Proc. of the Fourth Annual Conference on Wireless On demand Network Systems and Services,
Obergurgl, Austria, 24-26 January, 2007, pages 122 - 129

Performance of Scalable Source Routing in Hybrid MANETs

Thomas Fuhrmann *
Computer Science Department, Technical University Munich, Germany
fuhrmann@net.in.tum.de

Abstract

Scalable Source Routing (SSR) is a novel routing approach for large unstructured networks such as mobile ad hoc networks, mesh networks, or sensor-actuator networks. It is especially suited for organically growing networks of many resource-limited mobile devices supported by a few fixed-wired nodes. SSR is a full-fledged network layer routing protocol that directly provides the semantics of a structured peer-to-peer network. Hence, it can serve as an efficient basis for fully decentralized applications on mobile devices.

SSR combines source routing in the physical network with Chord-like routing in the virtual ring formed by the address space. Message forwarding greedily decreases the distance in the virtual ring while preferring physically short paths. Thereby, scalability is achieved without imposing artificial hierarchies or assigning location-dependent addresses.

1. Introduction

The complexity of planning and managing distributed systems quickly rises with their size and often requires experts. Nonetheless, an increasing number of consumer products are being sold with networking capabilities, be it wired or wireless. Networks installed by consumers should require as little planning and management as possible, or be completely self-organizing. The same requirements apply to disaster recovery scenarios where rapid deployment of a backup communication infrastructure is needed.

We consider a scenario where consumers incrementally create a network of electronic appliances such as portable computers, consumer electronics, embedded controllers, and so forth. The multi-hop network mixes wired and wireless links, as well as fixed and mobile devices. Most devices in such networks are highly available, while others may have volatile connectivity and be prone to failure. For example, the user could program lighting, air conditioning, and security systems using a portable computer. The network enables automation in and across buildings without a structured and managed infrastructure.

A basic functional requirement in a multi-hop computer network is routing. Although ad hoc networks have been extensively researched, and many ad hoc routing protocols have been developed, they are still limited to a few hundred nodes. Furthermore, distributed applications need basic services such as naming and service directories. As a result of their self-organization property, structured peer-to-peer (P2P) overlays such as Chord [10] are an attractive choice for the implementation of these basic services. However, running a P2P protocol on top of an ad hoc routing protocol incurs a high cost.

SSR is an integrated solution to both problems. It creates a Chord-like, ring-structured P2P network at the network layer. Thereby, SSR efficiently provides network-layer connectivity, and at the same time the indirect routing primitive of a structured P2P network. In this paper, we demonstrate the feasibility and scalability of SSR in the scenario outlined above. Compared to ad hoc routing protocols such as AODV and DSR, SSR delivers a substantially larger fraction of packets at a lower cost. SSR can cope with low levels of network mobility and with moderate node failure rates.

The remainder of this paper is structured as follows: The next section gives an overview of the SSR protocol. Section 3 evaluates its performance in IEEE 802.11 network with and without additional fixed wire links. Finally, Section 4 concludes the paper.

2. Protocol Overview

SSR provides routing in a potentially global, flat address space. Node addresses can be derived from MAC addresses, cryptographic keys, or other globally unique identifiers, for example by hashing an applica-

*Work done while at Universität Karlsruhe. Supported by Deutsche Forschungsgemeinschaft under grant FU448-1.

tion level key. Furthermore, SSR provides not only network layer routing, it can also serve as efficient basis for fully decentralized applications such as *distributed hash tables* (DHT): Like with an overlay DHT such as Chord, an SSR node *A* manages all data objects whose identifiers fall between *A*'s address and the address of the node *B* whose address is the smallest of all nodes with addresses larger than *A*'s. Node *B* is called *A*'s *successor* and consequently, node *A* is node *B*'s *predecessor*.

For correct routing in both, Chord and SSR, it is a necessary and a sufficient condition that each node knows its correct successor. This successor relation implicitly places all the nodes of the network on one virtual ring that is completely unrelated to the actual physical topology of the network. In particular, SSR makes no effort to align the virtual with the physical structure.

Nodes route packets by forwarding them in increasing direction of the virtual ring (i. e. address space) until the distance between the address of the current node and the address of the destination node (or identifies of the data object) cannot be minimized any further. If the network is inconsistent — i. e. , if some nodes do not know their correct successors — requests for data objects may get delivered to nodes that have no knowledge of the requested objects.

