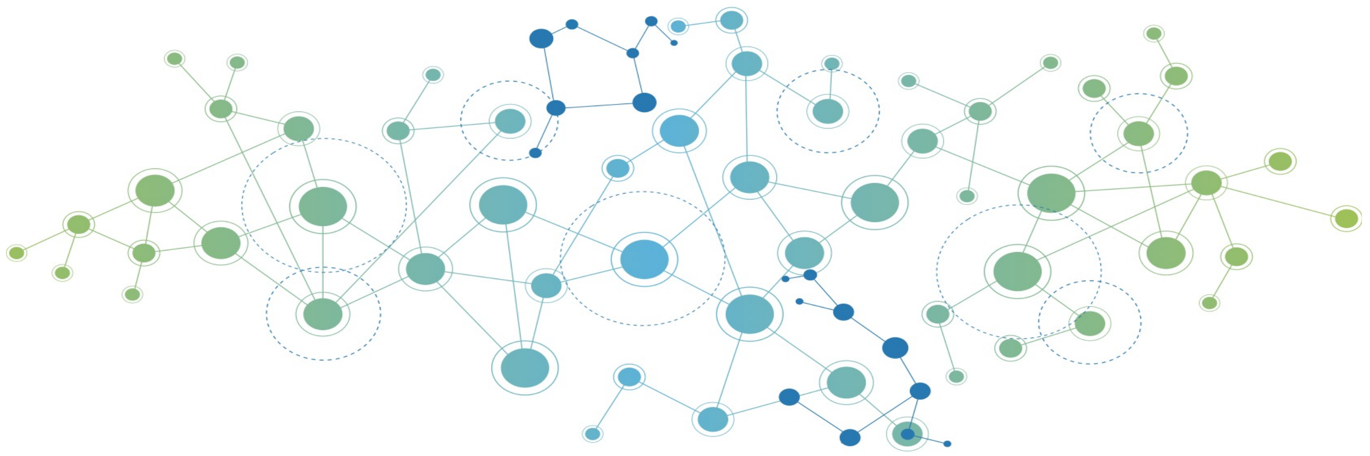
Feedback aided distributed routing based throughput optimization

BEN VAN VIEGEN, University of Twente, The Netherlands



This research proposes a light weight distributed routing system using arrival rate feedback to optimize the throughput of a wireless sensor network (WSN). The nodes have close to no knowledge about the system, the only information they have is their own hop count to the gateway and the hop count of their neighbors. Using this information nodes are able to route data traffic to the gateway of the network, however the throughput of the system is rather low. Using periodic feedback about the arrival rate of packets at the gateway nodes try to improve the network efficiency by avoiding slow or congested nodes. The system shows a maximum throughput increase of 45%, showing that the small amount of information that the feedback provides allows nodes to significantly improve network throughput. The system is not able to compete with systems that utilize knowledge about network topology with respect to throughput optimization. However, in systems where knowledge about the network topology is not feasible, this system could provide a significant performance increase with a very limited amount of feedback information.

Additional Key Words and Phrases: Distributed control, distributed routing, source routing, manet, arrival rate feedback

1 INTRODUCTION

In the case of wireless sensor networks, on which this research will be focused, it can be assumed that there is a single point where all the data flows towards (the gateway of the network). These networks do come with the challenge that it is not always feasible for all nodes to know the entire network topology, making efficient routing difficult. Although knowing the entire network topology might not be feasible, some very simple information about the network topology is assumed to be attainable. This information being the amount of hops needed to get to the gateway and the amount of hops that each direct neighbor needs to get to the gateway. The nodes will also receive periodic feedback from the gateway with information about the arrival rate of packets. With only this limited amount of data the question is whether the nodes will be able to make 'good' routing decisions resulting in efficient routing leading to a higher throughput of the network.

TScIT 41, July 5, 2024, Enschede, The Netherlands

2 PROBLEM STATEMENT

When scaling wireless multi-hop networks it may become difficult for each node or any entity to have a global overview of the network. Posing a challenge for nodes to make "good" routing decisions resulting in efficient routing. If the global network topology is not known then what should nodes use as information to base their decisions on. Simply sending messages to nodes with a lower hop count would result in the messages arriving at the gateway, but this might also lead to some nodes getting congested whereas some connections are left idle most of the time. This is of course problematic when trying to send large amounts of data through the network that requires close to the theoretical maximum throughput of the network. The congestion would lead to very large delays and packets being dropped due to their Time to Live running out or queues overflowing. Apart from the initialisation where nodes get to know the hop count of themselves and the hop count of their neighbors, no other control information will ever be exchanged between nodes, leaving nodes without any information regarding congestion or anything at all about the other nodes in the network. This makes it hard for a node to 'correctly' chose which neighbor(s) to use for forwarding messages to prevent unnecessary congestion of the network. To give nodes a bit more information to work with, the gateway of the system periodically sends out how much packets it has received from each node. If packets do not arrive at the gateway this could point to the network being congested and nodes should

