Using Link-Layer Broadcast to Improve Scalable Source Routing

Pengfei Di
Computer Science Department,
TU Munich, Germany
di@in.tum.de

Thomas Fuhrmann
Computer Science Department,
TU Munich, Germany
fuhrmann@net.in.tum.de

ABSTRACT

Scalable source routing (SSR) is a network layer routing protocol that provides services that are similar to those of structured peer-to-peer overlays.

In this paper, we describe several improvements to the SSR protocol. They aim at providing nodes with more up-to-date routing information: 1. The use of link-layer broadcast enables all neighbors of a node to contribute to the forwarding process. 2. A light-weight and fast selection mechanism avoids packet duplication and optimizes the source route iteratively. 3. Nodes implicitly learn the network's topology from overheard broadcast messages.

We present simulation results which show the performance gain of the proposed improvements: 1. The delivery ratio in settings with high mobility increases. 2. The required per-node state can be reduced as compared with the original SSR protocol. 3. The route stretch decreases. — These improvements are achieved without increasing the routing overhead.

Categories and Subject Descriptors

C.2.2 [**Computer Communication Networks**]: Network Protocols—*Routing protocols*

General Terms

Algorithms

Keywords

Routing, Peer-to-peer, Mobile Ad-hoc Network

1. INTRODUCTION

Routing in mobile ad-hoc networks (MANETs) has been a challenging research topic for more than a decade. Such networks operate in a difficult environment: The quality of the radio channel is typically unstable. Nodes move during operation (node mobility). Nodes may leave the network ungracefully (node churn). As a result, routing information quickly becomes outdated, and a node might not be able to forward a given packet as requested.

In most routing protocols the forwarder determines a packet's next hop based on its local routing information. In source routing protocols, the sender typically determines the entire hop sequence. In both cases, the routing information is cached and used for a while, even when it has been acquired in response to actual demand. Thus, the nodes often do not use the most up-to-date information, and they cannot select the most suitable next hop. As a result, the forwarding path is sub-optimal. In the worst case, a destination may become unreachable until new routing information has been acquired.

In this paper, we describe an extension of the SSR protocol that exploits the broadcast nature of radio channels. As we will show, it improves SSR's robustness against node mobility and node churn without increasing SSR's traffic overhead and per-node state. The key idea is to combine SSR with a joint forwarding decision among all the nodes in the local radio range. Thereby, each individual packet will be forwarded to a node that provides the best trade-off between channel quality and reachability of the destination.

We consider this improvement important especially for mobile peer-to-peer applications. SSR provides the key based routing service of a structured peer-to-peer overlay directly in the network layer. Thereby it avoids the overhead of running a peer-to-peer overlay such as Chord [13] on top of a network layer routing protocol such as AODV [11].

This paper is structured as follows: Section 2 briefly reviews other MANET routing protocols that use link-layer broadcast. Section 3 gives a brief overview of SSR. Section 4 describes our proposed enhancement of SSR. Section 5 evaluates the resulting performance improvement. Finally, section 6 concludes this paper.

2. RELATED WORK

Virtual ring routing(VRR) [3] and SSR [5] are two DHT-inspired routing protocols. Both protocols are based on the concept of a virtual ring. Each node maintains paths to its virtual neighbors in the ring. VRR stores the path information along the path, like AODV does. SSR stores the paths at the respective node, and uses source routing like DSR [9]. Both protocols achieve better scalability than AODV and DSR because each node has just a limited knowledge of the network topology. When routing to an unknown destination, the sender selects a known intermediate node that is virtually closer to the destination than the sender. This continues recursively until the packet has reached its final destination. Since each next intermediate destination is virtually closer to the destination, it is guaranteed that the routing process always makes progress. The resulting path, however, will typically be longer than the direct path AODV and DSR would have discovered.

