

Load Balancing in Grid Networks

Jacob Honoré Broberg, Kartheepan Balachandran, Søren Hede, Jesper Pedersen, M. Tahir Riaz, Jens Myrup Pedersen
Aalborg University, Department of Electronic Systems, DK-9220, Aalborg East, Denmark
{jbroberg, bkarthe, sthede, jellped, tahir, jens}@es.aau.dk

Abstract—A load balancing algorithm functioning as an extension to the topological routing scheme is proposed. The algorithm is designed to evenly distribute the traffic load in a 4-regular grid network. This is done by letting each node calculate how loaded it is with regular intervals and feed this information to its neighboring nodes, which then use it to derive link probabilities applied in probabilistic routing. The traffic load is defined as the average link utilization of each link and the average expedition delay for each output queue, i.e. the waiting time of packets in queues. The algorithm has been tested on the i3 Demonstrator, a 6x6 grid network testbed, using different traffic patterns. Results show that better performance is achieved when a high level of traffic including hotspot traffic is applied. Furthermore, packet drops and lengthy packet delays are avoided. With low traffic levels or with a uniform-like P2P traffic pattern, the performance is neither improved nor degraded noteworthy.

Index Terms—Networks, internet, routing, load balanced routing, network testing, topology, traffic control (communication), feedback communication, probabilistic routing

I. INTRODUCTION

Internet traffic is increasing in both size and complexity. Furthermore, the demands for QoS (Quality of Service) are also increasing in many kinds of internet traffic. To fulfill these QoS demands without a massive routing overhead it has been proposed to use regular graph structures as network topologies because of their good global structural properties [1]. Using this approach routing in a network with e.g. a 4-regular grid topology can be carried out using topological routing, which is a table-free routing scheme based only on the knowledge of the overall structure and addressing scheme [2][3]. Networks with this topology are not yet installed in real-world networks, but are very popular study subjects and are used in parallel computing systems [4]. Furthermore, internet backbones are often connected as fiber rings, which can be modeled as other topologies on the optical layer with different wavelengths acting as logical links.

This paper presents a load balancing algorithm with the objective to dynamically control the distribution of load in the entire network, with a low overhead and without causing the load to oscillate between network nodes. The algorithm presented here is an extension to the existing topological routing scheme. One of the primary motivations to perform load balancing, besides supplying good QoS, is to utilize the resources and capacity of the network to the full extend.

Because of the dynamic behavior of internet traffic an automatic load balancing routing algorithm would diminish the need for manual ongoing adjustment of the network traffic.

Load balancing is often discussed in connection with source routing either when using server clusters or in multipath networks [5]. Such load balancing schemes are based on making *decisions* using *global* knowledge and routing packets (*migrating*) through a *global* route, using the terminology in [6]. Furthermore, load balancing schemes can be divided into those who set fixed deterministic paths and the adaptive/dynamic schemes [7]. The contribution of this paper is to present an adaptive scheme, using only local knowledge to decide where to send packets and using hop-by-hop routing (local migration). In conjunction to topological routing the algorithm uses probabilistic routing, which is similar to that proposed in other load balancing schemes [8]. The current proposal is partly based on an earlier study concerning load balancing using topological routing [9]. A detailed description of the proposal is given in [10].

In Section II the performance metrics used in the design and evaluation are described together with the concept of topological routing and the testbed used for testing. In Section III the outline of the proposed load balancing solution is given, including the notion of *load factors* and the use of these. In Section IV the traffic patterns used for testing are described along with the performance results. Finally, the paper is concluded in Section V.

II. METHODS

In the following the metrics used in the problem formulation are outlined along with the evaluation parameters. A description of topological routing and the i3 Demonstrator testbed is also given.

A. Performance Metrics

To evaluate the performance of the proposed load balancing algorithm metrics have been defined. These metrics are also used in the design as measures for adaptively estimating how loaded the nodes and links in the network are, in order to affect the routing decisions. The metrics are measures with respect to a specific output link X and the unit of the metric is shown in brackets: $[z]$.

