Performance Evaluation of Gossip Protocol in Peer-to-Peer Mesh Networks

Tingzhi Li, Marco Falke, and Jinming Mu

***Abstract*—In the real world, the actual network infrastructure can be dynamic. With the help of network virtualization, we are able to perform a remote sensing task regardless of physical network changes. For a given network, if we tried to perform any task, the first crucial step is to spread the task requirement in the network in an efficient manner. In short, a task message needs to be broadcasted quickly. One way to do so is using gossip protocol; meaning for each node in the network, they randomly "gossip" the task message to neighbors. In this paper, we extened the ICMP to support gossip protocol control message, developed gossip protocol inside ns-3, and measured the performance of gossip protocol using random generated topology files which can be classified as peer-to-peer mesh networks.**

***Index Terms*—gossip protocol, IoT, virtual sensor networks, distributed protocol, mesh networks.**

I. Introduction

Along with the development of the information technology, the price of the broadband connectivity becomes affordable. Devices are trending to be smaller and more powerful. People start to explore ways to connect devices to the network for better control and monitor the status of them. Devices such as smartphone, smartwatch, and smart thermostats are able to connect to a network and these devices could communicate with each other. If these devices are connected to the Internet as well, we call them the *Internet of Things*. There is no doubt that IoT is an innovative paradigm [1] because this idea combined the Internet with our everyday gadgets. Either from the perspective of private user or from the perspective of business user, there are infinite possible ways to exploit IoT. As of today, research in the area of IoT is emerging rapidly and many open questions remain to be answered.

Remote sensing is one of the possible services of IoT. Remote sensing allows users to get the data from the device through the network instead of physically read the data. Remote sensing involves the search and selection of IoT devices to form a virtual sensor network. Subsequently, the selected devices estimate the wanted property collaboratively and report the result to the remote cloud agent. IoT assigns a unique MAC address to physical object which do not have an identification, so people could track the specific objects status and control it remotely. The advantage of remote control is that it requires less manpower needed to monitor and manage the object. And it could further increase devices usage rate.

However, the main challenge here is that IoT devices are often resource limited meaning they have limited bandwidth and power, and restricted by their mobility. Due to the characteristics of IoT devices, topology of physical network formed from IoT is dynamic. The distributed network protocol which virtualizing the physical network has to deal with this dynamic nature of IoT networks.Those dynamic networks require a robust way to spread information across the whole network while minimizing the burden placed on the network. An efficient protocol is the gossip protocol [2], which spreads information similar to how an infection spreads in a population.

II. Problem Statement

The goal of our project is to evaluate the gossip protocol in the simulation tool ns-3. First, we need to customize the gossip protocol control messages based on ICMP. Second, we need to develop the gossip protocol and implement it in ns-3. Finanlly, we are supposed to evaluate the performance of the gossip protocol in a mesh network of peer-to-peer connected IoT devices by running simulations in ns-3. In the mesh network, each node usually has multiple edges and we assume that there is no isolated node in the network.

There are three essential performance metrics we would like to measure.

- Average number of control messages sent per node

- Maximum amount of hops needed to spread the message
- Maximum amount of time needed until the message is spread

The average number of control messages sent per node could help engineers to understand the average load of the nodes. If the number of messages send per node is quite large, then further improvement of protocol need to be developed to reduce the load in each node.

The maximum amount of time needed to spread the message could be used to evaluate the time complexity of gossip protocol. The correlation between the maximum amount of hops and spread time is very important to characterize this protocol. If they are strongly correlated, then we probably would observe that as the maximum amount of hops increases, the spread time will increase proportionally. Under that circumstance, new mechanism need to be implemented to reduce the total number of hops for each node to get the message.

In order to obtain meaningful data, our performance testing plan is to run simulation under different number of nodes cases. And for each case, we would use 100 random generated topology files to further assure the randomness of the network topology. In this project, we hope to verify that the time complexity of the gossip protocol is $O(\log N)$, where N is the number of nodes.

III. Related Work

In this section, we introduce the topic of Virtual Networks formed using a subset of a large group of IoT devices. Relevant papers covering the topic of Virtual Networks are summarized. In addition, one paper about WiFi Direct technology is summarized as well.

A. Virtual Networks

The following sections will present an overview over Virtual Sensor Networks, as well as network virtualization using the existing internet.

