A Proximity-based Dynamic Path Shortening Scheme for Ubiquitous Ad Hoc Networks

Masato Saito[†] Hiroto Aida[†] Yoshito Tobe[‡] Hideyuki Tokuda[†,§]

[†] *Graduate School of Media and Governance, Keio University*
5322 Endo, Fujisawa, Kanagawa 252-8520, Japan
E-mail: {masato, haru, hxt}@ht.sfc.keio.ac.jp
[‡] *Department of Information Systems and Multimedia Design, Tokyo Denki University*
E-mail: yoshito@unl.im.dendai.ac.jp
[§] *Faculty of Environmental Information, Keio University*

Abstract

This paper describes the design, implementation, and evaluation of a proximity-based dynamic path shortening scheme, called DPS. In DPS, active route paths adapt dynamically to node mobility based on the "local" link quality estimation at each own node, without exchanging periodic control packets such as Hello packets. Each node monitors its own local link quality only when receiving packets and estimates whether to enter the "proximity" of the neighbor node to shorten active paths in a distributed manner. Simulation results of DPS in several scenarios of various node mobility and traffic flows reveal that adding DPS to DSR and AODV (conventional prominent on-demand ad hoc routing protocols) significantly reduces the end-to-end packet latency up to 50-percent and also the number of routing packets up to 70-percent over DSR, particularly in heavy traffic cases. We also demonstrate the more simulation results obtained by using our two novel mobility models which generate realistic node mobility: Random Orientation Mobility and Random Escape Mobility models. Finally, simple performance experiments using DPS implementation on FreeBSD OS demonstrate that DPS shortens active routes in the order of milliseconds (about 5 ms).

1. Introduction

Wireless ad hoc networks are expected to play a significant role in ubiquitous computing and communications in the home, office, and any other places. We envisage that wireless ad hoc networks are embedded in everywhere in the various network forms (from *personal area networks* and *sensor networks* to *intelligent transport systems*); i.e., "ubiquitous ad hoc networks." A key challenge to succeed in such communications is adapting to mobility. A mobile ad hoc network is a group of mobile computing devices (nodes) which communicate with each other using multi-hop wireless links. It does not necessarily require any stationary infrastructure such as base stations. In such a network, one important issue for achieving efficient network resource utilization is to update route information reactively depending on a change of network topology and connectivity. Since node mobility in an ad hoc network causes frequent, unpredictable and drastic changes to the network topology, it is especially important for communicating nodes to grasp the changes of the network topology and find efficient routes between two communicating nodes.

A number of research projects for mobile ad hoc networks have developed their routing protocols (e.g., AODV [11], DSR [7], OLSR [2], TBRPF [8]). These routing protocols can be classified into main two types: *proactive* and *reactive*. Proactive protocols attempt to continuously evaluate the routes within the network, so that when a packet needs to be forwarded, the route is already known and can be immediately used. On the other hand, reactive protocols invoke a route determination procedure on an *on-demand* basis. Some comparisons between these different protocols have been published in [1] and [5]. Both reported results based on simulations show that the reactive protocols perform better than traditional proactive protocols (e.g., DSDV [10]) in most situations. We also envisage reactive routing protocols will be spread in future ubiquitous ad hoc networks thanks to the low-cost and on-demand nature.

The previous on-demand routing protocols accommodate route changes only when an active path is disconnected. They cannot dynamically adapt to the change of network topology even if another route with less hop count becomes available by the movement of intermediate nodes without any link disconnections. Given this factor, we propose Dynamic Path Shortening (DPS) scheme that tunes up active paths by not any link disconnection events, but automatically adaptation to node mobility based on Smoothed Signal-to-Noise Ratio (SSNR) as a link quality indicator.

In order to shorten an active path, we introduce the concept of *proximity* that represents the "nearness" of

two communicating nodes. Each node determines to shorten an active path by using *proximity* based on the local SSNR value obtained from their own network interfaces. This local SSNR value is soft state using the internal state of their local network interfaces. DPS is particularly suitable for our conventional situation under slow node mobility (e.g., pedestrian and slow vehicle in campus computing) or dense mobile ad hoc network. In addition, since DPS operates only when forwarding or receiving data packets, it does not require periodic HELLO messages or advertisements.

The rest of this paper is organized as follows. Section 2 briefly describes some related previous work in mobile ad hoc networks. Section 3 presents the design and detailed description of DPS. Then, we present the evaluation results of detailed simulations and some experiments, and the novel realistic node mobility models in Section 4. Finally, we state conclusions and discuss some future work in Section 5.

2. Related Work

This section describes the related previous on-demand routing protocols which do path shortening in mobile ad hoc networks.

