Load Balancing by Distributed Optimisation in Ad Hoc Networks*

André Schumacher, Harri Haanpää, Satu Elisa Schaeffer, and Pekka Orponen

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

Abstract. We approach the problem of load balancing for wireless multi-hop networks by distributed optimisation. We implement an approximation algorithm for minimising the maximum network congestion as a modification to the DSR routing protocol. The algorithm is based on shortest-path computations that are integrated into the DSR route discovery and maintenance process. The resulting Balanced Multipath Source Routing (BMSR) protocol does not need to disseminate global information throughout the network. Our simulations with the `ns2` simulator show a gain of 14% to 69% in the throughput, depending on the setup, compared to DSR for a high network load.

1 Introduction

Ad hoc networks are communication networks formed by a number of nodes, which are small radio devices with limited computational capacity [1]. Perhaps the most significant advantage of ad hoc networks – and simultaneously an important design goal – is the ease of deployment. Ideally, it should be possible to deploy the nodes in the area of operation and have them self-organise to route traffic as necessary. Such a setup would be useful in a variety of environments ranging from military operations and disaster relief to commercial applications.

Ad hoc networks also present challenges. Nodes are usually battery-operated, as they should not depend on an external energy supply, and battery life is often a limiting factor. The radio transmission channel is limited in bandwidth and shared among nearby nodes. Determining and maintaining the network topology in a distributed fashion is a challenging problem, particularly if the network topology changes during operation due to addition, removal, or mobility of nodes.

Two properties of algorithms are particularly desirable in an ad hoc context. First, an algorithm should be mathematically justified. Analysing an algorithm mathematically gives insight into when it can be expected to work and when not. Linear and integer programming formulations can typically be applied in this approach to gain optimal solutions for small problem instances or good

* This research was supported by the Academy of Finland under grants 209300 (ACSENT) and 206235 (ANNE).

J. Cao et al. (Eds.): MSN 2006, LNCS 4325, pp. 873–884, 2006.

approximate solutions for larger instances. Such methods have been applied to optimisation of sensor-node coverage [2] and lifetime in energy-constrained networks [3], but these approaches typically require collecting state information to a central location to perform the optimisation, adding undesired hierarchy and a point of failure.

Second, an algorithm should be distributed and non-hierarchical. Each node should follow a simple set of rules to cooperate in computing the optimum. Neither the size nor the number of messages should grow rapidly with the size of the network. Such approaches have been used for bandwidth optimisation [4]. Certain energy-aware modifications of routing protocols such as AODV or DSR also fall into this category. However, formal analysis of heuristic optimisation methods is difficult and usually only simulation-based analysis is applicable.

In mathematically justifiable distributed algorithms, the nodes typically compute graph-based properties, such as shortest paths or spanning trees, in a distributed and iterative manner. This enables theoretic analysis of the expected quality of the solution and the convergence of the algorithm towards the optimum. Such methods have been applied to adjusting transmission power levels based on lowest-cost energy paths [5] and routing around congested nodes based on node potentials and the steepest gradient method [6].

In this paper, we present the Balanced Multipath Source Routing (BMSR) protocol that extends the Dynamic Source Routing Protocol (DSR) [7] to use *multipath routing* for balancing data traffic. Multipath extensions to DSR have been previously studied: Nasipuri, Castañeda, and Das [8] introduce alternate routes to the route discovery process, whereas Wu and Harms [9] propose a heuristic redirection of `RREP` messages to gain alternative routes. The focus has been primarily on the computation of node or link-disjoint paths, as they provide a higher fault tolerance in the presence of failures. Ganjali and Keshavarzian [10] state that multipath routing *alone* can not improve load balancing: as node density increases, the choice of shortest paths connecting any pair of nodes leads to congestion in the centre of the network. They conclude that additional incentive is needed to push traffic away from the centre.

Multipath-based network optimisation has been studied extensively for wired networks. Vutukury and Garcia-Luna-Aceves [11] propose an algorithm to minimise delay by heuristic redirection of flow over multiple paths. Basu, Lin and Ramanathan [6] present a potential-based routing method that forwards packets using steepest gradient search and propose a traffic-aware routing algorithm. This approach relies on a link-state routing algorithm for the dissemination of link information throughout the entire network.

