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

Load balancing by distributed optimisation in ad hoc networks

Published in:
Mobile Ad-hoc and Sensor Networks Second International Conference, MSN 2006 Hong Kong, China, December 13-15, 2006

DOI:
10.1007/11943952_73

Published: 01/01/2006

Document Version
Early version, also known as pre-print

Please cite the original version:
Schumacher, A., Haanpää, H., Schaeffer, S. E., & Orponen, P. (2006). Load balancing by distributed optimisation in ad hoc networks. In I. S. Jiannong Cao (Ed.), *Mobile Ad-hoc and Sensor Networks Second International Conference, MSN 2006 Hong Kong, China, December 13-15, 2006* (pp. 873-884). https://doi.org/10.1007/11943952_73

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 multihop networks by distributed optimisation. As an example of a distributed optimisation algorithm for ad hoc networks, we use an approximation algorithm for minimising the maximum network congestion and implement it as a modification of the DSR routing protocol. The algorithm is based on shortest-path computations that are integrated into the DSR route discovery and maintenance process. Therefore, it does not rely on the dissemination of global information within the entire network. The simulation results obtained by 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]. As the size and cost of such devices is no longer prohibitive, the advantages of ad hoc networks have attracted much interest. Perhaps the most significant advantage of ad hoc networks – and simultaneously an important design goal – is their easy 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 easy setup would be most advantageous in a variety of applications ranging from military operations and disaster relief to commercial applications.

Ad hoc networks also present challenges. For easy deployment, the nodes should not depend on an external energy supply, so they are usually battery-powered, and battery life is often a limiting factor. The radio transmission channel is limited in bandwidth and must typically be shared between nearby nodes. Determining and maintaining the network topology in a distributed fashion is a most challenging problem, particularly if the network topology can change during operation, for example due to adding or removing nodes or node mobility.

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 an algorithm 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 approximate solutions for larger instances (provided that there exists such a formulation). Such methods have been applied to optimising sensor node coverage [2] and maximising the lifetime of energy-constrained networks [3], but these approaches typically require that information about the state of the network be collected to a central location, where the optimisation is then carried out, which adds undesired hierarchy and a point of failure to the network.

Second, an algorithm should be distributed and non-hierarchical. Nodes should cooperate in computing the optimum so that each node follows a simple set of rules, and neither the size of the messages nor their number should grow too quickly as the size of the network increases. Such approaches have been used for bandwidth optimisation [4] and determining the location of the nodes based on the estimated location of their neighbours [5]. Certain energy-aware modifications of routing protocols such as AODV [6, 7] or DSR [8] also fall into this category. However, such heuristic optimisation methods are difficult to analyse mathematically, and often the only analysis is based on simulation.

There are distributed algorithms that are mathematically justifiable. Typically the nodes compute some graph-based properties, such as shortest paths or spanning trees, in a distributed and iterative manner. This enables a theoretic discussion about the expected quality of the solution and an estimate on the convergence of the algorithm towards the optimum. Such methods have been applied to adjusting transmission power levels based on lowest-cost energy paths [9] and routing around congested nodes based on node potentials and the steepest gradient method [10].

In this paper, we present a distributed approximation algorithm for load balancing in an ad hoc network. Load balancing can be advantageous for increasing reliability and network throughput. Recent proposals also apply load balancing to network life-time maximisation [11]. Our approach relies on modifications of the Dynamic Source Routing (DSR) [12] protocol. We extend DSR to use multiple source-destination paths, typically referred to as *multipath routing*, to balance data traffic. Our simulations show that it is possible to achieve a gain of 14% to 69% in the throughput, depending on the setup, compared to DSR by balancing the traffic over the nodes.

The rest of the paper is organised as follows. In the next section, we give an overview of the basic operation of DSR and describe how our approach relates to optimisation algorithms relying on multipath routing that have been proposed recently. In Section 3, the approximation algorithm is discussed and its implementation as a modification of DSR is presented. Section 4 includes the results obtained by network simulations using the `ns2` [13] network simulator. Finally, Section 5 presents our conclusions and outlines future research directions.