1) *Link Utilization* (lu_X) [%]: The relationship between the link throughput and the link capacity for the output link X , both measured in bits/s. The average link utilization can be seen as how much the link is used over time on average. The link utilization for each of the two directions of a full-duplex link has to be described.

2) *Expedition Delay* (ed_X) [s]: The time interval between a packet is received on any input link on a node until the packet is routed and forwarded on a specific output link X . The expedition delay is measured according to which output link the packet is forwarded to. The expedition delay covers the time a packet spends both in the input queue and output queue and the processing time of the routing algorithm.

B. Problem Formulation

The objective of the load balancing algorithm is to evenly distribute traffic load in a 4-regular grid network, i.e. the difference between the average link utilization of any link in the network and that of any other link should be minimized, and the difference between the average expedition delays for all output links on any node in the network and that of any other node should be minimized. Furthermore, the variance of the link utilization and expedition delay over time of any link in the network should be minimized. This is the definition of evenly distributed traffic load used here.

C. Evaluation Parameters

To evaluate the performance of the algorithm the following parameters are used. For each link in the network, denoted i, j, X using the notation explained in Section III-A, the mean and variance over time of both the link utilization and expedition delay are derived. These results are graphically plotted as columns for each link as shown in Section IV-B. Furthermore, for both link utilization and expedition delay scalar values are derived for the entire network. For instance, the variance of the mean link utilization: $\text{var}(\text{mean}(lu_{i,j,X}) [])$ describes how much the average utilization varies when comparing all links. Likewise, the variance of the mean expedition delay: $\text{var}(\text{mean}(ed_{i,j,X}) [])$, the average link utilization variance: $\text{mean}(\text{var}(lu_{i,j,X}) [])$, the average expedition delay variance: $\text{mean}(\text{var}(ed_{i,j,X}) [])$, and the overall average expedition delay: $\text{mean}(\text{mean}(lu_{i,j,X}) [])$ are also analyzed.

The average packet delay distribution (i.e. average delay when sending from any node to any other node) and jitter distribution are also analyzed, based on logged packet timestamps set by the source and destination nodes, together with the amount of dropped packets pr. node.

D. Topological Routing

In topological routing network traffic is routed based on knowledge about the network topology. Several regular

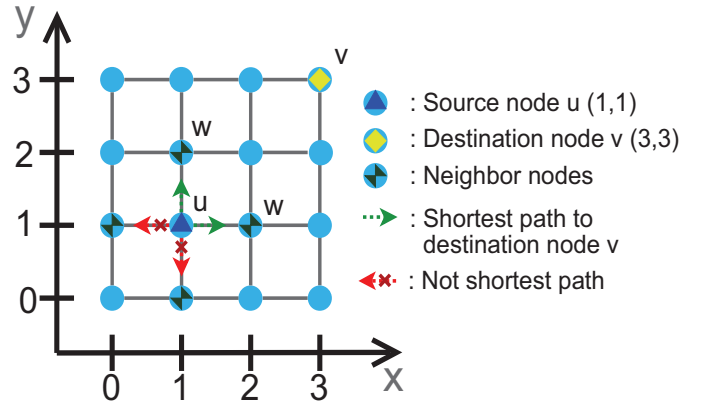


Figure 1. Source node u can only forward a packet to neighbor nodes w in order to reach destination node v and obey the shortest path constraint.

topologies can be used but here the 4-regular grid structure, also known as 2D mesh structure, is used. This structure has been proposed as a topology for large-scale networks in [2] [3]. Note that it is not fully regular, since the edge and corner nodes have degrees 2-3. With this structure the addressing scheme can be based on Cartesian coordinates. Using topological routing a packet routed from source node u to destination node v will use the shortest path through the network [2].