Although these two routing protocol solve the scalability problem for routing in ad-hoc network, they don't exploit the medium's potential broadcast property. In this paper, we show that by using broadcast messages, we can increase the performance of SSR significantly.

The idea to use link-layer broadcast to improve routing performance has first been proposed in the context of geographic routing protocols. In general, geographic routing protocols forward a packet to the neighbor that is geographically closest to the destination. If the nodes move quickly, the respective forwarder might not have the most accurate information about the (relative) position of its neighbors. Thus, it is beneficial to determine the next hop node on the fly.

The Implicit Geographic Forwarding (IGF) [2] is a stateless geographic routing protocol. It modifies the 802.11 DCF medium access control (MAC) protocol [8] such that information about the destination will be added to the RTS/CTS handshaking frames. The closer a node is to the destination, the earlier it sends a CTS. The primary sender will then send the DATA packet to the node that first sent its CTS.

Beacon-Less Routing (BLR) [4, 7] use link-layer broadcast to send a data packet to the next hop. The closer a node to the destination [7] or to the forwarder [4], the earlier it forwards the data. Thereby, it also suppresses further forwarding of that packet. In order to ensure that all other nodes can overhear the resending, BLR requires the specification of a so-called forwarding area, where the nodes can hear each other. Only nodes in this area are allowed to resend the data packet.

In error-prone networks, several opportunistic routing protocols [1, 12] have been proposed to reduce the packet loss probability.

Biswas et al. proposed the Extremely Opportunistic Routing (ExOR) [1] protocol. Here, the sending node assembles its known neighbors into a candidate list. This list is sorted such that nodes that are known to have a smaller hop distance to the destination are preferred. All candidates that receive the packet send an acknowledgment. Then the candidate with the highest priority forwards the packet. If two candidates cannot hear their respective acknowledgment, the packet will be duplicated. Thus, to avoid loops, packets are uniquely identified and transmitted only upon their first arrival.

Sanchez et al. [12] suggest a DATA-first handshaking sequence to avoid retransmissions in error-prone networks. Nodes that received a DATA packet acknowledge it according to a delay function. Again, the delay function prefers nodes closer to the packet's destination. The sender sends a selection packet on the first ACK, thereby suppressing data duplication.

2.1 Contribution of This Paper

Geographic routing protocols require all nodes to know their position. This is at least difficult in many scenarios such as indoor communication. Furthermore, since these protocols consider only geographic distance, they are unable to deal efficiently with obstacles and wireline shortcuts.

Opportunistic routing protocols need to know the global network topology. This is difficult in large networks. It is especially difficult in networks where nodes are mobile and the radio channel quality varies quickly.

Peer-to-peer protocols that run on top of a MANET routing protocol suffer significantly from such problems. Structured P2P systems spend much effort for state maintenance. The more churn in the network layer, the worse that maintenance overhead. Moreover, in order to be efficient, P2P protocols need to consider the physical proximity of the peers. Measuring proximity at the overlay layer, for example, by using a ping-like mechanism, cannot provide timely results.

Scalable source routing combines structured peer-to-peer overlay routing with network layer routing. In this paper, we propose the use of link-layer broadcast to improve the performance of SSR in MANET scenarios. We present a light-weight and fast selection mechanism. It avoids the long delays and packet duplication that occur in geographic and opportunistic routing.

We show that this enhancement allows SSR to operate robustly in networks with high node mobility and node churn. Furthermore, we show that our improvement allows to further reduce the required per-node state as compared to the unmodified SSR without introducing additional overhead. Depending on the network size we find a possible reduction of about 30% in the nodes' route caches. In addition, this enhancement is also applicable to VRR.

3. SCALABLE SOURCE ROUTING

Scalable source routing (SSR) provides DHT-inspired routing at the network layer. It combines source routing with Chord-like routing in an address space which has the structure of a virtual ring.

