

SAFAR: An Adaptive Bandwidth-Efficient Routing Protocol for Mobile Ad Hoc Networks

Jigar Doshi and Prahlad Kilambi

Sri Venkateswara College of Engineering
University of Madras
Pennalur, Sriperumbudur 602 105
jigar@doshi.com, prahlad@acm.org

Abstract. A mobile ad hoc network suffers from the same cost constraints as most wireless networks. In particular bandwidth constraints of wireless links are severe. We present a scalable adaptive fitness-based routing protocol, SAFAR, for mobile ad hoc networks in which we try to optimize the usage of this bandwidth at every stage. The protocol is hybrid, i.e. it makes use of both proactive and reactive procedures for routing in an attempt to reduce route acquisition latency. Using a fitness function, a node decides how many other nodes can be proactively maintained by it. Each node tries to know the best(most fit) nodes in its neighborhood. Hence, high bandwidth nodes are well known. Most of the traffic is routed through these nodes and hence performance is optimized. We present simulation results to substantiate the protocols performance. We also extend this protocol to show how it can be used for power aware routing.

Keywords: MANETs, Hybrid, Adaptive, Fitness.

1 Introduction

Mobile Ad hoc Networks (MANETs) are multi-hop wireless infrastructure less networks. All nodes are capable of movement and can be connected dynamically in an arbitrary manner. Nodes in these networks function as routers that discover and maintain routes to other nodes in the network. Applications of ad-hoc networks have been widely studied and they find extensive use in emergency services. Mobile ad hoc networking can support robust and efficient operation in mobile wireless networks by incorporating routing functionality into mobile nodes. The topology in such networks is dynamic and sometimes rapidly changing. Therefore a protocol for such a network has to be robust as well as efficient.

Mobile networks have many interesting characteristics, which differ from traditional wired networks as described in detail in [1]. In particular Wireless links will have significantly lower capacity than their hardwired counterparts. The throughput of wireless communications has to take into account the effects of multiple access, fading, noise, and interference conditions, etc. Therefore, transmission rate is bound to suffer [1]. One effect of the relatively low to moderate link capacities is that congestion is typically the norm rather than the exception.

Existing routing protocols can be classified into two - proactive routing protocols and reactive routing protocols. Proactive routing protocols in general have not been favored in ad hoc networks because of the volume of routing information exchange (overhead) in a volatile environment. Proactive protocols such as [2] are limited in their application by this overhead. Reactive protocols on the other hand, aim to solve this problem by discovering routes as and when necessary in an on-demand fashion. However these suffer from high route acquisition latencies Data packets have to wait while a route to the destination is found. Reactive protocols such as [3,4], and [5] also cause excessive network traffic when the number of routes required is more. A few hybrid protocols like [6] which, try to combine the best of both worlds, have also been proposed. But all of them are homogenous in their view of an ad hoc network. Nodes of varying bandwidth and power characterize ad hoc networks [1] and these parameters themselves vary over time. Bandwidth, hence, becomes a prime factor of optimization for an ad hoc routing protocol. They do not take into account this diversity in bandwidth and power capacity.

We present a hybrid protocol, Scalable Adaptive Fitness-based Ad hoc Routing (SAFAR), which maintains a restricted active view of the surroundings and uses route discovery for nodes, which are not in the active routing neighborhood. SAFAR is essentially a hybrid protocol, which routes based on the concept of ‘node fitness’. Each node is assigned a fitness value, which can be its bandwidth, power or a cost metric (like the weighted average of the bandwidth and power). The protocol then uses the node’s fitness to decide its role in routing and the extent of its proactive nature. Thus the protocol can dynamically adjust to network characteristics. As the node’s fitness changes so does its role in routing.