2 Overview of DSR and Multipath Extensions

DSR [12] is an on-demand *source routing* protocol, so 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 by packets forwarded through neighbouring nodes. Source routing comes along with benefits, such as loop freedom or the possibility for route shortening, as source nodes have knowledge about the entire route.

2.1 DSR Operations

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. For each route discovery, any 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 minimal or close to minimal in terms of the hop count. Therefore, DSR chooses routes not significantly longer than the shortest route between source and destination. Although in principle multiple routes to the same destination might be contained in the route cache, e.g. by overhearing other routes, the nodes select always 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 that are initiated only 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. DSR also contains an optional *flow-state extension*, which reduces the route overhead by omitting source route information for packet flows that are considered to be static, assuming intermediate nodes keep track of the route in use.

2.2 Multipath-based Network Optimisation

Multipath extensions to DSR for load balancing have been previously studied: Nasipuri, Castañeda, and Das [14] introduce alternate routes to the route discovery, whereas Wu and Harms [15] 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. Multipath routing helps in increasing reliability and throughput as well as load balancing and energy conservation in ad hoc networks, although the choice of the path metric is crucial [16].

Ganjali and Keshavarzian [17] state that multipath routing alone can not improve load balancing, as with an increasing density of nodes, the choice of shortest paths connecting any pair of nodes leads to congestion in the centre of the network. They conclude that an additional incentive has to be given to push traffic away from centre nodes to avoid congestion.

Multipath-based network optimisation has been studied extensively for wired networks. Vutukury and Garcia-Luna-Aceves [18] for example propose an algorithm to minimise delay based on the heuristic redirection of flow over multiple paths. Basu, Lin and Ramanathan [10] present a potential-based routing methodology that forwards packets using steepest gradient search. They also propose a traffic-aware routing algorithm that uses queue lengths to determine congestion. Both approaches rely on a link-state routing algorithm for the dissemination of link information throughout the entire network. Su and de Veciana [19] study dynamic multi-path routing to minimise network congestion.

However, most proposals are not directly applicable to ad hoc networks because of the limitations described above. In the work presented here, we obtain multiple source-destination routes by a linear programming approximation algorithm that minimises flow congestion [20]. 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 DSR routing protocol. The actual data flow is uniformly distributed over these precomputed routes.

3 Distributed Load Balancing

In this section we describe a modification to DSR for load balancing by multipath routing. We model choosing a set of source routes as a min-max congestion multicommodity flow problem and describe how an approximation algorithm can be used for solving the problem in the context of an ad hoc network.

As explained in Section 2, DSR normally uses one route from the source node to the destination node. However, extending DSR to use more routes is relatively easy and has potential benefits in increased reliability, throughput and load balancing.

3.1 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 i to j.

For every commodity c in a multicommodity flow problem, there is an associated supply or demand $t^c(i)$ at each node $i \in V$; these must satisfy $\sum_{i \in V} t^c(i) = 0$ for all c. The task is to find a set of flows $x_{ij}^c \geq 0$ over the edges $(i, j) \in E$ such that each commodity is routed from the supply nodes (with $t^c(i) > 0$) to the demand nodes (with $t^c(i) < 0$), that is,

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

Typically the total flow on an edge (i, j) is limited by the capacity u_{ij} of the edge: $f_{ij} = \sum_c x_{ij}^c \leq u_{ij}$.

In applying the multicommodity flow model to routing, each commodity c represents one data stream of traffic volume v^c from the source s^c to the destination d^c. In this case, $v^c = t^c(s^c) = -t^c(d^c)$ and $t^c(i) = 0$ for all other nodes i. Within these constraints we choose x_{ij}^c to *minimise the maximum congestion*:

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

Many algorithms exist for solving linear optimisation problems such as ours in the case when the whole state of the network is known. In contrast, here we desire an optimisation algorithm where the individual nodes cooperate to determine the optimum while only passing a reasonable number of messages of reasonable size. The approximation algorithm [20] in Fig. 1 has these properties. In this formulation it is assumed that 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

$$I \geq \lceil 4m \log m / \epsilon^2 \rceil \tag{3}$$

iterations, where m is the number of edges.