1) Virtual Sensor Networks: The ongoing technological progress further and further improves the computation, connectivity and sensing capabilities of various devices, sometimes mobile ones. [3] This enables a huge variety of opportunities in sensor networks. For example, devices in a sensor network could be assigned tasks based on their constraints in computation, power usage or networking potential. In contrast to dedicated sensor networks, where the participating nodes serve a single application, Virtual Sensor Networks (VSN) take advantage of the nodes technological progress. When a VSN is formed on top of a Wireless Sensor Network, only a subset of all available nodes is part in the VSN. Furthermore, several VSNs can exist simultaneously in on Wireless Sensor Network. [3] That is, one subset of the nodes forms a VSN and relies on the remaining nodes to communicate between its nodes. In some cases, physical nodes of one VSN even could be completely cut off from communication due to their spatial distribution and must rely on the other nodes. Usually the different VSNs pursue completely unrelated sensing tasks and the nodes in each VSN behave like they are on their independent Sensor Network. Figure 1 based on [3] depicts a visualization of two VSNs formed on top of an Wireless Sensor Network. This logical separation helps to simplify the implementation of applications significantly. [3] Further advantages of VSNs are enhanced performance and better scalability.

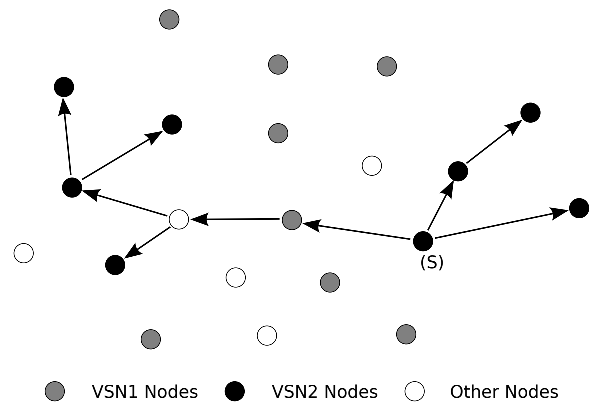

Fig. 1. Broadcast path from a node (S) in VSN2.

The development of algorithms and protocols to support the grouping of VSNs on top of Sensor Networks, is still an ongoing research topic. Those need to consider how the available time and frequencies should be fairly distributed for intra network communication. Moreover, it should be possible for nodes to change their membership in VSNs.

2) Virtual Networks on Top of the Internet: It is important to realize that the Internet, due to so many different participants with sometimes opposing interests, is hard to modify and only possible small and slow steps, if at all. Therefore, Virtual Networks are often the only way to realize innovation. To implement a Virtual Network using the existing Internet, several things need to be considered. First, the characteristics of the networking technology determine the attributes of the Virtual Network. For instance, a wired network yields a more scalable and bandwidth flexible Virtual Network than a wireless network would do. [4] Second, the layer of virtualization (referring to the OSI layer model) impacts the flexibility of the Virtual Network. That is, the lower the layer of virtualization, the more flexibility will be possible. Specifically, so-called overlay networks, mostly realized in the application layer, are limited in their ability to support fundamentally different architectures. [4] Moreover, virtualization on top of IP is fixed to the network layer protocol and cannot deploy IP independent mechanisms. [4] Lastly, an important consideration in the non-comprehensive list is also about security and privacy in virtual networks. Thus, attack vectors such as denial-of-service or distributed denial-of-service against the underlying physical network will have impact on all simultaneously virtualized networks.

3) Virtualization Algorithm: Though, it is possible to form a VSN of mobile IoT devices by having access to all relevant data such as availability, sensor capabilities or sensor mobility, a more efficient solution is to assume the managing cloud agent does not have full knowledge of every sensors properties. [5] The cloud agent even may not be connected to all nodes but only to a subgroup of them. The presented algorithm also takes into account mobility of the devices which sometimes leads to nodes being unavailable for some time. [5] This virtualization algorithm will search and select appropriate sensors from the whole network to form the virtual network which then executes the sensing task.

B. WiFi Direct

The goal of WiFi Direct technology is to improve direct device to device communications in Wi-Fi [6]. The main strength of this technology is that it doesnt require the presence of an Access Point [6]. Actually direct device to device connection was already possible by the ad-hoc mode. But due to the power consumption issue, it is not popular [6]. Instead WiFi Direct uses WiFi infrastructure mode [6]. It let devices negotiate who will be the Access Point in this network [6]. By doing so, it obtains all the enhanced QoS, power saving, and security mechanisms originally developed for the WiFi infrastructure mode [6]. After first establish the network, new devices could connect to this network just like connecting to an Access Point [6].

