アドホックネットワークにおける動的経路短縮方式

斉藤 匡人[†] 間 博人[‡] 戸辺 義人[‡] 田村 陽介[‡] 徳田 英幸[‡,§]

[†] 慶應義塾大学 総合政策学部, [‡] 大学院 政策・メディア研究科, [§] 環境情報学部
{masato, haru, tobe, tamura, hxt}@ht.sfc.keio.ac.jp

本論文では、アドホックネットワークにおける動的経路短縮方式である、"OR2" について述べる。従来のアドホックネットワーク経路制御方式は、通信経路確立時と通信リンク切断時にのみ経路の再構築を行う。しかし、これではノードの移動に伴うノード間の近接を認識できないため、よりホップ数の少ない経路へと適応できない。OR2 はデータ通信中においても、リンク品質を基にして、ノードの移動に適応的な経路制御を行うことで最適なマルチホップ経路を再構築する。本稿では、FreeBSD OS 上における OR2 プロトタイプの実装と、実環境における評価結果及び有用性を示す。

A Dynamic Path Shortening Scheme in Ad Hoc Networks

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

[†]Faculty of Policy Management, [‡]Graduate School of Media and Governance,
[§]Faculty of Environmental Information, Keio University

This paper presents "OR2", an adaptive path shortening scheme for mobile ad hoc networks. In OR2, the active path adapts dynamically to node mobility without any link failures based on the local link quality. Most conventional routing protocols accommodate the change of network topology only when the link fails. By introducing the notion of *proximity* that indicates the nearness of two communicating nodes, OR2 skips the neighbor upstream node in a proximity area and reduces the hop count of an active path, and continues to shorten an active path as possible. We have implemented OR2 as an extension to DSR on FreeBSD. The experimental results have shown that OR2 is effective in enhancing TCP throughput and reducing end-to-end delay for all relevant flows.

1 Introduction

As popularity for mobile computing increases, cooperative communications with wireless devices are becoming an attractive technology. A key challenge to succeed in such communications is adapting to node mobility. A mobile ad hoc network is a group of mobile computing devices (nodes) which communicates with each other using multi-hop wireless links. It does not require any stationary infrastructure such as base stations. Each node in the network can act as both a host and a router forwarding data packets to other nodes.

There are several scenarios where ad hoc networks are useful. One major application is a military-use communication in a battlefield where a centralized configuration is difficult. Another application is emergency communication in disaster areas. In addition to these large-sized applications, we can use ad hoc networks when several people have meetings with computers that are equipped with wireless interfaces. Also, it can be interesting research for supporting intelligent transport systems and sensor networks.

One important issue for achieving efficient network resource utilization is to update route information 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 change of the network topology and find an efficient route between two communicating nodes. A number of research for mobile ad hoc networks has focused on the development of their routing protocols (e.g., DSR [1], AODV [5], LAR [3], SOAR [6]). The key advantage behind on-demand protocols is the reduction of routing overheads so that on-demand routing protocols maintain only active paths to those destinations to which data must be sent. Minimizing the routing overhead is effective in such a dynamic environment of ad hoc networks due to limited available bandwidth, unpredictable nodes mobility, battery outages, interference and high bit error rates.

These above on-demand routing protocols accommodate route changes only when an active path is disconnected. They cannot adapt to the change of network topology even if another route with less hop count becomes available by the movement of intermediate nodes unless any link is disconnected. DSR protocol [1] only has the mechanism that shorten an active path which is not driven by link failures but by overhearing packets by operating the network interfaces in promiscuous receive mode. This promiscuous mode, however, requires greater CPU cycles, power consumption and sending delay due to overheard packets. In contrast to the conventional protocols, we propose Optimized Reconfigurable Routing (OR2) algorithm that tunes up an active path adaptive to node mobility without any link disconnection based on Smoothed Signal-to-Noise Ratio (SSNR) as a link quality value indicator. OR2's adaptation to node mobility leads to the reduction of a hop count and path delay which significantly improves the performance of Transmission Control Protocol (TCP) flows. Since TCP is the de facto standard for reliable unicast data transport in the Internet today, its use over ad hoc networks is also a certainty.