Unlike typical link-state and distance-vector routing protocols, SSR does not maintain state information for all destinations in the network, and does not have to flood the network to discover the path. A node is only required to maintain source routes to its virtual neighbors in the address space (cf. the Chord ring). These source routes are constructed using an iterative consistency algorithm [10]. SSR assumes that a node provides additional memory that can be used to cache further source routes. These source routes may be copied from the packets that a node forwards, and be aggregated in a tree format. Each entry in the route cache can be considered as a finger in a Chord ring. In [5], we showed that a small route cache containing only 256 entries suffices even for very large networks of up to 100 000 nodes.

SSR uses two distance metrics for its forwarding decisions: firstly D_{phy}, the physical distance to the nodes in route cache, measured normally by "hop", and secondly D_{vir}, the virtual distance of two nodes as determined by the absolute value of the numerical difference of the nodes' addresses.

SSR packets contain a source address, a destination address, and a source route. The selected source route does not necessarily span the entire path from the source to the destination. It could prematurely end at some intermediate node. In that case, this intermediate node (i.e., the last node contained in the source route) will try to append another source route (from its route cache) such that the source route either covers the entire way up to the destination (if such a route is available in the intermediate's cache), or the next intermediate is virtually closer to the destination than the current intermediate. Typically, a node has many potential next intermediates available in its cache. In this case, it will prefer the node which is physically closest to the current intermediate.

Note that by construction of this routing process SSR is guaranteed to make progress in each step in a consistent network. The routing process will end at the node that is virtually closest to the destination address. This property is called *key based routing*.

4. IMPROVING SSR

Like opportunistic routing, our proposed SSR enhancement uses a prioritized forwarding candidate set. The sender sends this set before the data packet. The nodes in the candidate set that have successfully received the data packet, acknowledge their receipts. They do so until the sender has determined one of the candidates to forward the data. The acknowledgments include a quality assessment of the candidates' ability to forward the data packet. This enables the sender to learn who is the best candidate to forward the data. Note that in most cases, the sender selects the node that first sent its acknowledgment. This minimizes forwarding delay.

With our enhancement proposal, all packets[1] are forwarded using link-layer broadcast. Before each DATA packet a CTS packet conveys the candidate list. (In fact, the CTS is piggybacked on the DATA packet such that they share one link-layer broadcast frame.) The node sequence in the candidate list specifies the response sequence of the candidates. It is filled by the forwarding node following the original SSR's routing metrics. The first slot in the candidate list is called *default candidate*.

Energy-constrained nodes should appear in the candidate list only as default candidate. If the MAC layer protocol could be modified so that a non-candidate will not receive the DATA, this would save energy.

After receiving the DATA packet, each of the candidates calculates its forwarding ability for this packet and will respond to the receipt of the DATA packet with an ACK packet after $T = \delta t * n$, where n is the position in the candidate list, starting from 0. Note that applying a discrete delay for ACKs avoids packet collisions and the value of δt depends on the respective values of the MAC protocol. Each ACK includes the the acknowledging node's forwarding ability of the DATA packet.

Each candidate node caches the DATA packet until it receives the RTS from the sender or a timeout occurs if the RTS was lost. (The cache size depends on the individual node's capacity. The timeout value depends on the size of the candidate list and the respective properties of the MAC protocol.) The RTS contains the ID of selected candidate and its routing ability. The candidates except for the selected one drop the DATA packet and use the ability value to decide whether to drop the scheduled ACK or reschedule it.

If the *default candidate* fails, other candidates have their chance to quickly stand in for that packet by sending back the scheduled ACK. Now the sender can either immediately trigger the forwarding of the DATA packet by broadcasting a respective RTS (cf. Fig. 1(a)), or it can wait for further ACKs before deciding which node shall forward the packet(cf. Fig. 1(b)). Note that in the first variant, the pause between the ACKs must be sufficiently large, in order to wait for a potential RTS; while in the second variant, a smaller pause between the ACKs is allowed.