IV. Solution

To achieve the goals outlined in section II and implement them in ns-3, we took three vital steps.

- Extend the Internet Control Message Protocol (ICMP) to support transmitting three simple control messages needed for our protocol.
- Develop a new application to be installed on network nodes. This application implements the gossip protocol.
- Import nodes information from a given topology file and export simulation results for performance evaluation.

A. ICMP Extension

ICMP stands for Internet Control Message Protocol. The most common use of ICMP is for error reporting [7]. A ICMP message contains two parts: 8-byte header and data section. The first 4 btyes of the header have fixed format. However, the last 4 bytes vary and depend on the type or code of the ICMP packet [8]. The first and second byte of the header is the type field and code field respectively. And the third and fourth byte are checksum field. The format of the header is shown in table I.

TABLE I
ICMP Header Structure

Octet	0	1	2	3
	Type	Code	Checksum	
Octet	4	5	6	7
	Rest of Header			

Table II here presented some selected ICMP message types.

TABLE II
Control Messages

Type	Code	Description
0	0	Echo reply
8	0	Echo request
9	0	Router Advertisement
10	0	Router discovery/selection/solicitation
42 to 255		Reserved

As we could see in table II, type 42 to 255 are reserved for further development. So we decided to extend ICMP by defining type 42, 43, and 44 to represent ackownlegement, request, and data respectively. The detail is shown in table III.

TABLE III
GOSSIP PROTOCOL CONTROL MESSAGES

Type	Code	Description
42	0	Send Acknowledgment
43	0	Send Request
44	0	Send Data

Upon these new control message types, we could further develop gossip protocol in ns-3.

B. Gossip Protocol

Gossip protocol is a computer communication protocol which inspired by the social activity – gossip. This protocol accomplishes to synchronize a message in a system that does not need real-time synchronization. It provides $O(\log n)$ time to synchronize the message to the network, where n means the number of nodes in the network. There are three packet types for the message protocol: Data packet, ACK packet and request packet. The data packet contains the message from a certain source node that wants to spread this message to the whole network. The ACK packet is a packet to notice the node that the destination already contains the message. The request packet is the packet that request the message from destination node. There are two states for each node, running and stop.

The pseudo code of gossip algorithm is given in figure 2.

The explanation of the pseudo code:

All the node in network have same behavior and they are all independent. When we want to spread a message x to the whole network, we assign the x to message and run this algorithm. Node 1 start with running state, and node 1 has a message x, so it will find a random neighbor every five milliseconds and send a data packet which contain message x to its random neighbor. If node 1 receive a ACK packet it will go to stop state, otherwise it will keep doing these steps. Other nodes which do not have the message x will wait for packet, if a node 1 send a data packet to the node 2 which do not have message x, node 2 will update the message in node 2 and not return anything to node 1. If node 2 do not have message 2, it will send a request packet to a random neighbor every 5 second. If node 2 receives a data packet after sending the request packet, it will update its message to x. The nodes in stop state would always waiting for other node send message to it and it will send data packet or ACK packet back depends on what packet it receive.

```
switch(state):
  case running:
    if message is not null:
      every 5 milliseconds:
        find a random neighbor R
        send data packet to R
      if receive a packet:
        if packet is ACK:
          state <- stop
        if packet is data:
          send ACK to packet source
    else:
      if receive a packet:
        if packet is a data packet:
          update the message
      every 5 seconds:
        find a random neighbor R
        send a request packet to R
        if receive a packet:
          if packet is a data packet:
            update the message to data

  case stop:
    if receive a packet:
      if packet is request:
        send data to the source node
      if packet is data:
        send ACK to the source node
```

Fig. 2. The pseudo code of the gossip algorithm.

C. Gathering Simulation Data

To evaluate the performance of the gossip protocol, we use several randomly generated topology files with the number of nodes as variable. Those topology files are derived from a random geometric graph network, which was created by uniformly and randomly placing nodes into a space and then connect nodes whose distance is smaller than some given radius.

Each topology file contains the number of nodes as well as all the edges, which are connections between nodes. As an example, the content of a simple topology file is shown in figure 3. We assumed in our work, that each nodes is connected

```
#Nodes
0
1
2
#Edges
(0, 1)
(1, 2)
```

Fig. 3. A topology file of a linear topology with three nodes.