try to decrease this congestion in the next period until settling on an 'optimal' solution for the current data traffic.
It is under these circumstance, where nodes do not know more than the hop count of themselves and their direct neighbors and only receive periodic feedback about the arrival rate of their packets from the gateway, that the questions arises, whether nodes could be able to make individual routing decisions resulting in efficient routing. If this would be the case this might possibly provide better scalability for wireless sensor networks.

2.1 Research Question

To what extend can feedback from the gateway about the arrival rate of packets (grouped by the first hop a packet took) help individual nodes make routing decisions to maximize the throughput of the network?

2.2 sub-research questions

- How would such a feedback aided approach influence the latency of the network?
- What is the effect of limiting the percentage of change that nodes can apply to their sending ratios[1] per iteration?

3 RELATED WORKS

There is already quite some research done regarding distributed routing protocols. For instance aimed at improving queue stability to meet more strict quality of service constrained. One of the researches suggests a random access scheduling approach with probabilities dependent on the local backlog [6]. Similar to this research the referred research assumed nodes to be almost entirely blind, meaning that nodes do not know much about the rest of the network. In contrast to this research the referred research proposes an optimization approach that adjusts the medium access control strategy to improve the throughput of the network. This however, leaves performance on the table when it comes to prioritizing more efficient paths. A combination of the research of T. Yang et al. [6] and this research could provide a stronger solution, optimizing both the medium access control and the routing could result in the cumulative improvement of both systems.
T. Yang et al. [9] propose a distributed power control algorithm that uses the inherent properties of wireless networks to estimate a centralized maximum weight scheduling algorithm. The approach uses a combination of routing, link scheduling and resource allocation to meet Quality of Service (QoS) requirements while keeping energy consumption minimal. They aim to provide a solution that allows for less decrease in performance when scaling a heterogeneous network. Both this research and the research performed by T. Yang et al. do not utilize knowledge about the network topology. In contrast to this research their research does not utilize the gateway for providing feedback.
T. Padmapriy et al. [7] propose a system that uses hierarchical Channel Strength Index feedback, to inform nodes about link quality. They show an improvement in performance and a decrease in overhead. Their approach is similar to this research, showing that such a feedback system allows for performance improvements in a network. Different kinds of feedback are not explored in the paper. This research provides further exploration in the domain of feedback aided network optimization, covering feedback regarding packet arrival rates instead of channel quality. A combination of both approaches could provide improved performance in comparison to both individual systems and could be an extension of both researches.
S.J. Douglas et al. [1] and L. Shuang et al. [5] propose a system that uses the Expected Transmission Count (ETX) metric to find optimal paths to the sink. Both show an improvement in network performance (with regards to throughput), in comparison to a directed diffusion approach. Both researches do meet different service requirements than this research, where sinks send their data requests through the network and nodes that can provide the requested data will respond. Whereas, in this research all nodes simply send all their data to the single gateway of the network. The challenge of efficiently routing data to the sink is similar in both cases. Suggesting that the performance increase that S.J. Douglas et al. and L. Shuang et al. show, could also improve the feedback system covered in this research, by expanding or replacing the feedback about arrival rate, with ETX feedback.

4 HYPOTHESIS

When nodes would receive feedback from the gateway of the network about the amount of arrived packets and via which first hop they went, this would give nodes some crucial information about via which nodes packets would get lost, because of slow links or congestion. When a single node in the network would receive such feedback and make a change based on the information it would be expected that it would result in the network achieving the same or higher throughput, since the node would reroute the amount of lost packets to a better performing node that possibly still has some extra capacity for sending extra packets. When allowing all nodes in the system to make adjustments to their routing protocol simultaneously at the end of each iteration, this could introduce oscillatory behaviour. Many different nodes could chose to reroute a lot of their packets to the same node which would result in that node being overloaded and congested, so in the next iteration they would then switch back only for the cycle to repeat. A solution to this problem would likely be to limit how much each node can change per iteration or to only let some nodes change per iteration. If this would resolve the oscillatory behaviour then over time the throughput of the network would likely increase because of the changes since over time slow links would be utilized less and less and packets would be rerouted more via faster paths.