3.2 Integration in DSR

Our shortest-path methodology enables a simple integration into the DSR routing protocol; the computation of shortest paths is similar to that of the original protocol. Our approach differs from standard DSR in that DSR initiates route discovery when necessary, while our approach uses an initial setup phase to proceed through the iterations of the balancing algorithm. Each source obtains one *balanced route* to the destination for each iteration of the balancing algorithm. Some routes may occur more than once. After the setup phase, for every packet to be sent the source will choose one of the routes in its cache at random. Unlike in DSR, the routes are not removed from the cache even in the presence of link failures, as in this setup these can be only caused by temporarily congested links.

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

$$w_{ij} := \left(1 + \epsilon \sum_c y_{ij}^c\right) w_{ij} \quad . \tag{4}$$

3. Scale the total flow by letting $x := x/I$.

Fig. 1: Approximation algorithm for min-max congestion multi-commodity flow [20] that 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.

We implement this algorithm by modifying DSR's route discovery and route maintenance operations. For this purpose, the two basic 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 in Fig. 1. For clarification, we refer to these modified messages by `BREQ` and `BREP`.

Route Discovery Instead of computing shortest routes on the basis of 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 that lie on the route.

Each node keeps track of the weight of its incoming links and the flow on them. Incoming links are those links, which a node may use to receive a `BREQ` message from one of its neighbours. `BREQ` messages carry, in addition to the list of addresses of nodes that re-broadcasted the message, the accumulated route cost from the source to the current intermediate node. 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 second `BREQ` has a lower cost route from the source to the intermediate node than the previous one, the intermediate node re-broadcasts it.

When the destination receives a `BREQ` packet, it must wait a short while for possible other `BREQ` packets with lower cost. 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 pa-

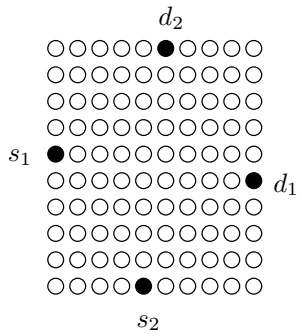

Fig. 2: The simulation setup: two source-destination pairs (s_1, d_1) and (s_2, d_2) are placed "off-by-one" on the opposite sides of the grid. The source nodes s_1 and s_2 send data packets at constant bit rate to their respective destination nodes d_1 and d_2. Instead of utilising only the direct shortest path, we occupy the longer paths to balance the load. Each node may communicate with the nodes beside, above or below it.

rameter 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 for a longer data stream.

Route Maintenance 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.

4 Experiments

The considered scenario is a stationary grid network with source and destination pairs. The chosen traffic pattern resembles an *emergency relief scenario*, where a large amount of constant bit rate (`CBR`) data is to be transfered through an already congested network. When a sudden demand arises to transmitting a large amount of data between a dedicated pair of nodes, e.g. between the control centre and rescue teams, one aims to deliver as much of the critical data as possible. The means to achieve this include balancing the traffic among the nodes and utilising the network capacity to maximise throughput over source-destination pairs.

We compare our algorithm to DSR by using `ns2` to simulate it on a 10 by 10 grid with two `CBR` flows, from s_1 to d_1 and from s_2 to d_2; see Fig. 2 for the network setup. Both `CBR` sources are transmitting with a previously determined rate and packet size. See Tab. 1 for the particular parameter values.

Prior to initiating the `CBR` traffic, we run the balancing algorithm of Fig. 1 for a chosen value of ϵ and a chosen number of rounds I to select routes that give an

Table 1: The parameters used in ns2 simulations.

Parameter	Values used	Parameter	Values used
CBR packet size	256 B, 512 B, 1024 B, 2048 B	MAC bandwidth	1 Mbit
CBR data rate	160 Kbit/s	MAC protocol	802.11 (with RTS/CTS handshake)
Antenna type	OmniAntenna	Propagation model	TwoRayGround
Max. source route length	22	Max. IFQ length	50
Network size	2400 m × 2400 m	Node count	100
Simulation time	1500 s	Balancing setup	500 s

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. We use the same source-destination setup to transmit data using the DSR implementation provided in the ns2 standard distribution.