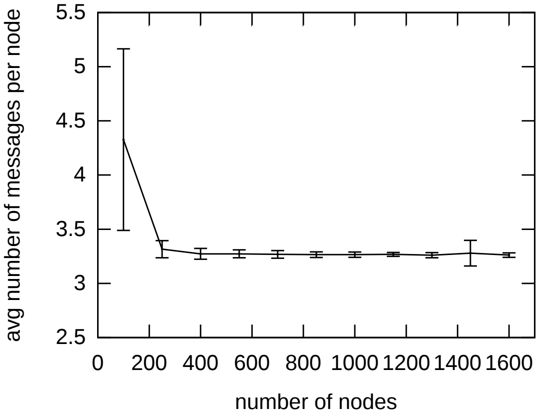

Fig. 4. The average number of messages each node sent over the duration of a whole simulation.

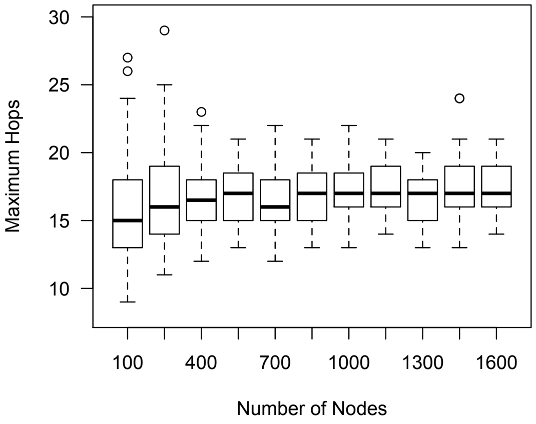

Fig. 5. The maximum amount of hops the message experienced over the duration of a whole simulation.

at least once to the network. We wrote a parser in C++ to create the given number of nodes in ns-3 and installed the gossip protocol application on them. Thereafter, all edges are parsed and created accordingly. Each node holds an ns-3 Ipv4Interface for every connection and stores the address of his interface and the one of his neighbors interface.

For our simulation, we set the link rate for all connections to 100 Mbps and the link delay to 2 ms. Also, all nodes are instructed to execute the gossip process of sending out data periodically every 5 ms. The interval of requesting new data is set to 5 s.

To allow the performance analysis, all nodes count the number of ICMP messages they sent. Also, they track how many hops the data message experienced before reaching them and they record the time when they received the data message. For one single simulation, we collect the information from the nodes and determine the average amount of ICMP messages sent. Moreover, the highest number of reported hops is saved, as well as how long it took for the message to reach the "last" node.

This information is determined and stored for each of the several hundred topology files. It should be noted that due to time limitations each topology file was only simulated once. Finally, we did statistical analysis upon those collected data in the hope of verifying the assumptions we made in section II.

V. Performance Evaluation

The random generated topology files are the input of our simulations. There are 11 different cases where the number of nodes vary from 100 to 1600 , increasing with 150 nodes step. And for each case, we sampled 100 topology files to run the simulations.

First, we will have a look at the average number of ICMP messages each node sent, depicted in fig. 4. It is important to note that the the collected average number for one topology file was again averaged over all reported values produced by topology files with the same number of nodes. Thus, the error bar is an indicator how consistent the average number is. It can be deduced from fig. 4 that this value is a constant roughly equal to 3.3. Only when a very litte number of nodes is observed, the mean and variance of the average amount of sent messages increases.

Second, the maximum number of hops a message experiences is analyzed. Figure 5 shows that with a growing number of nodes, the average maximum hops slightly increases as well. But the standard deviation has the tendency to decrease which is a positive sign since we hope the gossip protocol performance metrics would converge as the network grows. Nonetheless, the overall impression is that the number of hops is generally speaking, more or less constant and mostly is found to range from 15 to 20.