In contrast to well-known link-state and distance-vector routing protocols, SSR does not maintain state information for all destinations in the network. A node's forwarding information base is comprised of the node's physical neighbors, the node's virtual predecessor and successor, and opportunistically cached information.

Upon joining the network, a node acquires the routes to its predecessor and successor through a proxy that is already connected to the network. During operation the nodes update these routes whenever they detect an inconsistency. If the network bootstraps simultaneously or if partitioned network united, the successor and predecessor routes are obtained by a simple iterative process that we describe in Section 2.2.

SSR trades off shortest paths for a reduced amount of state information and therefore less maintenance overhead. Depending on the scenario, paths on average are 20%–200% longer than the shortest paths.

The protocol maintains its forwarding information base as a source route cache. This cache stores source routes using a least-recently-used (LRU) policy (see Section 2.3). It locks the routes to a node's virtual predecessor and successor and thereby avoids flushing them. The next section explains how SSR routes packets on the basis of this cache.

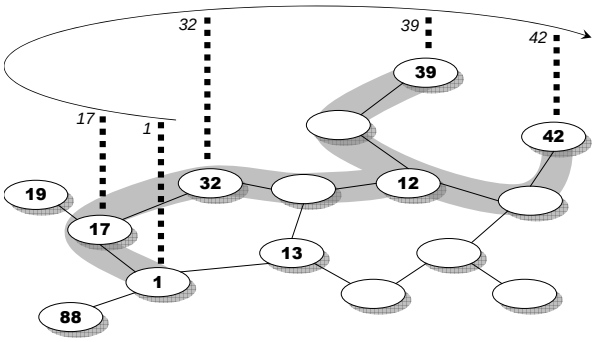

Figure 1. Illustration of the routing process

2.1. Packet Forwarding

SSR uses two distance metrics for making its forwarding decisions. The physical distance of two nodes *A* and *B* is measured in hops, that is the length of a source route between *A* and *B*. The virtual distance of two nodes is the absolute value of the numerical difference between the nodes' addresses.

Data packets contain a source address, a destination address, and a source route. The included source route does not have to span the entire path from the packet source to the packet destination. In that case, the last node in the source route tries to append a source route from its local cache. Again, the appended path need not end at the packet's destination, but at a node *virtually closer* to the final destination.

When appending a source route, a node also considers physical node proximity in its selection of the next virtual hop. In particular, if a node with address I_1 does not have a source route to the packet's destination *D* in its cache, it selects a so-called intermediate destination I_2 from its cache, so that I_2 is virtually closer to *D* than I_1. Typically, several such nodes exist. If so, I_1 selects the one that is physically closest to I_1. If more than one of these nodes exist, the protocol chooses the virtually closest to *D*.

For example, in Figure 1, node 1 wants to send a packet to node 42. The packet is first forwarded to node 17 because that node is physically closest to node 1. Node 17 is preferred over node 13 since 17 is virtually closer to 42 than 13 is. For the same reasons, node 17 forwards the packet to node 32. Node 32 forwards the packet to its successor, node 39, which in turn forwards the packet its successor that coincides with the destination, node 42.

As a result of the indirectness of routing, a packet is delivered to the node whose address is virtually closest to the destination address *D*. If *D* identifies an existing node, the packet arrives at *D*.

2.2. Virtual Neighbor Acquisition

SSR guarantees consistent routing if and only if all nodes have valid source routes to their respective virtual neighbors, that is its predecessor and successor in the address ring. During normal operation, a joining node acquires source routes to its virtual neighbors through one of its physical neighbors. This neighbor acts as proxy for the newly joining node's request for its own address.

If the network is inconsistent, for example, when many nodes are bootstrapping simultaneously or when partitioned networks reunite, nodes acquire source routes to their virtual neighbors through an iterative process. As a prerequisite for this process, the nodes are assumed to have information about their physical neighborhood. They gather this information, for example, from the periodic sending of "hello" beacons. This information is stored in a node's route cache and is locked, so that it is not discarded by the LRU strategy.

In the following, we describe the iterative process for discovering a path to a node's successor.

1. A node A selects a successor candidate B from its cache. In the beginning, B is one of A's physical neighbors and unlikely to be the globally correct successor of A. A sends B a *successor notification* message. It can do so since by construction the cache contains the source routes to all of its address entries.