2 Related Work

There have been two main approaches to hybrid routing. One approach is to use node election to elect a landmark for a zone. This approach is used in [7]. The landmarks are then proactively maintained. The other approach involves forming overlapping zones like those used in [6] with each node maintaining a proactive zone thus distributing the work of the landmark leader. The disadvantage of the first approach is that the election of the landmark may consume resources and may have to be repeated as topology changes, thus significantly increasing overhead. In the case of [6], its performance depends on the selection of the zone radius, which cannot be done dynamically but instead is set by some administrative means. Unlike [7] our protocol does not use leaders. And unlike [6] our protocol does not use a statically set zone radius. The extent of proactive routing is wholly dynamic.

Many power efficient routing schemes have been proposed as in [8,9]. However these schemes mainly optimize transmission power by using longer routes. Such protocols are again not dynamic and cannot be adapted to topologies where performance and low latency are the overriding concerns.

[10] introduces an adaptive hybrid protocol. Our protocol is similar to [10] in that we dynamically adjust the proactive region of the protocol. However

our protocol differs from [10] in various ways. In [10] the proactive region is uniform depending on the zone radius. [10] adjusts the zone radius dynamically. In our protocol the zone is not uniform. Each node selectively adds only the best (most fit) nodes in its neighborhood to its proactive region. In [10], the adjustment of the zone is based on an approximation cost model. [10] adjusts the proactive region in order to make a node more accessible. We change the proactive region in order to reduce route acquisition latencies. Unlike [10], we use the concept of FITNESS (a Genetic Algorithm-based technique) to determine the node's participation in proactive routing. This yields a more realistic proactive region as it takes into account the changing environment of a node.

This paper is organized as follows. In section 3, we give an overview of the proposed protocol, bringing out its salient features. Section 4 contains the complete functional description of the protocol. Section 5 contains simulation results where we investigate performance of our protocol and the issues of scalability and route acquisition latencies. In section 6 we investigate adapting the protocol for power oriented routing.

3 Overview of SAFAR

Our protocol uses a fitness-based routing table buildup scheme. The fitness of a node is based on node characteristics like power or bandwidth value and can change over time. We query nodes, which have a "good" idea about its surroundings. This minimizes the overhead to maintain the routes when compared to proactive protocols and other hybrid protocols. In comparison to purely reactive schemes, our protocol reduces route acquisition latency. Existing protocols are not adaptive in terms of overhead. The disadvantage of a non-adaptive approach is that the diversity of the nodes is completely ignored. A node with low bandwidth (or power - in battery time left) cannot afford the overhead involved in having actively maintained routes. But a node with good bandwidth, which can afford this, helps boost its own performance as well as that of its neighborhood. This is an example of an adaptive "overhead" scheme.

For routing we use a two-stage approach with a limited discovery scheme in the first stage. There is a very high probability of finding a route in the first stage itself. Thus the overhead of a reactive route discovery stage is avoided. If the route is not found in the first stage then we use a route discovery procedure similar to the on demand protocols. In our protocol, data is routed between two mobile nodes. We do not explore multicast operation of SAFAR. Each mobile node maintains a routing table. The routing table contains the node's address, next-hop address, bandwidth (in kbps) and power (in battery minutes remaining). Ancillary information like neighbor and update lists, required by the protocol, need to be maintained separately or as part of the routing table itself. Each node also maintains a node heap which is a max heap maintained in terms of a cost value (power or bandwidth). This heap is used during the table buildup procedure explained in 4.2. Power optimizations will be explained in section 6.

4 Protocol Details

The protocol has four main subdivisions. In section 4.2, we explain the neighbor discovery mechanism, which leads to the table buildup procedure(Section 4.3). In section 4.4, we describe how routing is carried out and the route discovery procedure. Section 4.5 describes how routes already acquired are maintained.

Assumptions:

- Every node has information about its own bandwidth in terms of link capacity from the link layer.
- Every node has information about its own power at any point of time. This information is available in terms of battery time remaining.
- All links are bi-directional

4.1 Fitness Function

Our protocol uses the concept of fitness function applied extensively in [11]. We use this because each node exists in a diverse environment (nodes have differing bandwidth and power attributes). Thus, it draws a parallel to genes of differing fitness value existing in the population of a Genetic Algorithm(GA). GAs use fitness functions to determine which members from the population are to be selected.