5 PROPOSED SYSTEM

5.1 prerequisites

The proposed system is based on some prerequisites. Firstly, the simulated network allows nodes to have different links with different capacities. If all nodes would have the same sending speed then there would be nothing to optimize since a particular path would not be any better than any other path (with the same hop count). Also this difference in connection speeds represents the differences

[1]The percentage of the incoming message load that a node sends to each of its neighbors

in connection speeds caused by environmental factor in the real world. Secondly, the wireless communication protocol is assumed to work as follows: when a node send a packet it will use the full channel capacity for the entire duration of sending that packet. This means that there is no capacity left for a node that is sending or receiving a packet to send or receive anything else. This results in a node being blocked from sending or receiving anything if another node decide to send a packet to it. So if a node has two neighbors, one with a 10kb/s connection and one with a 150kb/s connection then the incoming speed of that node would not be 150+10=160, but instead 0.5*10+0.5*150=80, since the sending time is shared fairly between the two neighbors. Thirdly, there should be a certain level of fairness for the system to function (not implemented by default in The One). If this is not implemented it could be the case that a slow node takes all the sending time and does not leave any sending time for the fast nodes. This could result in a slow connection having a higher effective throughput than a fast link, which would make it hard to distinguish between a good (a lot of utilization of fast links and little utilization of slow links) and a bad (a lot of utilization of slow links and little utilization of fast links) configuration, since a bad configuration could result in a higher throughput due to the lack of fairness. Lastly, the system assumes that nodes do know their own hop count and the hop count of their neighbors. This is done to mitigate the need for network discovery and make sure that it is at least clear where the nodes should send the packets to get to the gateway, allowing this research to focus on only the data flow optimization. This information is also assumed to be quite easily attainable in a distributed network.

5.2 Functioning of the system

Since all data is routed to the gateway, the gateway would be able to record how many messages it has received from each node and communicate this back to those nodes. Instead of only counting the amount of message that arrive from a particular node, the gateway also records the first hop that a message took from that node. In a situation where node *A* would have neighbors *B* and *C*, feedback from the gateway would be the following: in the past period x message where received via node *B* and y messages where received via node *C*. This gives nodes an idea of how many message arrive via each possible neighbor (with lower hop count). Since the node knows itself how many message it has tried to send via that node it would be able to calculate the success rate of that node. If the success rate of a neighbor is 100% then it would mean that no messages are dropped when sending via that neighbor and it might even have capacity left to handle more messages. When the success rate of a neighbor is less then 100% it means that either that neighbor itself is overloaded or somewhere along the path from that neighbor to the gateway a node is overloaded. Regardless of which is the case, it would be a good choice to try and alleviate the node in question forwarding less message via that node.

5.3 Reasoning For The Proposed System

In a system where fairness with regards to sending time is provided, nodes with fast connections will be able to provide a higher effective throughput than slower nodes. If a large amount of message is sent through the network without keeping the link speeds in mind, this would lead to slow links being overloaded, resulting in packets being lost whereas fast links are not used to their full capacity. Since the slow nodes would drop packets, other nodes that send via these slow nodes will decide to reroute a part of the packets via other nodes that do have a 100% success rate and can presumably still handle even more. This would result in slow links being alleviated and used less for forwarding messages, instead faster nodes would be used which would be able to handle the increased load. This system would not only alleviate the slow links in the system but it would also prevent congestion in general. Even fast nodes can get congested if they have a larger amount of packets coming in then going out. As soon as this happens nodes will notice that sending via a particular neighbor has gotten a lower success rate so they will reroute a part of the packets. If it would happen that the congestion is somewhere down the line, all the nodes upstream will try to reroute. But, lets say the first node upstream of the congested node is still able to take all the load and reroute it to another neighbor then the following will happen. At first a couple of packets will be dropped because the node is congested and all nodes upstream will reroute a part of their packets. In the next iteration the first upstream node of the congested node is able to handle the same load as before by rerouting via another node. If the nodes upstream have issues cause by the previous reroute attempt, they will try to route it back the original way since this path now has a success of 100% again.

6 METHOD

The system used to perform the research is a combination of The One and a custom python wrapper. The python wrapper is responsible for generating the network topology and converting that to a configuration file that the simulator can use to simulate the network. Once the simulator has run the simulation, it will create result files which are read by the python wrapper and processed to create a new configuration after which the cycle repeats. Both the python wrapper and the extensions made to The One together with The One itself, are published on the GitLab server of the university of Twente [8]

6.1 The One

To simulate network communication an open source project, made by Ari Keränen, [4] was used. The project is called The One (Opportunistic Network Environment) and was originally created for delay tolerant networks (dtn's), but due to the simplicity of the simulator it allows for easy adjustments allowing it to be used for more than only the intended use case.