In addition to throughput, we are interested in the extent the balancing algorithm yields a more uniform selection of routes over the nodes. Therefore, we study the number of forwarded CBR packets to measure the load distribution over the nodes. We expect most packets to be forwarded by nodes located within 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 burdened with forwarding a significantly larger number of packets than nodes lying on slightly longer paths.

A more balanced selection of nodes should also have a positive effect on the number and the occurrence of 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 [21] claim 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 network throughput. Although we do not expect the number of collisions to be significantly lower compared to the standard DSR route selection, we would expect a more even distribution over the nodes, preventing the formation of bottlenecks. Figure 3 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 have been excluded from the figure for the sake of 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.

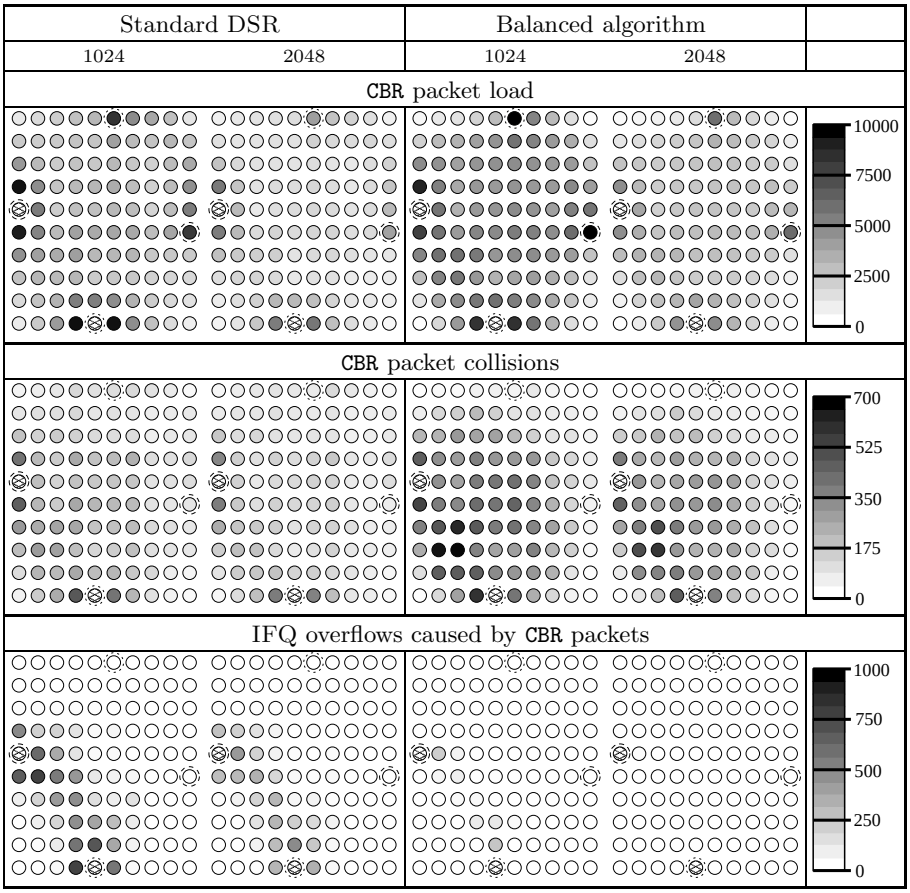

Fig. 3: Performance measures of the standard DSR and the balancing routing algorithm for $I = 160$ iterations and $\epsilon = 0.05$ for `CBR` packet sizes 1024 B and 2048 B. The colourings are averages over five `ns2` runs, but variations from one run to another were observed to be negligible. Sources and destination nodes are indicated by dashed circles.

One might expect the route selection of DSR to favour shorter routes, yielding an increased network load within the centre of the grid that results in inter-

ference and a low network throughput. The balancing algorithm should recognise areas of larger congestion and after initially selecting shorter routes, select routes that avoid the potentially congested areas.