However, most proposals are not directly applicable to ad hoc networks due to the aforementioned limitations. Our proposed BMSR protocol constitutes modifications of the DSR protocol and obtains multiple source-destination routes by a linear programming approximation algorithm that minimises flow congestion [12]. The algorithm relies on the computation of shortest paths determined by an adaptive cost metric using link weights. Using distributed weight updates we

avoid dissemination of global information. In our simulations we achieve a gain of 14% to 69% in the throughput, depending on the setup, compared to DSR.

The paper is organised as follows. In the next section, we describe the linear programming approximation algorithm, followed by a brief overview of the basic operation of DSR. Thereafter, we describe extensions that were made to DSR in order to implement the approximation algorithm, which constitute the proposed BMSR protocol. Section 3 presents simulation results with the `ns2` [13] network simulator. Finally, Section 4 concludes the paper and outlines future research directions.

2 Distributed Load Balancing

In this section we describe the BMSR protocol, which is an extension to DSR for load balancing by multi-path routing. DSR normally uses one route from the source node to the destination node. However, extending DSR to use more routes is relatively easy and may improve reliability, throughput, and load balancing. We model choosing a set of source routes as a min-max congestion multicommodity flow problem and describe the implementation of the approximation algorithm BMSR is based on.

2.1 Approximation Algorithm

We model the ad hoc network as a directed graph $G = (V, E)$ with vertices representing the radio nodes of the network and edges representing links between the radio nodes. For two vertices $i, j \in V$, we have a directed edge $(i, j) \in E$ if there exists a link from radio node i to j.

We consider routing as a multicommodity flow problem: each commodity c represents one data stream of traffic of volume v^c from the source s^c to the destination d^c. When $t^c(i)$ represents the supply of commodity c at vertex i, we have $v^c = t^c(s^c) = -t^c(d^c)$ and $t^c(i) = 0$ for all other nodes i. The task is to find flows x_{ij}^c of commodity c along each edge (i, j) that satisfy the flow requirements

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

for each commodity c and vertex i. Within these constraints we choose x_{ij}^c to *minimise the maximum congestion*:

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

where $f_{ij} = \sum_c x_{ij}^c$ is the total flow along edge (i, j) and u_{ij} is its capacity.

Many algorithms exist for solving such linear optimisation problems when the whole state of the network is known. In contrast, we need an optimisation algorithm that can be implemented so that the individual nodes cooperate to determine the optimum by passing only a reasonable number of messages of reasonable size. The approximation algorithm that we use for min-max congestion multi-commodity flow is from [12]; it computes a flow x over a set of paths,

taking as input a graph $G = (V, E)$ and a list of flows of volume v^c from source s^c to destination d^c, with parameters I and ϵ.

1. Initialise $w_{ij} = 1$ for each edge $(i, j) \in E$. For each edge (i, j) and every commodity c (with source node s^c and destination node $d^c \in V$), set the flow $x_{ij}^c = 0$.
2. For each of the I iterations, do the following computation:
   (a) For each source-destination pair of nodes s^c and d^c, compute the shortest path $p(s^c, d^c)$ with respect to the edge weights defined by w.
   (b) Let y^c be the flow vector resulting from routing v^c units of flow on the shortest path $p(s^c, d^c)$. For each edge $(i, j) \in E$, assign $x_{ij}^c := x_{ij}^c + y_{ij}^c$ and
   $$w_{ij} := \big(1 + \epsilon \sum_c y_{ij}^c\big) w_{ij} \quad . \tag{3}$$
3. Scale the total flow by letting $x := x/I$.

In this formulation, each edge has the same capacity u. To obtain flows for which the maximum congestion is at most $(1 + \epsilon)$ times the optimal value, it suffices to run the algorithm for

$$I \geq \lceil 4m \log m/\epsilon^2 \rceil \tag{4}$$

iterations, where m is the number of edges.

2.2 DSR Operations

DSR [7] is an on-demand *source routing* protocol: the source includes the whole route in every packet sent. This property eliminates the need for actively maintaining routing information at intermediate nodes and enables an easy integration of multipath routing. Nodes keep routing information in their *route cache*, which can also contain routing information that was overheard from neighbouring nodes.

