Shortest Path Awareness in Delay-Based Routing

Mia Heinz, Christoph Schwarzenberg*, Florian Wiedner*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: mia.heinz@tum.de, schwarzenberg@net.in.tum.de, wiedner@net.in.tum.de

***Abstract*—The back-pressure routing (BP) algorithm is proveably throughput optimal which makes it a very promising algorithm, but it struggles with high end-to-end delay. In this paper, we will compare existing approaches on delay-based routing, shortest-path-aided back-pressure routing and possible combinations of both to determine how they decrease the end-to-end delay while maintaining throughput optimality.**

***Index Terms*—back-pressure, shortest path, delay metric, end-to-end delay**

1. Introduction

Efficient and fast routing is becoming increasingly important especially in real-life applications, like streaming where a large end-to-end delay is very noticeable.

The BP algorithm promises throughput optimality which means that it guarantees system stability in a network. The network is stable meaning that queue occupancy does not increase endlessly.
Under a high traffic load the algorithm works comparably well as it exhausts all possible paths and therefore distributes the traffic over the whole network instead of concentrating it on one or few paths.
If the traffic is lighter, however, this becomes a problem. With a low traffic load the algorithm chooses randomly between all possible paths, therefore sending data over unnecessary long paths or even loops causing high end-to-end delay2.

First, the paper describes the BP algorithm in Section 2. Section 3 gives an overview of existing optimization methods and in Section 4, we focus on shortest-path-awareness and delay-based algorithms.

2. Background

The BP routing algorithm was first proposed by Tassiulas and Ephremides in [1]. It was originally developed for wireless multihop radio networks but can easily be transferred to wired multihop networks.

The algorithm works with congestion gradients in queueing networks. Each node has a queue for each destination in the network. A flow is a sequence of packets belonging to one activity. A link weight is calculated by the difference of queue length for a flow and the neighbor's queue length for the same flow. The biggest weight signifies the least congested path. Then in each time slot, the scheduling decision is made to maximize the sum of the weights for activated links.

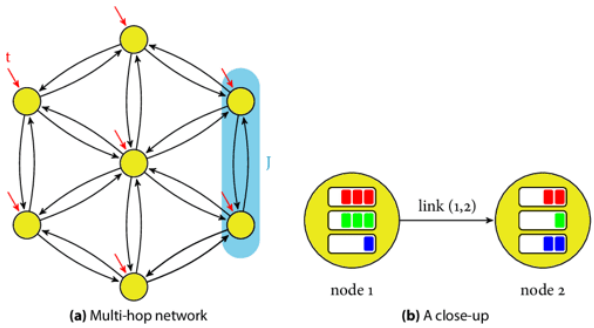

Figure 1: General principle of the BP algorithm [2]

Figure 1 shows an example illustrating a multihop queueing network. For the shown link $(1, 2)$ in this example the BP algorithm would choose the green flow, as the difference in queue length is the highest.

Another way to understand the principle is imagining the links of a network as pipes and the data as water. The water is put into the pipe network in one place and can only escape at its destination. If it is only a little bit of water it will randomly distribute over the pipes. If more water is introduced to the network, the water pressure pushes the water out at the destination. With the BP algorithm pressure builds up leading the data to its destination the same way water pressure builds up and pushes the water out of the pipes at the destination.

The BP algorithm is provably throughput optimal and stable. But challenges are the high end-to-end delays, especially in low data load traffic because it always exploits all possible paths.
According to Hai et al. in [3], the high end-to-end delay is caused by three main factors:

1) The *initial delay* describes the initial startup time it takes to build up pressure for the BP algorithm to work.
2) The *random walk delay* is the delay that is created when two links have the same weight, so a random choice is made which can lead to looping or unnecessary long paths.
3) The *last packet delay* describes a delay of the last packet of a flow. If no packets to the same destination are following the last packet of a flow, the queue remains lightly filled and the packet may starve for an undefined amount of time.

The following section analyses approaches to reduce the end-to-end delay.

doi: 10.2313/NET-2022-11-1_07

3. Related Work

Since the first introduction of the BP algorithm there have been many attempts to improve it.