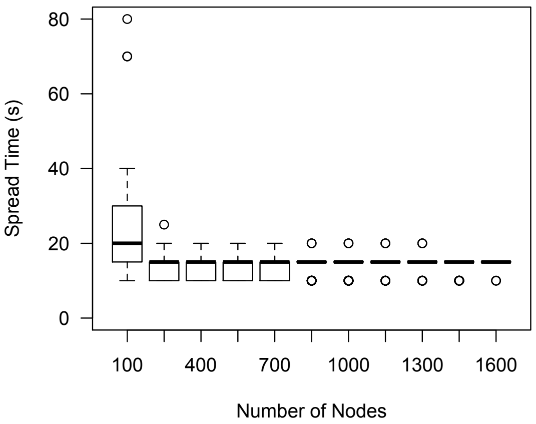

Fig. 6. The spread time under different number of nodes cases

Moreover, figure 6 illustrates the time needed to spread the message across the whole network. Again, the mean is found to be more or less constant and slightly less than 15 s. Due to the huge difference in the gossip-interval-time (5 ms) and solicit-interval-time (5 s), only the influence of the solicit-interval can be deduced from the results. One can see, that the time needed to spread the message is fluctuating due to the random nature of the gossip protocol, especially for the case of a number of nodes of 100, where the standard deviation is the largest.

We assume that the reason that the results being not a function of the number of the nodes but rather seem more or less constant, is due to the increasing edge density with higher numbers of nodes. As seen in fig. 7, the number of edges is strictly linear depending only on the number of nodes in the topology. We also consider this as the reason we could not verify the logarithmic complexity suggested by [2].

Another interesting observation is that accross these statistical analysis, the case of 100 nodes always has different behavior than other cases. We wondered if that has anything to do with not eough experiment trials. So we run each topology file of 100 nodes case one time, which is defalut in our performance evaluation setting, and 50 times just to see if the standard deviation would decrease as number of experiment trails increase. As we can see in fig. 8 and fig. 9. There is no sign that in this case, the standard deviation would decrease as the number of experiment trials increases.

Section VII proposes further work which can be

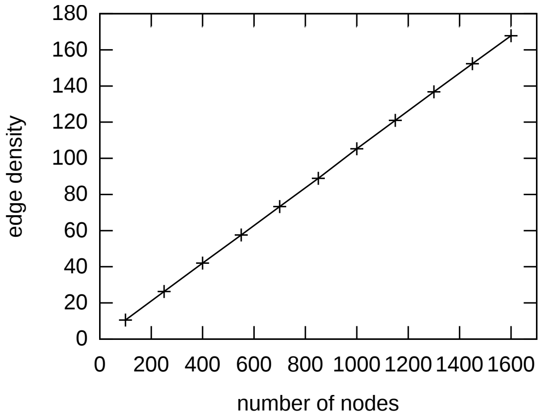

Fig. 7. The density of the nodes for each batch of topology files.

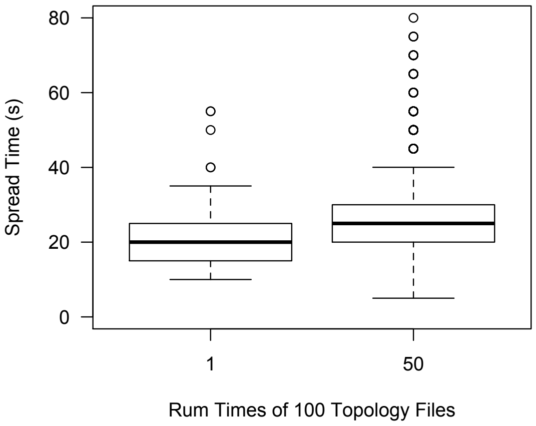

Fig. 8. Spread Time Under 1 and 50 Run Times for 100 Node Cases

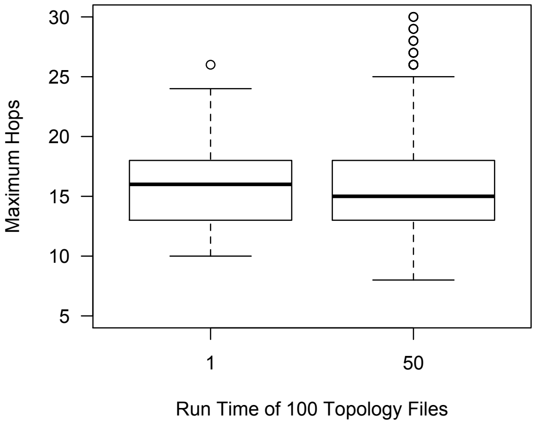

Fig. 9. Maximum Hops Under 1 and 50 Run Times for 100 Node Cases

done to gain a more thorough evaluation.