In Fig. 3, we only observe minor differences for the balanced algorithm and standard DSR. Depending on the averaging of packet load over the rather long simulation run, the load for standard 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 standard DSR, but the overall network load for the balanced algorithm is higher than for standard DSR, which can be explained by the higher throughput, discussed later in this section.

Due to higher load, the balanced route selection method encounters more collisions compared to standard DSR. A remarkable effect is the concentration in the quarter 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 the balanced algorithm 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 always 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. Standard 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 standard DSR and the balancing algorithm: the single-path routing of standard DSR leads to the formation of bottleneck nodes due to congestion in the bottom left quarter 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. The balancing algorithm shows hardly any IFQ overflows at all, except within the vicinity of the sources.

Figure 4 shows the average throughput over both source-destination pairs over time. Comparing throughput for standard DSR and the balancing algorithm, one observes larger fluctuations for standard DSR. The stability of the throughput for the balancing algorithm mainly results from the fact 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 standard DSR, we consider the throughput over 1000 s from the time when `CBR` transmissions have been initiated to compare both algorithm in the following. For both packet-sizes the balancing algorithm clearly outperforms standard DSR.

We also studied the effect of the I and ϵ parameters on the performance of the algorithm. The results are summarised in Fig. 5 and are mostly as expected;

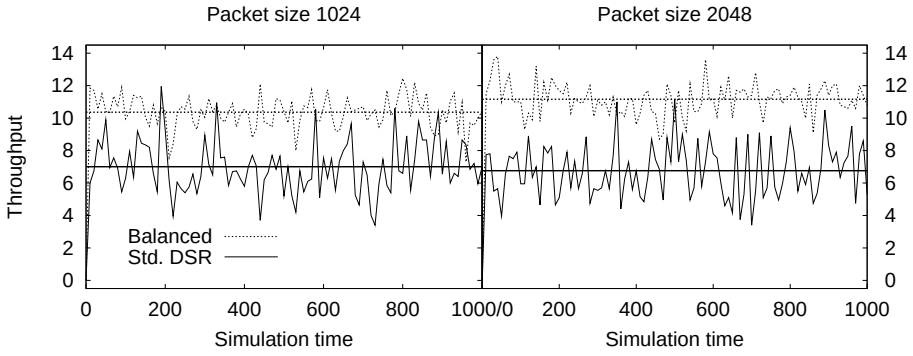

Fig. 4: Average throughput of both source-destination pairs in KB/s versus simulation time for a single run of DSR and the balancing algorithm. Note that the setup stage for the balancing algorithm is omitted from the plot. The parameter values for the balancing algorithm were $I = 160$ and $\epsilon = 0.05$. Average throughput over the entire simulation run is shown for each plot.

already for a modest number of iterations we obtain throughput that is clearly superior to DSR. There is a dependency of the throughput on ϵ and I: for a larger value of ϵ, fewer iterations are necessary to obtain a fairly good throughput, but running a large number of iterations with a small ϵ yields, in the end, 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 number of iterations, 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 all $\epsilon \leq 0.05$, the peak performance had not yet been reached for $I = 160$.

However, in our experiments we ran considerably fewer iterations than recommended by Eq. 3. For small values of ϵ, in the first iterations the weight of each edge remains at approximately 1, and thus the paths found by the algorithm 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 Eq. 3.

The results shown in Fig. 3 indicate that there is a qualitative difference in the performance of standard DSR and the balancing algorithm for different packet sizes. Figure 6 shows throughput and packet delay for various packet sizes. It seems that the performance of standard DSR increases up to a point

