Balanced Multipath Source Routing*

Shreyas Prasad, André Schumacher**, Harri Haanpää, and Pekka Orponen

Laboratory for Theoretical Computer Science, Helsinki University of Technology,
P.O. Box 5400, FI-02015 TKK, Finland
Shreyasp@tcs.hut.fi, Andre.Schumacher@tkk.fi,
Harri.Haanpaa@tkk.fi, Pekka.Orponen@tkk.fi

Abstract. We consider the problem of balancing the traffic load ideally over a wireless multihop network. In previous work, a systematic approach to this task was undertaken, starting with an approximate optimisation method that guarantees a provable congestion performance bound, and then designing a distributed implementation by modifying the DSR protocol. In this paper, the performance of the resulting Balanced Multipath Source Routing (BMSR) protocol is validated in a number of simulated networking scenarios. In particular, we study the effect of irregular network structure on the performance of the protocol, and compare it to the performance of DSR and an idealised shortest-path routing algorithm in setups with several source-destination pairs. For all network scenarios we consider, BMSR outperforms DSR significantly. BMSR is also shown to be more robust than the shortest-path algorithm, in that it can distribute the traffic load more evenly in cases where shortest-path routing is impeded by radio interference between proximate paths.

1 Introduction

Consider a scenario whereby a wireless multi-hop network is used to set up communications between a disaster recovery area and an operations centre, or is needed to replace a broken segment of a high-throughput fixed network. One issue that then arises is how to optimally allocate the total transmission bandwidth of the wireless network to carry the high volume of end-to-end traffic. Unless the traffic pattern and the design of the wireless network are fully predictable, some load balancing multipath routing scheme should be used to avoid congestion. Such schemes also improve reliability of the network.

A considerable amount of work exists on multipath routing in wireless networks. E.g. Nasipuri, Castañeda, and Das [1] extend the DSR [2] route finding process to consider alternate routes for a given destination. Wu and Harms [3] modify the procedure of forwarding route-reply messages back to the initiator of a route-request in order to discover alternative routes. See [4] for a discussion of different approaches to multipath routing in wireless networks.

* This research was partially supported by the Academy of Finland under Grant No 209300 (ACSENT).

** Corresponding author.

Most of the existing literature on this topic takes as its starting point some natural heuristic for multipath routing and investigates its behaviour either analytically or by means of simulation studies. The recently introduced *Balanced Multipath Source Routing* (BMSR) protocol [5], however, takes a different approach. Here the starting point is a linear programming approximation algorithm [6, 7] that *provably guarantees* a desired bound on the congestion performance of the routing scheme, and this algorithm is then given a *distributed implementation* by extending the DSR protocol. Since in DSR, alternate routes can easily be collected to the source node, it provides a simple and lightweight platform for multipath routing extensions, as also noted by other authors [1].

The present work validates the behaviour of the BMSR protocol in several networking scenarios, using the `ns2` network simulator [8]. In the original article [5], average throughput and packet delay provided by BMSR were compared to those of DSR on a simple square grid of 10×10 nodes, with one source-destination pair generating traffic left-to-right and another bottom-to-top on the grid. In this paper, we first investigate how the placement of nodes on a grid influences the behaviour of the algorithm, as opposed to a random placement with a similar node density. Observing that the effect is not significant, we continue with a grid placement in order to eliminate one source of random effects in the simulation results. The following sets of simulation experiments contrast the behaviour of BMSR to DSR and an idealised shortest-path routing scheme using various placements of source-destination pairs, and with attention to the scalability of the different algorithms as the number of source-destination pairs increases.

Overall, for all the network setups we considered, BMSR outperforms DSR significantly. BMSR also performs better than the shortest-path algorithm when source-destination pairs are placed densely or the shortest-paths are not disjoint and there are only few sources simultaneously active in the network.

The rest of the paper is organised as follows. In the next section, we first give a brief overview of DSR and describe its basic operation. Thereafter, we describe the basic linear programming congestion-control approximation algorithm [6, 7] and its implementation as an extension to DSR. Section 3 presents the `ns2` simulations conducted and compares the performance of BMSR with DSR and a global shortest-path routing algorithm. Finally, Section 4 presents some conclusions and outlines future research directions.

2 The BMSR Protocol

The recently proposed BMSR protocol [5] extends the Dynamic Source Routing (DSR) [2] protocol with a distributed approximation algorithm to optimise network congestion.