The basic DSR protocol consists of two operations: *route discovery* and *route maintenance*. If a source node wishes to send a packet 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`, nodes consult their route cache and can decide to send a *route-reply* (`RREP`) message back to the source. If they do not know a route to the destination, they append their own address to the list of nodes in the `RREQ` and forward the request further, until it eventually reaches the destination. The destination obtains a route from the source to itself by consulting the list of nodes that forwarded the `RREQ`. 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 limited forwarding of `RREQ`'s by intermediate nodes. In route discovery, 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 (close to) minimal in terms of the hop

count. Therefore, DSR chooses routes not much longer than the shortest route between source and destination. Although in principle multiple routes to the same destination may be contained in the route cache, e.g. by overhearing other routes, the nodes always pick the shortest route from the cache.

The basic route maintenance includes reliable packet transmissions from one hop to the next, e.g. utilising link-layer acknowledgements. Additionally, there are other operations initiated on-demand. If a source route breaks, the source is notified by an intermediate node detecting the break. The source can then choose to select an alternative route to the destination by consulting its route cache, or initiate a new route discovery. In the case that 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.

2.3 The BMSR Protocol

The shortest-path methodology of the approximation algorithm enables a simple extension to DSR; the computation of shortest paths is similar to that of the original protocol. Our approach differs from DSR in that DSR initiates route discovery when necessary, while BMSR uses an initial setup phase to proceed through the iterations of the balancing algorithm. Each source obtains one *balanced route* to the destination per iteration. Some routes may occur more than once. After the setup phase, every packet sent by the source follows a randomly chosen cached route. Unlike in DSR, the routes are not removed from the cache when link failures occur, as the failure may be due to temporary link congestion.

We implement BMSR by modifying DSR's route discovery and route maintenance operations. The DSR route control messages `RREQ` and `RREP` are extended to include *iteration-index*, *cost* and *flow-value* fields. These fields correspond to the variables needed for the algorithm of Section 2.1. For clarity, we refer to these modified messages by `BREQ` and `BREP`. Instead of computing shortest routes based on hop-counts, the nodes compute the minimum-cost route for each iteration of the balancing algorithm and each source and destination pair. The cost of a route is the sum of the link costs w on that route. Each node keeps track of the weight of and the flow on each incoming link (i.e., those links that it may use to receive a `BREQ` message from a neighbour). `BREQ` messages carry, in addition to the list of addresses of nodes that re-broadcasted the message, the accumulated route cost from the source. An intermediate node adds the cost of the incoming link on which it received the `BREQ` to the accumulated route cost of the `BREQ` upon re-broadcasting it. Later, however, an intermediate node may receive another `BREQ` packet with the *same* iteration index. If the new `BREQ` has a lower-cost route from the source than the previous one, the intermediate node re-broadcasts it.

When the destination receives a `BREQ` packet, it must wait a short period for possible lower-cost `BREQ` packets. The destination only replies with a `BREP` to the `BREQ` with lowest cost. The flows and weights are updated along the route used when the destination sends the `BREP` packet back to the source. As the link weights and therefore the least-cost routes are subject to change at

each iteration, the balanced routing algorithm can not rely on DSR's caching mechanism to narrow down the dissemination of `BREQ` messages in the network. Therefore, `BREQ`'s have to spread by flooding through the network. Since the parameters ϵ and I can be used for a trade-off between route-control overhead and quality of the solution, this effect can be adjusted to the network setup. Additionally, the setup phase is only performed once even for long data streams.

As mentioned above, routes that are broken due to temporarily congested links *stay* in the cache and do not get invalidated. For a larger number of iterations the effect of a single link failure diminishes, as the source randomly selects balanced routes from the cache.

3 Experiments

We consider a stationary grid network with source and destination pairs. The chosen traffic pattern resembles a mesh-network scenario, where a large amount of constant bit rate (CBR) data is transfered through an already congested network. When a sudden demand arises for transmitting a large amount of data between a dedicated pair of nodes, e.g. between a control centre and rescue teams, one aims to deliver as much of the critical data as possible. For this purpose one must balance the traffic among the nodes and utilise the network capacity to maximise throughput over source-destination pairs.