Dynamic Source Routing (DSR) [7, 13] is an on-demand routing protocol which uses aggressive caching and source routing headers to obtain the topology information. A DSR node is able to learn routes by overhearing packets not addressed to it by operating its network interfaces in promiscuous receive mode. This scheme also automatically shortens the active paths while sending data packets as well as our DPS scheme. The feature can achieve the dynamic multi-hop path shortening, thus it leads the drastic improvement of the packet latency. However, this scheme requires an always-active transceiver mode of the network interfaces and more CPU cycles to process overheard packets addressed to the other nodes, which may be significantly power consuming. This is especially inefficient in environments where battery power is a scarce resource. Also, because DSR does not take the link quality into account, it possibly leads to inefficient and frequent route change and the great degradation of the link quality.

Roy [12] presents the source-tree on-demand adaptive routing protocol (SOAR) based on link-state information. SOAR incurs much less overhead of control routing packets than DSR under various scenarios, ranging from high mobility to low mobility. SOAR has the mechanism to shorten the active paths, but it achieves that by *periodically* exchanging link-state information in which a wireless router communicates to its neighbors the link states of only those links in its source tree that belong to the path it chooses to advertise for reaching destinations with which it has active flows. As this partial topology broadcast algorithms exchange relatively large control packets including the minimal source tree, total byte overhead due to SOAR control packets has been found to be $2-3$ times more compared to the previous ad hoc routing protocols (e.g., even DSR). High control packet overhead is undesirable in low-bandwidth wireless environments.

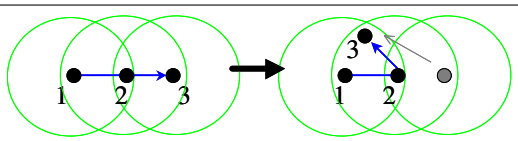


Figure 1: *Node 1 sends packets to node 3 via node 2. Next, node 3 moves into the cell of node 1 without link failures. Although node 1 can directly send packets to node 3, node 1 still sends packets to node 3 via node 2.*

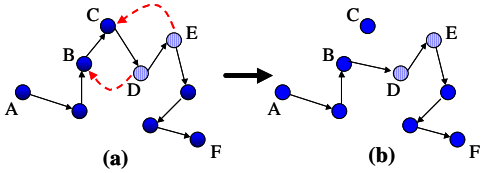


Figure 2: *Node A sends packets to node F in a multi-hop network.*

Additionally, periodical exchanging messages could collide with the data streams, thereby may degrading the performance. The SOAR mechanism to maintain and synchronize the minimum source tree of its own neighbor nodes with the varying network conditions is fairly complex and computational overhead.

In contrast to the above several work, "DPS" does not lead to the weak-connectivity shortened routes or inefficient frequent route switching since it is based on local link quality, and does not need periodic information advertisements or any overheard packets by making the network interfaces promiscuous receiving mode. DPS adapts effectively to node mobility using local link quality in wireless ad hoc networks which are scarce bandwidth and battery environment.

3. Dynamic Path Shortening

This section describes the detailed design of DPS. First, we explain some scenarios in which DPS effectively shorten active paths. Second, we introduce the notion of proximity to identify two near nodes by using link quality and discuss the link quality. Finally, we explain DPS protocol performing active one-hop shortening in detail.

3.1. Problem of Path Inefficiency

In a mobile ad hoc network, due to node mobility, we encounter a situation shown in Fig. 1. In this case, we pay attention to node mobility without link disconnection. For such node mobility, we possibly find the less hop route (i.e., direct hop route shown in Fig. 1) than the current route in use.

Fig. 2 shows the more complicated scenario. Some less hop routes are available in the active path from source to destination. If each neighbor node simultaneously shortens the active path (in Fig. 2, $D \rightarrow B$ and $E \rightarrow C$), it leads to the isolated routes and deadlocking. As a result, the active path from source to destination is failed and the sender node must re-initiate a

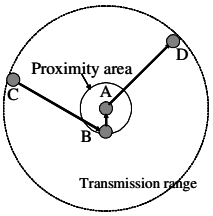


Figure 3: A proximity area

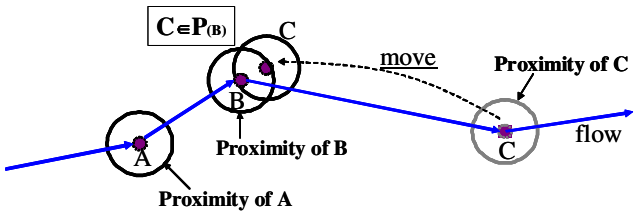


Figure 4: A→B→C route shortens A→C.

new route discovery. We describe how to overcome this problem later.