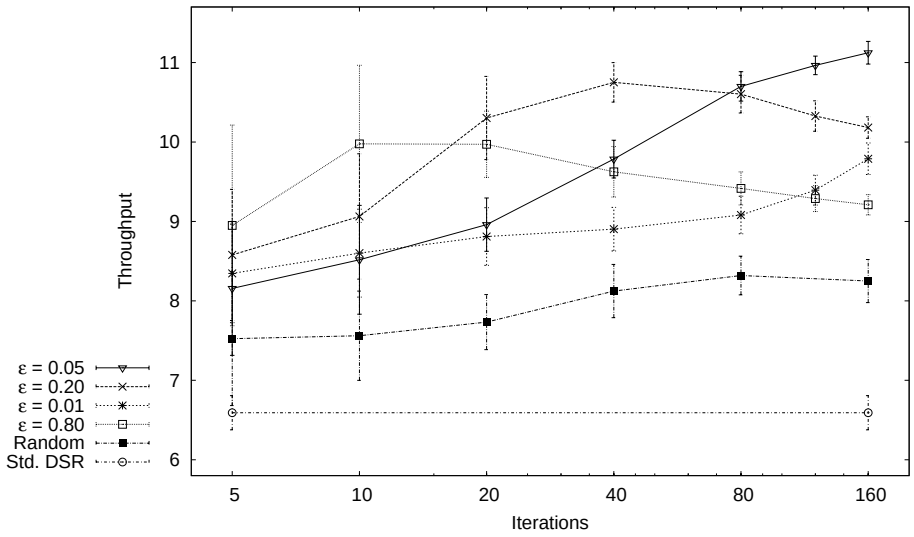

Fig. 5: Average throughput in KB/s over both source-destination pairs and 1000 s as a function of I and ϵ for `CBR` packet size 2048: The plot also contains values for standard DSR and random route selections of varying size. These are formed by selecting I routes on the basis of a random walk from the sources to the destinations. All values are averages over at least 15 repetitions, standard deviations are shown with error bars. The legend ordering corresponds to the throughput value at $I = 160$.

where the packet size has reached a critical value. Please note that the `CBR` rate is 160 Kbit/s for all runs. Increasing the packet size even further has a negative effect on DSR's performance. We assume that this response is caused by the interdependence of the two main reasons of packet loss: collisions of `CBR` packets due to interference and IFQ overflows.

The balancing algorithm aims at decreasing link congestion, which successfully reduces the number of IFQ overflows, as indicated in Fig. 3. We further assume that increasing the packet size reduces the negative effect of collisions on throughput for the balancing algorithm: Increasing packet size for a constant `CBR` rate reduces the number of packets and therefore also 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.

`CBR` packet delay rises almost linearly when increasing the packet size. Note that Fig. 6 uses a logarithmic scale for the packet size. The balancing algorithm shows a significantly lower delay than standard DSR, which even decreases for 2048 bytes. We studied the effect of disregarding all route errors on `CBR` packet delay by letting the algorithm always choose the same of all possible shortest routes for both source-destination pairs. The results supported our assumption

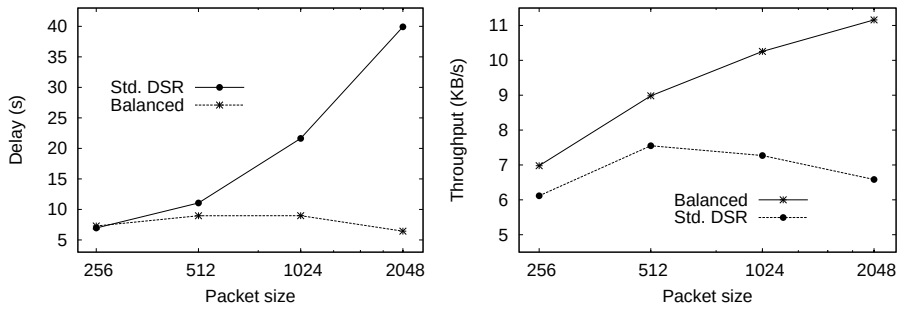

Fig. 6: On the left, delay in seconds versus packet size, and on the right, throughput in KB/s versus packet size. In both plots, the balanced algorithm had parameter values $I = 160$ and $\epsilon = 0.05$. All values are averages over at least 15 repetitions.

that the decrease in packet delay mainly results from this property of the balancing algorithm.

5 Conclusions and Future Work

In this paper, we have 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 graph given as problem instance. Even though the original algorithm does not necessarily provide for a distributed implementation, we have integrated it into the DSR route-discovery process and obtain a significant increase in network throughput for the studied topology.