We compare BMSR to DSR by using `ns2` to simulate it on a 10 by 10 square grid with two CBR flows, from s_1 to d_1 and from s_2 to d_2; see Fig. 1 for the network setup. Both CBR sources are transmitting with a previously determined rate and packet size. See Table 1 for the particular parameter values. Prior to initiating the CBR traffic, we run the balancing algorithm of Section 2.1 for a chosen value of ϵ and a chosen number of rounds I to select routes that give an approximately balanced flow in the sense of minimising the maximum congestion. We run a series of long simulations to obtain estimates of the throughput of the network, defined as the average rate of CBR data that was received by the destinations. In the following we will refer to this metric as the performance for the particular choice of parameters. We use the same source-destination setup to transmit data using the DSR implementation provided in `ns2`.

Table 1. The parameters used in `ns2` simulations

CBR packet size (B)	256, 512,1024, 2048	MAC bandwidth	1 Mbit
CBR data rate	160 Kbit/s	MAC protocol	802.11 with `RTS/CTS`
Antenna type	OmniAntenna	Propagation model	TwoRayGround
Max. IFQ length	50	Max. route length	22
Network size	2.4 km × 2.4 km	Node count	100
Simulation time	1500 s	Balancing setup	500 s

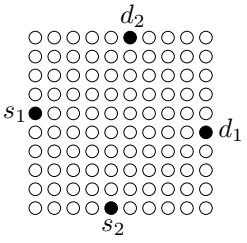

Fig. 1. The simulation setup: two source-destination pairs (s_1, d_1) and (s_2, d_2) are placed "off-by-one" on the opposite sides of the grid. The source nodes s_1 and s_2 send data packets at constant rate to their respective destination nodes d_1 and d_2. Each node may communicate with the nodes beside, above or below it.

In addition to throughput, we study the distribution of routes over the nodes by calculating the number of forwarded CBR packets at each node. We expect most packets to be forwarded by nodes located near the centre of the network, as these routes are shortest and the algorithm initially prefers shorter routes over longer ones. However, the central nodes should not be loaded much more heavily than those on slightly longer paths.

A balanced network load should also reduce collisions and *interface queue* (IFQ) overflows in the network. The IFQ contains packets that are scheduled to be transmitted over the network interface. Hou and Tipper [14] observed that one of the main reasons for the decline in throughput for congested networks running DSR is the overflow of the IFQ of congested nodes. Besides queue overflows, collisions of the *media access control* (MAC) layer control messages and CBR packets are expected to degrade the performance. Although we do not expect the number of collisions to be significantly lower compared to the DSR route selection, we would expect a more even distribution over the nodes, preventing bottleneck formation. Figure 2 shows simulation results for two CBR packet sizes.

We use the following measures: CBR *packet load*; the number of CBR packets sent by the MAC layer of the node. Note that there are in total 20000 and 10000 packets per source for packet sizes of 1024 and 2048 bytes respectively. This value does not correspond to the actual number of successfully forwarded packets, as drops and collisions have to be subtracted. Sources were excluded from Fig. 2 for clarity. CBR *packet collisions*; the number of CBR MAC layer collisions caused by interference that occurred at each node, excluding the sources. These numbers do not necessarily coincide with the number of dropped packets, as the MAC layer uses a retransmission scheme. IFQ *overflows caused by* CBR *packets*; the number of IFQ overflow events that occurred at each node.

One might expect DSR to favour shorter routes, yielding an increased network load within the centre of the grid that results in interference and a low network throughput. BMSR should recognise areas of higher congestion and after

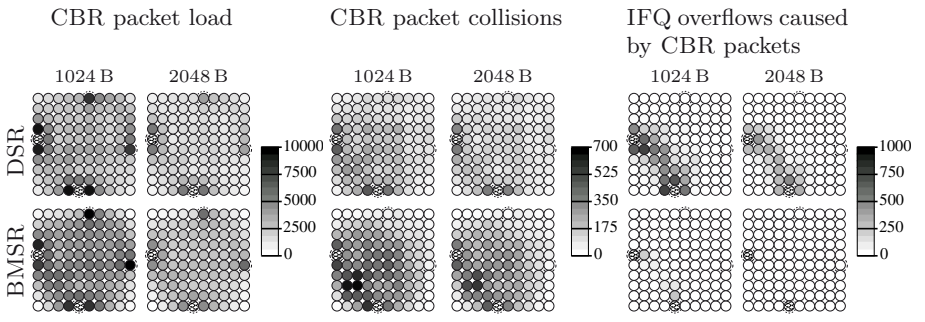

Fig. 2. Averages over five runs for the performance measures of DSR and BMSR for $I = 160$ iterations and $\epsilon = 0.05$ for two CBR packet sizes; variations were negligible. Source and destination nodes are indicated by dashed circles.