The One does not simulate the physical layer, so interference is not simulated. The One does support checking whether nodes are already busy with sending or receiving. So even though, there is no signal interference it is possible to only allow nodes to send or receive to/from one node at the time. This also means that if a slow node is sending it is keeping the receiving node busy, so it can't receive from other nodes. This means that slow nodes might keep a node busy preventing fast nodes from using more send time, therefore bringing down the efficiency of the network. There are also no fairness guarantees implemented by default, when running a simulation without this it can be observed that slow nodes take up most of the sending time, making the difference in throughput between fast and slow links almost unnoticeable.

6.2 The One Extensions

In order to make the One useful for this research a few extensions had to be added. Firstly support for individual node settings was added. This allowed the system to simulate a network where each node had its own independent values for parameters such as speed, and send timeouts. This system was also used to give the individual nodes information about which nodes their direct neighbors were and how much of their message load they should send to each neighbor. This was needed for the custom router.
The One did not yet contain a router that could send out the messages to their neighbors based on ratios, so this had to be implemented. For this, the individual configuration system was used to give each node a list of neighbors and a list of ratios, indicating how much the node should send to each neighbor. When the router has to forward a message it picks a neighbor by change based on the given ratios.
Lastly The One was extended further by adding a custom report to give the feedback system information about how many messages each node actually forwarded via which node. This report is necessary since the actual messages send out may not perfectly match up with the ratios, since the forwarding is chance based.
Since The One does not guarantee any fairness and in fact seems to favour certain nodes over others, a timeout system was added to make sure that "favoured" nodes could not keep a receiving node occupied. The timeout allows nodes to send for a certain time period and then blocks the node from sending anything for the same period to allow other nodes to send. A timeout of 50ms was chosen to make sure that nodes timeout often enough, so the chance based nature of selecting the next node that can send would provide a fair enough distribution.

6.3 Python Wrapper

The python wrapper consists of two parts. The first part handles the generation of the network topology and the second part handles processing the feedback from a simulation for each node and writing a new configuration based on that. Topology generation is done as follows: a specified number of node objects are created with random coordinates. Then for each node its neighbors are determined based on the specified range and after this the hop count of each node is determined. After this a configuration file is generated which contains all the information for the individual nodes and an events configuration file is generated for setting up the connections between the nodes. Nodes themselves also have a certain speed. To make sure there is a significant difference between connections, nodes do not get a random speed but instead they get one of two specified speeds. Either 150kb/s or 10kb/s. When a node is assigned a certain speed it can only transmit at that speed, it can receive at other speeds without issue. This is to simulate some nodes having better connections then other, due to for instance better placement (high above the ground) or other environment factors. The low

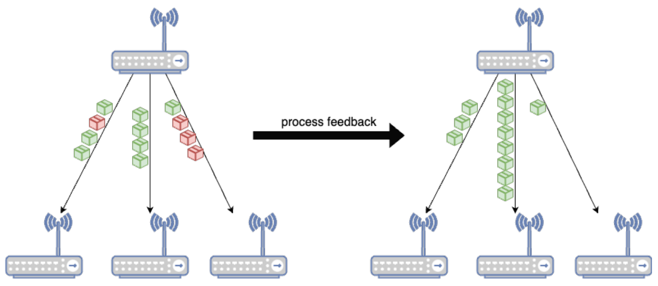


Fig. 1. diagram of system behaviour

speed was chosen to allow for exactly 1 packet transmissions within a sending period (50 ms) and the high speed is chosen such that the theoretical throughput of the network is more than the applied load (if only the fast link could and would be used).
When a simulation has finished there are 2 reports that the python wrapper uses to adjust the current configuration. The first one is the *DeliverMessagesReport*. This report contains a list of all the message that have been send through the network including the path that each message took from its source node. In a real network this file would be generated by the gateway receiving all the message and logging them to this file. The file is then processed to create a map for each node indicating how many packets the sink has received from that node and the amount of messages that where received via each neighbor of that node. In a real network this map would be the feedback that a node would receive. When the "feedback" map is generated the distributed feedback processing is simulated. Each individual node will use the custom *nodeSendingReport* to determine how many messages it has send via each neighbor in the last run and then compares these values to the feedback from the controller to determine what the success rate of each neighbor is. These success rates are then used to tweak the configuration for the next iteration. This tuning process, where it reduces the packet ratio that it send to nodes that are loosing packets and reroutes them to one or more nodes that do not lose any messages, can be seen in figure 1.
Nodes have a parameter that determines how much each node can change it output ratios per iteration. This was done to minimize anticipated oscillations that would occur within the network due to the distributed nature of the control system. In a case where more than one neighbors are overloaded, the node will make changes to the output ratios in such a way that each overloaded node gets a fair share of the maximum change amount. In the case where neighbor A would lose 20% and neighbor B would lose 10% and the maximum allowed amount of change is 10%, then 6,67% of messages going to node A would be rerouted to not overloaded nodes and 3,33% of messages going to node B would be rerouted to not overloaded nodes. This would effectively reduce the load to node A by 6,67% and the load to node B by 3,33%. This should result in less load on node A and B resulting in less congestion and thus less packet loss and therefore a higher success rate (if the not overloaded nodes still had extra capacity to handle the extra messages).