2. Let C be the predecessor of B before receiving the *notification* message. Upon reception, B checks whether C's address lies between that of A and B. In this case, C is a better successor candidate for A. Otherwise, A is a better successor candidate for C. In both cases, B sends *update* messages to A and C and inserts the source route from the *notification* message into its cache. The source routes for the *update* messages are obtained by concatenating B's source routes to A and C. In the trivial case where $A = C$, only one message is sent. This message is called an *acknowledgment* message.

3. Upon reception of an *update* message, A has learned the source route to C, which either is a better successor, or a potentially better predecessor. In the former case, A sends a *successor notification* to C. In the latter case, A either agrees with C being its predecessor or A knows a better successor for C and accordingly sends either a *predecessor notification* or a *successor update* message.

For example, assume that in Fig. 1 node $A = 1$ searches its successor. The node among A's neighbors that is virtually closest to A is $B = 13$. Furthermore assume that B has a path to its correct predecessor $C = 12$. Thus, A sends a *successor notification* to B and since $A < C < B$, node B sends *update* messages to both A and C. As a consequence, A sends its next *successor notification* to C. If C knows of a node D with $A < D < C$, the process continues.

Mutual correctness of the successor-predecessor relation does not imply correctness from a global viewpoint. In order to reduce the risk of global inconsistency and to accelerate network convergence in general, any node forwarding *notification* and *update* messages may check their caches to detect inconsistencies. In sufficiently large, well connected networks the risk of an undetected inconsistency is negligible. If global consistency must to be guaranteed nevertheless, all nodes whose address is larger than the address of their respective successor have to periodically broadcast their address in the network. Since in each disjoint ring, there is only one such node, the resulting traffic is small.

2.3. Route Cache Maintenance

While forwarding a packet from its source to its destination, the intermediate nodes append source route fragments (see sec. 2.1). In many cases an appended fragment contains a node that already is part of the packet's source route. This would result in a loop in the source route. Hence, after appending fragments, the nodes cut out such loops so that the resulting path is the shortest path contained in the concatenated source route path. In Fig. 1, the source route from node 1 to node 42 can be pruned so that the stub path between node 12 and node 39 is not contained in the resulting path.

In practice the intermediate node performing the concatenation operation is likely to be cut out in this pruning process. For forwarding, packets therefore not only contain one source route but also a stub path that connects the most recent intermediate node to the main source route. Moreover, even when the main route has been reached, the stub is kept in the packet so that each subsequent node can retrieve a source route to the intermediate node that appended the fragment.

This is important to ensure that route caches can be updated when routes break. Assume that a packet contains a source route which has become invalid, for example because the next hop failed. In this case, the detecting node sends an *update* message back to the most recent intermediate node which can then delete the broken link from its cache. If the route can be salvaged, another *update* will be created to inform that intermediate node of the new route.

Upon reception of an *update*, only the part between the intermediate node and the destination is entered into the cache, since only that part has just been traversed by the packet. Together with the *update*s indicating broken links, this allows SSR to work without regular probing.

When a source route has become invalid, it remains in the cache until it is used and then causes *update* messages. As a consequence, SSR is quiescent, i. e. it does not create any control messages in the absence of payload traffic.

3. Evaluation

SSR provides both, direct any-to-any routing in unstructured networks, and the semantics of a structured peer-to-peer overlay. The latter eliminates the need to run an overlay network protocol such as Chord on top of an ad hoc routing protocol such as AODV. As has been shown elsewhere [6] such a combination performs badly.

In the remainder of this section, we report simulation results measuring SSR's performance for routing in MANETs. In particular, we compare SSR to AODV and DSR using their GloMoSim [1] implementation as benchmark. Owing to the scenario of providing peer-to-peer services for embedded devices, the route cache size was limited to 255 nodes only, that is, the union of all nodes in the cached source routes contained at most 255 distinct nodes.

We simulated networks with 50 – 125,000 nodes, where the area is increased likewise so that the networks have a constant density of 50 nodes per square kilometer. With the chosen radio model (see below) this corresponds to 9.8 nodes per radio range disk. With this node density, there is a high probability for the network to be connected with potentially a few unconnected nodes [3, 12]. Other node densities were studied separately (see below).