The fitness function, is given by, $F = \frac{\delta}{2\Delta}$ where, δ is node's cost metric and Δ is average cost metric of surrounding network environment. The choice of δ ,can vary in a volatile ad hoc environment. In most cases where neither bandwidth nor power is an overbearing concern, the most versatile choice would be the weighted average of bandwidth and power, $\delta = \frac{\alpha B + \beta P}{\alpha + \beta}$ where, $\alpha, \beta > 0$, B is the node's own bandwidth(in kbps) and P is the node's remaining power (in battery time left). But in certain cases, when bandwidth(or power) alone may require optimum usage, β (or α) could be set to 0. For the remainder of the paper, we describe the protocol behavior in terms of optimizing bandwidth alone ($\alpha = 1$ and $\beta = 0$). The choice of Δ is obvious and is given by $\Delta = \frac{\bar{\delta}}{n}$ where, n = no. of nodes in the immediate vicinity of the current node. A more effective choice for Δ might be the average of averages from different environments. This may be given by, $\Delta = \sum_{k=1}^N \frac{\Delta_k}{N}$ where, Δ_k is the average of node cost metric δ for environment k, and N is the number of environments considered.

4.2 Neighbor Discovery

Initially when a node (say A with bandwidth 1500 kbps) joins a network (Fig. 1), it sends a hello packet to its neighbors by means of a broadcast as it does not know their addresses. The hello packet carries A's address and cost metric(bandwidth) value.

A's neighbors (B, C, D, E, F), on receiving this hello packet, update their routing tables, adding A as a neighbor. They then reply, with a hello reply

packet that reports their address and bandwidth value. They also have the option of preventing A from actively maintaining information on them using a blocking bit(explained subsequently). For example in Fig. 1, node F sets the blocking bit and hence prevents A from querying it although it has high fitness. Node A, now adds all the neighbors who responded into its neighbor list and those with blocking bit not set to the node heap. Now the node begins the table buildup procedure. Node A, for the remainder of its lifetime, keeps polling its neighborhood with hello messages to stay informed of its neighbors.

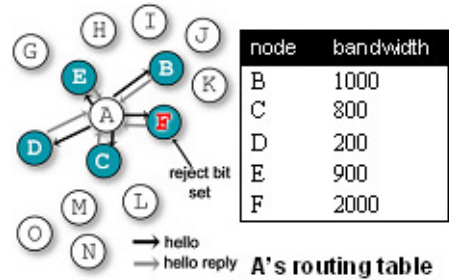


Fig. 1. Node A's immediate neighborhood on its entry into the network

Use of the Blocking bit: This bit may be set if a node finds the ratio of routing information to data information exceeds some threshold. This safeguards the proactive routing overhead of the protocol and hence a node with high fitness is prevented from being overloaded (Node F in Fig. 1). The blocking bit is also set when the environment around that node is volatile. This forces nodes around it to switch over to reactive routing, which performs better in such scenarios. In a stable environment, nodes will be encouraged to use active information. Thus the protocol adapts to changing environment characteristics.

4.3 Routing Table Buildup

APPLICATION OF FITNESS FUNCTION

We extend the concept of fitness function, discussed earlier, to MANETs. Each node uses the fitness function to find its role in the environment and how many nodes it can query. If it has higher fitness, it has to assume a role of facilitator and allow less fit nodes to communicate. This function $F(\text{fitness})$ is now used to select the number of nodes to be queried (M) from subsequent hop nodes using the formula,

$$M = \min(\text{round}(F \times \exp^{-i^2} \times n), n1) \tag{1}$$

where, i is the iteration number ($0, 1, \dots, n$), n is the sum of number of hop ($i+1$) nodes to be queried and number of nodes not selected up to i iterations and $n1$ is the number of nodes currently in heap. In Fig.2, Node A selects B, C and E based on this fitness function as they have sufficiently high bandwidth. It then starts the next iteration of the table buildup by querying these selected nodes. This is explained in the next paragraph(Querying Fit nodes). The exponential fall-off in the order of 2 is used because it becomes that much more expensive to maintain nodes proactively with increasing hop radius. This is multiplied by n , as n is dynamic and varies from hop to hop. Thus the fitness is effectively 'scaled' [11].