3.2. Proximity Area

To argue the "nearness" of two nodes more formally, we introduce the notion of proximity. It is based on the relationship between the distance of two nodes and the received SNR. In short, the proximity area of a node is determined from the received SNR value more than a pre-defined threshold. In this proposal, we assume that SNR resources are available from wireless MAC layers. Most of current "wireless" NIC drivers have the reporting function of SNR. Fig. 3 demonstrates the proximity area around the active intermediate node A. In this figure, node B enters the proximity area of node A.

To discuss the proximity area logically, we define the following symbols.

- $S_{(AB)}$: The SNR (SSNR) value observed at node B for received data packets from node A.
- S_{max}: A threshold value of SNR (SSNR).
- $P_{(A)}$: The proximity of node A.
- $R_{uf}(A)$: The upstream adjacent node of node A for flow f on a route.
- $R_{df}(A)$: The downstream adjacent node of node A for flow f on a route.

We hypothesize that $S_{(AB)} = S_{(BA)}$. Here, we assume all the nodes in an ad hoc network are homogeneous, it means they have the same wireless network interfaces. This assumption is strictly unrealistic, but in DPS design we try to cope with that by using DPS handshake protocol which is explained later. If $S_{(AB)} \geq S_{max}$, node B is said to be in the proximity of node A, or $B \in P_{(A)}$. Based on the above hypothesis, if $B \in P_{(A)}$, then $A \in P_{(B)}$. Fig. 4 shows a flow traverses node A, B, and C in this order. This can be written as $A = R_{uf}(B) = R_{uf}(R_{uf}(C)) = R^2_{uf}(C)$. If $C \in P_{(B)}$, there is a possibility that the path of the flow can be shortened: $A = R_{uf}(C)$. As shown in Fig. 4, each node is associated with its own proximity. When node C moves to the proximity of node B, node A can directly send data packets to C. In practice, we need a hysteresis mechanism around the threshold value to avoid unnecessary oscillation.

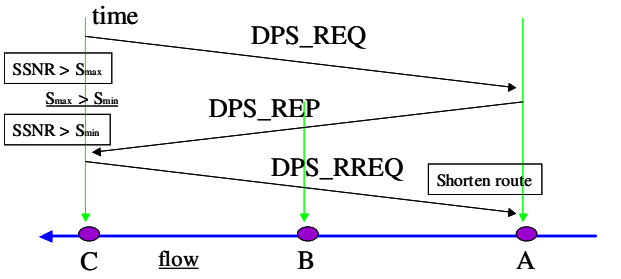
3.3. Link Quality

From the point of view of measuring the link quality, DPS uses a smoothed value of Signal-to-Noise Ratio (SNR) in a time domain since the SNR could change dynamically with a high frequency due to electro-magnetic effects. This value, Smoothed SNR (SSNR), can be computed using a weighted moving average technique as follows: $ssnr = (1 - \alpha) * old\_ssnr + \alpha * cur\_snr$, where cur_snr and old_ssnr represent the value of SNR on receipt of a packet and the previously computed SSNR, respectively. The constant value of α is a filtering factor and is set to $1/8$ in this paper. It is because we could adapt to the large fluctuation of SNR and use a shift operation in our experimental implementation. In DPS, the filter calculates SSNR whenever a node receives the frames.

3.4. Design of DPS

We set two design goals to DPS: reducing the hop count of a path, and minimizing the number of additional control packets. The first goal is obvious in the context of the problem aforementioned. In addition to the first goal, we aim at a scheme not producing periodic control packets. This is an important consideration for an ad hoc network since nodes in the network need to save their power consumption. We design our scheme so that control packets are transmitted only when a node determines that a path should be changed based on the proximity. In DPS, each node in ad hoc networks has the original routing information concerning upstream two-hop-away nodes. Since a node attempts to transmit the control packet to the upstream two-hop-away node, the node needs to retain the route information of its upstream two-hop-away nodes of the active flows as well as its neighbors.

We introduce the fundamental messages passed among three nodes. DPS uses three kinds of messages: DPS_REQ, DPS_REP, and DPS_RREQ; they are shown in Fig. 5. DPS_REQ and DPS_RREQ are newly generated control packets, while DPS_REP can be piggybacked on a data packet. Let us assume that $A = R_{uf}(B)$ and $B = R_{uf}(C)$ for flow f as shown in Fig. 5. When node C determines that it has moved into the proximity of node B, it sends DPS_REQ to node A. The purpose is to observe whether or not a packet can be directly exchanged between node A and C. Upon receipt of



DPS_REQ	Request to $R^2_{uf}(k)$ as a packet to probe direct route
DPS_REP	Reply to $R^2_{df}(k)$ via $R_{df}(k)$ for DPS_REQ (piggybacked)
DPS_RREQ	Redirect Request to $R^2_{uf}(k)$ via $R_{uf}(k)$ to shorten route

Figure 5: Three DPS control packets

DPS_REQ, node A sends DPS_REP to node C. Unlike DPS_REQ, DPS_REP is rather piggybacked by th data packet of flow f than sending as a single control packet. Therefore, DPS_REP reaches node C via node B. By receiving DPS_REP, node C knows that node A can send packets directly to node C and sends DPS_RREQ to node A to initiate a change of route.