In Figure 1 the routing scheme principle is shown. The packet is routed on a hop-by-hop basis. The $d_{u,v}$ denote the distance between source node u and destination node v and is calculated using (1).

$$d((x_u, y_u), (x_v, y_v)) = |x_u - x_v| + |y_u - y_v| \quad (1)$$

When node u sends a packet to node v , it can only forward the packet to a neighboring node w where $d_{w,v} < d_{u,v}$, i.e. a path which will decrease the distance to node v , as shown in Figure 1, thereby obeying the shortest path constraint. A node routing a packet will therefore never have more than two possible directions, to which the packet can be forwarded. In the basic topological routing scheme using no load balancing, packets are always routed along the y axis if possible, otherwise along the x -axis. If load balancing is used, a routing choice is made adaptively.

E. i3 Demonstrator

The Intelligent ICT Infrastructure (i3) Demonstrator is a testbed, which has been constructed at Aalborg University to work with a prototype of a 4-regular grid network. The i3 Demonstrator has been used in various research projects, including a project on load balancing in 2006 [9] and has been used to test the algorithm proposed here as well. A detailed description of the i3 Demonstrator implementation and setup has been given in [11]. The testbed consists of 36 node computers (ordinary PCs) peer-to-peer connected

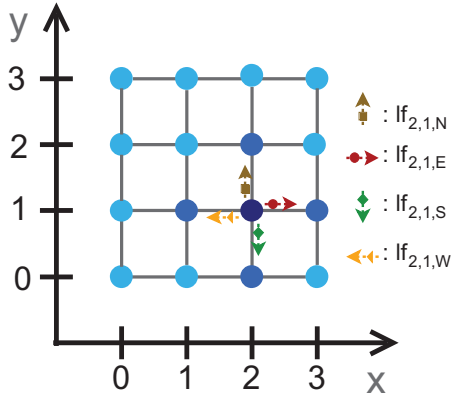


Figure 2. The 4 link load factors calculated by node (2, 1).

using Ethernet. They are connected in a 6x6 grid according to the 4-regular grid topology. Because the routing algorithm is implemented in user space, testing is carried out in a virtual network layer on top of UDP/IP sockets. All the nodes communicate with a master computer through TCP/IP sockets using a gigabit Ethernet sideband network. Traffic send into the network is controlled by the master. The time on all the node computers is synchronized using NTP.

III. SOLUTION

The proposed load balancing algorithm is partly based on the principle of feedback control. Each node can thereby be modeled as a system and the output is how loaded the node and its output links are. Each of the neighboring nodes act as a controller, since they control how much traffic is send to the node, which is directly proportional to the load. Feedback control is therefore achieved by letting the node feed back measurements of how loaded it is - *load factors* - to its neighboring nodes, which then use the received information to decide where to route traffic. In the following the proposed solution is outlined.

A. Calculation of Load Factors

A load factor represents the current load of an output link, which is updated with a certain interval. A node will update up to four load factors at each update interval, as shown in Figure 2. Each load factor represents the load experienced when a packet is received and send out on a specific output link. The load factors are named as follows: $lf_{i,j,N}$ where i, j are the node coordinates and N means that this load factor describes the load regarding the output link upwards (North), using the naming scheme of a compass.

A load factor is a scalar value based on aggregated measured metric values. Between every update the node measures the size and expedition delay of packets forwarded on each of its output links. Based on the measurements from the preceding

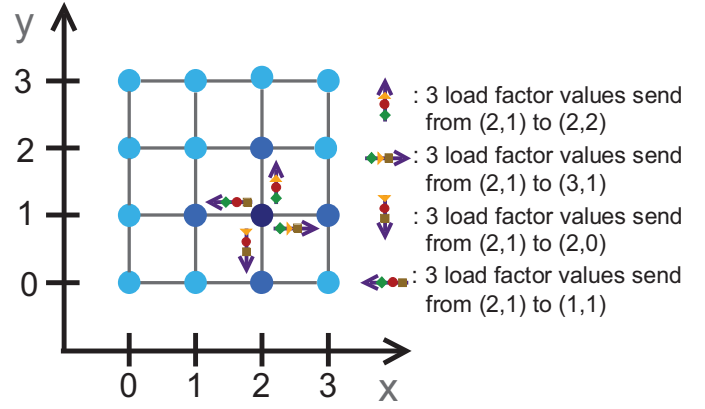


Figure 3. The load factor distribution, where node (2, 1) sends the updated load factors to its neighbors, using colored shapes as in Figure 2.