The first group of optimization methods is based on queue structure and management.

One approach is the clustering of nodes, which was explored by Ying et al. in [4]. The idea is that nodes are combined in clusters and if the source and destination are already in the same cluster, the standard BP algorithm is applied. If they are in different clusters, the BP algorithm is used to reach one gateway of the cluster containing the destination. This not only reduces the end-to-end delay but also the memory complexity, as fewer queues have to be maintained. The main problem of this algorithm is the question of how to cluster the nodes.

In [5] Alresaini et al. introduce a BP algorithm with adaptive redundancy (BWAR). If a node has a queue backlog below a certain threshold, it duplicates the packets it is sending in another buffer. Then when the queue is empty those copies are sent. By creating several copies of one packet, more pressure is built up and the chance that one arrives at the destination is higher. When one of them arrives at the destination, all other copies have to be deleted. Deleting all copies is a problem which can be solved with timeouts.

Another approach is combining the BP algorithm with data science and machine learning. In [6] Huang et al. propose a predictive BP algorithm that predicts and preserves the arriving packets based on a lookahead-window.

Furthermore, approaches that reduce the delay by decreasing the path length or the number of hops, like avoiding or reducing loops [7] or including shortest-path-awareness [8], will be considered in detail in Section 4.1.

Delay-based algorithms, that use metrics other than the queue length to make the routing decisions, can also be used to reduce the delay, which will further be explained in Section 4.2.

4. Approaches

In the following we will have a closer look at the approaches using shortest-path-awareness and delay-based routing.

4.1. Shortest-Path-Awareness

In [8], Ying et al. combine the traditional BP algorithm with shortest-path-awareness based routing which decreases end-to-end delay while maintaining throughput optimality.

The end-to-end delay is dependent on the hop count of packet i.e., the path length and the time it spends in the queues. Using an algorithm like Dijkstra or Bellman-Ford the shortest path to each destination can be calculated for every node.

If the routing algorithm would now always use the shortest path, the path becomes congested very fast and this leads to a very high queue delay as the packets are buffering and waiting in the queues.

Therefore, in the proposed algorithm not only the shortest path is used but all paths with a specified maximum path length. This hop constraint gets adjusted as the data rate in the network rises.

First, Ying et al. propose an algorithm for a specified hop constraint h, which uses the BP algorithm but only exploits paths with a length $\leq h$. So every node knows the minimum hops for every destination and every flow has a hop constraint h. Then the path (a, b) is only a valid option if node b is less than $h - 1$ hops away from the destination. From all possible hops, the one with the least congestion is picked via the BP algorithm.

In a network we usually do not have a set hop constraint but the maximum path length is dependent on the data rate and h has to be chosen dynamically. They introduce a variable K which is the price for taking a longer path, so a larger K minimizes the average number of hops but leads to a larger queue delay as not that many paths can be used. A smaller K means that longer and therefore more paths can be used, better distributing the flows and leading to a smaller queue delay.

Using this K, the optimal h is calculated dynamically to minimize the tradeoff between hops and queue delay.

The combination of the BP algorithm and the shortest-path-aided BP algorithm significantly reduces the initial delay as in the beginning no pressure is needed to guide the data to its destination and just the shortest path is used. The random walk delay is also decreased because if links have the same weight, now the shorter path is chosen and loops are prevented by limiting the hop constraint to the number of nodes.

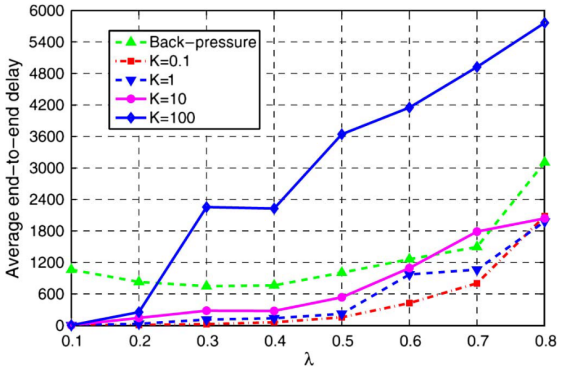

Figure 2: Simulation results comparing end-to-end delay in traditional BP algorithm and shortest-path-aided BP algorithm with different values for K using OMNeT++ with λ arrival rate of flows in packets/time slot [8]