There is concern about race condition: simultaneous attempts by each adjacent nodes to shorten the same path may occur as shown in Figure 2. We solve this problem in the way similar to TCP three-way handshake but more delicate way to handle mutual exclusion (see, Fig. 5).

If we assume that $A = R_{uf}(B)$, $B = R_{uf}(C)$, and $C = R_{uf}(D)$ for flow f. When $S_{(BC)} \geq S_{max}$, node C sends DPS_REQ to node $R^2_{uf}(C)$ (i.e., node A) to locate the direct hop route. As long as $S_{(BC)} \geq S_{max}$, node C continues to send DPS_REQ every time DPS timer expires until node C receives DPS_REP. Upon successful receipt of DPS_REP, node C sends DPS_RREQ to node A to ask for the redirection of the path of flow f. Upon success in the above process, node A can directly send data packets to node C.

4. Performance Evaluation

We show the detailed simulation and some experimental results of DPS. First, we describe the simulator implementation and performance results of DPS adding to DSR and AODV in the *ns-2* network simulator environment [14]. We simulate DPS on several large mobile topologies to quantify the scaling behavior of DPS. In addition, to study how DPS scheme performs in *realistic* node mobility patterns, we measure the effectiveness of DPS using our two practical mobility generation models which are based on the *random way-point model* [6] used in almost the previous simulation research. Finally, we demonstrate building a small wireless ad hoc network testbed and performing a simple preliminary experiment.

4.1. Implementation Decisions

We have implemented the two DPS schemes in our simulation: the fast shortening mode (FAST) and robust shortening mode (ROST). In ROST, DPS_REQ sender obtains the IP address of the two-hop upstream neighbor from the source routing header of receiving data packets. Since DSR is the source routing protocol, we should report the changes of the active route by sending GRATUITOUS_REPLY [7] to the source node when DPS completes shortening of active paths. We also implemented the timer routine to reset the above state to compensate such cases as the shortening failures or incompletion due to DPS control packets loss or limitations of link quality. This mode is the same as DPS implementation in our real environments.

In FAST, DPS realizes faster shortening at the cost of the completion probability of the operation. The difference is that the node receiving the DPS_REQ shortens its own active route promptly to forward directly data packets to the node which send the DPS_REQ. Since FAST operates based on the relatively optimistic policy, it cannot necessarily assure the consistency of the shortening. Although the performance of FAST was better than ROST in our preliminary simulations, we chose ROST scheme based on the conservative nature of ROST and our heuristic.

To illustrate that DPS is not specific to DSR protocol, we have incorporated the dynamic path shortening mechanism into AODV. The original AODV does not have methods for shortening active routes. The modifications we had to make for AODV were somewhat different than those incorporated for DSR. Specifically, data packets do not carry the full source route in their header since AODV is the distance vector routing algorithm. Thus, the two-hop upstream neighbor information is not available from their header. However, we can easily cope with this problem to let the one-hop upstream neighbor node forward DPS_REQ packets. In other words, DPS_REQ packets travel two-hop journey via the upstream neighbor node as the intermediate node, instead of one-hop direct communication. In the intermediate node, to distinguish its own particular route, we have implemented the matching scheme based on the final destination and the next hop node. AODV in *ns-2* is implemented as a user-land application daemon, the final destination node cannot initiate the shorten route, unlike the cases in DSR. For comparison with DSR based DPS, we chose to implement DPS on AODV-LL (Link Layer) [1] using only link layer feedback from 802.11 as in DSR, completely eliminating the standard AODV HELLO mechanism.

4.2. Detailed Simulation

On *ns-2*, the Monarch research group in CMU developed support for simulating multi-hop wireless networks complete with physical, data link and MAC layer models [1]. The distributed coordination function (DCF) of the IEEE standard 802.11 for wireless LANs is used as the MAC layer. The radio model uses a shared-media radio with a nominal bit-rate of 2 Mb/sec and a nominal radio range of 250 meters.