update interval, the current link utilization and average expedition delay of the link can be derived. The load factor is calculated as a linearly weighted sum of these as shown in (2). The weights c_{lu} and c_{ed} should be tuned to the needs of the network in question, depending on which is the most important: balancing link utilization (resource efficiency) or expedition delay (QoS).

$$lf_{2,1,N_n} = k_{dyn} \cdot (c_{lu} \cdot lu_{2,1,N} + c_{ed} \cdot \text{mean}(ed_{2,1,N} [])) + c_{neigh} \cdot \text{mean}(lf_{2,2} [])) + (1 - k_{dyn}) \cdot \left(\frac{\sum_i \exp\{-i \cdot c_{prev}\} \cdot lf_{2,1,N_{n-i}}}{\sum_i \exp\{-i \cdot c_{prev}\}} \right) \quad (2)$$

Besides the measured load metrics the calculated load factor is also based on the average of the latest received load factors from the neighbor connected to the output link in question as shown in (2). Thereby the load factors of a node will increase if its neighbors are heavily loaded. To avoid an unstable network where traffic oscillates because of rapidly changing load factors the previously calculated load factors are also included. These values are exponentially weighted, so that more recent values count more than older ones. The coefficient k_{dyn} determines how dynamic the load factor calculation is and how much influence the older values have.

B. Load Factor Distribution

When the link load factors have been updated they are distributed to the neighboring nodes (i.e. to a maximum of four nodes). As shown in Figure 3 each neighbor receives three out of the four load factors, e.g. node (1,1) receives the load factors $lf_{2,1,N}$, $lf_{2,1,E}$ and $lf_{2,1,S}$ as illustrated with small colored shapes on the purple arrow. Node (1,1) does not need to know the fourth load factor ($lf_{2,1,W}$), because it describes the load of sending from node (2,1) to node (1,1). In case the node is an edge or corner node only one or two load factors will be distributed to each neighbor, respectively.

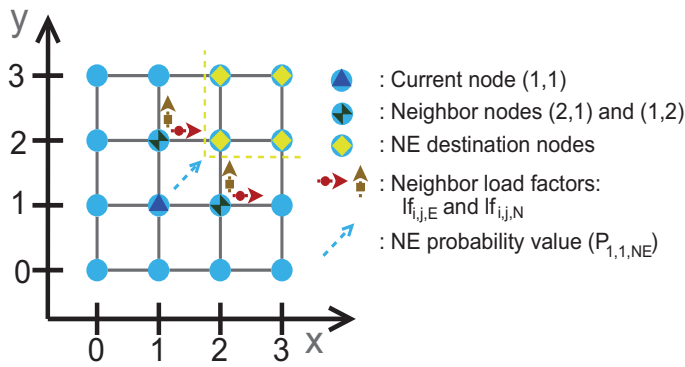


Figure 4. The load factors from the neighbors used to derive the probability value $P_{1,1,NE}$.

The load factors are only distributed once after each update, using one of two different methods: **a)** appending the load factor values to existing traffic forwarded to a neighbor by including it in the packet header, i.e. piggybacking, or **b)** by sending out load factor control packets with the only purpose to deliver the updated load factor to the neighbor. The latter is only used if no existing traffic is forwarded within a timeout interval, thereby ensuring that the updated load factors will be distributed within a certain time interval. Piggybacking is used as a first priority distribution method to avoid too much overhead traffic. The values could be appended to the packet header the way extension headers are used in IPv6, thereby only extending the packet size when necessary.