In this section we first outline the basic operation of DSR and those aspects of it that we will modify. Thereafter, we describe how we obtain BMSR by integrating the approximation algorithm for load balancing into DSR.

2.1 DSR

DSR is an on-demand source routing protocol: the source includes the whole route in every packet sent, and routes are discovered only when required. The basic DSR protocol consists of two operations: *route discovery* and *route maintenance*. If a source node wishes to send to a destination to which it does not have a route in its route cache, it initiates the route discovery process by broadcasting a *route-request* (`RREQ`) message to its neighbours. Upon receiving the `RREQ`, if a node knows of a route to the destination it can send a *route-reply* (`RREP`) message back to the source; otherwise it will append its own address to the list of nodes in the `RREQ` and forward the request. Upon receiving the `RREQ`, the destination obtains a route from the source to the destination, and in the presence of bidirectional links it can simply reverse this route and use it for sending a `RREP` message along this route to the source. A sequence number mechanism ensures that a node only forwards each `RREQ` at most once. Since shorter routes require fewer hops, the first `RREQ` to reach the destination is likely to have taken a route that is minimum or close to minimum in terms of the hop count.

If a source route breaks, the source is notified by the intermediate node that detects the break. The source may then resend the packet using an alternative route in its route cache, or initiate a new route discovery. If the intermediate node has a different route to the destination in its own cache, it can initiate *packet salvaging* and forward the packet using this alternative route.

2.2 BMSR

In this section we describe a method for obtaining multiple source-destination routes for a given source-destination pair by a linear programming approximation algorithm that minimises flow congestion [7]. The algorithm relies on the computation of shortest paths determined by an adaptive cost metric using weights on the links. The weight updates are distributed to avoid dissemination of global information. Each shortest path computed becomes a source route for the BMSR protocol. The data flow is then equally distributed over these pre-computed routes.

We model balancing the traffic in the network as a multicommodity flow problem in a directed graph $G = (V, E)$ where the set of vertices V corresponds to the radio nodes in the network. There is a directed edge $(i, j) \in E$ from vertex i to vertex j if the radio node corresponding to j is within transmission range of the radio node corresponding to i; this is the well known *unit disk graph model*. Each source-destination pair is modelled as a commodity so that there is a supply of the commodity at the source node and a demand, modelled as a negative supply, at the destination node. With x_{ij}^c denoting the flow of commodity c along the edge from vertex i to vertex j and $s^c(i)$ representing the supply of commodity c at vertex i, to route the transmissions from the respective sources to destinations we must satisfy

$$s^c(i) + \sum_{(j,i)\in E} x_{ji}^c - \sum_{(i,j)\in E} x_{ij}^c = 0. \tag{1}$$

The version of the algorithm we use requires that each radio link (and thus every edge $(i, j) \in E$) has the same fixed capacity u_{ij}, but we do not limit the total flow $f_{ij} = \sum_c x_{ij}^c$ along an edge; rather our aim is to balance the flow in the network by minimising the maximum congestion in the network:

$$\min \max_{(i,j)\in E} \frac{f_{ij}}{u_{ij}} \ . \tag{2}$$

The optimisation method that we will implement as an extension to DSR is based on the work of Young [6], as summarised by Bienstock in [7]. To route a flow of rate $s^c(i)$ of commodity c from the source node i to a destination, the weight of each edge is initialised to 1. We then run I iterations: in each iteration, for each commodity c, the least-weight path from the source to the destination is determined, the flow of commodity c along each edge on the path is increased by $s^c(i)/I$, and the weight w_e of each edge e on the path is updated as

$$w_e \leftarrow (1 + \epsilon s^c(i))w_e. \tag{3}$$

where $0 < \epsilon \leq 1/2$ is a parameter of the algorithm. It can be shown that if the number of iterations I is sufficiently large, the result of this algorithm is optimal to a factor of $(1 + \epsilon)$ [6, 7].

Our routing method, BMSR, is an implementation of this algorithm on top of DSR. Each radio node maintains a record of the weights of its incoming links. Our balanced route request packet, `BREQ`, is a modified version of the standard route request packet (`RREQ`) of DSR that additionally contains a record of the total weight of the path the `BREQ` has taken so far. Contrary to DSR, if a node receives a `BREQ` packet related to a route request it has already seen, it may resend it if the second packet has a lower weight than the previously seen `BREQ` packets related to the same route request. This allows us to find routes of minimum weight.