Table 1: DSR simulation parameters

Time between retransmitted Route Requests (exponentially backed off)	500 ms
Size of source route header carrying n addresses	$4n + 4$ bytes
Timeout for non-propagating search	30 ms
Time to hold packets awaiting routes	30 s
Max rate for sending gratuitous Replys for a route	1/s
Max rate for sending DPS Request for a route	3/s

Table 2: AODV-LL simulation parameters

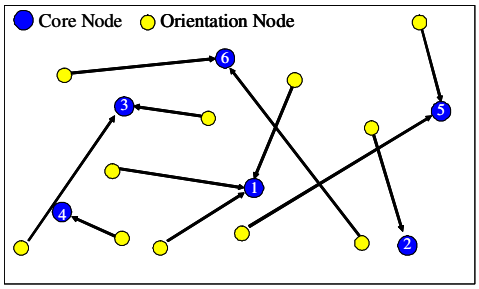
Time for which a route is considered active	50 sec
Lifetime on a Route Reply send by destination node	1 sec
Number of times a Route Request is retried	3
Time before a Route Request is retried	10 s
Time for which the broadcast id for a forwarded Route Request is kept	6 sec
Time for which reverse route information for a Route Reply is kept	10 sec
Time before broken link is deleted from routing table	3 sec
MAC layer link breakage detection (HELLO Packets OFF)	yes
Max rate for sending DPS Request for a route	3/s

DSR and AODV protocols detect link breakage using feedback from the MAC layer. A signal is sent to the routing layer when the MAC layer fails to deliver a unicast packet to the next hop. In this evaluation, no additional network layer mechanisms such as *HELLO Messages* [11] are used. With regard to SNR, DPS extracts the values from the MAC Layer every time a data frame receives. In all the later simulations, we set S_{max} simulation parameter to 0.000008 which was obtained from our preliminary analysis and experiments of SNR.

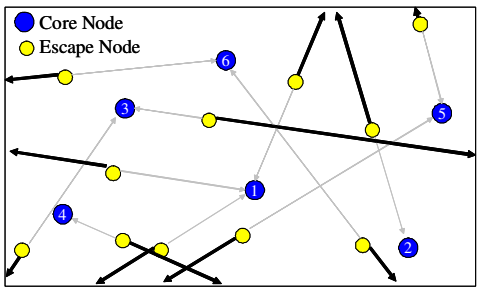
Table 1 and 2 provide all the simulation parameters of both the protocols extended by DPS. These parameters are remained default in the *ns-2* current distribution except DPS parameters.

4.2.1. Traffic and mobility models We have used traffic and mobility models similar to previous published results using *ns-2* ([1], [5], [3]) for appropriate performance comparisons. Traffic sources are Constant Bit Rate (CBR). The source and destination pairs are spread randomly over the network. Only 512 byte date packets are used. The number of source-destination pairs and the packet sending rate in each pair are varied to change the offered load in simulated networks.

To investigate how DPS scheme performs in the *realistic* node mobility pattern, we propose the two node mobility models: "random orientation mobility model (ROM)" and "random escape mobility model (REM)." These models are based on the *random way-point model* [6] used in most of the previous simulation research. In the *random way-point model*, each node begins the simulation by remaining stationary for



(a) Random orientation mobility (ROM)



(b) Random escape mobility (REM)

Figure 6: Examples of random orientation and escape movement

pause time seconds. It then selects a random destination in the specified field space and moves to the destination at a speed distributed uniformly between *0* and some maximum speed. On reaching the destination, the node pauses again for *pause time* seconds, selects another destination, and proceeds there as previously described, repeating this behavior for the duration of the simulation.

In contrast, our two node mobility model generate more realistic movement patterns. ROM is assuming people pursuing something (e.g., *peace*, *money*, *hope*) or attracted something(e.g., *gravity*, *power*). On the other hand, REM is literally assuming people are escaping from something (e.g., *disaster*, *ghost*). The movement patterns are shown in Fig. 6(a) and 6(b). In our proposed models, mobile nodes are classified into three types: CORE_NODE (CN), ORIENTATION_NODE (ON), and ESCAPE_NODE (EN). CNs move around simulation field based on the random way-point model. On the other hand, ONs select one destination from the coordinate positions of CNs randomly instead of a perfectly random destination, and pursue the CN at a speed distributed uniformly between *0* and some maximum speed (e.g., *not all people pursue money*). If a ON reaches the selected destination, then it selects another destination among that of CNs again. This cycle continues until the end of simulation. In REM, ENs desperately try to leave from one of the CNs. ENs choose

the exact opposite side of the destination position of a randomly selected CN as the destination, and move towards the destination. In this model, we assume human mobility in situations as disaster to where ad hoc networks expect to apply. Note that, when node mobility files are generated for simulation, we need to specify the ratio of ONs or ENs to CNs as one argument. If the stated ratio is 0.0, the generated node mobility pattern is accurately based on the random way-point model.

Recently, another realistic node mobility model is proposed in [4]. This model attempts to model the behavior of nodes in a realistic environment where there exits any number of obstacles that obstruct data forwarding paths (e.g., buildings, vegetation). While this model is reasonable from the micro view of realistic environments, our models are rather targetting at the macro model of human mobility. The similar approach incorporating obstacles is partly described in [5].