As we can see in Figure 2 for a low data rate the delay of the BP algorithm first decreases before increasing again with higher data loads. This is due to the initial packet delay. In the shortest-path-aided BP algorithm the end-to-end delay is significantly reduced compared to the traditional BP algorithm, as especially under a low data rate excessively long paths are not explored. But also for higher data rates, there is an improvement as the random walk delay is reduced and loops are prevented. However, the problem of the last packet delay is not solved as starvation is still possible.

But it all depends on the value of K. If K chosen is too large, the algorithm always chooses the shorter path independent from the queue backlogs, because the penalty of choosing a longer path is too high. Hence, the delay

doi: 10.2313/NET-2022-11-1_07

caused by buffering in the queues is not considered. If K is too small, however, unnecessary long paths are taken even if the shorter path would have been faster. The optimal value for K is specific to the network and the problem of how to choose it is still open.

In [7], Rai et al. developed a loop-free BP (LFBP) algorithm. By giving the links in the network a direction a directed acyclic graph (DAG) is created. Then in this DAG the BP algorithm is applied. If it now overloads at some point in the network, the directions of the links pointing from the non-overloaded nodes to the overloaded nodes are reversed which creates a new DAG. This procedure prevents congestion in the network by routing packets away from overflowing areas and towards less busy areas of the network.

Another hop minimizing algorithm is presented in [9] by Bui et al. The Enhanced Backpressure (EBP) algorithm uses shortest path heuristics to first use short paths and only add longer ones if the links are overloaded.
This algorithm also uses shadow queues and only maintains queues for the direct neighbors, which reduces the memory complexity as well as the delay.

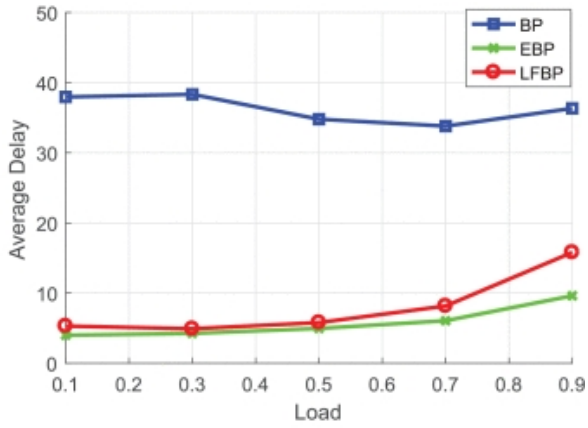

Figure 3: Simulation results comparing end-to-end delay in traditional BP algorithm, EBP and LFBP [7]

As we can see in Figure 3, both the EBP and the LFBP have a far smaller delay than the traditional BP algorithm with the EBF showing slightly better results. One can especially see that both algorithms do not suffer from the initial packet delay.

4.2. Delay-Based Algorithms

In [3], Hai et al. introduce a delay-based optimization method, which combines the queue length with the packet delays to calculate link weight. They introduce a metric called the sojourn time backlog (STB) and an STB-based BP algorithm (STBP). Instead of the length of the queue the sum of sojourn time, the time passed since a packet arrived in the network, of all packets in the queue is used.

If every packet just has the weight 1, like in the traditional BP algorithm, there cannot be a prioritization of packets, which already have a high delay. Giving the packets different weights by assigning the STB as the link weight, leads to more pressure on packets with a higher delay, preventing starvation and unnecessarily long paths, therefore reducing the average end-to-end delay. Hai et al. introduce an implementation of this algorithm using First-In-First-Out (FIFO) queues and virtual queue management.