C. Probabilistic Routing

When a node receives a set of load factors from one of its neighbors it uses these to update its link probabilities, i.e. the probability of using a specific output link when forwarding a packet towards its destination. Due to the shortest path constraint of topological routing no more than two output links are ever considered. If the destination is on the same axis as the current node, only one output link is considered and load balancing is not used. If the destination is in e.g. the North-East direction, i.e. forwarding on either the North or East output link will bring the packet closer to the destination, the choice of output link is made based on the probabilities of these two links.

An aggregated probability value is calculated for the North-East direction: P_{NE} (and likewise for the directions North-West, South-East and South-West). The probability of forwarding on link N is P_{NE} and the probability of forwarding on link E is $1 - P_{NE}$. This value is derived purely on the load factors received from the neighbors, not the load factors calculated by the node itself. To derive $P_{1,1,NE}$ (P_{NE} for node (1,1)) the sets of load factors received from the neighbors in this direction are used (i.e. $lf_{1,2}[]$ and $lf_{2,1}[]$). Only two of the three received load factors from each neighbor are used, namely the ones representing the N and E links, as shown in Figure 4. $P_{1,1,NE}$ is derived as shown in (3), where the

average of the squared neighbor load factors are used. Using this formula the probability of forwarding traffic to a node will decrease, when its load increases (given that the load of the other neighbor stays constant).

$$P_{1,1,NE} = 1 - \frac{\text{mean}(lf_{1,2}[]^2)}{\text{mean}(lf_{2,1}[]^2) + \text{mean}(lf_{1,2}[]^2)} \quad (3)$$

IV. PERFORMANCE EVALUATION

In the following the performance evaluation of the load balancing algorithm is presented, by describing the applied traffic patterns and the test results together with a discussion.

A. Traffic Patterns

To test the algorithm on the i3 Demonstrator, artificial internet traffic was generated. According to an overview of internet traffic studies given by Williamson [12], internet traffic consist mainly of TCP streams.

A method for generating packets in streams has been devised based on stochastics. The interval length of the streams and the intervals between packet transmissions are based on Poisson processes, each with their own rate. These rates define how much traffic a given node will send to another node or basically when to toggle a stream of packets. The rates are randomly generated between a set of max and min values. Different kinds of traffic pattern can thus be generated. If the max/min value are equal uniform traffic is generated, however this pattern is not used.

More recent studies, like one from 2008 by Sandvine [13], points out the fact that P2P traffic accounts for 43.5% of the internet traffic. With current ISPs promoting IPTv, these servers have to be taken into account when planning backbone networks, as they will act as hotspots [14].

The P2P traffic is modeled by letting all nodes communicate with all other nodes by establishing streams with random rates to all nodes and sending packets according to the above described traffic generating model. Hotspot traffic is modeled by letting a few selected nodes receive and send a lot of traffic. In the tests this traffic was symmetric, however this need not be the case for hotspot traffic. In the implemented traffic generator on the testbed a merge of P2P and Hotspot is used. All generated packets were unisized due to the testbed implementation.

Each test run was carried out during a 3 min interval, with 150 packets/s capacity for each link. During tests each node locally logged the expedition delay and size of transmitted packets. These local log files were then transferred to the master computer after the test for post-processing and analysis in Matlab.

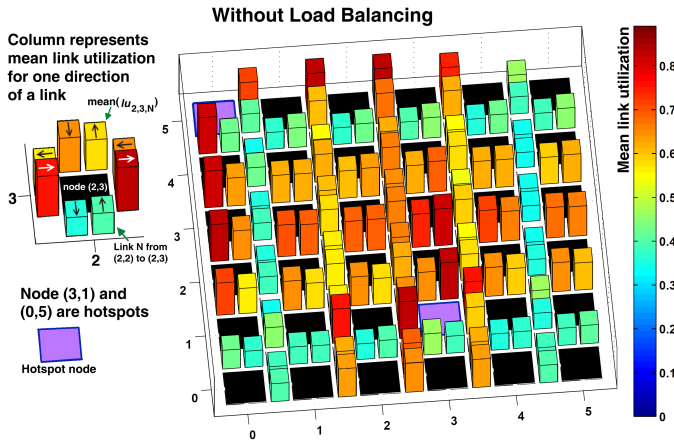


Figure 5. The mean link utilization without load balancing in a test with 2 hotspots at node (3, 1) and node (0, 5)