In order to shorten an active path, we introduce the notion 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 from their local network interfaces. Thus, we can change the active path while preserving stable link connectivity. OR2 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 OR2 operates only when forwarding or receiving data packets, it does not require periodic HELLO messages or advertisements when there are no link connectivity changes in the data path. In this work, we have designed OR2 as an extension to DSR [7] which is one of the best performing on-demand routing protocols and implemented OR2 on FreeBSD. The experimental results have shown that OR2 is effective in enhancing TCP throughput and reducing end-to-end delay for all relevant flows. The overhead incurred with OR2 is sufficiently negligible.

The rest of this paper is organized as follows. Section 2 briefly describes some proposed protocols in mobile ad hoc networks. Section 3 describes a design and a detailed description of OR2, and Section 4 explains the implementation of it. Then, we present the results and analysis of several experiments in Section 5. Section 6 discusses another usage of SSNR. Finally, we present our conclusions and discuss some future works in Section 7.

2 Related Work

This section describes conventional on-demand routing protocols in mobile ad hoc networks and other protocols using link-state information.

Dynamic Source Routing (DSR) [1, 7] 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 as well as our OR2 scheme while sending data packets. However, this scheme requires an always-active transceiver mode of the network interfaces and more CPU cycles to process overheard packets, 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 [6] presents the source-tree on-demand adaptive routing protocol (SOAR) based on link-state information. SOAR has the mechanism to shorten the active paths, but it achieves that by *periodically* exchanging the link-state information consisting of the minimal source trees in the paths with its neighbors while sending data packets. These periodical messages could lead to the collision with the data streams in wireless networks.

In contrast to the above works, "OR2" does not lead to the weak-connectivity shortened routes or inefficient frequent routes 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. OR2 adapts effectively to node mobility using local link quality in wireless ad hoc networks which are scarce bandwidth and battery environment.

3 OR2

This section describes the details of OR2. Firstly, we explain a case in which a data path becomes redundant due to the movement of nodes. Secondly, we introduce the notion of proximity to identify two near nodes by using link quality. Finally, we explain the design of OR2 based on the identification of nodes in proximity.

3.1 Path Inefficiency

In a mobile ad hoc network, due to node mobility, we encounter a situation shown in Figure 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 Figure 1) than the current route in use.

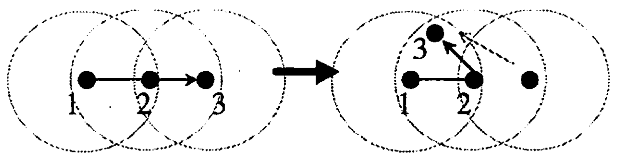


Figure 1: Node 1 sends packets to Node 3 through Node 2. At the next step Node 3 moves into the cell of Node 1 without link failures. Although Node 1 can directly send packets to Node 3, Node 1 still sends packets to Node 3 through Node 2.

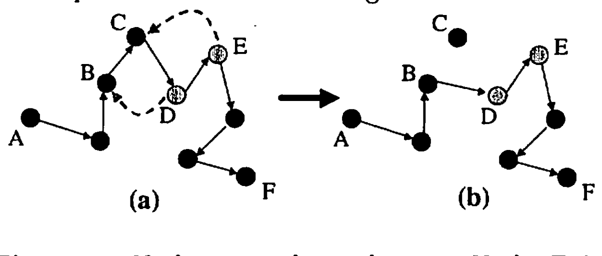


Figure 2: Node A sends packets to Node F in a multi-hop network. By using OR2's algorithm, Node D and E can shorten the path currently in use preserving the consistency of the active path.

