad Hoc Networks," Preprint, 1997.

[6] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in Mobile Computing (ed. T. Imielinski and H. Korth), Kluwer Academic Publishers, 1996.

[7] P. R. Kumar and P. Varaiya, *Stochastic Systems: Estimation, Identification and Adaptive Control*, Prentice Hall Inc., 1986.

[8] S. Murthy and J. J. Garcia-Luna-Aceves, "A routing protocol for packet radio networks," in Proceedings of 1st Annual ACM International Conference on Mobile Computing and Networking, 1995, pp. 86-95.

[9] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," Computer Communications Review, vol. 24, no. 4, (ACM SIGCOMM '94), pp. 234–244.