Clearly, if the default candidate manages to acknowledge the DATA packet, none of the other candidates would have a chance to forward the *current* DATA packet. However, if a candidate detects that it would have been better suited to forward the packet than the default candidate, it may send an additional ACK after the forwarder has sent the DATA packet (cf. fig. 1(c)). This enables the sender to optimize its candidate list for *future* DATA packets. Note that the ACK has to be rescheduled so that this ACK is not likely to collide with ACKs and the RTS of the next forwarding hop.

4.1 Ability Calculation

SSR chooses its next intermediate node based on the candidates' virtual and physical distance. We propose to extend the choice of forwarding candidates by another parameter: the node's liveliness. In detail, we define the *routing ability* of a candidate with respect to a given destination as follows:

$$Q = \begin{cases} D_{phy} + T_{nextHop} \\ \qquad \text{if the candidate can provide a complete path} \\ D_{phy} + PENALTY + D_{vir} \qquad \text{otherwise} \end{cases}$$

where D_{phy} is the physical distance to the destination (or the next intermediate) measured in hops, D_{vir} is the virtual distance between the intermediate and the final destination normalized to the unit interval $[0, 1)$, $T_{nextHop}$ is the liveliness of the next hop measured as the ratio of the time since the respective node last sent a packet and the length of the SSR hello interval (1 second in our implementation), and a penalty value ($PENALTY = 50$ in our implementation).

Considering the effect of this formula, we see that a complete path is valuated much more than a path to an intermediate. (Note: The smaller Q, the better the node's ability to route the packet.) Thus, the liveliness $T_{nextHop}$ is only taken into consideration when choosing between several complete paths of the same length. If none of the candidates can provide a complete route to the destination, the ability calculation corresponds to a distributed application of the SSR routing algorithm. To this end, D_{phy} and D_{vir} reflect the respective SSR requirements.

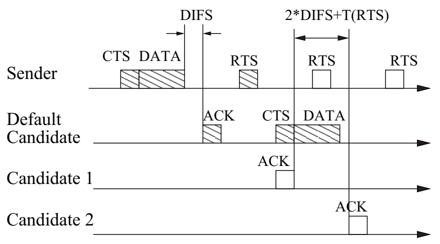
Note that $T_{nextHop}$ is quite important in scenarios with mobility or churn. It enables the nodes to quickly learn of nodes that can provide a path to the destination. In the original SSR proposal, broken links were only detected after the timeout of a link-layer unicast packet. Depending of the setup of the 802.11 card this is typically on the order of 300ms.

We illustrate the ability calculation and its effect with an example: Assume node 1 wants to route a packet to node 20. Node 1 can produce the source route 1-4-17 from its route cache. Thus, node 4 becomes the default candidate.

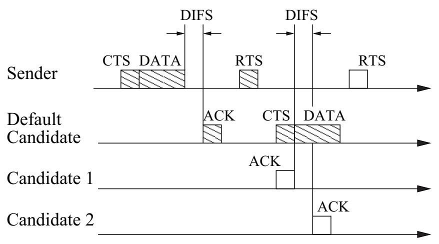
Node 1 also knows that it has two more physical neighbors: node 2 and node 3. Both become further candidates in the candidate list.

Node 2 has the source route 2-6-9-20 in its cache. It has

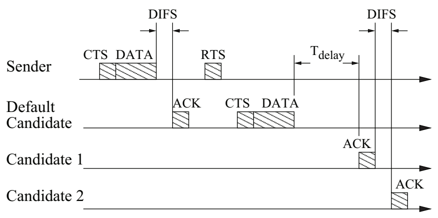
[1]Note that throughout this section, we use the terms DATA, ACK, RTS and CTS to name the link-layer broadcast packets of our protocol. We chose these terms in analogy to the well-known IEEE 802.11 frames.



(a) The sender always waits for the first ACK. If the default candidate responds, candidate 1 and 2 will cancel their ACKs upon receiving the RTS; otherwise, candidate 1 and 2 will send their ACKs in the predetermined sequence unless they receive an RTS.