Choice of fitness function: The ratio $F(\text{fitness})$, remains fairly constant throughout the table buildup procedure. Hence, it makes for a stable fitness function. The node is likely to query half the nodes if its bandwidth matches the average.

QUERYING FIT NODES

This involves querying the selected nodes once the fitness function has been applied. In Fig. 2, Node A queries nodes B and C with a send table message. This message is a request to these nodes to return their neighbor lists back to the source node A. B and C are added to the proactive list (the proactive field in the routing table is set to true). As illustrated in Fig. 2, nodes B, C and E, on receiving a send table message from node A, generate a transfer table packet which reports address and bandwidth information of its neighbors (I, J, K, L, M, G, H respectively) back to A. It also adds A to its update list (i.e. the update field in the routing table is set to true). This list is explained in section 4.4. Node A, on receiving a table transfer message, updates its routing table and its heap with the newly found nodes (I, J, K, L, M, G and H). These nodes will be considered in the next iteration of fitness function application along with the rejected nodes from the previous query (D). The node A calculates average bandwidth again using new data in its routing table and recalculates number of nodes to query. As shown in Fig. 3, nodes J and M are selected.

It has to be noted that the M nodes can be chosen from any hop on the basis of maximum bandwidth value. Thus the radius of the active neighborhood is not maintained uniformly. The procedure of querying fit nodes is applied again on J and M (Fig. 3). At this stage the number of nodes to query becomes zero and hence the table build-up procedure ends.

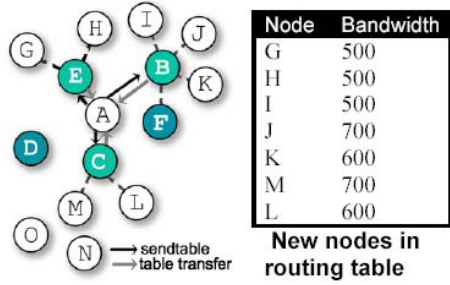


Fig. 2. Node A, relaying a send-table message to nodes B,C and E

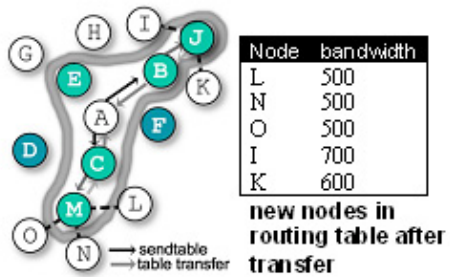


Fig. 3. Formation of the proactive zone

4.4 Routing

When a node requires a route to the destination (say X), it starts a route discovery procedure, which follows a two-stage mechanism as given below.

TYPE-1 MESSAGING

The node, in need of the route, sends a type-1 message to everyone in its proactive list. If anyone of these queried nodes has a path to the destination, it responds with a Type-1 reply packet. This packet also reports the maximum cost (lowest

bandwidth) link along the route it has to the destination (it would be able to calculate this from its own routing table).

The first Type-1 reply for the destination X, triggers an entry being added for the destination in the source node's routing table with the responding node's address set as the next hop. For subsequent Type-1 replies for the same destination, the response with a higher bandwidth is chosen and the routing table is updated appropriately. An intermediate node, which receives a Type-1 reply, adds an entry for destination X to its routing table before forwarding the packet to the source A. In the event of no response from any node (within a timeout period), route discovery proceeds to the next stage, otherwise, it ends here.

TYPE-2 MESSAGING (This is an on demand route discovery, following the same pattern as [5] or [6]).