It is well known that pure ad-hoc networks are limited in their capacity [7]. Thus, in most of the simulations, we equipped 5% of the nodes with additional 1 MBit/s point-to-point links. This models fixed nodes with a wireline communication infrastructure, for example powerline, cable-modem or DSL connections. Links between these nodes are chosen according to the Barabási-Albert model [2]. This is a so-called preferential connectivity model that is known to produce a small-world topology.

Nodes move according to a random direction model with a maximum node velocity of 1 m/s, the velocity of a pedestrian. Other maximum velocities were studied separately (see below).

The traffic model for our simulation is derived from a scenario where SSR is used for a fully decentralized directory service: Every minute each node sends a 500 byte packet to a randomly selected destination. Other packet sizes were simulated, too, but found to have little influence on the protocol's performance. The results therefore are not reported here. For bootstrapping, all nodes are switched on with an empty cache at equally distributed points in time during the first minute of the simulation.

If not otherwise stated, each simulation run spans one hour of simulated time. During the run, we measure the delivery rate, the end-to-end delay for the delivered packets, and the network load created by the payload, overhead, and control messages. The plotted values are averages of one-minute intervals, or, in case of time series, averages of 10 simulation runs. For the analysis of the runs, we delete an initial transient period of 10 min to allow the network to settle. Additional simulations compare the route stretch, which is the average ratio of the achieved path length to the shortest path length, and the traffic distribution.

3.1. Radio- and Link-Layer Model

Packet level simulations of large networks are difficult since the detailed simulation of the entire protocol stack requires sufficiently fast simulation machines. Especially, we found the GloMoSim IEEE 802.11 model to be incapable of simulating sufficiently large networks for our study. Therefore, we resorted to a mixed-mode simulation that models PHY and MAC with the help of a fluid model based on experimental results from the literature:

Kim and others [9] show that PHY and MAC of IEEE 802.11 wireless LANs can be modeled by a sequence of collision, transmission, and idle periods such that the system's throughput is a function of the frame size and the number of nodes M simultaneously attempting to access the medium. Moreover, the authors found the throughput to be almost independent of M when RTS/CTS was applied.

He and others [8] show how the throughput depends on the number of nodes in the transmission range, the carrier sense range of the sender, and the number of hidden nodes. In absence of hidden nodes, the entire throughput of all nodes is independent of the number of nodes attempting to access the medium [9]. In the presence of hidden nodes, the throughput drops when the offered load is increased beyond a certain threshold, where the extent of this drop depends on the node density. However, with the chosen node density, the omission of considering hidden nodes has only a minor effect [8]. Simulation and measurement results are in accordance with these results [5]. Following these results, we based our PHY and MAC simulation on a fluid model using the IEEE 802.11 system parameters from [9]. The reception power threshold was chosen such that the radio range was 250 m.

3.2. Comparison with AODV and DSR

In the following sections, we compare SSR to AODV and DSR with respect to their scalability and robustness in face of mobility and node churn.

3.2.1. Varying Network Size. Figs. 2–4 show the results for a pure mobile ad-hoc network scenario, a hybrid MANET with 1% fixed-wired nodes, and a hybrid MANET with 5% fixed-wired nodes. Since the GloMoSim implementations of AODV and DSR are not optimized for large networks, they frequently broke down in simulations with more than a few hundred nodes due to internal problems, such as memory leaks and array overflows. However, since we intended to use GloMoSim as a widely accepted benchmark, we refrained from more than basic modifications of the simulation code. Hence, we included those runs that simulated at least 20 minutes, but could not acquire AODV and DSR results for even larger networks.

Nevertheless, the results already show that SSR is capable of routing a large fraction of the packets in large networks, while AODV and DSR are not. In the pure MANET scenario, increasing the network size up to 600 nodes does not have an effect on the delivery ratio of SSR, which successfully routes about 90% - 95% of all messages (cf. Fig. 2). AODV and DSR, in contrast, have delivery ratios that continuously drop to less than 50%. In the hybrid MANET scenarios, the delivery ratios of AODV and DSR are comparable to those of the pure MANET scenario (cf. Figs. 3 and 4). SSR, however, can maintain a high delivery ratio for much larger networks, namely up to 1200 nodes in the scenario with 1% fixed nodes and up to more than 12,000 nodes with 5% fixed nodes. (See below for even larger networks.)