(b) The sender always waits for the first ACK. If the default candidate responds, candidate 1 and 2 will cancel their ACKs upon receiving the RTS; otherwise the sender waits for the ACKs of all candidates before it selects the forwarder.



(c) If the default candidate succeeds to respond, other candidates may nevertheless acknowledge to improve future forwarding actions. Note the ACKs are rescheduled with $T_{delay} \geq 3 \cdot T_{DIFS} + T_{ACK} + T_{RTS}$.

Figure 1: Packet delivery sequence

last heard of node 6 half a second ago. Thus $Q_2 = 3 + 0.5 = 3.5$.

Node 3 has the source route 3-8-19 in its cache. Assuming an address space $0 \ldots 99$ for this example, we find $Q_3 = 2 + 50 + 0.01 = 52.01$.

Node 4 has the source route 4-18 in its cache. Thus $Q_4 = 1 + 50 + 0.02 = 51.02$.

Node 4 acknowledges the DATA packet, receives the RTS and forwards the DATA packet. Node 3 cancels its ACK because $Q_4 < Q_3$. At least T_{delay} after node 4 has forwarded the DATA packet, node 2 sends its ACK. The reason is that $Q_2 < Q_4$. Thus in future, node 1 will choose node 2 as default candidate.

4.2 Discussion of the Design Rationale

As stated afore, if the default candidate doesn't fail, the joint selection mechanism takes effect not for the current packet, but only for the future packets. One might ask why we do not recommend to wait for all the ACKs. This would have optimized the route for the first DATA packet already. However, it would introduce a larger overhead and delay all packets. This is not advisable, especially for small DATA packets.

In our approach, both the acknowledgment sequence and the actual forwarder are determined by the sender. This centralized approach avoids the duplication of the DATA packet. Furthermore, it does not introduce any additional packet delay as compared to the standard MAC-layer RTS/CTS handshake. Swapping the RTS/CTS and DATA order increases the probability of a successful DATA delivery on error-prone links, owning to several potential forwarders as stated in [12].

If packet errors were mostly caused by hidden terminals, our proposal could easily be adapted to that situation, too. In that case, a node would not send the DATA packet until the next hop forwarder has sent its CTS.

5. PERFORMANCE EVALUATION

In this section, we evaluate our proposal by the simulation of three scenarios: a static scenario, a scenario with node mobility and a scenario with node churn. We evaluate these three scenarios with 100 or 225 nodes arranged in a grid. The node distance (50 meter) is chosen so that each node has connections with its eight neighbors.

We use the OMNeT++ simulator [14] with the IEEE MAC80211 module from the INET framework. We use the default parameters of that MAC module (from the ad-hoc example scenario), except for the radio range, which is set to 75 meters. The application packet size is set to 120 bytes. Before beginning with application-layer messages, the system is simulated for 180 seconds. In this time SSR bootstraps and fills up the nodes' route caches. After the initialization the simulation runs for another 600 seconds of simulated time.

5.1 Static Scenario

In this scenario, we test our approach against the original SSR protocol using two static scenarios with 100 and 225 nodes respectively. Each node generates 5 consecutive packets to one randomly chosen destination every 50 seconds. Note that this corresponds to a light traffic load in order to match our assumption that transmission errors dominate over packet collisions. We run the simulation several times with a route cache size ranging from a minimal size of 30 (45) nodes to a maximal size of 100 (225) nodes.