As synchronization in networks is very hard to achieve, another option Hai et al. propose is taking the hop count as the delay instead of the actual time. Using this hop approach the delay of the packets is not increasing while buffering, therefore packets can be stuck in queues for a very long time and starve without their priority increasing. It still reduces the average number of hops and the end-to-end delay because if a packet was already transmitted over many hops, it is then prioritized and the remaining number of hops is therefore on average smaller.

Both versions decrease the initial delay and the random walk delay, but the last packet delay is only decreased by the STBP using the actual time and not the hops because the starvation problem is not fixed.
Considering flow dynamics the last packet delay is especially important as all packets of a flow have to be transmitted. So in this context the STBP algorithm using the actual time has a significant advantage over the hop count based version.

In [10], McKeown et al. introduce another weight metric. It uses the sojourn time only of the head-of-line (HOL), the first packet in the queue. The so-called oldest cell first (OCF) algorithm, which uses HOL delay as a metric to calculate the link weights, achieves less delay compared to the traditional BP algorithm.
In the BP algorithm queues with a short length can be starved if the length remains small and other queues receive new packets regularly, therefore being prioritized over the shorter queues. This problem is solved with the OCF algorithm as the waiting time of the HOL increases in each time slot. This way the HOL delay gets bigger until it is eventually served.
The OCF algorithm also reduces the last packet delay as it prevents starvation of shorter queues by only considering the first packet.
Another HOL delay-based algorithm is introduced by Ji et al. in [11].

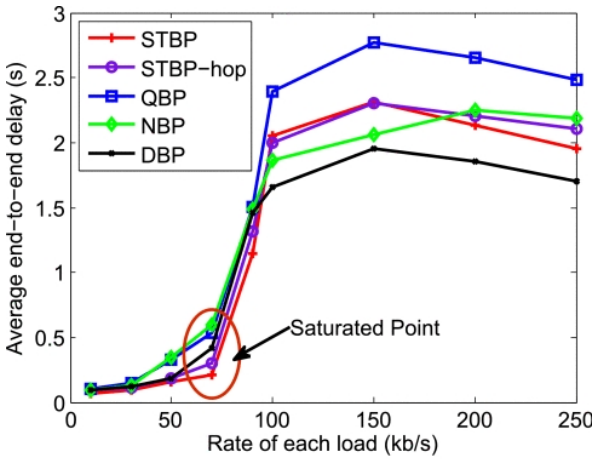

Figure 4: Simulation results comparing end-to-end delay in traditional queue-based BP (QBP), the STBP, the hop based STPB (STPB-hop) and an HOL delay based algorithm (DBP) using NS-2 network simulator [3]

In Figure 4 the network saturation point at about

doi: 10.2313/NET-2022-11-1_07

70kb/s is visible. After this point, the delay of all algorithms increases sharply. According to Hai et al. after the saturated point the network is no longer able to stabilize the data rate and operates in an overload.
When looking at the range where the network is still stable, we can clearly see the biggest improvement in end-to-end delay is achieved with the STBP but the hop-based STBP and the HOL delay-based algorithm are still better than the traditional BP algorithm.

When comparing the STBP, the HOL delay-based based algorithm and the traditional BP algorithm it becomes obvious that the STBP, which is a combination of the other two has the best performance. The end-to-end delay gets higher the longer packets are buffering in the queues.
The HOL algorithm ignores very long queues if the first packet only has a small delay, as the algorithm does not consider queue length. As the delay increases constantly, these longer queues are not starved. The longer a packet is not served, the higher the delay gets and the more likely it will be served in the next time slot. It can however lead to a slightly bigger delay for the waiting packets.
The queue-length-based BP on the other hand ignores short queues even if the packets in it already have been waiting for a long time. The STBP prioritizes long queues and highly delayed packets combining the advantages of both algorithms.

Another approach is the LIFO-backpressure which is explored by Huang et al. in [12]. They show that by simply combining FIFO and Last-In-First-Out (LIFO) queues a significant improvement in delay can be achieved as the packets with the highest delay can be served first regardless of if they are at the head or tail of the queue.

4.3. Comparison and Combination

The algorithm reducing the path length and delay-based algorithms focus on different aspects.
Both algorithms still have unsolved problems for example how to choose K in the shortest-path-aided BP algorithm or how to achieve synchronization but both approaches already accomplish a significant improvement compared to the traditional BP algorithm.