When the members of the proactive list do not respond positively, a Type-2 request is created. The Type-2 request packet contains a request-ID, a bandwidth field and a node list. The source adds its address to this list and broadcasts the message. This prompts a route discovery among the nodes that receive this message. Every intermediate node G (say), that receives this message checks the source-destination-request-ID tuple to see if it has already handled this packet. If it has, it discards the packet, otherwise, it adds this tuple to its list. This also prevents loop formation.

- If G does not have knowledge of X, it adds its own address to the node list of the Type-2 message, compares its bandwidth to the bandwidth field of the packet and sets the bandwidth field of the packet to the lower of the two values. This helps to maintain the minimum bandwidth link along the path so that the source node can choose the maximum least bandwidth reply in case of multiple responses for destination X. This is because the minimum bandwidth along a path is its bottleneck. Node G, now forwards the packet to its neighbors.
- If G has knowledge of X, it simply reverses the node list of the Type-2 request and adds it to the Type-2 reply message that it generates. It also copies the value of the bandwidth field from Type-2-request to the reply. It then forwards this message to the first node on the node list. The source just adds an entry to its routing table choosing the least cost route if it has multiple routes.

In case there are multiple replies for the same destination, the source chooses the response with the highest value in the bandwidth field as explained earlier. The source then transmits the data packets based on this entry. In case there is a change in the network configuration over this period and the route no longer exists, the node that is not able to find the destination (say Z) in its routing table, upon receipt of the data packet, returns a "route error" message to the source. This is likely to occur if the destination leaves the routing table of Z before the arrival of the data packet but after the type-2 reply has been dispatched.

Advantage of the two-stage approach: The probability of finding the destination node using a Type-1 message is high as the members of the proactive list have a high bandwidth. Hence, they are likely to be well known to others. Therefore, in the average case, the route is likely to be found at the first stage itself. Even if it is not found at this stage, the overhead is negligible compared to the gain

in performance. It has to be noted that although the node exchanges neighbor information when it is queried during the table buildup procedure, it does not exchange the complete routing table. This is why a Type-1 message is required. Exchanging the complete table would cause problems during update because of chained updates. Also, Type-2 relies on source routing in its reply. Hence, the routing overhead is reduced.

4.5 Route Maintenance

PROACTIVE REGION MAINTENANCE

If a node A leaves the neighborhood of node X, it is removed from both the neighbor list and routing heap and all entries with A as the next hop are removed. If a node B sends X a send table packet, then X adds B to its update list before sending A its neighbor list. In case of any change in the immediate topology of X, i.e. a neighbor being added or removed, an update message will be sent to all nodes in the update list (including B). Thus, the proactive routes are maintained.

In a mobile scenario, where the topology is bound to change rapidly, nodes will be continuously added and deleted from the proactive zone. This presents a challenge because we take the trouble to dynamically buildup our “proactive zone”. Hence we describe a zone replacement strategy, as follows: The node X keeps monitoring the status of its routing table and the number of nodes proactively maintained. Assume, initially the number of nodes in the proactive list is n . Over time, due to addition and deletion of nodes, n may fall below a certain threshold. In such a case, X deletes nodes from the heap and applies the table buildup procedure again to rebuild its proactive list from scratch. This threshold value is calculated as follows: The node X keeps querying neighbors with hello messages and also updates its lists when it gets an update packet. When the total number of proactively maintained nodes drops below 40% of the initial value, the node may begin the table buildup again. The threshold can be tweaked according to the environment in concern. In a highly mobile environment, the threshold may be decreased because repeatedly applying the table buildup procedure will be expensive while in an environment which is expected to be stable the threshold value can be increased. The whole buildup procedure is repeated instead of incremental updates because as all the nodes are mobile, it is assumed that the node will find itself in a totally “alien” environment after some time.

HANDLING ROUTE ERRORS

If a node forwarding a data packet finds out that there is no route to destination then it tries a local route repair. This differs from traditional reactive schemes because we resort to the Type-1 discovery mechanism. If the Type-1 discovery does not yield a route to the destination, we return a route-error message to the source. In the very unlikely case, that we do not have a route to the source itself, we drop the packet.