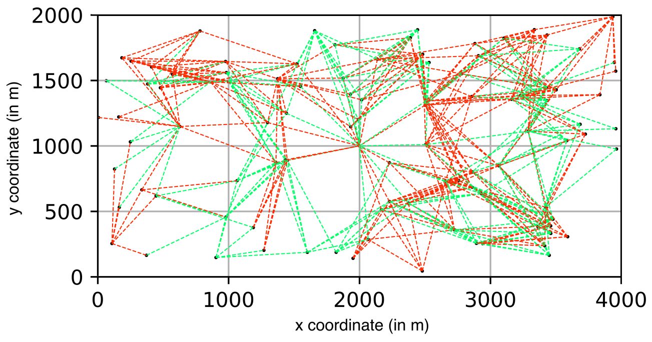


Fig. 2. Network topology with speeds

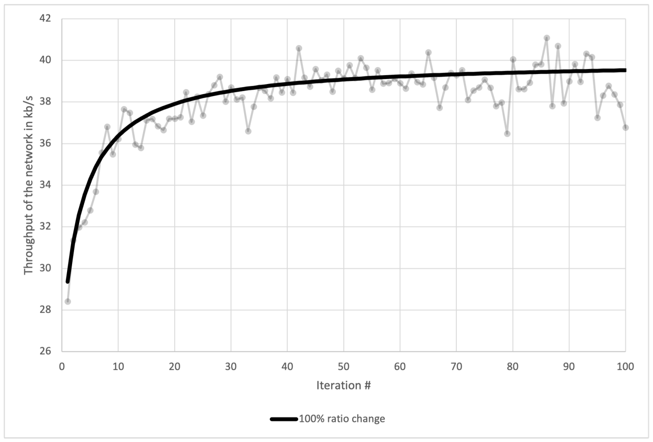


Fig. 3. Throughput of the network over time

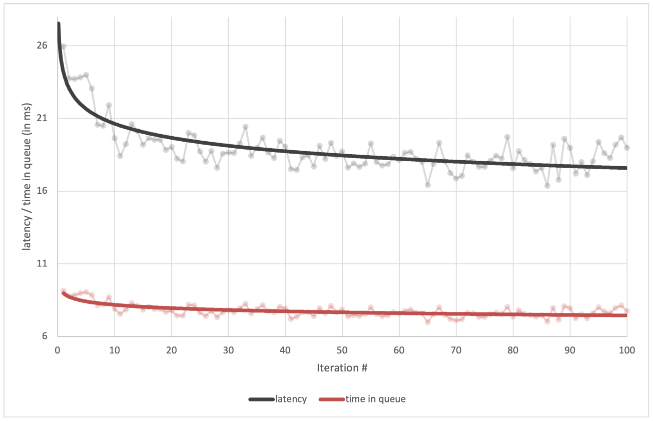


Fig. 4. Average latency of the network over time

6.4 Simulations

During the simulation each node produces packets of 500 bits at a 1 second interval. The network consists of 100 nodes and one gateway, meaning that the total load on the network is 100 * 500 bits * 1 packet/s = 50kb/s. This means that if the network would only utilize the slow links then the throughput of the network would be 10kb/s which is only 20% of the applied load. If only the fast links would be used this would result in a throughput of 150kb/s which is 300% of the applied load. In practice however it is very unlikely that all nodes have a path to the gateway that consists of only fast links, meaning that even if the network would find a perfect optimum some slow links would still be utilized.
The topology of the network with the links can be seen in figure 2. Here the green links represent (fast) links with a speed of 150kb/s and the red links represent (slow) links with a speed of 10kb/s.

7 RESULTS

For the results 5 different change rates were chosen to see what the effect of the change rates would be. These 5 cases were run for 100 iterations each and several metrics of each iteration were recorded, in particular the arrival rate of packets which can be used to deduct the throughput and the average latency. Also for each iteration a visual representation of the network was saved with a color based indication of the link utilization

7.1 Throughput

As can be seen in figure 3, the system seems to be able to achieve a significant performance gain, going from a throughput 28,4kb/s to surpassing and hovering around 40kb/s after around 45 iterations when leaving the allowed amount of change per iteration uncapped. The max throughput that the network manages to achieve is 41kb/s of the applied network load, which is a 45% increase compared to the throughput at the start. This shows that using feedback from a central gateway to adjust forward ratios, does allow a network to reach a more efficient state than it started at. The achieved throughput is, however, still not even close to the theoretical maximum throughput of the network.