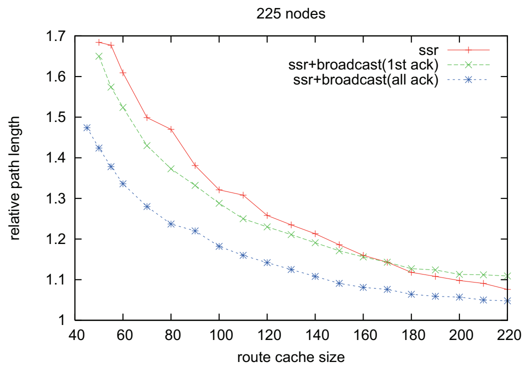
From Figure 2(a) we can see that our proposal achieves a shorter path length than the original SSR protocol. Due to the space limit, we skip the similar figure of 100 nodes here. The improvement is especially significant for small route cache sizes. Conversely, we find that with our proposal we can reduce the route cache by about 30% and still achieve the same route stretch as the original SSR protocol.

One would expect that the reduced path length resulted in a reduced delay. This is not the case, as shown in Figure 2(b). The reason is the additionally introduced handshaking overhead in network layer. Note that we use broadcast packets even though we term the packets in analogy to the respective native link-layer frames. Thus these packets create more overhead than the link-layer handshaking in the unmodified SSR protocol.

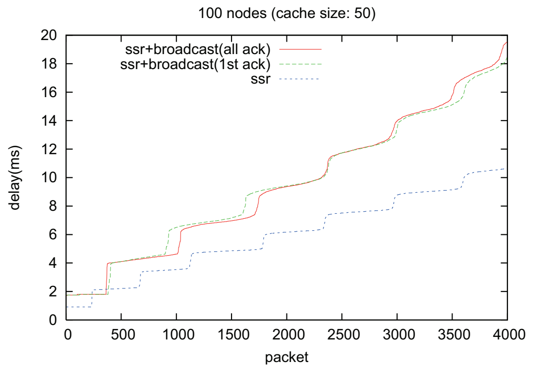
Table 1 shows the entire traffic that is generated at the different layers. As we can see, our proposal generates a little more overhead in the network layer and physical layer than the original SSR protocol, despite reduced path length. Again, this is due to the fact that our protocol actually includes some MAC layer functions and that we use broadcast packets for the handshaking. For example, an ACK frame in 802.11 MAC consists of 14 bytes and a minimal DATA frame contains 34 bytes. In our implementation,

Sent/Rcvd bits	Original SSR	enhanced SSR(1st ack)	enhanced SSR(all ack)	AODV
Application layer	570k (1.0) / 569k (1.0)	580k (1.0) / 566k (0.98)	575k (1.0) / 559k (0.97)	578k (1.0) / 255k (0.44)
Network layer	9455k (16.6) / -	9798k (16.9) / -	9807k (17.1) / -	13146k (22.7) / -
Physical layer	35489k (62.3) / -	41249k (71.1) / -	41292k (71.8) / -	158660k (275) / -

Table 1: Traffic in network stack in static scenario for one minute.(100 nodes, route cache size: 50)



(a) path stretch



(b) sorted packet delay (the 4000 smallest packet delay)

Figure 2: Static scenarios

the CTS/RTS/ACK packets are all 802.11 DATA frames, that introduces more overhead and delay than the native CTS/RTS/ACK frames. But anyway, the overhead of any SSR versions is much smaller that from AODV, thus result in a significant higher packet delivery ratio. (For a complete performance comparison between SSR, AODV and DSR, cf. [6].)

Thus, we argue that both the increased delay and the increased traffic overhead at the physical layer are the result of the way 802.11 realizes broadcasts. We believe that if we moved part of our protocol (e. g. the RTS/CTS/ACK packets) into the MAC-layer, or use a different MAC protocol, these effects would vanish.

5.2 Scenario with Mobility

In this scenario, one node moves randomly (based on the random way point model) inside the network area. Randomly selected static nodes send messages to the moving node. In order to eliminate the potential effect of a limited route cache, the route cache size is set such that the caches can hold all nodes of the network.