Figs. 2–4 (middle) show the average end-to-end delays for the three scenarios. Note that the drop in delivery ratios for SSR coincides with a rise in the round trip time. Higher round trip times indicate contention on the wireless links. Both, the higher round trip times and the packet loss caused by overflowing queues in a congested network affect the ability of SSR to quickly adopt to changes in the network. In the pure MANET case, SSR can keep the delay below 400 ms for networks of up to 600 nodes. AODV and DSR have significantly larger delays. With DSR, the delays are also erratic, leading to huge error bars.

The reason for the erratic delays is that the distribution of delays is heavy-tailed. Fig. 5 shows the distribution of the delays for the 5% fixed nodes scenario and a total of 612 nodes. While SSR follows a power-law distribution with a single distribution parameter, AODV and DSR exhibit a slightly different behavior: both follow two different distributions, one for values below about 40 ms and a different one for the larger delays.

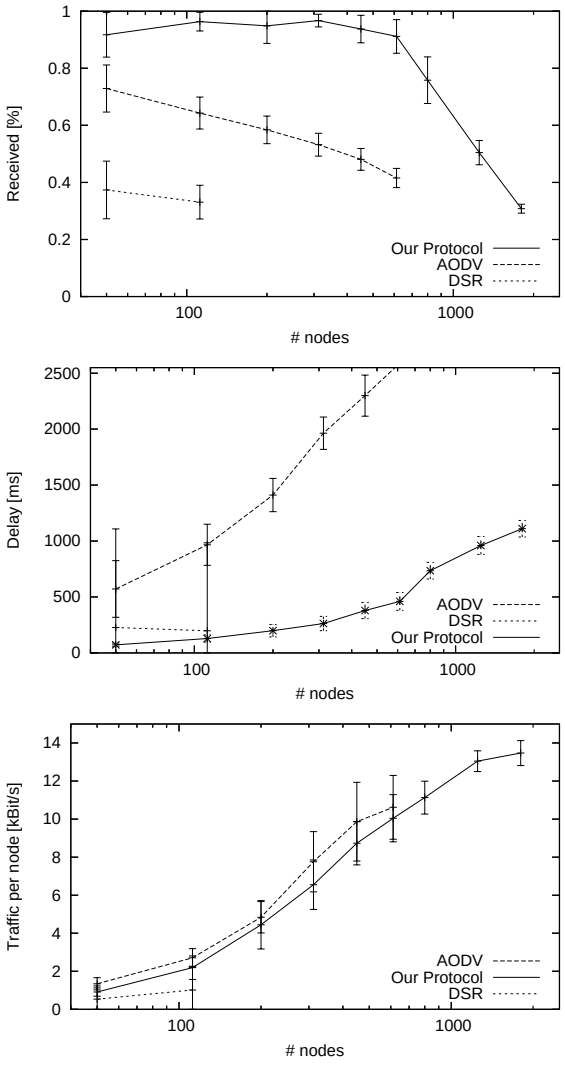

Figure 2. **Pure MANETs**

The reason for the two different parameters is that AODV and DSR need to acquire or produce state in the network before they can route the actual payload packet. Thus we can distinguish the case where that state already exists from the case where the state needs to be established. In the former case, AODV and DSR lead to a lower delay than SSR. In the latter case, they lead to a much higher delay. The different cases correspond to the different slopes in the AODV and DSR delay distribution in the double-logarithmic plot in Fig. 5.

Coming back to the relation of delay and delivery ratios, we see that in all cases SSR can maintain its high delivery ratio as long as the delay stays below 500 ms. We also see that the delay increases roughly logarithmi-