7.2 Latency

The latency graph that can be seen in figure 4 shows a similar result; there is a significant improvement over time, but the system does manage to reach results near the theoretical minimum. With the average hop count of the system being 2 the theoretical minimum latency would be 2*(0.5/150)=6.67 ms (average hop count * (packet size / link speed) + average hop count * time in queue). The average time that messages spend in the queue is also recorded and is also shown in figure 4. There is only about a 10% decrease in time spend in queue, which shows that there is not much less congestion. This means that most of the decrease in latency must be coming from the decrease in transmission delay due to the network favoring fast connections over slow connections. The fact that the average amount of time that messages spend in the queue did not decrease much does not necessarily point to the system failing to decrease congestion. It could be caused by the fact that there is not much congestion to begin with. At the start there is an average time in queue of 10 ms, which would mean that there are on average only 3 packets before it in the queue if the node has a fast connection and only an average of 0.2 packets before it in the queue if the

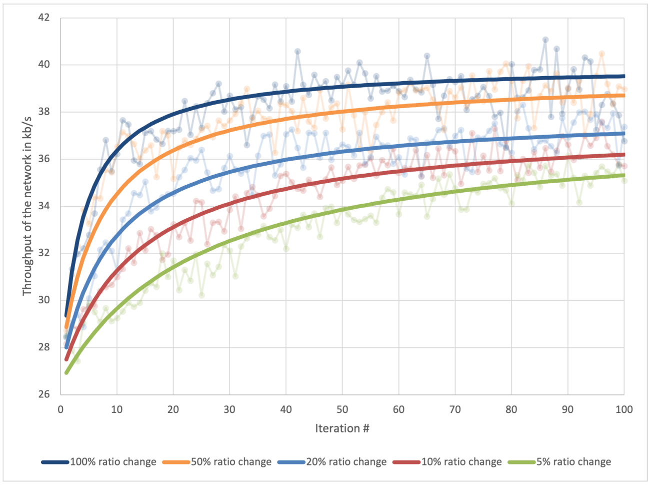


Fig. 5. Throughput over time with limited amount of change per iteration

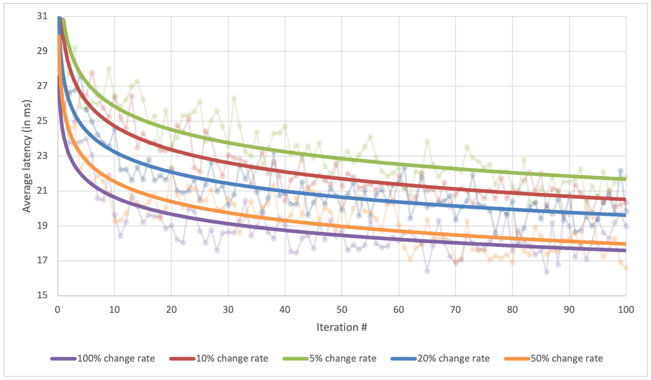


Fig. 6. Average latency over time with limited amount of change per iteration

node has a slow connection. There did occur some congestion at the neighbors of the sink, since those nodes are the once responsible for the most amount data transfer. This did not clear up as the network progresses since the network becoming more efficient causes these neighbors of the sink to get an even higher load. In the rest of the network there is close to no congestion. With a max buffer size of 20 packets and only the (17) neighbors of the sink being congested it makes sense that there are around 3 packets before each packet in the queue on average.

7.3 Network Convergence

Against the expectations, leaving the allowed amount of change uncapped did not cause the network to oscillate and not converge. It actually results in the network converging much faster, which leads it to achieve a way higher increase in efficiency at the end of the 100 iterations. As can be seen in figure 5 and figure 6 capping the amount of percentage change that nodes are allowed to make to their forwarding ratios, only negatively impacts the speed at which the network converges.