After receiving a `BREQ` packet the intended destination waits for a while, aiming to make sure that lower weight routes represented by `BREQ` packets that arrive later are taken into account in choosing the route. The destination then sends a route reply packet back to the source, and all nodes on the path update the weights of the edges on the path according to (3).

To obtain a good selection of routes, a source node may run a large number $I$of route requests. The iteration number I also depends on the chosen approximation quality parameter ϵ: for larger ϵ fewer iterations are necessary in order to obtain an acceptable selection of routes, but the resulting flow may be further away from the optimum.

In BMSR, routes that are broken due to temporarily congested links are not invalidated. With each source having a balanced collection of routes to the destination the effect of a single link failure diminishes, as the source distributes the traffic equally among the routes in its the cache.

3 Simulation and Performance Evaluation

To experimentally validate the BMSR algorithm, we performed simulations for a number of different network topologies using the `ns2` network simulator [8]. An initial set of experiments was used to investigate the algorithmic effect of arranging the simulated nodes in a grid structure, as opposed to the arguably more natural placement uniformly at random in a corresponding area. Thereafter, the number and position of source-destination pairs was varied to determine the effect of differing traffic patterns on the algorithm's performance, as well as to get an intuition how BMSR scales with the number of source-destination pairs.

As a performance metric in the evaluations, we have used the average throughput over the simulated time, taken over all source-destination pairs in use at each simulation. Using this metric, we compare BMSR to standard DSR and an idealised shortest-path routing (SPR) algorithm.

3.1 Review of Previous Experiments and Random Node Placement

As part of previous work [5] we tested BMSR on a simulated 2160 m × 2160 m grid network of 100 nodes. We placed two pairs of constant bit rate (`CBR`) traffic source and destination nodes at the boundary of the grid, so that the direct connections between both pairs would approximately form a cross shape. In this setup each node may communicate with the nodes beside, above or below it. The supply value s^c for both source nodes was chosen to be 1. However, one should note that due to the update rule given in (3), the particular choice for s^c has only a scaling effect on the approximation quality parameter ϵ. Therefore, s^c will be also fixed to 1 for all further experiments discussed in this paper.

We then compared DSR to BMSR for a range of `CBR` packet-sizes and BMSR parameters ϵ and I. BMSR clearly outperformed DSR, both in terms of packet delay and network throughput. The simulations showed a performance gain of 14% to 69%. We also analysed the effect of BMSR on network load and on collisions due to interference and interface queue (IFQ) overflows. BMSR led to a more balanced distribution of the load in the network, which effectively reduced packet loss resulting from IFQ overflows. A choice of $\epsilon = 0.05$ and $I = 160$ showed the best average throughput over both source-destination pairs.

In the present set of experiments, we first evaluate the impact of the regular network structure on the performance of BMSR and DSR, by considering nodes placed uniformly at random within a square area. The locations of the source and destination nodes, as well as the total number of nodes, are kept the same as in the experiments in [5]. In order to maintain connectivity, we roughly doubled the density of nodes within the network by scaling the network down by a factor of approximately $\sqrt{2}$, yielding a network size of 1530 m × 1530 m. We then compared results for `CBR` traffic throughput of a grid with the throughput of networks with randomly placed nodes using the same dimensions and location of sources and destinations. The simulation parameters are summarised in Tab. 1. The `ns2` default value for transmission range, 250 m for our TwoRayGround

Table 1. The parameters used in `ns2` simulations.

`CBR` packet size:	2048 B	MAC bandwidth:	1 Mbit
`CBR` data rate:	160 Kbit/s	MAC protocol:	802.11 with `RTS/CTS`
Antenna type:	OmniAntenna	Propagation model:	TwoRayGround
Transmission range:	≈ 250 m	Interference range:	≈ 550 m
Max. source route length:	22, 26	Max. IFQ length:	50
Node count:	100, 200	Network size:	1530 m × 1530 m, 2160 m × 4320 m
Simulation time:	1500 s	Balancing setup time:	500 s
BMSR parameters:	$I = 160$, $\epsilon = 0.05$		