VI. Conclusion

We introduced the topics of Internet of things and sensor networks. The opportunities resulting by virtualization of those sensor networks have been elaborated. A distributed algorithm to spread data across a mesh-network has been implemented. We proposed several performance measures to evaluate this algorithm.

We have implemented the gossip protocol and run the simulation in ns-3. The result from ns-3 shows that the number of messages a node expects to send is not a function of the total number of nodes in the network. Further, we see that the maximum number of hops is slightly increasing for an increasing number of nodes. And lastly, our results imply that the time until all nodes have the data, is mostly only depended on how the solicit-interval is chosen.Unfortunately, we could not verify that the time complexity of the algorithm is $O(\log n)$ as outlined in [2]. Possible causes to consider are discussed in the next section.

VII. Further Work

In further work, various alterations to the gossip protocol can be covered and examined. For example, a node can improve routing (or decrease the number of hops) by notifying neighbors if there exists a shorter route to the owner of a message. To accomplish this, they need to keep track of the number of hops and then compare if the owner of a received message is one of their immediate neighbors. In case this is true and they will notify the source of the message, that the *true* (or ideal) number of hops would be 1.

Another option for further work, would be to change the parameters to create the topology file. As demonstrated in fig. 7, the number of edges per node is not a constant. We believe that making this value constant, will yield different results which possibly verify the logarithmic complexity suggested by [2].

And lastly, the effect of reducing the solicit interval should be investigated. Clearly, having a faster rate to request new data will add an overhead to the network. Further work may determine an appropriate interval for the solicit interval.

Due to time limitations we could not address the huge spectrum of possible analysis. Further work is needed to examine the effects when the wired peer-to-peer links between nodes are replaced by wireless ones. Power limitations and mobility of the nodes should be included in such a scenario. Thus, the list of neighbors for each node changes over time and a topology file is not needed anymore. In the wireless case, Wi-Fi Direct can be used to connect devices with each other as well.

Acknowledgment

We would like to thank Dr. Bechir Hamdaoui for his advising. We also would like to thank Sherif Abelwahab for providing useful information and answers to our questions in numerous discussions. In addition, our team would like to thank Kendall Bailey for her suggestion about automating simulation process.

References

[1] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey,” *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] K. Jenkins, K. Hopkinson, and K. Birman, “A gossip protocol for subgroup multicast,” in *Distributed Computing Systems Workshop, 2001 International Conference on*, pp. 25–30, IEEE, 2001.

[3] A. P. Jayasumana, Q. Han, and T. H. Illangasekare, “Virtual sensor networks-a resource efficient approach for concurrent applications,” in *Information Technology, 2007. ITNG’07. Fourth International Conference on*, pp. 111–115, IEEE, 2007.

[4] N. M. K. Chowdhury and R. Boutaba, “A survey of network virtualization,” *Computer Networks*, vol. 54, no. 5, pp. 862–876, 2010.

[5] S. Abdelwahab, B. Hamdaoui, and M. Guizani, “Cloud-assisted remote sensor network virtualization for distributed consensus estimation,” *arXiv preprint arXiv:1501.03547*, 2015.

[6] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano, “Device-to-device communications with wi-fi direct: overview and experimentation,” *Wireless Communications, IEEE*, vol. 20, no. 3, pp. 96–104, 2013.

[7] F. K. James and W. R. Keith, *Computer networking a top-down approach featuring the internet*. Addison-Wesley, Reading, 2004.

[8] A. B. Forouzan, *Data Communications & Networking (sie)*. Tata McGraw-Hill Education, 2006.

Appendix A
Division of Responsibilities

Our team of three members had different responsibilities in fullfilling the project’s goal. All of us did reasearch in the field of IoT, Sensor Networks and distributed algorithms. Each of us wrote different sections of the paper and helped improving all other sections by proof-reading.

The two graduate students – Marco Falke and Tingzhi Li – wrote the section about related work.

Marco Falke wrote about Virtual Networks and Tingzhi Li covered WiFi Direct.

The different coding parts were assigned as follows. Jinming Mu did the initial setup of ns-3 and coded the topology creationg part of the code. Marco Falke had the responsibility to implement the Gossip protocol in form of a ns-3 application. Furthermore, he wrote the topology file parser to read a given topology file. Lastly, he coded the connection between the ICMP level of ns-3 and application level. Tingzhi Li did modifications to the ICMP part of ns-3 to allow transmission of modified control messages. Moreover, he wrote code to process the produced data and create plots.