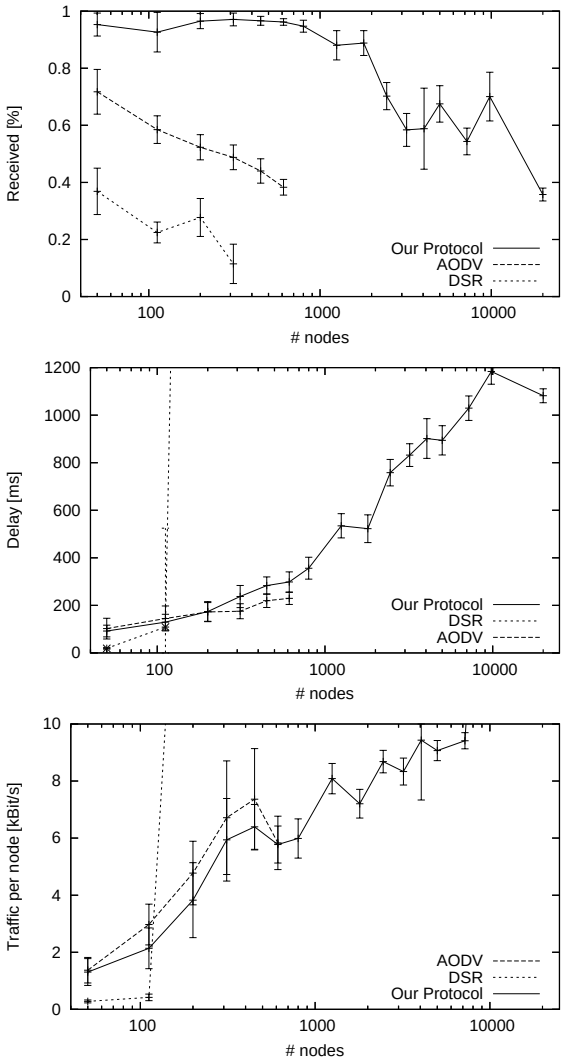

Figure 3. **Hybrid MANETs with 1% fixed-wired nodes**

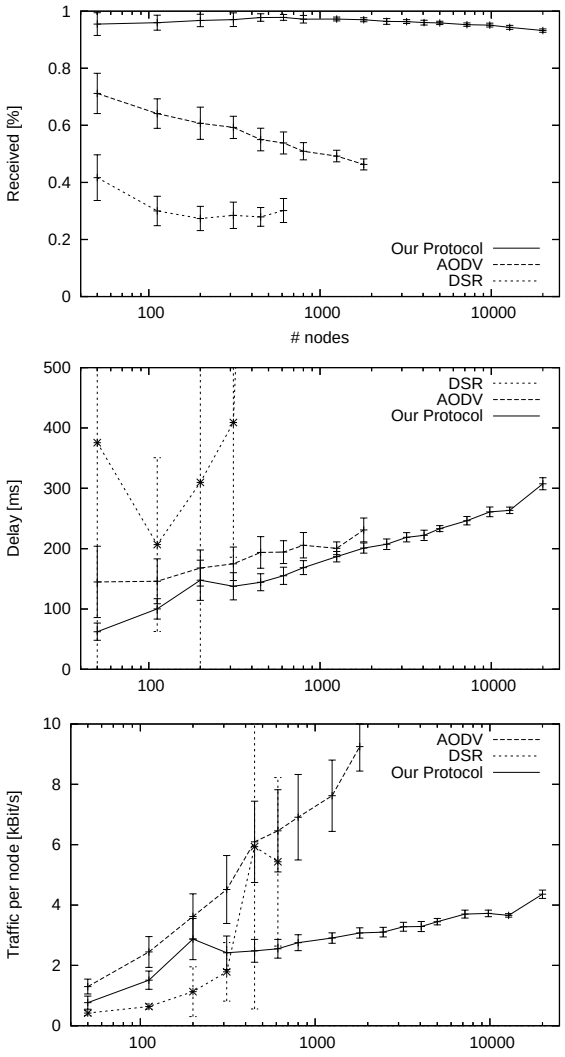

Figure 4. **Hybrid MANETs with 5% fixed-wired nodes**

cally with the network size. This is caused by the growing path lengths (not shown). Such a scaling behavior of average node distances is typical for small-world networks [11, 4].

Besides its high delivery ratio, SSR also keeps the network traffic low. Figs. 2–4 (bottom) show the average traffic per node. As can be best seen in the pure MANET case, in small networks SSR produces a similar traffic volume as AODV. DSR achieves slightly lower traffic volumes. In the hybrid cases, however, SSR is able to keep the traffic volume low while the AODV and DSR traffic quickly drive the network into congestion, presumably because of their intensive use of flooding.

3.2.2. Varying Node Density. In order to complete the comparison, we have also studied the effect of varying node densities and maximum node velocities. Simulations varying the message size did not yield any particularly remarkable effect and are thus not reported here. Due to the frequent problems with the DSR simulation, we compared SSR only to AODV.