From Figure 3(a) we can see that the packet delivery ratio of the original SSR decreases rapidly with increasing node speed. The reason is that plain SSR cannot distinguish between a broken link and a link with packet loss. Thus, SSR waits for several MAC-layer retransmissions before it starts to seek an alternative forwarding path. (Note that using the delivery fail information from the 802.11 link-layer would generate an extremely large delay because of the exponential back-off.)

In our proposed enhancement a broken link can be quickly replaced by another link that is known to be available. As we can see from the simulation, our enhanced SSR protocol achieves a delivery ratio of more than 80% even for a node speed of 20m/s. Our version that waits for all the ACKs before determining the new link when default candidate fails, achieves a delivery ratio of even 97% at that speed.

5.3 Scenarios with Churn

Both in MANET and in sensor Ad-hoc network, node churn can occur any time and any where. In this experiment, we let 10 (and 50) of the 100 nodes suddenly become inactive for the time interval between 300 and 330 seconds. Note that we simulated the scenarios with different random seeds, and chose the ones in which the remaining network was still connected during this time interval. Furthermore, we generated the traffic only between the remaining nodes during that time.

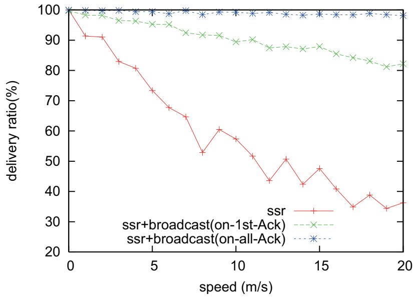
From Figure 3(b) we find that with a moderate churn rate (10%), the packet delivery ratio of the original SSR drops dramatically to 65%, while our enhanced SSR is almost not be influenced. Clearly the performance drop of the original SSR results from its deterministic forwarding mechanism on potentially outdated neighbor information, while in our enhanced SSR, timely neighbor information can be obtained and used.

Figure 3(c) shows the performances in an extreme churn rate (50%), we find that at the beginning of the churn period the packet delivery ratios (PDR) of both the original SSR and our enhanced SSR are affected. The PDR of the original SSR drops to 20% and then slowly converges back to 100% at the end of churn time. Impressively, even in this excessive case, both versions of our enhanced SSR manage to keep the PDR above 70%.

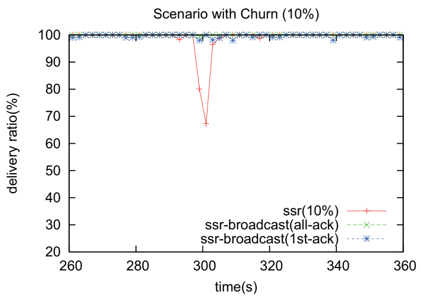
This shows that at this point of time, most multi-hop paths are affected by the churn event. Thus, the original SSR protocol can only route packets to destinations that happen to be close to the source node. Our enhanced SSR protocol is only little affected by the churn.

6. CONCLUSIONS

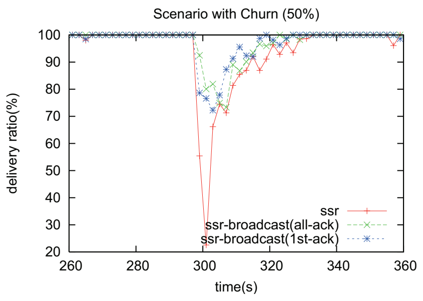
In this paper, we have proposed an enhancement of the scalable source routing protocol. SSR provides the key based routing service of structured peer-to-peer overlays in the network layer. Moreover, it requires only a small per-node state. Therefore, SSR has been considered to be well-suited for large ad-hoc networks.



(a) Mobility scenarios



(b) Churn scenarios (churn rate 10%)



(c) Churn scenarios (churn rate 50%)

Figure 3: Mobility scenario and churn scenarios

Our proposed enhancement further improves the performance of SSR in MANET scenarios. It uses link-layer broadcast when forwarding SSR packets. Together with a few handshaking messages this allows a node to include its physical neighbors into its routing decision, and thus to optimize the routes subsequently. (Clearly, there is a trade-off in the joint decision process between the computing/memory cost and the performance improvement. Thus only the powerful nodes should participate in the decision process.) In addition, the concept of a *default candidate* minimizes the forwarding delay.