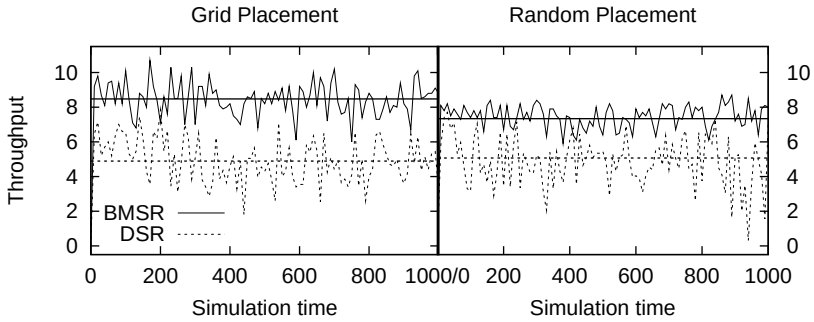

Fig. 1. Average throughput of both source-destination pairs in KB/s versus simulation time for a single run of DSR and BMSR, setup stage for BMSR is omitted from the plot. Performance over 15 runs is shown for each plot.

propagation model, was used, enabling nodes to communicate with the closest nodes located in their vicinity, including diagonal neighbours.

As can be seen in Fig. 1, random node placement does not affect the performance of either routing algorithm significantly. However, a slight performance decrease for BMSR is observable. This may be due to interference caused by the larger density of nodes in the network, as BMSR does not take into account interference between radio links.

Due to the only minor decrease and to prevent the introduction of an additional source of random effects, we focus further simulations on the grid topology.

3.2 Densely Placed `CBR` Pairs

In this section we describe experiments where the sources are located directly opposite to their respective destinations and the source-destination nodes are placed next to each other on the grid, as shown in Fig. 2. The width of the grid was doubled to determine the spread of routes over the network.

We also increased the bound for the number of hops in each route to facilitate the potential increase in route length. This value, which is a constant given in the `ns2` implementation of DSR, determines the spreading of routing-control

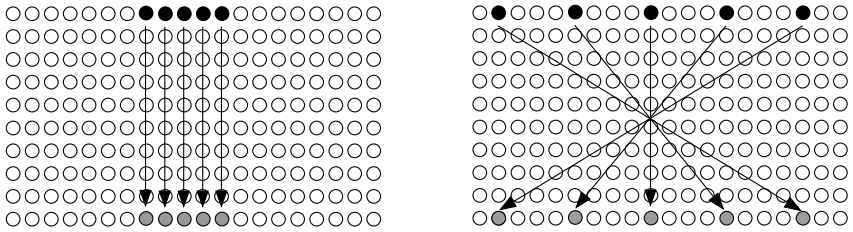

Fig. 2. Dense and twisted sparse network setup: Left unidirectional for all source and destination pairs, on the right side the row of destinations is twisted around.

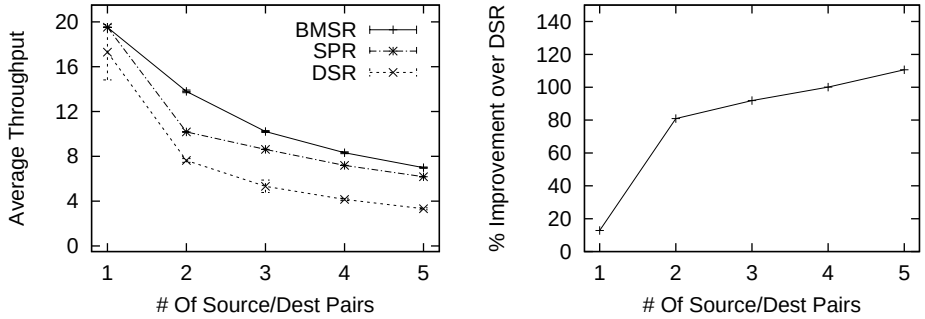

Fig. 3. Average throughput of multiple densely-placed source-destination pairs in KB/s using routing algorithms DSR, BMSR, and SPR. Errorbars represent the standard deviation over 15 repetitions.

packets, such as `RREQ`, in the network as well as the connectivity between nodes. However, `ns2` resource consumption forced us to choose a rather conservative value of 26 hops. We then explored the performance of DSR, BMSR and SPR.

The SPR algorithm was initialised to use a route of minimum length from the source to the destination node for all packets during the simulation run, independent of route failures. In this sense it behaves similar to BMSR, which determines routes in the setup phase of the algorithm. The algorithm was chosen to evaluate the benefit from choosing multiple routes over a single shortest route for a given network setup.