5 Simulation

We simulate SAFAR under varying conditions of a mobile ad hoc network and show that it achieves its objectives of being highly adaptive. We also evince how SAFAR routes data packets expeditiously whilst remaining bandwidth efficient. The simulator is multi threaded and has been developed in C++. The simulator is, essentially, real time in nature and uses preprocessing of mobility information to relieve some burden off its operation.

The number of nodes in our simulation has been fixed at 50 to replicate the scenarios in [12]. The initial position of nodes is chosen from a uniform random distribution over an area of [670mx670m]. The simulation itself runs for a period of 900 seconds. Every cycle in the simulation corresponds to 1/10 second. The node movement follows the random way point model of mobility. The nodes are assumed to be moving with constant velocity. They move from one location to another by setting an initial velocity, which is maintained throughout their movement. Once they reach this location the mobile host pauses for a random amount of time before moving to another destination. Within the next few time cycles, a change of direction occurs.

Each node is assumed to be equipped with a transceiver whose transmission range (or radius) T_R can be varied. It is assumed that all transmissions within this radius are reliable and have a specified channel error rate. Using the transmission radius we calculate the adjacency lists of each node during any point of the simulation and store it in files, which are accessed by the real time simulator at every clock cycle. Thus in each clock cycle a node knows which nodes are adjacent to it without any expensive calculations.

A unique number known as node-id identifies each node. Every node differs from the other in terms of its bandwidth. The nodes are randomly assigned bandwidth values from a distribution that runs very close to the values present in actual mobile networks. Other than this, all nodes are homogenous in their functional characteristics, i.e. an individual thread handles every node. Each thread has data structures private to it, which simulates the node's internal data structures. All nodes also access shared global structures at every time instant to simulate the channel access and contention characteristics of the MAC layer. In particular, the channel may be busy, in which case, a node cannot send a message and has to wait till it becomes free. The lower layers (MAC and physical) have been simulated with a packet drop rate of around 5%. Data packets are handled differently from other routing packets. Their size is randomly generated between 64 bytes and 1024 bytes. Each node has sufficient buffer capacity to handle data packets of varying sizes.

Assumptions:

- The link layer can report varying bandwidth conditions in the environment.
- The hello and hello reply packets used are not part of the routing overhead. This is because these packets are generated, transmitted and received by the link layer. Any change to the bandwidth information is made through a shared structure, which is accessible at a higher layer.

We measure the performance of the SAFAR protocol against the following performance metrics:

Size of the Proactive Zone

Since the proactive zone is adaptive, the size of the proactive zone varies according to the node’s bandwidth and also its neighborhood. We show change in average size of the proactive zone of the node with varying standard deviations of bandwidth under fairly constant average bandwidth.

The graph (Fig. 4), shows the variations in number of nodes queried by a node for proactive maintenance during a simulation run to the standard deviations of bandwidth. Each node in the simulation run is given a different bandwidth value, with the standard deviation of all 50 nodes being shown in the graph. The average bandwidth of nodes remains constant. As shown in Fig. 4, it has been found that the maximum number of nodes queried shows a sharp increase with increase in standard deviation of bandwidth. This is to be expected since with a larger standard deviation, there will be nodes whose bandwidth is far greater than its surroundings.

Thus the factor F(fitness) increases and it queries more nodes. The average number of nodes remains fairly constant(as expected), since the average bandwidth is held constant. However, the difference between the average and maximum value shows a steady increase. This shows that the fitness function allows adaptive table buildup.

Type-1 Success

This is a measure of the percentage of successful routes discovered in stage 1 of routing. It is measured against bandwidth of the node that initiates this node discovery.