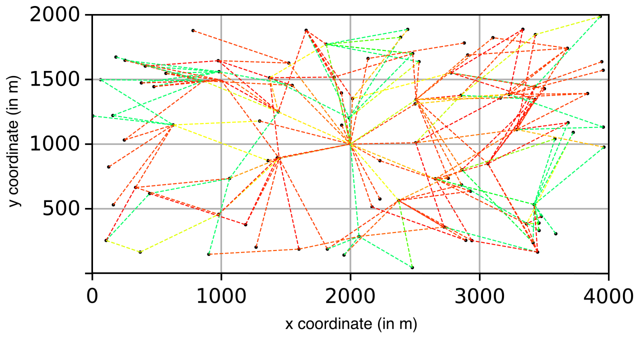


Fig. 7. Link utilization in the final iteration

7.4 Link Utilization

Figure 7 shows the link utilization that the network has settled on at iteration 100 with an uncapped change amount per iteration. The figure only includes the links that are actively used by the network, so links that are not used at all are not displayed. The color coding is used to indicate how much connections are used: red means they are used very little (in most cases only for the generated messages of a single node but not for any forwarding of messages of other nodes) and green means that the link is used a lot. When comparing the link speeds (that can be seen in figure 2) and the link utilization (figure 7), there is a similarity apparent especially with regards to the links close to (directly connection to, or one hop away from) the sink. It can also be observed that nodes that have a slow link to the sink receive very little incoming packets (no incoming links or red incoming links), whereas nodes with a fast link appear to be receiving a lot more data from their neighbors to forward to the sink. This shows that the system is able to select nodes with fast links for sending, even without any node knowing anything about connection speeds.

At the start links far from the sink will likely only have neighbors with a less then 100% success rate and thus are not allowed to adjust their send ratios. Only after the nodes close to the sink have optimized enough to handle more data transfer, there become paths that are still able to handle more data allowing nodes further away from the sink to optimize as well by choosing these paths. This results in the system exhibiting expanding behaviour where first the nodes close to the sink optimize and over time nodes further away from the sink also get the opportunity to optimize. This behaviour is only present if the network is slightly overloaded with data. If the data load would be low, there could already be successful path from the edge of the network to sink at the start meaning that even nodes close to the edge can already optimize and use this "successful" path. In this case it might be beneficial to assign limits to the amount of change that nodes further away from the sink can make to their routing ratios. This would help the nodes close to the sink to optimize first without having rapidly changing incoming data rates. The nodes close to the sink should get this priority since they are responsible for relaying the most amount of data and should therefore optimize as fast as possible.

The other way around, if the network would be overloaded with too much data then nodes close to the edge of the network would never be able to optimize, since the nodes close to the sink simply cannot handle all the data, even when optimized, resulting in no "successful" path from the sink to the edge of the network. Leaving nodes at the edge of the network with only neighbors with a less then 100% success rate and thus not being able to optimize.

7.5 Impact of Individual Nodes

The impact that an individual node has on the throughput of the network strongly depends on how much data from other nodes a particular node has to relay. When the network has not optimized yet this impact is strongly related to the hop count, since nodes close to the sink receive data from other nodes further away from the sink. As the network converges slow nodes get less data send to them and fast nodes get more data. This results in fast nodes that are close to sink being responsible for even more data relaying, therefore making their routing decision very important for the throughput of the network. Adversely, slow nodes close to the sink receive less data, meaning that their routing decisions do not impact the throughput of the network much. Due to this distribution of impact, it is expected to be desirable to first have the nodes close to the sink optimize and expand outwards from there (as discussed in the previous section).

8 DISCUSSION

8.1 Performance comparison

The fact that the system does not achieve a throughput even close to the theoretical maximum, together with the large amount of packets the system needs to reach close to its achievable max throughput (at around iteration 45), makes it very unlikely that this system could outperform other already existing systems. Douglas et al. [1] proposes a system where the expected transmission count is used to improve performance and showed significantly better performance in comparison to a minimal hop approach which was used in this research. The system proposed by Douglas et al. [1] showed a more than 2 times increase in throughput, instead of the 45% increase achieved with the feedback system, with much less data traffic needed to achieve these results. This however, does not necessarily mean that the feedback mechanism is flawed, it could be that the ETX metric is a better metric to optimize on than arrival rate. Adjusting the proposed system to use the more promising ETX metric instead of arrival rate feedback, could yield improved results.

8.2 Transmission Delay

In the current system only arrival success rate is taken into account and not the transmission time. This does result in some performance gain but it does not allow the system to chose a fast link over a slow link if neither are dropping packets, even though this might boost network throughput since this would leave more time for other node to send. A possible way to incorporate this would be to also take the cumulative transmission delay of packets into account and trying to minimize that. This would result in faster connections being favoured over slow connections even if the slower connections are not dropping message due to being overloaded.