We use the above-mentioned three mobility model in a rectangular area. One field configurations are used - (i) $1500m \times 300m$ field with 50 nodes. Thus, each node starts its travel from a random location with a randomly chosen speed (uniformly distributed between $0-20$ m/sec except in REM). We vary the pause time, which affects the relative speeds of the mobile nodes: in this paper, we used the following pause times (0, 30, 60, 120, 300, 500 [sec]). Simulation are run for 500 simulated seconds for 50 nodes. Each data point represents an average of ten runs with identical traffic models, but different randomly generated mobility scenarios.

4.2.2. Performance Results

Light Traffic Loads First, we perform experiments using the light traffic loads to study the behavior of DPS added to DSR (DSR+DPS) and AODV (AODV+DPS) and compare with the pure DSR and AODV. Note that, as described in Section 2, DSR can automatically shorten the active paths by operating its network interfaces in promiscuous receiving mode. Normally, while we have to compare DSR+DPS with the promiscuous receiving DSR, our preliminary simulation experiments comparing DSR with and without the promiscuous mode showed that the use of it did achieve a significant improvement of network performance in terms of the packet delivery ratio, end-to-end data delay, and routing packet overhead. However, we think that this feature requires an active receiver in the nodes which can be fairly power consuming. In ad hoc networks where nodes have limited power and CPU, the resources should be limitedly used as possible. Thus, we decide to compare DSR+DPS with the pure DSR. However more exhaustive simulations should be made taking into account power and CPU consumption.

For the 50 nodes experiments we used 10 traffic sources and a packet rate 4 packets/sec. We found that DPS has improved the end-to-end delay as expected and reduces the routing control packet overhead effectively (see Fig. 7(a), 7(b) and 7(c)). However, in the packet delivery ration (PDR), DPS loses about 5 - 10% packets. While we are currently working the accurate reason of lost packets, we think that the reason is by the failures of the path shortening. In Fig. 7(c), note that the packet delivery fractions for DSR are more than 100 %, we think that it may be caused by the retransmissions of data packets performed by DSR protocol.

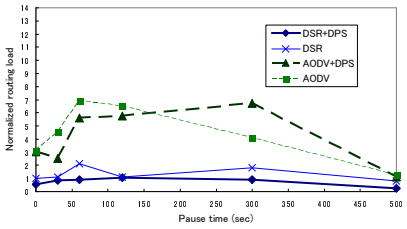
Heavy Traffic Loads To stress the traffic loads to DPS, we used 30 traffic sources. The other configuration parameters are the same as the above the light traffic load experiments. In Fig. 8(a), 8(b) and 8(c), we can see that DSR+DPS achieves the significantly reduction of the packet delay by about 1.5 seconds and routing overhead. Additionally, DSR+DPS also has the high performance of PDR. It is because the number of shortening paths increases more than the light traffic case, thanks to the increased density of active (sending data) nodes, which leads to avoid interference and congestion points, and long-distant hops routes. However, AODV+DPS does not show the improved performance as salient as DSR+DPS, in even the light traffic case. We think that one of the reasons is the potential feature of AODV protocol; AODV node holds many state information and uses the timer-based routine frequently. Hence, AODV and DPS may not be compatible well, we are now investigating this results and reason more deeply. For reference's sake, such the simulation results were pointed out in the previous research work [1, 3].

Random Orientation Mobility Model This mobility model tends to make several independent networks and node congestion points. We can expect that the effectiveness of DPS will be much high since active shortenings could occur frequently. To mainly focus the effect of mobility models to DPS, we have used 10 traffic sources. In Fig. 9(a), 9(b) and 9(c), we see that the improved delay reduction is significant. To generate heavy mobility loads, we have set the ratio of ONs to CNs to *0.8*. For instance, in the case of 50 mobile nodes, the number of ONs is *40*.

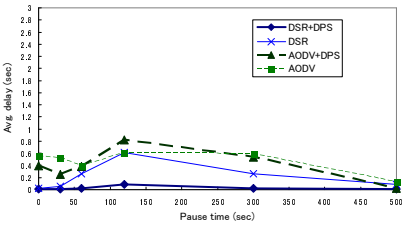
Random Escape Mobility Model This mobility model inclines to make some network partitions intentionally. Thus, the mobile ad hoc nodes suffer from frequent link failures. In this simulation, we also have used 10 traffic sources. As Fig. 10(a), 10(b) and 10(c) show, DPS has still improved the performance of DSR. However, some simulations of AODV protocol caused long-lived routing loops, so that we could not perform the evaluation of AODV and AODV+DPS appropriately. It seems that AODV does not perform well in the situations where several network partitions exist frequently. We are currently trying to improve the *ns-2* AODV simulation code. In this evaluation, we used the ratio of ENs to CNs to *0.8*.