This scenario in particular is likely to occur frequently in the realistic environment with pedestrian or slow vehicle speed in our daily life. However, the most of the previous routing protocols cannot accommodate the change of network topology without any link failures. Thus, there exists the path inefficiency in respect to the hop count, network capacity and power consumption while communicating with other nodes in an ad hoc network. We eliminate the inefficiency by using local link information and the concept of *proximity* in the next section. On the other hand, OR2 also finely accommodates large-scale and dense networks since it is decentralized algorithm using local link quality information. Figure 2 shows a more complicated scenario in which OR2 is tuning up the active path adapting to node mobility. In the figure, some less hop routes are available in the active path from source to destination. If each neighbor node simultaneously shortens the active path (in Figure 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 new route discovery. We describe how to overcome this problem later.

3.2 Link Quality

It is desirable for a node on a path to determine whether or not it can shorten the path based on some indicators of the quality of the link between the node and its neighbors on the path. For such an indicator, we use the Signal-to-Noise Ratio (SNR) of the link associated with receiving packets. By definition, SNR represents a channel condition and is expressed as the ratio of signal to noise in electrical power. When the value of SNR becomes higher, the link communication quality will also be relatively higher. However, it should be noted that the SNR could change dynamically with a high frequency due to electro-magnetic effects. From the point of view of measuring the link quality, we rather obtain a smoothed value of SNR in a time domain. 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 implementation. In OR2, the filter calculates SSNR whenever a node receives the frames.

Let consider the situation which two nodes approach each other. If the distance between two nodes is associated with the SSNR of the link between the two nodes, one of the pair can determine whether the other one is near the own position. In order to investigate this assumption (i.e., the relationship between distance and SSNR), we transmitted a constant rate UDP stream at 1 Mbps between two wireless nodes with IEEE 802.11b NIC and changed the location of the receiver. As seen in the Fig.3, the value of SSNR rises considerably as the distance becomes smaller (< 10m). This experiment was performed in our flat rectangular ground (300m x 300m) with no obstacles or walls. By deciding some optimal threshold value of receiving SSNR, it is possible to define the nearness of the other node. In addition, we found that the values of SSNR larger than some particular value (e.g., 10 dB) were highly stable in terms of throughput as shown in Fig.3 of the second experiment result with WaveLAN NICs [9]. Thus we possibly assume that some highly receiving SSNR value indicates the nearness of the two nodes or the good condition of the link. OR2 assumes that transmission power can not be varied and all nodes in an ad hoc network have the same network interfaces.

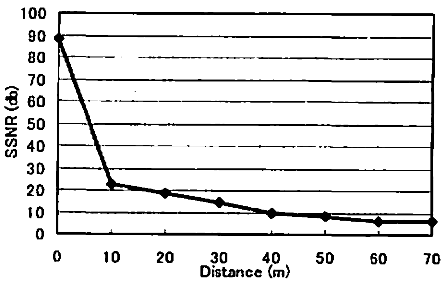


Figure 3: SSNR vs. Distance

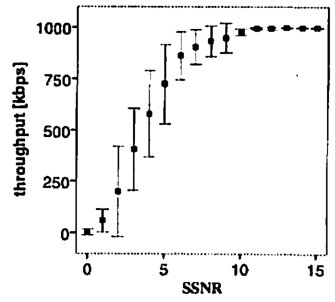


Figure 4: SSNR vs. throughput

3.3 Proximity

To argue the "nearness" of two nodes more formally, we introduce the notion of proximity based on the observation of the relationship between the distance and the SSNR between two nodes. Let us define the following symbols.

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

We hypothesize that $S_{(AB)} = S_{(BA)}$. This is not impractical since homogeneous nodes are assumed in many mobile ad hoc networks. We will discuss a case in which this assumption does not hold in the future. 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 $A \in P_{(B)}$.

Let assume that a flow traverses Node A, B, and C in this order. This can be written as $A = R_{uf}(B) = R_{uf}(R_{uf}(C)) = R_{uf}^2(C)$. If $C \in P_{(B)}$, there is a possibility that the path of the flow can be changed: $A = R_{uf}(C)$. As shown in Figure 5, 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. This motivates us to design our scheme described in the next section. In practice, we need a hysteresis mechanism around the threshold value to avoid oscillation.