For analyzing the Type-1 success ratio, nodes of different bandwidth values are analyzed as they try to route packets to random destinations. The percentages of such requests that are successful are logged. As seen in Fig. 5, the Type-1 success ratio increases with the bandwidth as expected. The graph also shows that a very high percentage of destinations are found with a Type-1

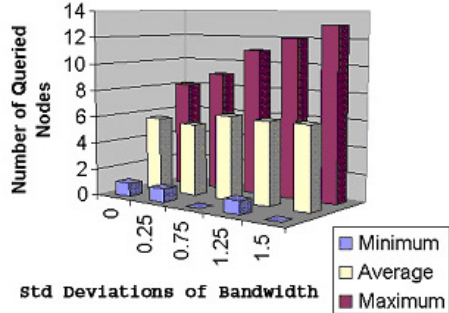


Fig. 4. Number of nodes in proactive zone vs Standard deviation of Bandwidth

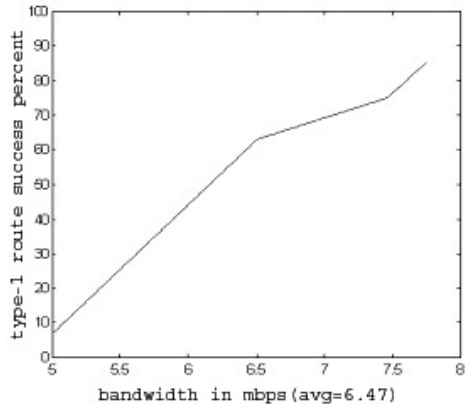


Fig. 5. Percentage of Type-1 queries successful vs Node Bandwidth

query itself thus reducing the need for an expensive Type-2 query mechanism and hence decreasing route acquisition latency.

Packet Delivery Ratio

This is defined as the ratio of the number of data packets delivered successfully to the total data packets sent. We have considered the busy transmission channel condition, which might affect this ratio. The ratio thus, gives us a measure of the reliability of a route and is measured against a varying transmission radius.

The percentage of data packets routed successfully shows that SAFAR finds routes quickly and accurately. The percentage of packets routed successfully increases with transmission range. This is because there is more probability of loss in a route with many hops, as the movement information might not have been registered.

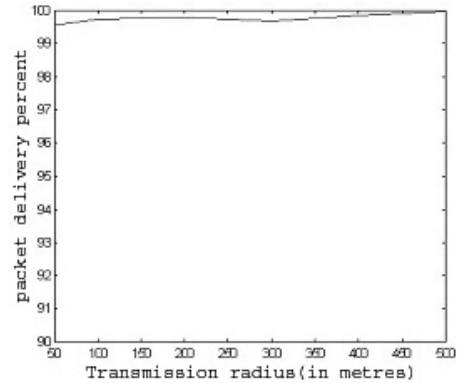


Fig. 6. Packet Delivery Ratio vs Transmission Radius

Route Acquisition Latency

It is defined as the delay between generating a route query and receiving the corresponding reply. During our simulation, we assume that the processing delay of messages is negligible but take into consideration the delay due to queuing of messages in the transmission buffer.

The route latency graph (Fig. 7) shows, how the latency of discovering routes decreases with increasing bandwidth. This substantiates the advantages of SAFAR. By being highly adaptive, it allows a high bandwidth node to use its capacity effectively. It knows more destinations and hence can route faster and more effectively. Even if it does not have a route in the routing table, it can query using a Type-1 packet, which is sent to more nodes. Thus, a high bandwidth node need not resort to Type-2 querying which slows the protocol down and increases latency. Hence, the average route acquisition latency drops with increasing bandwidth.

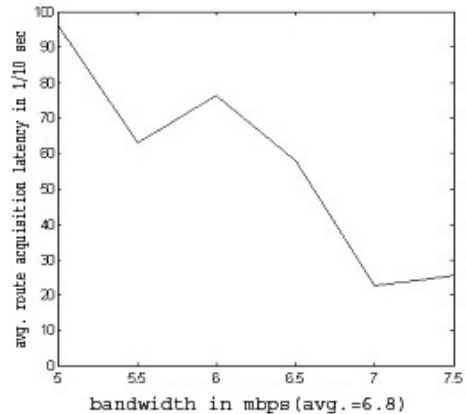


Fig. 7. Route Acquisition Latency vs. Bandwidth

6 Optimization of Power

As mentioned earlier, our protocol can switch over from the bandwidth to the power domain during its operation. In this section, we show that bandwidth and power are interchangeable.