initially selecting shorter routes, select routes that avoid the potentially congested areas. In Fig. 2, we only observe minor differences for BMSR and DSR. Depending on the averaging of packet load over the rather long simulation run, the load for DSR appears to be well balanced. The reason is that within the congested network, rediscovered routes will typically be different from recently broken routes. There is a slightly higher utilisation of boundary nodes by DSR, but the overall network load for BMSR is higher than for DSR, which can be explained by the higher throughput, discussed later in this section.

Due to higher load, BMSR encounters more collisions compared to DSR. A remarkable effect is the concentration in the quadrant of the network formed by the square with the sources on its diagonal. The effect is apparent for both algorithms and packet sizes, but emphasised for BMSR and 1024-byte packets. Nodes within this part of the network may be relaying packets from both sources in roughly opposite directions. Hence they have to transmit packets in more diverse directions than nodes within the vicinity of the destinations.

As the MAC layer transmission of a CBR packet includes a *request to send* (`RTS`)/*clear to send* (`CTS`) handshake, collisions are more likely to occur when nodes are transmitting in different directions than when the packets travel roughly in the same direction. DSR always uses the shortest known route to the destination. Therefore, subsequent packets for the same destination are less likely to interfere with each other. The distribution of IFQ overflows follows basically the same principle. We, however, observe a major difference between BMSR and DSR: the single-path routing of DSR leads to the formation of bottleneck nodes due to congestion in the bottom left quadrant of the network. As DSR prefers shorter paths, such overloading of nodes is restricted to the band of nodes between the sources. The effect is stronger for smaller packet sizes, explained by the increased MAC layer overhead. BMSR shows hardly any IFQ overflows at all, except within the vicinity of the sources.

Figure 3 shows the performance of both routing protocols over time. Comparing throughput for BMSR and DSR, one observes larger fluctuations for

DSR. A major reason for the throughput stability of BMSR is that broken links do not cause route invalidation. Therefore, its performance is determined during the initial setup phase of the algorithm. To compensate the fluctuations of DSR, we consider the throughput over 1000 s from the time when CBR transmissions have been initiated to compare both algorithms in the following. For both packet-sizes BMSR clearly outperforms DSR.

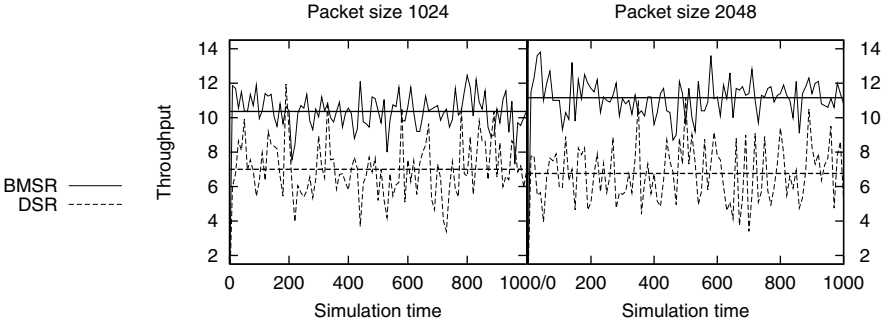

Fig. 3. Average throughput of both source-destination pairs in KB/s versus simulation time for a single run of BMSR and DSR. Note that the setup stage for BMSR is omitted from the plot. The parameter values for the balancing algorithm were $I = 160$ and $\epsilon = 0.05$. The horizontal lines are averages over the entire simulation.