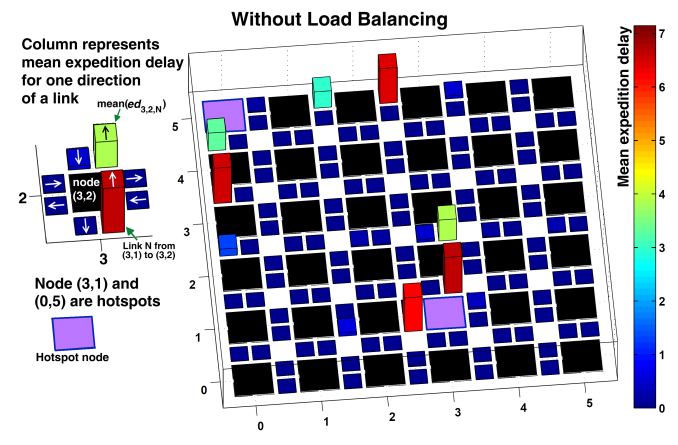


Figure 7. The mean expedition delay without load balancing in a test with 2 hotspots at node (3, 1) and node (0, 5)

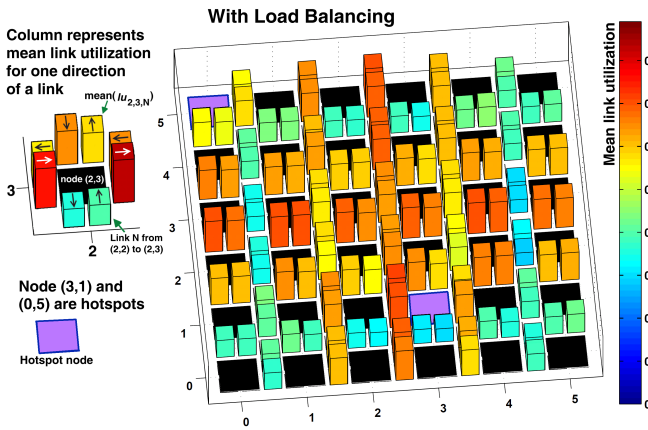


Figure 6. The mean link utilization with load balancing in a test with 2 hotspots at node (3, 1) and node (0, 5)

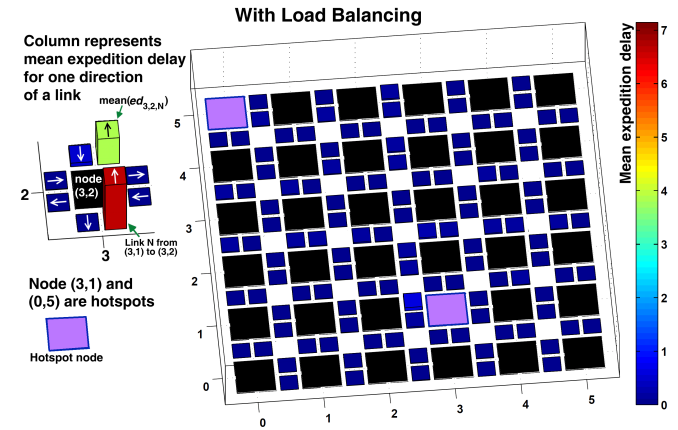


Figure 8. The mean expedition delay with load balancing in a test with 2 hotspots at node (3, 1) and node (0, 5)

B. Results and Discussion

15 test-runs have been conducted using different traffic levels (average link utilization ranging from 10 – 80%) of both pure P2P traffic, P2P + 1 hotspot, and P2P + 2 hotspots. Each test-run was carried out both with and without load balancing for comparison. A test-run with 2 hotspots at node (3,1) and node (0,5) and a medium-high traffic level (average link utilization of 53%) showed the following results, using the results without load balancing as a reference (100%). The $\text{var}(\text{mean}(lu_{i,j,x}))$ was only 80% and the $\text{mean}(\text{var}(lu_{i,j,x}))$ was 201% - the link utilization is therefore more smoothly balanced on average, but varies more over time when using load balancing. In Figure 5 and in Figure 6 the mean link utilization is compared for the two tests, where the links are between the black nodes. The peaks have been smoothened in the load balancing graph.