Because the source and destination nodes are packed closely together, there exist a large number of nodes to the left and to the right, respectively, of the leftmost and rightmost source and destination nodes. In this setup, one can observe from Fig. 3 that BMSR outperforms both DSR and SPR. The latter two routing methods tend to use routes that are close to each other, so that up to three shortest paths can interfere with each other. Route interference, in turn, causes collisions and packet drops due to IFQ overflows.

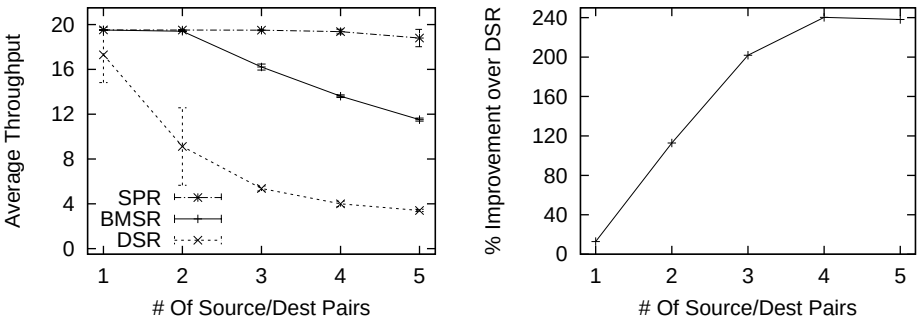

Fig. 4. Average throughput of multiple sparsely-placed source-destination pairs in KB/s using routing algorithms DSR, BMSR, and SPR. Errorbars represent the standard deviation over 15 repetitions.

Note that DSR buffers packets scheduled to be sent over the wireless interface in a queue. After several retransmissions fail, these packets are dropped and removed from the IFQ, causing a decrease in throughput. Hou and Tipper [9] observed that one of the main reasons for the decline in throughput for congested networks using DSR is the overflow of the IFQ of congested nodes.

In the dense setup, as the number of source-destination pairs increases, shortest-path routes can be expected to be the most favourable in terms of causing less interference than any other choice of routes. Thus the comparative advantage of BMSR with respect to SPR decreases. This trend is observable in Fig. 3. However, as densely packed routes are subject to a higher rate of collisions and retransmissions, SPR also shows a decreasing performance for an increasing number of `CBR` pairs.

3.3 Sparsely Placed `CBR` Pairs

Subsequently, we separated the source and destination nodes by three intermediate nodes. Because of the spacing, adjacent shortest-path routes do not conflict with each other. As a consequence, one can see from Fig. 4 that the performance of SPR remains constant for an increasing number of source-destination pairs.

In this setup, BMSR is able to take advantage of the additional nodes between adjacent shortest-paths and shows an increased throughput compared to the dense placement of source-destination pairs, while DSR does not perform significantly better than for the dense setup. As DSR heavily relies on cached routing information, which is updated by routes overheard from neighbours or taken from forwarded packets, nearby sources tend to share parts of their routes over the long run.

In order to evaluate the performance of BMSR for a scenario with a large number of route-intersections, we created a worst-case scenario for route intersections by 'twisting' the aforementioned setup. Figure 2 depicts the resulting

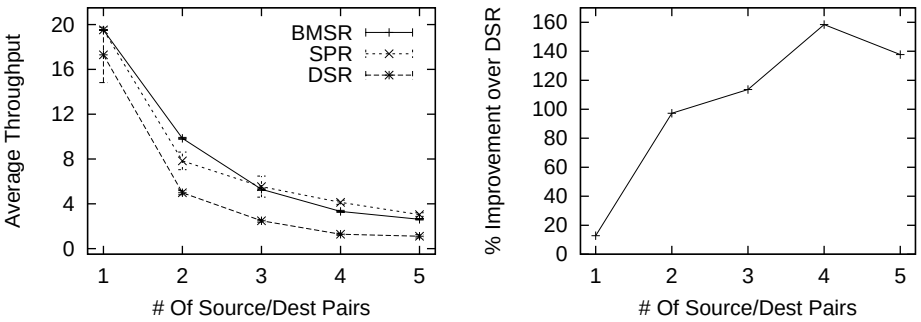

Fig. 5. Average throughput of multiple sparsely-placed source-destination pairs with twisted destination arrangement in KB/s using routing algorithms DSR, BMSR, and SPR. Errorbars represent the standard deviation over 15 repetitions.

network. It is easy to see that the number of pairwise route intersections is $n(n-1)/2$, where n is the number of source-destination pairs.