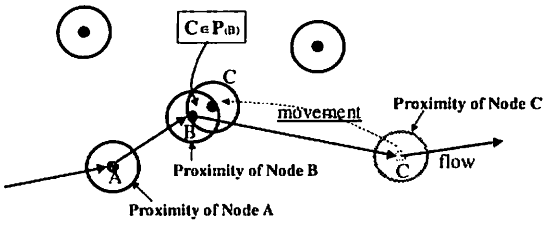
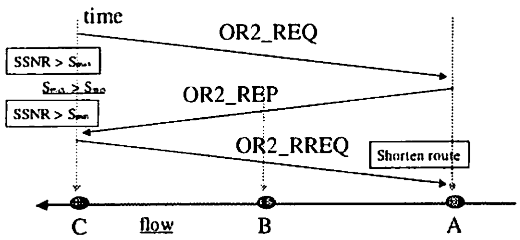


Figure 5: Proximity of node

3.4 Design of OR2

We set two design goals to OR2: 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 that does not produce many control packets. In particular, we do not allow transmission of control packets when active flows do not exist. This is an important consideration for an ad hoc network since nodes in the network need to reduce their power consumption. We design a scheme in which control packets are transmitted only when a node determines that a path should be changed based on the proximity. We call this scheme OR2. In designing OR2, we made an assumption that 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. In this work, we confine ourselves to apply OR2 to a source-routing protocol such as DSR. In the following, we describe the details of OR2.



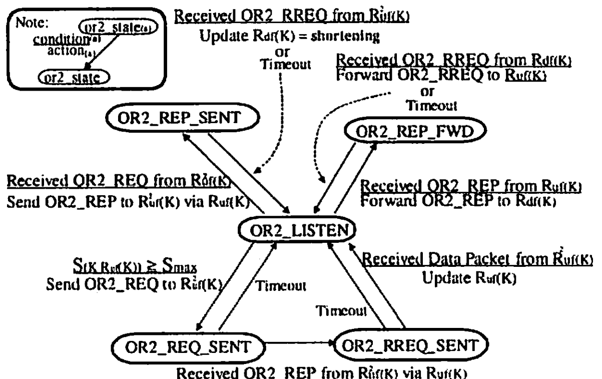
OR2_REQ	Request to $\dot{R}_{uf}(k)$ as a packet to probe direct route
OR2_REP	Reply to $\dot{R}_{df}(k)$ via $R_{df}(k)$ for OR2_REQ (piggybacked)
OR2_RREQ	Redirect Request to $\dot{R}_{uf}(k)$ via $R_{uf}(k)$ to shorten route

Figure 6: Three OR2 control packets

Let us now explain the fundamental messages passed among three nodes. OR2 uses three kinds of messages: OR2_REQ, OR2_REP, and OR2_RREQ; they are shown in Figure 6. OR2_REQ and OR2_RREQ are newly defined control packets, while OR2_REP is piggybacked on a data packet as a DSR-header option. Let us assume that $A = R_{uf}(B)$ and $B = R_{uf}(C)$ for flow f as shown in Figure 6.

When Node C determines that it has moved into the proximity of Node B, it sends OR2_REQ to Node A. The intent is to observe whether or not a packet can be directly exchanged between Node A and C. Upon receipt of OR2_REQ, Node A sends OR2_REP to Node C. Unlike OR2_REQ, OR2_REP is not sent as a single control packet. Rather, Node A inserts it as a DSR option header into the data packet of flow f. Therefore OR2_REP reaches Node A via Node B. By receiving OR2_REP, Node C knows that Node A can send packets directly to Node C; Node C sends OR2_RREQ to Node A to initiate a change of route. The extra packets of OR2_REQ and OR2_RREQ may temporarily interfere with data packets. However, the overhead incurred with the packets is still negligibly small compared with an alternative scheme using HELLO messages.