Fig. 6 shows the delivery rate in a 450 node pure MANET where the size of the network has been varied to yield node densities between 5 and 22 nodes per radio disk. As expected, the probability of successful delivery drops rapidly when the node density falls below 10 nodes since the network partitions [12]. Conversely, higher node densities show no effects on the delivery ratio. We did however not explore very high node densities since with our traffic model they would

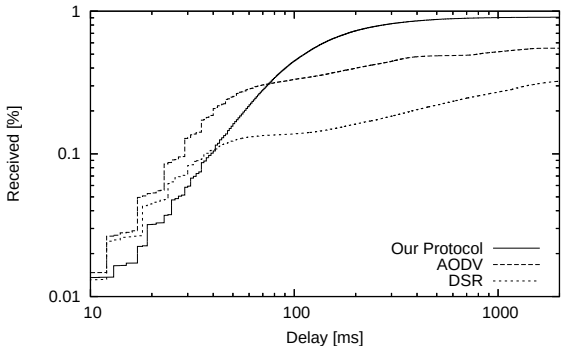

Figure 5. **CDF of the end-to-end delays**

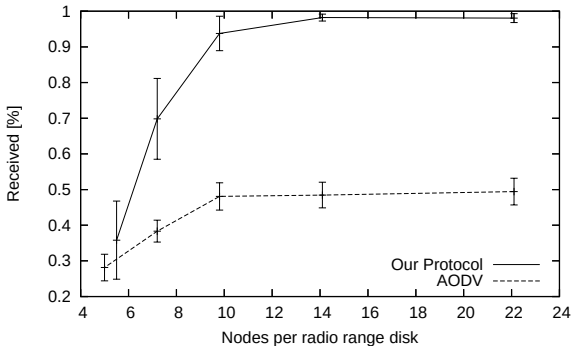

Figure 6. **Varying the node density, max. velocity 1 m/s**

immediately cause congestion in the network.

3.2.3. Varying Node Mobility. Figs. 7 show the effects of varying the maximum node velocity in our 450 node pure MANET model. As can be seen, SSR is most effective at low node velocities. Up to about 1 m/s, the velocity of a pedestrian, the delivery rate can be sustained at more than 90%. When the node speeds are further increased, the delivery rate significantly drops. For maximum node velocities of more than about 2 m/s, SSR performs worse than AODV. The reason is that the proactive aspects of SSR cannot cope with the speed of route changes while the reactive AODV protocol can obtain routes, albeit at the expense of a high delay. This is different from the congestion collapse in a large pure MANET, as can be seen from the comparatively low delay for SSR and the low traffic load (cf. Figs. 7 mid and bottom).

Finally, we have compared the paths lengths produced by SSR and AODV to the shortest paths as they could be produced by an ideal omniscient routing instance (Fig. 8). As expected, SSR does not yield shortest paths.

Nevertheless, this comparison yields some remarkable insights, too. Since AODV discovers routes by means of flooding, the resulting paths are shortest paths, or, if repair mechanisms apply, close to shortest paths. The larger the distance, the higher the probability that AODV fails to deliver the path. If in the shown simulation, SSR would give up after a maximum of eight hops, the corresponding line would level with AODV's line.