An evaluation of our proposal in simulations shows that it is able to greatly improve SSR's performance in scenarios with high node mobility and node churn. It can maintain its service even in scenarios where a node moves with 20m/s or where suddenly 50% of all nodes fail. Moreover, it can achieve the same routing efficiency as plain SSR with a significantly smaller per-node state. Depending on the scenario the route cache size can be reduced by about 30% without compromising SSR's routing performance. These properties recommend the enhanced SSR protocol to provide the key based routing service of structured P2P overlays also for unstable mobile ad-hoc networks.

Since some MAC functions have to implemented in our protocol, some additional costs (e.g., delay) are introduced. We believe that these side-effects will be eliminated when using other MAC protocols or implementing the handshake functions into the MAC layer. Nevertheless, our enhancement shows a great performance improvement in the scenarios of the node mobility and node churn, hence increases the adaptivity of SSR in WSN and MANET.

Currently, we are exploring the performance of our proposed SSR enhancement in a real-world set up. There, the protocol is used to provide communication among embedded controllers which control mobile robots performing joint actions.

7. REFERENCES

[1] S. Biswas and R. Morris. Opportunistic routing in multi-hop wireless networks. In *Proc. ACM SIGCOMM '05*, Philadelphia, PA, USA, August 2005.

[2] B. Blum, T. He, S. Son, and J. Stankovic. IGF: A state-free robust communication protocol for wireless sensor networks. Technical report, Department of Computer Science, University of Virginia, USA, 2003.

[3] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron. Virtual Ring Routing: Network Routing Inspired by DHTs. In *Proc. ACM SIGCOMM '06*, Pisa, Italy, Sept. 2006.

[4] M. Chawla, N. Goel, K. Kalaichelvan, A. Nayak, and I. StojmenovicIvan. Beaconless Position-based Routing with Guaranteed Delivery for Wireless Ad hoc and Sensor Networks. In *Proc. 19th IFIP World Computer Congress*, Santiago de Chile, Chile, August 2006.

[5] T. Fuhrmann. Scalable routing for networked sensors and actuators. In *Proc. 2nd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, Sept. 2005.

[6] T. Fuhrmann, P. Di, K. Kutzner, and C. Cramer. Pushing Chord into the Underlay: Scalable Routing for Hybrid MANETs. Technical report, Universität Karlsruhe(TH), Germany, Fakultät für Informatik, 2006.

[7] M. Heissenbüttel, T. Braun, T. Bernoulli, and M. Wächtli. BLR: Beacon-less routing algorithm for mobile ad-hoc networkssigcomm. *Elsevier's Computer Communications Journal (ECC)*, 27(11):1076–1086, 2004.

[8] IEEE 802.11 Working Group. IEEE Std. 802.11-1999: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications., 1999.

[9] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing*, 353:153–181, Feb. 1996.

[10] K. Kutzner and T. Fuhrmann. Using linearization for global consistency in ssr. In *Proc. 4th Int. Workshop on Hot Topics in P2P Systems*, Mar. 2007.

[11] C. E. Perkins and E. M. Royer. Ad hoc On-Demand Distance Vector Routing. In *Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, USA, Feb. 1999.

[12] J. A. Sanchez, R. Marin-Perez, and P. M. Ruiz. Beacon-less geographic routing in real wireless sensor networks. In *Proc. 4th IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems (MASS)*, 2007.

[13] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. ACM SIGCOMM '01*, pages 149–160, 2001.

[14] A. Varga. The OMNeT++ discrete event simulation system. In *Proc. European Simulation Multiconference (ESM'2001).*, Prague, Czech Republic, June 2001.