There is concern about a 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 a way similar to TCP's three-way handshake but in a more delicate way to handle mutual exclusion. Considering the above problem, let us describe the protocol of OR2. The state transition at node K of flow f is shown in Figure 7 and the handling of the race condition is shown in the following Figure 8.



Note:
or2_state(a)
condition(a)
action(a)
or2_state
Received OR2_RREQ from $\dot{R}_{uf}(K)$
Update $R_{df}(K)$ = shortening
or
Timeout
Received OR2_RREQ from $R_{df}(K)$
Forward OR2_RREQ to $R_{uf}(K)$
or
Timeout
OR2_REP_SENT
OR2_REP_FWD
Received OR2_REQ from $\dot{R}_{df}(K)$
Send OR2_REP to $\dot{R}_{uf}(K)$ via $R_{uf}(K)$
Received OR2_REP from $R_{uf}(K)$
Forward OR2_REP to $R_{df}(K)$
OR2_LISTEN
$S_{(K\ R_{uf}(K))} \geq S_{max}$
Send OR2_REQ to $\dot{R}_{uf}(K)$
Timeout
Received Data Packet from $\dot{R}_{uf}(K)$
Update $R_{uf}(K)$
Timeout
OR2_REQ_SENT
OR2_RREQ_SENT
Received OR2_REP from $\dot{R}_{df}(K)$ via $R_{uf}(K)$
Send OR2_RREQ to $\dot{R}_{uf}(K)$ via $R_{uf}(K)$

Figure 7: OR2 state transition diagram at node K

```
node K :

Received OR2_REQ, OR2_REP or OR2_RREQ
if (timeout)
   goto OR2_LISTEN
else if ($or2\_state_{(a)}$&$condition_{(a)}$)
     exec $action_{(a)}$
else drop the message
```

Figure 8: OR2 solution of race condition

Let us 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 OR2_REQ to Node $R^2_u(C)$ (i.e., Node A) to locate the direct hop route. As long as $S_{(BC)} \geq S_{max}$, Node C continues to send OR2_REQ every time OR2's timer expires until Node C receives OR2_REP. Upon successful receipt of OR2_REP, Node C sends OR2_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.

Let us consider a case in which Node D is also attempting to make a short cut between Nodes B and D. Node D sends OR2_REQ to Node B. When Node B receives OR2_REQ, the state of flow f at Node B moves to OR2_REP_SENT. If there is an OR2_REP message from Node A to C, it traverses Node B. When this OR2_REP message reaches Node B and the state is OR2_REP_SENT, the message is discarded since the short cut between Nodes B and C is on-going. Thus the short cut from Node B to C is prioritized. In contrast, if an OR2_REP message from A to C reaches Node B ahead of an OR2_REQ message from B to D, the state of B changes to OR2_REP_FWD and suppresses the short cut from Node B to C.

The pseudo-code summarizing the salient feature of our algorithm is shown in Figure 9. This figure mainly depicts the action of the initiator which starts OR2's procedures.

OR2 Notations:

```
is_reply()     : Is this OR2_REP?
probe          : Flag indicating probing now.
Smax           : SSNR max threshold of Proximity.
Smin           : SSNR min threshold for hysteresis.
conform()      : Send OR2_RREQ for shorting route.
probe_route()  : Send OR2_REQ.
```

Shortening path:

```
Each packet arrives
reply_flag = is_reply();
if (!reply_flag && probe)
   return;
SSNR = calc_SSNR();
if (reply_flag && SSNR > Smin)
   conform();
else if (SSNR > Smax)
   probe_route();
   probe++;
```

Figure 9: Pseudo-code of OR2

4 Implementation

OR2 scheme is built on off-the-shelf wireless LAN technology. We have implemented OR2 as an extension to DSR developed by the Monarch project [7]. Since the implementation of DSR is for FreeBSD 3.3-RELEASE and mainly WaveLAN [9] cards, in our implementation, DSR is ported to FreeBSD 4.2-RELEASE and modified to retrieve the SSNR values from IEEE 802.11b [2] wireless LAN cards. In addition, we extended "wi" driver: */sys/i386/isa/if_wi.c* to obtain the pair of the source IP address and the SSNR which is supplied by the register on IEEE 802.11b cards when a frame arrives. Since the SSNR value is retrieved every time a frame arrives, the SSNR value accurately reflects up-to-date link quality. 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 is relatively adaptive to the data send rate.