8.3 Longer Paths Might Be Better

The current system only has the option to send to nodes with a lower hop count than the node has itself. This means that with each hop the message should get closer to the sink, resulting in a message always taking one of the paths with the lowest amount of hops. It could be that there exist a path that takes more hops, but would still be faster than any path with the minimum amount of hops, for instance if the path with more hops has a lot more fast link than the other paths. When exploring the impact of allowing other paths than the ones with the minimal amount of hops, it would also be interesting to see the impact of using the ETX metric to determine path quality as this showed promising results in other papers [1] [5]. Allowing nodes to also forward messages to neighbors with the same or a higher hop count would allow the system to find more optimal paths that are not necessarily the paths with the least amount of hops. This, however, would likely take longer to converge and have a lower throughput at the start since packets are more likely to be bounced around or get lost in the network, especially during the first iterations, but it would likely ultimately converge to a state where the throughput of the network would be higher than that of the current system.

8.4 Decreased Congestion

As stated in the results section the system did not manage to decrease the time that message spend in the queue by much. This was likely caused by the fact that messages did not spend much time in the queue to begin with, so there was not much to optimize. This means that it is still quite unclear whether this approach would be good at mitigating congestion. This could be tested by increasing queue sizes allowing for more congestion to build up. This would have to be done in another simulation environment, since The One becomes to slow when increasing queue sizes.

9 CONCLUSION

As can be concluded from the results, providing nodes with feedback about how many message have arrived from that node and via which hop they went, does allow the individual nodes to adjust their forwarding ratios to improve the overall throughput of the network significantly. However, the system does not seem to be capable of finding a configuration that performs close to the theoretical maximum throughput using the limited amount of information obtained from the feedback of the gateway. This means that the proposed system provides only rather limited optimization capabilities, with regards to network throughput.

With regards to the latency of the network the conclusion is not more optimistic, the system is again able to provide some improvement, but not reach even close to the theoretical minimum.

Regarding the amount of allowed change per iteration, it can be concluded that a higher amount of allowed change results in the network converging quicker, without the expected oscillatory behaviour. All in all, providing a small amount of information regarding the arrival rate of packets at the gateway, allows individual nodes to adjust their forwarding ratios and all together achieve a significant increase in throughput of 45%. Even though the system does not manage to achieve close to the theoretical throughput of the network, the simplistic nature of the behaviour of individual nodes, allows the system to be implemented on nodes with very limited processing power. This could mean that the system could be used in networks where other options may not be possible, providing a significant increase in throughput. Furthermore, extra research regarding the combination of this research and other optimization strategies[1][5][6], might lead to better results.

10 ACKNOWLEDGEMENTS

10.1 AI acknowledgement

During the development of the python wrapper copilot [2] was used to facilitate faster development. It has only been used to complete single lines, working as a more advanced auto complete than the one build in to the python language extension of Visual Studio Code [3]. Copilot was never used to generate more than one line of code at a time.

10.2 Supervisor

The research was supervised by Alessandro Chiumento, who provided helpful constructive feedback and great guidance.

REFERENCES

[1] De Couto, Douglas S. J. (Douglas Seraphim James), and 1975. 2004. *High-throughput routing for multi-hop wireless networks.* Thesis. Massachusetts Institute of Technology. https://dspace.mit.edu/handle/1721.1/16695 Accepted: 2005-05-17T14:57:49Z ISSN: 5737-7502.

[2] Github Inc. 2024. Github Copilot. https://github.com/features/copilot.

[3] Microsoft Inc. 2015. Visual Studio Code. https://code.visualstudio.com/.

[4] A. Keränen. 2015. the-one. https://github.com/akeranen/the-one.

[5] Shuang Li, Alvin Lim, Santosh Kulkarni, and Cong Liu. 2007. EDGE: A Routing Algorithm for Maximizing Throughput and Minimizing Delay in Wireless Sensor Networks. In *MILCOM 2007 - IEEE Military Communications Conference.* 1–7. https://doi.org/10.1109/MILCOM.2007.4454781

[6] P. Marbach. 2007. Distributed Scheduling and Active Queue Management in Wireless Networks. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications.* 2321–2325. https://doi.org/10.1109/INFCOM.2007.273

[7] Thirumalai Padmapriya and Vaitilingam Saminadan. 2015. Improving throughput for downlink multi user MIMO-LTE advanced networks using SINR approximation and hierarchical CSI feedback. *International Journal of Mobile Network Design and Innovation* 6, 1 (2015), 14–23. https://doi.org/10.1504/IJMNDI.2015.069213 arXiv:https://www.inderscienceonline.com/doi/pdf/10.1504/IJMNDI.2015.069213

[8] B.S.M. Viegen. 2024. Feedback aided distributed routing based throughput optimization. https://gitlab.utwente.nl/s2737779/feedback-aided-flow-control.

[9] Ting Yang, Jiabao Sun, and Amin Mohajer. 2024. Queue stability and dynamic throughput maximization in multi-agent heterogeneous wireless networks. *Wireless Networks* (April 2024).