4.3. Experimental Study

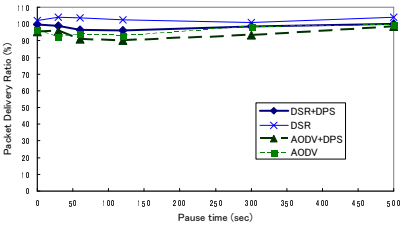
We have implemented DPS as an extension to DSR developed by the Monarch project [13]. Since the Monarch DSR was implemented on FreeBSD 3.3-RELEASE OS platform with old WaveLAN [9] driver, we have performed porting DSR to



(a) Normalized routing load

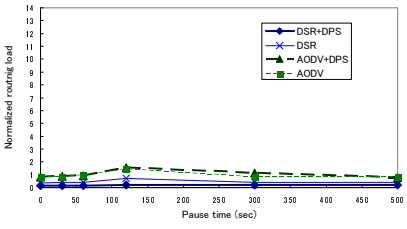


(b) Average data packet delay

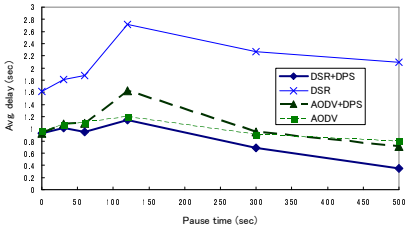


(c) Packet delivery fraction

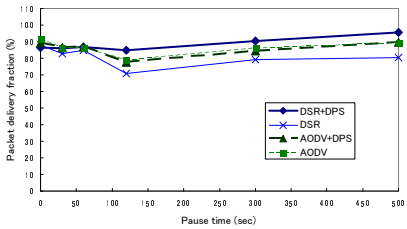
Figure 7: 10 sources model



(a) Normalized routing load

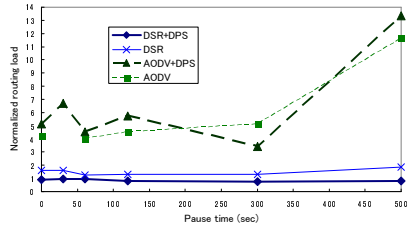


(b) Average data packet delay

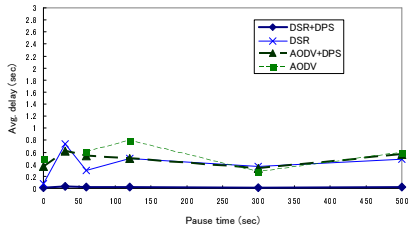


(c) Packet delivery fraction

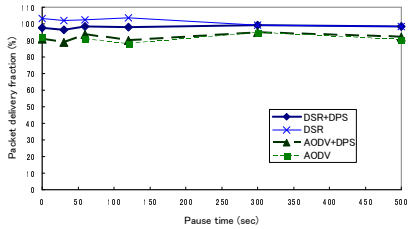
Figure 8: 30 sources model



(a) Normalized routing load



(b) Average data packet delay



(c) Packet delivery fraction

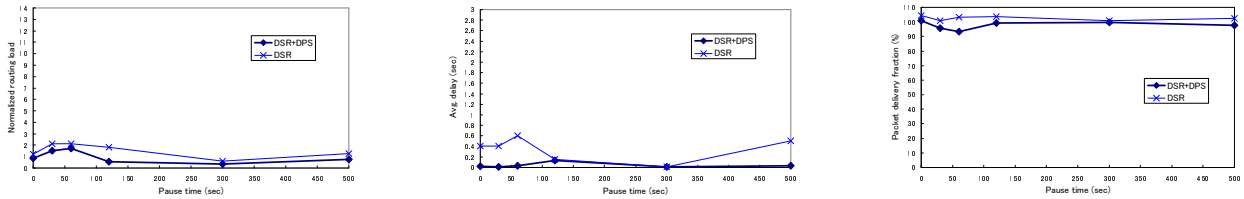
Figure 9: Orientation model

FreeBSD 4.4-RELEASE OS with modern IEEE wireless LAN drivers. We have installed DSR extended by DPS on FreeBSD OS running on laptops. Specifically, as the control packets to initiate the shortening of active paths, we added DPS_REQ, DPS_REP, and DPS_RREQ DSR header options. DPS_REP is always piggybacked on data packets. DPS needs to hold the two state information to shorten active paths: IP addresses of the skipped node and the DPS initiator node (the first DPS_REQ sender).

We briefly state the implementation detail of local proximity checking mechanisms retrieving SNR values from wireless network interfaces. Since SNR is obtained every time a frame arrives, the SNR value accurately reflects up-to-date link quality. To control the oscillation of SNR values, we compute Smoothed SNR (SSNR) using a weighted moving average technique as described in the Section 3. We compare the SSNR with S_{max} n times to initiate path shortening. This n parameter is currently set to 10. Interestingly, the n parameter has the relatively correlation to the frame data receiving rate.