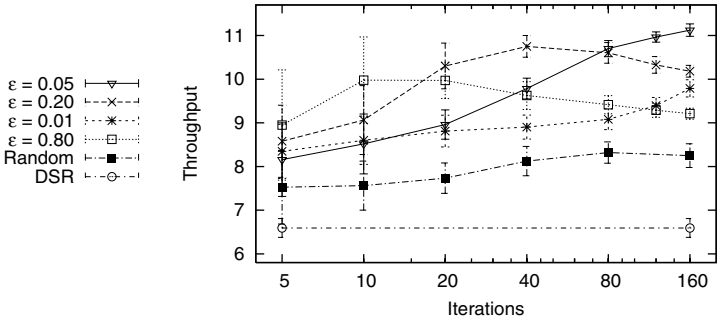

Fig. 4. Performance in KB/s as a function of I and ϵ for CBR packet size 2048: BMSR, DSR, and random route selections of I routes between source-destination pairs. All values are averages over at least 15 repetitions (standard deviations shown). The legend ordering corresponds to the throughput value at $I = 160$.

We also studied the effect of the I and ϵ parameters on the performance. The results are summarised in Fig. 4 and are mostly as expected; already for a modest number of iterations we obtain throughput superior to DSR. There is a dependency of the throughput on ϵ and I: for larger values of ϵ, fewer iterations

are needed to obtain a good throughput, but running a large number of iterations with a small value of ϵ yields a slightly better throughput.

A curious phenomenon in the results is that for a given value of ϵ, the throughput first increases rapidly as I increases but after reaching a maximum, the throughput starts to decline gradually. We can only offer a heuristic explanation of this phenomenon. As the optimisation algorithm progresses, the weights of the most congested edges will come to completely dominate the search for the least cost route from the source to the destination. With a large enough iterations count, the algorithm only seeks to balance the flow on those edges without any regard for the traffic situation in the rest of the network. We also ran the tests for other values of ϵ, but omitted some from the figure for clarity; for $\epsilon \leq 0.05$, the peak performance had not yet been reached for $I = 160$.

However, in our experiments we ran considerably fewer iterations than recommended by (4). For small values of ϵ, in the first iterations the weight of each edge remains at approximately 1, and thus the paths found by BMSR will be essentially fewest hop paths. It seems plausible that instead of only optimising the hop count, or only balancing the flow along the most congested edges, good results could be obtained by taking both factors into consideration – and we hypothesise that this is what happens when the number of iterations I is less than recommended by (4).

The results shown in Fig. 2 indicate that there is a qualitative difference in the performance of BMSR and DSR for different packet sizes. Figure 5 shows throughput and packet delay for various packet sizes. The throughput performance of DSR seems to increase until a critical packet size, after which increasing the packet size further decreases DSR's performance. We assume this to be caused by the interdependence of the two main reasons of packet loss: collisions of CBR packets due to interference and IFQ overflows.

BMSR aims at decreasing link congestion; it reduces the number of IFQ overflows, as shown in Fig. 2. We conclude that increasing the packet size reduces the negative effect of collisions on throughput for BMSR: increasing packet size

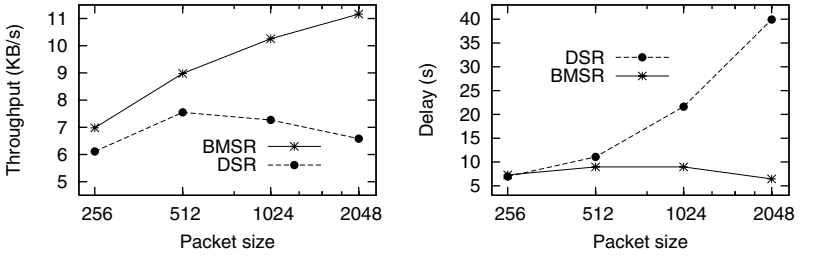

Fig. 5. On the left, throughput in KB/s versus packet size, and on the right, delay in seconds versus packet size, for parameter values $I = 160$ and $\epsilon = 0.05$. All values are averages over at least 15 repetitions. Note the logarithmic scale for the packet size and that the CBR rate is 160 Kbit/s for all runs.

for a constant CBR rate reduces the number of packets and therefore the total MAC layer overhead. However, the time frame required for the transmission of a single packet increases correspondingly and retransmissions become more costly. Still the effect of losing larger packets due to IFQ overflows seems to outweigh the impact of collisions.

As Fig. 5 indicates, DSR packet delay grows nearly linearly with packet size, whereas BMSR shows a clearly lower, approximately constant delay. This is most likely due to the fact that after the initial setup phase BMSR uses a static routing scheme.