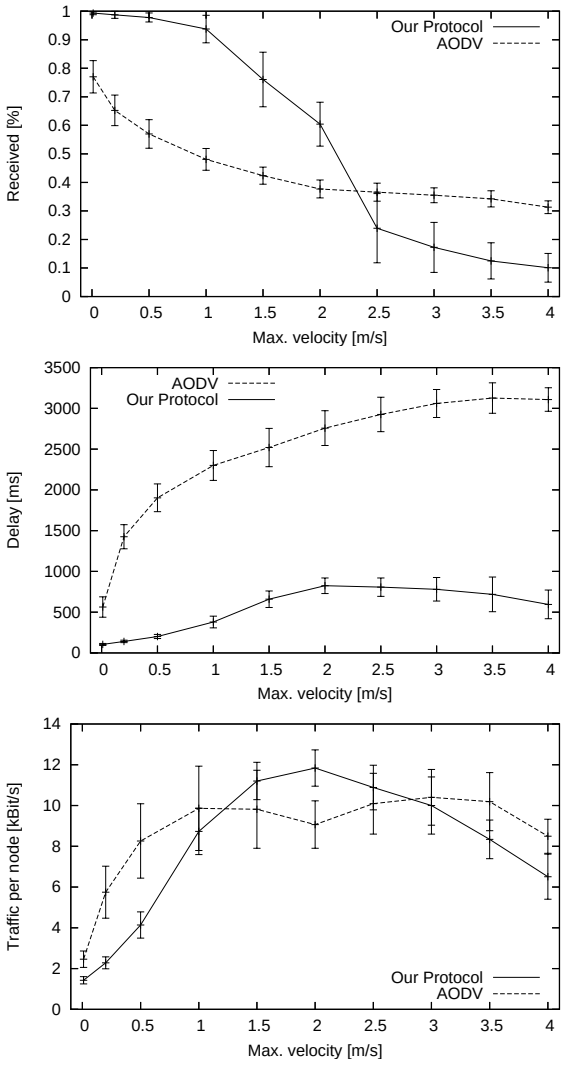

Figure 7. **Varying node mobility in a pure MANET**

4. Conclusion

Scalable routing in unmanaged, unstructured, and self-organizing networks is a basic requirement for scenarios where consumers use many resource-limited devices that communicate with each other. We have argued that, although ad hoc networks have been extensively researched, in the scenarios that we envision,

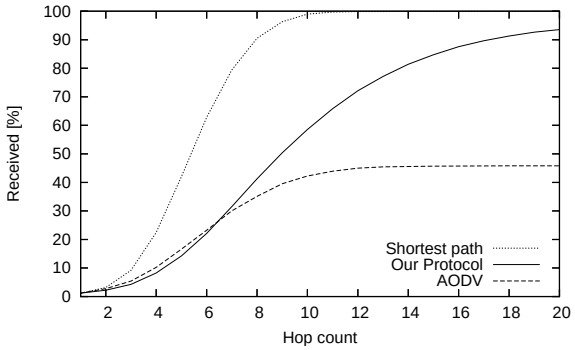

Figure 8. **Paths lengths for an 800 node hybrid MANET with 5% fixed-wired nodes**

MANET protocols are not practical due to their scalability problems. Especially, combining structured peer-to-peer overlays with MANET protocols is known [6] to lead to very bad performance.

The *scalable source routing* (SSR) protocol combines ideas from the Chord overlay network with source routing in ad hoc networks. Thereby, SSR provides scalable any-to-any communication in large unstructured networks as well as the semantics of a structured peer-to-peer overlay. Thus, SSR can serve as a basis for distributed applications in ad hoc networks.

In this paper, we have evaluated SSR by means of simulations and have found that it delivers a substantially larger fraction of packets than AODV and DSR, while achieving a lower average delay. In hybrid scenarios, SSR can efficiently use fixed-wired links and thereby avoid congestion collapse. With low node mobility and low node churn, SSR achieves delivery rates of more than 90% in networks with up to 125,000 nodes.

References

[1] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla. GloMoSim: A Scalable Network Simulation Environment. Technical Report 990027, UCLA, Computer Science Department, May 1999.

[2] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, October 1999.

[3] C. Bettstetter. On the Connectivity of Ad Hoc Networks. *The Computer Journal*, 47(4):432–447, July 2004.

[4] B. Bollobas. *Random Graphs*. Cambridge University Press, 2001.

[5] S. Choi, K. Park, and C. kwon Kim. On the performance characteristics of WLANs: revisited. In *Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 97 – 108, Banff, Alberta, Canada, June 2005.

[6] C. Cramer and T. Fuhrmann. Performance evaluation of chord in mobile ad hoc networks. In *Proceedings of the First International Workshop on Decentralized Resource Sharing in Mobile Computing and Networking (ACM MobiShare)*, Los Angeles, CA, Sept. 2006.

[7] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, IT-46(2):388–404, Mar. 2000.

[8] J. He, D. Kaleshi, A. Munro, Y. Wang, A. Doufexi, J. McGeehan, and Z. Fan. Performance investigation of IEEE 802.11 MAC in multihop wireless networks. In *Proceedings of the International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pages 242 – 249, Montral, Quebec, Canada, October 2005.

[9] H. Kim and J. C. Hou. A fast simulation framework for IEEE 802.11-operated wireless LANs. In *Proceedings of the Joint international conference on Measurement and modeling of computer systems*, pages 143 – 154, New York, NY, USA, June 2004.

[10] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proceedings of the SIGCOMM 2001 conference*, pages 149–160. ACM Press, 2001.

[11] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[12] F. Xue and P. R. Kumar. The Number of Neighbors Needed for Connectivity of Wireless Networks. *Wireless Networks*, pages 169–181, 2004.