So far we have applied the algorithm to a rather static and regular network topology, i.e. ignoring mobility and a non-uniform spatial distribution of nodes within the network. As future work we are interested in considering more general network topologies. 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 be more applicable. Therefore, we are especially interested in studying the effect of node-based metrics on the balancing algorithm, such as optimising for node instead of link congestion.

One example are networks with scale-free topologies [22] in terms of their underlying graph model. The corresponding network topology would contain some nodes having far more links than most other nodes. We would expect edge congestion to serve well in uniform topologies, but a node-based approach to give better results in scale-free topologies, where the load on single hubs easily becomes very 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, i.e. by enabling a calculation of the edge weights depending on the present edge flow. Further applications of the balancing algorithm, 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.

Acknowledgements

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

References

1. Perkins, C.E., ed.: Ad Hoc Networking. Addison Wesley, Reading, MA, USA (2001)
2. Meguerdichian, S., Koushanfar, F., Potkonjak, M., Srivastava, M.B.: Coverage problems in wireless ad-hoc sensor networks. In: Proc. of 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM). (2001) 1380–1387
3. Floréen, P., Kaski, P., Kohonen, J., Orponen, P.: Lifetime maximization for multicasting in energy-constrained wireless networks. IEEE Journal on 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. of 2nd ACM International Workshop on Wireless Mobile Multimedia. (1999) 26–33
5. Meguerdichian, S., Slijepcevic, S., Karayan, V., Potkonjak, M.: Localized algorithms in wireless ad-hoc networks: Location discovery and sensor exposure. In: Proc. of 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc), New York, NY, USA, ACM Press (2001) 106–116
6. Senouci, S.M., Naimi, M.: New routing for balanced energy consumption in mobile ad hoc networks. In: Proc. of 2nd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, New York, NY, USA, ACM Press (2005) 238–241
7. Park, S., Shin, J., Baek, S., Kim, S.C.: AODV-based routing protocol considering energy and traffic in ad hoc networks. In: Proc. of International Conference on Wireless Networks. (2003) 356–361
8. Chen, W.T., Pan, M.S.: DSR-based energy efficient routing protocol in mobile ad hoc networks with transmit power adjustment. In: Proc. of International Conference on Wireless Networks. (2003) 350–355

9. Kawadia, V., Kumar, P.: Power control and clustering in ad hoc networks. In: Proc. 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM). (2003)
10. Basu, A., Lin, A., Ramanathan, S.: Routing using potentials: A dynamic traffic-aware routing algorithm. In: Proc. of the ACM SIGCOMM 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, New York, NY, USA, ACM Press (2003) 37–48
11. Cho, H.K., Kim, E.S., Kang, D.W.: A load-balancing routing considering power conservation in wireless ad hoc networks. In: Proc. of 16th International Workshop on Database and Expert Systems Applications. (2005) 128–132
12. Johnson, D.B., Maltz, D.A., Hu, Y.C.: The dynamic source routing protocol for mobile ad hoc networks (DSR). Tech. report, IETF (2003) IETF Draft, July 2004, work in progress.
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. Nasipuri, A., Castañeda, R.C., 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
15. 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
16. 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
17. Ganjali, Y., Keshavarzian, A.: Load balancing in ad hoc networks: Single-path routing vs. multi-path routing. In: Proc. of 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM). (2004)
18. Vutukury, S., Garcia-Luna-Aceves, J.J.: A simple approximation to minimum-delay routing. In: Proc. of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, New York, NY, USA, ACM Press (1999) 227–238
19. Su, X., de Veciana, G.: Dynamic multi-path routing: asymptotic approximation and simulations. In: Proc. of the 2001 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, New York, NY, USA, ACM Press (2001) 25–36
20. 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)
21. Hou, X., Tipper, D.: Impact of failures on routing in mobile ad hoc networks using DSR. In: Proc. of Communication Networks and Distributed Systems Modeling and Simulation Conference. (2003)
22. Barabási, A.L., Albert, R., Jeong, H.: Mean-field theory for scale-free random networks. Physica A **272** (1999) 173–187