Most noteworthy, the $\text{var}(\text{mean}(ed_{i,j,x}))$ was only 0.4%, the $\text{mean}(\text{var}(ed_{i,j,x}))$ was 6% and the $\text{mean}(\text{mean}(ed_{i,j,x}))$ was 18% - the expedition delay is therefore more smoothly balanced, it varies less over time and

is reduced on average. Comparing the graphs in Figure 7 and in Figure 8, where the output links queues are plotted between the black nodes, it is found that these results are caused primarily by a few high peaks (the expedition delay variance graphs look almost the same). The high expedition delay peaks are caused by packets piling up in queues. As a consequence packets were therefore also dropped in four of the queues in the test without load balancing (up to 2500 drops), but not with load balancing. The packet delay and jitter was also much lower when load balancing was used.

Due to space limitations the results from the other test-runs are not given in detail, however the following conclusions can be made (the metrics of the results mentioned below are equal to those of the results discussed above in the same order):

- The expedition delay benefited more from load balancing than the link utilization as is illustrated by the values discussed above. In another test with medium-high traffic and 1 hotspot the results for link utilization were 104% and 130% and the results for expedition delay were 1.3%, 12% and 32% respectively. Other test results confirm this

conclusion. This difference has to do with the weighting coefficients used in (2), which favored expedition delay.

- When the pure P2P traffic pattern is used no real advantage is gained from using load balancing. In a test with medium high pure P2P traffic the results for link utilization were 107% and 100% and the results for expedition delay were 125%, 86% and 117% respectively. Load balancing primarily improves performance when hotspots are included as the results above show.
- When low traffic levels are used, the performance is slightly worse when load balancing is used. In a test with low traffic and 1 hotspot the results for link utilization were 105% and 71% and the results for expedition delay were 138%, 114% and 134% respectively. However, all the expedition delay results obtained with low traffic are so small that they can be neglected.
- Packet delay and jitter is improved with load balancing in the high traffic + hotspots tests, because the packet delay is correlated with the expedition delay.
- Packets were dropped in several of the high traffic + hotspots tests without load balancing, but not in the ones with load balancing.

V. CONCLUSION

The basic design of a load balancing algorithm, functioning as an extension to the topological routing scheme, has been presented. This algorithm is designed to distribute the load evenly across the network using feedback and probabilistic routing for the sake of better QoS support and resource utilization. Tests carried out on the i3 Demonstrator, a local testbed, have shown that a more even distribution of traffic load can be achieved when using the algorithm compared to not using load balancing. This conclusion is based on a lower variance of average link utilization over the network and a general reduction of expedition delay, both the average and the variance across the network. These results are obtained when a high level of traffic, which includes hotspot traffic, is routed in the network. When more uniform P2P traffic pattern or a lower level of traffic is used, the point in applying load balancing seems to vanish. However, the most important objective for a load balancing algorithm must be to deliver better performance in a network stressed with high traffic levels, since packets might be dropped or heavily delayed in this situation, resulting in poor QoS. To that extend these initial results prove the work so far to be successful.