4 Conclusions and Future Work

We studied the application of a linear programming approximation algorithm to distributively optimise network bandwidth in a wireless multi-hop network. The algorithm aims at minimising the maximum flow over any edge in the input graph. We integrated it into the DSR route-discovery process in a distributive manner and obtained significant increase in throughput for the studied topology. The topology considered was static and regular. As future work we are interested to consider more general network topologies, non-uniform spatial node distributions, and to incorporate mobility.

We believe that optimising link congestion proves successful also for other topologies with a uniform distribution of nodes and a relatively regular graph structure. For non-uniform topologies we expect the optimisation for node-based metrics to work better. We plan to study the effect of node-based metrics on the balancing algorithm, such as optimising for node congestion. We would expect edge congestion to serve well in uniform topologies, but a node-based approach to give better results in non-uniform topologies, where the load on single hubs may get heavy due to a high number of neighbouring nodes.

The static-network and the uniform-node-distribution assumptions are essential in the current formulation of the algorithm. Besides considering node-based optimisation metrics, we want to consider a steady-state formulation of the algorithm, e.g. by enabling a calculation of the edge weights depending on the present edge flow. Further applications of the BMSR protocol, such as energy-efficient routing, are to be considered as well.

The results presented in this paper show the potential of using mathematically justified distributed optimisation techniques for ad hoc networks. By utilising shortest-path computations integrated into the DSR route discovery, we obtain an improvement in throughput of 14% to 69% compared to DSR for a network with high load. The assumption of a static network with a uniform spatial distribution of nodes does not seem too restrictive. We are convinced that it can serve as a starting point for further investigating the potential of distributed optimisation for ad hoc networks.

References

1. Perkins, C., ed.: Ad Hoc Networking. Addison Wesley, Reading, MA, USA (2001)
2. Meguerdichian, S., Koushanfar, F., Potkonjak, M., Srivastava, M.: Coverage problems in wireless ad-hoc sensor networks. In: Proc. 20th Annual Joint Conf. of the IEEE Computer and Communications Societies. (2001) 1380–1387
3. Floréen, P., Kaski, P., Kohonen, J., Orponen, P.: Lifetime maximization for multicasting in energy-constrained wireless networks. IEEE J. Selected Areas in Communications **23**(1) (2005) 117–126
4. Aggelou, G., Tafazolli, R.: RDMAR: A bandwidth-efficient routing protocol for mobile ad hoc networks. In: Proc. 2nd ACM Int'l Workshop on Wireless Mobile Multimedia. (1999) 26–33
5. Kawadia, V., Kumar, P.: Power control and clustering in ad hoc networks. In: Proc. 22nd Annual Joint Conf. of the IEEE Computer and Communications Societies. (2003)
6. Basu, A., Lin, A., Ramanathan, S.: Routing using potentials: A dynamic traffic-aware routing algorithm. In: Proc. Conf. Applications, Technologies, Architectures, and Protocols for Computer Communication, (2003) 37–48
7. Johnson, D., Maltz, D., Hu, Y.: The dynamic source routing protocol for mobile ad hoc networks (DSR). Tech. report, IETF (2003) IETF Draft, July 2004.
8. Nasipuri, A., Castañeda, R., Das, S.: Performance of multipath routing for on-demand protocols in mobile ad hoc networks. Mobile Networks and Applications **6**(4) (2001) 339–349
9. Wu, K., Harms, J.: Performance study of a multipath routing method for wireless mobile ad hoc networks. In: Proc. 9th Int'l Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Washington, DC, USA, IEEE Computer Society (2001) 99–107
10. Ganjali, Y., Keshavarzian, A.: Load balancing in ad hoc networks: Single-path routing vs. multi-path routing. In: Proc. 23rd Annual Joint Conf. of the IEEE Computer and Communications Societies. (2004)
11. Vutukury, S., Garcia-Luna-Aceves, J.: A simple approximation to minimum-delay routing. In: Proc. Conf. Applications, Technologies, Architectures, and Protocols for Computer Communication, New York, ACM Press (1999) 227–238
12. 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, Norwell, MA, USA (2002)
13. 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/`.
14. Hou, X., Tipper, D.: Impact of failures on routing in mobile ad hoc networks using DSR. In: Proc. Communication Networks and Distributed Systems Modeling and Simulation Conf. (2003)