In addition, as control packets to initiate changes to the active path, we added OR2_REQ, OR2_REP, and OR2_RREQ header options as one of DSR header option types: */sys/dsr/ip6_opts.[h,c]*. OR2_REP is always piggybacked on a data packet. Additionally, we added some routines which send and receive OR2_REQ, OR2_REP, and OR2_RREQ in */sys/dsr/dsr_output.c* and */sys/dsr/dsr_input.c*. Specifically, the DSR option header is inserted following DSR header after IP header, followed by headers such as a transport layer header.

5 Experiments

In this section, we show some experimental results. In our experiments, mobile nodes are Pentium-based laptop computers running FreeBSD 4.2 and equipped with a MELCO IEEE 802.11b wireless network card. We installed DSR and OR2 in these nodes and conducted two preliminary experiments: measurement of the latency in re-routing paths, and quantification of improving TCP throughput by reducing the number of hops.

5.1 Latency

To observe the overhead associated with path shortening, we 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. For comparison, we measured the round-trip time (RTT) from C to A. The result of measuring the RTT is shown as "Ping" in Figure 10. To create the situation of path shortening, we moved Node C close to Node B. We measured the duration from the time at which Node C sent the first OR2_REQ message to the time at which Node C received data packets directly from Node A. As observed in Figure 10, the overhead incurred with the exchange of OR2's messages is sufficiently small; it is less than 5 ms.

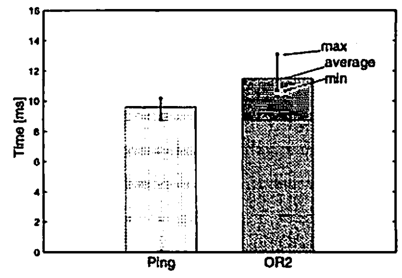


Figure 10: Network delay (ping) and latency time of OR2 to shorten an active path over two-hop route

5.2 TCP Throughput vs. Number of Hops

We also examined the relationship between TCP throughput and the number of hops. We used netperf [4] to send TCP flows. Figure 11 shows the obtained results. As seen in the figure, the TCP throughput dramatically decreases as the number of hops increases from 1 to 2. It is also observed that TCP throughput decreases monotonically with the number of hops. Therefore reducing the number of hops performed by OR2 will leads to significant improvement in TCP throughput.

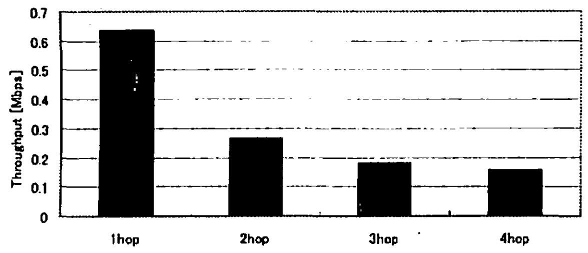


Figure 11: TCP throughput vs. number of hops

6 Discussion

Although we use SSNR as a metric of proximity in this work, SSNR also can be used as another metric. By monitoring the differential values of SSNR and rate of received packets, we can determine whether or not two nodes are moving apart. If these differential values decrease, we can know that the distance between these nodes is increasing. These values can be obtained even with TCP ACK packets from a downstream. When we hypothesis $S_{(R_{uf}(K)K)} = S_{(KR_{uf}(K))}$, we can estimate the downstream link quality of the route sending TCP data packets by the SSNR of TCP ACK packets from $R_{df}(k)$.