Figure 5 shows performance results obtained for this network setup. It is interesting to see that BMSR performs very similarly to SPR. This can be explained by the fact that each route will necessarily cross any other route, which causes collisions and congestion at intermediate nodes. However, due to the spreading achieved especially for the inner source-destination pairs, BMSR still outperforms SPR for a smaller number of active sources. As the maximum source-route length restricts the choice of routes for the balancing algorithm, the degree of freedom for outer pairs is much smaller than for inner pairs. In fact, for five source-destination pairs, the outmost pair will always route over shortest paths, determined by the maximum source-route length of 26.

DSR again shows the least performance for this setup. Congested nodes and collisions due to the heavy load in the centre of the network cause routes to fail repeatedly. These have to be rediscovered regularly, causing additional overhead of routing control messages.

4 Conclusions

In this paper, we have applied a linear programming approximation algorithm to the problem of load balancing in ad hoc networks. When integrated into the DSR routing protocol, the resulting BMSR protocol relies on a mathematical formulation of the underlying problem. This is an advantage over routing protocols which aim to achieve load balancing by introducing heuristic rules, for which there is little mathematical justification like that provided here for BMSR.

A limiting factor of BMSR is its non-adaptivity towards node and permanent link failures, as well as changes in the demands of source-destination pairs and the network topology in general. An adaptive version of the proposed protocol could adjust edge flows by locally rerouting flow for example. The second limiting

factor is the non-integration of edge interference into the model. This problem could be solved by considering clusters of interfering edges as units to be used for the load balancing procedure, rather than edges themselves.

BMSR is a lightweight, distributed implementation of a linear-programming approximation algorithm. Its integration into a standard routing protocol is enabled by the computation of shortest paths. In this paper, we have presented simulations that also indicate its robustness for different traffic patterns. For all simulations BMSR clearly outperforms DSR. BMSR also performs better than the idealised shortest-path algorithm when source-destination pairs are placed densely or the shortest-paths are not disjoint and there are only few sources simultaneously active in the network. Therefore, we conclude that load balancing based on source routing can provide benefits for stationary networks, e.g. mesh networks with a high throughput requirement.

References

1. Nasipuri, A., Castañeda, R., Das, S.R.: Performance of multipath routing for on-demand protocols in mobile ad hoc networks. Mobile Networks and Applications **6**(4) (2001) 339–349
2. Johnson, D.B., Maltz, D.A., Hu, Y.C.: The dynamic source routing protocol for mobile ad hoc networks (DSR). Technical report, IETF (2003) IETF Draft, July 2004, work in progress.
3. Wu, K., Harms, J.: Performance study of a multipath routing method for wireless mobile ad hoc networks. In: Proc. of 9th International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Washington, DC, USA, IEEE Computer Society (2001) 99–107
4. Mueller, S., Tsang, R.P., Ghosal, D.: Multipath routing in mobile ad hoc networks: Issues and challenges. In: Performance Tools and Applications to Networked Systems: Revised Tutorial Lectures, LNCS **2965**, Springer-Verlag (2004) 209–234
5. Schumacher, A., Haanpää, H., Schaeffer, S.E., Orponen, P.: Load balancing by distributed optimisation in ad hoc networks. In: Proc. of the 2nd International Conference on Mobile Ad-hoc and Sensor Networks (MSN 2006). (2006) To appear.
6. Young, N.E.: Randomized rounding without solving the linear program. In: SODA '95: Proc. of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (1995) 170–178
7. Bienstock, D.: Potential Function Methods for Approximately Solving Linear Programming Problems: Theory and Practice. Volume 53 of International Series in Operations Research & Management Science. Kluwer Academic Publishers, Norwell, MA, USA (2002)
8. McCanne, S., Floyd, S., Fall, K., Varadhan, K.: The network simulator `ns2` (1995) The VINT project, available for download at http://www.isi.edu/nsnam/ns/.
9. Hou, X., Tipper, D.: Impact of failures on routing in mobile ad hoc networks using DSR. In: Proc. of the Communication Networks and Distributed Systems Modeling and Simulation Conference. (2003)