By reducing the path lengths a packet is directed to its destination faster but packets can still starve and especially with a high data load almost as many paths are used as in the traditional BP algorithm.

In delay-based algorithms we create a prioritization of packets that already have a high delay but the path is still chosen randomly and loops can still occur.

So naturally one could try combining both algorithms to decrease the end-to-end delay even more and optimize the BP algorithm further. One idea would be using the joint algorithm from [8] but instead of the queue length the STB is used as the metric to make the routing decisions. By combining the algorithms it would be possible to profit from the shortest-path-aided BP algorithm reducing the initial packet delay and the STBP reducing the last packet delay and both algorithms reducing the random walk delay. This would decrease the overall delay even more. However, when combining the two algorithms one also has to consider the challenges of both algorithms.

Especially in flow-based routing we cannot only look at per packet delay but have to consider the overall delay until all packets of the flow have arrived. Therefore the last packet delay plays an important role, as all packets of a flow have to arrive.

5. Conclusion

We analyzed and compared the shortest path and delay-based approach. Both reduce the end-to-end delay and have their specific advantages and challenges. For shortest path based approaches the biggest improvement is the reduction of the hops in low data load and the biggest challenge is the tradeoff between minimizing the path lengths and the delay from buffering in the queues. But if an optimal parameter K is chosen for this, the end-to-end delay is also reduced for higher data loads, e.g. by avoiding loops.

Delay-based algorithms reduce the end-to-end delay by prioritizing already heavily delayed packets and bringing those to their destination first. The challenge with this is how to measure the delay as synchronization in networks is very hard to achieve and also while they are prioritized, it is still possible that the packets are transmitted in loops or overly long paths.

All in all a combination might benefit from both algorithms' strengths and lead to an even better optimization but we have to consider that we also have to deal with both algorithms' challenges.

References

[1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," in *29th IEEE Conference on Decision and Control*, 1990, pp. 2130–2132 vol.4.

[2] J. Kampen, "Route guidance and signal control based on the back-pressure algorithm," 2015.

[3] L. Hai, Q. Gao, J. Wang, H. Zhuang, and P. Wang, "Delay-optimal back-pressure routing algorithm for multihop wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2617–2630, 2018.

[4] L. Ying, R. Srikant, and D. Towsley, "Cluster-based back-pressure routing algorithm," in *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, 2008, pp. 484–492.

[5] M. Alresaini, M. Sathiamoorthy, B. Krishnamachari, and M. J. Neely, "Backpressure with adaptive redundancy (bwar)," in *2012 Proceedings IEEE INFOCOM*, 2012, pp. 2300–2308.

[6] L. Huang, S. Zhang, M. Chen, and X. Liu, "When backpressure meets predictive scheduling," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2237–2250, 2016.

[7] A. Rai, C.-p. Li, G. Paschos, and E. Modiano, "Loop-free back-pressure routing using link-reversal algorithms," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2988–3002, 2017.

[8] L. Ying, S. Shakkottai, A. Reddy, and S. Liu, "On combining shortest-path and back-pressure routing over multihop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 3, pp. 841–854, 2011.

[9] L. Bui, R. Srikant, and A. Stolyar, "Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing," in *IEEE INFOCOM 2009*, 2009, pp. 2936–2940.

[10] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Transactions on Communications*, vol. 47, no. 8, pp. 1260–1267, 1999.

doi: 10.2313/NET-2022-11-1_07

[11] B. Ji, C. Joo, and N. B. Shroff, "Delay-based back-pressure scheduling in multihop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 21, no. 5, pp. 1539–1552, 2013.

[12] L. Huang, S. Moeller, M. J. Neely, and B. Krishnamachari, "Lifo-backpressure achieves near optimal utility-delay tradeoff," in *2011 International Symposium of Modeling and Optimization of Mobile, Ad Hoc, and Wireless Networks*, 2011, pp. 70–77.

doi: 10.2313/NET-2022-11-1_07