For a reliable transport protocol like TCP, it is smarter to control the data transmission rate adaptively to the degradation of link quality and the possibility of link disconnection. Thus, by using the history of SSNR, we presumably identify and estimate the decrease of link quality and link disconnection to avoid making the link wastefully congested and prevent numerous packet loss. Additionally, we could switch to a more stable route in advance. However, to extract the information on such link quality, an extra overhead of processing is posed at each node. Evaluating this overhead remains our future work.

7 Conclusion and Future Work

We have proposed OR2, an adaptive path tuning algorithm for mobile ad hoc networks. Since most conventional routing protocols accommodate topology changes only when an active path is disconnected, it is not suitable for node mobility. Our approach is more adaptive to our conventional node mobility (e.g., pedestrian and slow vehicle in campus computing) using the wireless link quality value: SSNR. OR2 shortens an active path adaptive to node mobility by using the notion of *proximity*. This scheme achieves a significant reduction of a path delay while the links are still active. As a result, it is highly effective for overall network capacity and power consumption in limited resource environments such as multi-hop wireless networks. Also, reducing path delays is especially important for TCP flows. In OR2, each node monitors local link quality only when receiving packets and makes local decisions in a decentralized manner. There is no need to exchange periodic control information such as HELLO messages.

We have designed OR2 as an extension to DSR and implemented it on FreeBSD. The experimental results have shown that OR2 is effective in enhancing TCP throughput and reducing end-to-end delay for all relevant flows. Also, since the latency time of shortening an active path is on the order of tens of milliseconds, our scheme is appropriate for slow node mobility in our daily life.

The work presented in this paper is the preliminary phase. The experimental results are shown only for three nodes network, and so we need to verify the application of OR2 for large scale networks. Currently we are implementing our scheme OR2 using the *ns-2* [8] network simulator to investigate how OR2 performs in environments varying in network load, mobility and network size, particularly in a large-scale ad hoc network environment. This simulation results will enable us to compare it with our current experimental results. And also we still need to analyze the decision of the SSNR threshold S_{max} value and the comparing frequency because these factors have important impact on the effectiveness of OR2. We also will be re-designing our OR2 not to be dependent on a source-routing protocol model like DSR and for node K to know the IP address of node $R^2_{uf}(K)$. Finally, while OR2 targets the "one-hop path shortening" between nodes that are adjacent with a single intermediate node, we also need to consider how OR2 can be extended to cases where path shortening is not necessarily restricted to three adjacent nodes (i.e., "N-hop path shortening"). The limited promiscuous listening approach might be effectively introduced to the next OR2.

References

[1] BROCH, J., JOHNSON, D. and MALTZ, D. The Dynamic Source Routing Protocol for Mobile Ad Hoc Netoworks, IETF Internet-Draft [Work in Progress] (Mar. 2001).

[2] IEEE 802.11 STANDARD (IEEE COMPUTER SOCIETY LAN MAN STANDARDS COMMITTEE), *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications* (June 1999).

[3] KO, Y. and VAIDYA, N. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks, Proceedings of ACM MOBICOM'98 (June 1998).

[4] *Netperf*, http://www.netperf.org/.

[5] PERKINS, C., ROYER, E. and DAS, S. Ad Hoc On Demand Distance Vector (AODV) Routing, IETF Internet-Draft [Work in Progress] (Mar. 2001).

[6] ROY, S. and GARCIA-LUNA-ACEVES, J. Using Minimal Source Trees for On-Demand Routing in Ad Hoc Networks, Proceedings of IEEE INFOCOM'01 (Aug. 2001).

[7] The MONARCH Project at Carnegie Mellon University, http://www.monarch.cs.cmu.edu/.

[8] The VINT Project, http://www-mash.cs.berkeley.edu/ns *ns-2 network simulator (ver 2)*.

[9] *The WaveLAN Home Page*, http://www.wavelan.com (1998).