To observe the overhead associated with path shortening, we have conducted five trials of path shortening among three nodes, nodes A, B, and C. Node A sends UDP packets continuously to node C via node B. To build the situation of path shortening intentionally, we have moved node C close to node B. We have then measured the latency from the time at which node C sent the first DPS_REQ message to the time at which node C received data packets directly from node A. Consequently, the overhead latency (including packets processing time) incurred with the exchanges of DPS control messages was sufficiently small: it was the order of milliseconds (about 5 ms).

We have also examined the relationship between TCP throughput and the number of hops (up to 4 multi-hops including 5 nodes) in this testbed environment. We used netperf to send TCP flows. The result showed the one-hop to 4-hop TCP throughput were



(a) Normalized routing load (b) Average data packet delay (c) Packet delivery fraction

Figure 10: Random escape mobility model

0.64, 0.27, 0.18, and 0.16 Mbps. The TCP throughput decreases up to about three-fold as the number of hops increases from 1 to 2. It is also observed that TCP throughput decreases monotonically with the number of hops. In the future ubiquitous ad hoc networking environments, several multi-hop networks are seemed to be more frequently used in the form of integrating into the Internet (e.g., cellular networks supplemented with ad hoc networks) than the pure multi-hop ad hoc networks which have tens of hops routes, so we argue DPS approach is highly effective for ubiquitous ad hoc networks.

5. Conclusion and Future Work

We have proposed DPS, an adaptive route path compression scheme for mobile ad hoc networks. Our approach is highly responsive to the conventional human mobility (e.g., pedestrian and slow vehicle in our daily life) by using the wireless link quality value: SSNR. As a case study of DPS scheme, we embedded DPS in the prominent DSR and AODV protocols. Performance analysis by means of simulation demonstrates significant improvement with regard to the number of routing packets and end-to-end data packet latency in almost scenarios. We have also proposed two novel mobility models which generate more realistic node mobility than the random-waypoint model. Additionally, DPS shortens active routes in the order of milliseconds (about 5 ms) in our real DPS implementation.

We are currently working on studying the effects of DPS in case of using the promiscuous listening mode of network interfaces for shortening multi-hop active paths dynamically while taking into account its influences to power consumption. In this paper, though we consider devices operating in the ISM bands (such as Lucent WaveLANs or IEEE 802.11b wireless LAN cards), we hope to choose a flexible signal threshold value taking into account the situated environments since signal power of received packets is dependent on what wireless devices are used.

Acknowledgements

We would like to be extremely thankful to Yosuke Tamura for the valuable advice and discussions. We also thank Mika Minematsu and the anonymous referees for their constructive comments which helped to improve the quality of this paper.

References

[1] J. Broch, D. Maltz, D. Johnson, et al. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of ACM/IEEE MobiCom'98*, Oct. 1998.

[2] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626, Oct. 2003.

[3] S. Das, C. Perkins, and E. Royer. Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks. In *Proceedings of IEEE INFOCOM'00*, pages 3–12, Mar. 2000.

[4] A. Jardosh, E. Belding-Royer, K. Almeroth, and S. Suri. Towards Realistic Mobility Models For Mobile Ad Hoc Networks. In *Proceedings of ACM MobiCom'03*, Sept. 2003.

[5] P. Johansson, T. Larsson, N. Hedman, and B. Mielczarek. Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks. In *Proceedings of ACM MobiCom'99*, Aug. 1999.

[6] D. Johnson and D. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks*. In Mobile Computing, edited by Tomasz Imeilinski and Hank Korth, chapter 5, pages 153-181, Kluwer Academic Publishers, 1996.

[7] D. Johnson, D. Maltz, and Y. Hu. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. IETF Internet-Draft [Work in Progress], Apr. 2003.

[8] R. Ogier, M. Lewis, and F. Templin. Topology Dissemination Based on Reverse Path Forwarding (TBRPF). IETF Internet-Draft [Work in Progress], Oct. 2003.

[9] Orinoco, Inc. The WaveLAN Home Page. http://www.wavelan.com, 1998.

[10] C. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of ACM SIGCOMM'94*, Aug. 1994.

[11] C. Perkins, E. Royer, and S. Das. Ad Hoc On-Demand Distance Vector (AODV) Routing. RFC 3561, July 2003.

[12] S. Roy and J. Garcia-Luna-Aceves. Using Minimal Source Trees for On-Demand Routing in Ad Hoc Networks. In *Proceedings of IEEE INFOCOM'01*, Aug. 2001.

[13] The Rice University Monarch Project. Mobile Networking Architectures. http://www.monarch.cs.rice.edu/, 2003.

[14] The VINT Project. Network simulator - ns2. http://www.isi.edu/nsnam/ns, 2001.