To make SAFAR power centric, the bandwidth field of a node is replaced with a power field (in battery time remaining). Using this, it again selects nodes which will be ‘alive’ for a longer time.

Table build-up phase: This is in direct correspondence with a high bandwidth node, since a node with higher battery power would be able to sustain more data and routing traffic. This would also provide relief to nodes with low power, as they would not have to handle requests other than their own traffic. A node with low power would not be concerned with route efficiency and latency. Since it has low power, it would have few nodes in the proactive list and hence would not spend too much time with Type-1 querying. Instead, it proceeds directly to Type-2 querying which is more likely to yield a route, and also consumes less power in that node, than a combined Type-1 Type-2 mechanism, as it has to transmit more packets which corresponds to more transmission power. Again it has to be noted that this is achieved completely dynamically. In case of multiple responses in the power domain, the path having the maximum least battery time is chosen.

7 Conclusion

In this paper, we have presented and evaluated a bandwidth adaptive hybrid protocol that is well suited for operation in mobile ad hoc networks. The protocol uses a table build-up procedure whose little overhead is well used in the dynamic maintenance of neighboring nodes on the basis of bandwidth fitness. This also leads to reducing the route acquisition latency and hence reduces routing overhead. It was also seen that the protocol adapts to varying high traffic conditions, wherein a node is given the option of reducing its traffic by preventing others from using it. An improvement to the protocol will be in using a hybrid cost factor, which includes both power and bandwidth instead of one of them purely, making it a better estimate of the network’s performance constraint.

References

1. J Macker, S Corson: “Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations”, *Internet draft, January ’99*.
2. Charles E. Perkins and Pravin Bhagwat: “Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers”, pages 234-244, In *Proceedings of the SIGCOMM ’94 Conference on Communications Architectures, Protocols and Applications, August ’94*.
3. Charles E. Perkins and Elizabeth M. Royer: “Ad Hoc On Demand Distance Vector (AODV) algorithm”, In *Proceedings of WMCSA ’99, New Orleans, LA, February ’99*.

4. David B. Johnson and David A. Maltz: "Dynamic source routing in Ad hoc wireless networks", In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153-181, Kluwer Academic Publishers, '96.
5. Vincent D. Parka and M. Scott Corsonba: "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks-Temporally-Ordered Routing Algorithm(TORA)", In *Proceedings of Infocom '97*.
6. Zygmunt J. Haas, Marc R. Pearlman: "Zone Routing Protocol", *Internet draft*, <draft-zone-routing-protocol-00.txt>, MARCH '03.
7. Mario Gerla, Xiaoyan Hong, Guangyu Pei: "Landmark Routing for Large Scale Wireless Ad Hoc Networks with Group Mobility", In *Proceedings of Mobihoc 2000*, Boston, MA, November '00.
8. Suresh Singh, Mike Woo and C S Raghavendra: "Power Aware Routing in Mobile Adhoc Networks", pages 181-190, In *Proceedings of MobiCom'98*.
9. Javier Gomez, Andrew T. Campbell, Mahmoud Naghshineh and Chatschik Bisdikian: "Conserving Transmission Power in Wireless ad hoc Networks", In *Proceedings of IEEE 9th International Conference on Network Protocols*, Riverside, California, November '01.
10. Venugopalan Ramasubramaniam, Zygmunt J. Haas and Emin Gün Sirer: SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks, pages 303-314, In *Proceedings of Mobihoc '03*.
11. David E Goldberg: Genetic Algorithms in Search, Optimization and Machine Learning, pages 76-80, *Pearson Education - 1999*.
12. David B. Johnson, Josh Broch, David A. Maltz, Yih-Chun Hu, and Jorjeta Jetcheva: "A performance comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols", In *Proceedings of MobiCom '98*.