A. Further Research

The results presented here can be regarded as one of the first steps to prove the usefulness of topological routing and the load balancing extension. Further work could include performing more tests on the current testbed or in a different setting, preferably a real life internet backbone. In these tests additional work should be put into tuning weighting coefficients in the load factor formula according to the needs

of the current network. The need for including load factors from neighboring nodes and previously calculated load factors should also be verified. It would also be interesting to compare the proposed feedback based approach with an algorithm, where the nodes use locally calculated load factors in the derivation of probability values. Such an approach would have a smaller overhead, but would perhaps not provide as good results as the feedback based approach. A merge of the two approaches could be imagined. Furthermore, it would be interesting to design an algorithm that is not constrained to the shortest path, however it should then avoid routing packets in loops.

VI. ACKNOWLEDGMENT

The research work was conducted in the context of a master student project at the Department of Electronic Systems at Aalborg University, and as such supported by the Study Board at Aalborg University, see www.esn.aau.dk.

REFERENCES

- [1] O. B. Madsen, T. P. Knudsen, and J. M. Pedersen, "SQoS as the base for next generation global infrastructure," in *Proceedings of IT&T 2003, Information Technology and Telecommunications Annual Conference 2003*, vol. 1, October 2003, pp. 127–136.
- [2] J. M. Pedersen, T. P. Knudsen, and O. B. Madsen, "Topological routing in large-scale networks," in *Proceedings of IEEE/ICACT 2004, The Sixth International Conference on Advanced Communication Technology, Korea*, vol. 2, February 2004, pp. 911–916.
- [3] J. M. Pedersen, A. Patel, T. P. Knudsen, and O. B. Madsen, "Applying 4-regular grid structures in large-scale access networks," *Computer Communications*, vol. 29, pp. 1350–1362, August 2005.
- [4] Intel, "Tera-scale computing research program," 2009. [Online]. Available: <http://techresearch.intel.com/articles/Tera-Scale/1421.htm>
- [5] K. Gopalan, T.-c. Chiueh, and Y.-J. Lin, "Load balancing routing with bandwidth-delay guarantees," *IEEE Communications Magazine*, vol. 42, pp. 108–113, 2004.
- [6] R. Luling, B. Monien, and F. Ramme, "Load balancing in large networks: A comparative study," in *Proceedings of the Third IEEE Symposium on Parallel and Distributed Processing, 1991*, vol. 1, December 1991, pp. 686–689.
- [7] J. H. Upadhyay, V. Varavithya, and P. Mohapatra, "Efficient and balanced adaptive routing in two-dimensional meshes," in *Proceedings of First IEEE Symposium on High-Performance Computer Architecture*, January 1995, pp. 112–121.
- [8] K.-C. Leung and V. Li, "Generalized load sharing for packet-switching networks," in *Eighth International Conference on Network Protocols (ICNP'00)*, 2000, pp. 305–314.
- [9] P. B. Nejsun and S. Studstrup, "Load balancing in large-scale 4-regular grid networks," 7th Semester Project, Aalborg University, Department of Electronic Systems, Denmark, December 2006.
- [10] K. Balachandran, J. H. Broberg, S. Hede, J. Pedersen, and D. Shabo, "Load balancing in grid networks," 8th Semester Project, Aalborg University, Department of Electronic Systems, Denmark, June 2009. [Online]. Available: <http://kom.aau.dk/group/09gr872/report.pdf>
- [11] T. M. Riaz, J. M. Pedersen, J. G. Lopéz, R. H. Nielsen, and O. B. Madsen, "A test-bed for topological routing in 4-regular grid structures," in *Proceedings of ICACT 2008, The 10th International Conference on Advanced Communication Technology*, vol. 1, February 2008, pp. 550–554.
- [12] C. Williamson, "Internet traffic measurement," *IEEE Internet Computing*, vol. 5, no. 6, pp. 70–74, November 2001.
- [13] T. Spangler, "Study: 44% of internet traffic is peer-to-peer," *Multichannel News*, June 2008. [Online]. Available: <http://www.multichannel.com/article/CA6572401.html?nid=3408>
- [14] S. Labeaga, "Design of a new IT infrastructure for the region of Nordjylland - the backbone network," Master's thesis, Aalborg University, Department of Electronic Systems, Denmark, June 2008.