OREGON STATE UNIVERSITY

Energy-Aware Gossip Techniques for Wireless Broadcasting

by

Tingzhi Li

A thesis submitted in partial fulfillment for the degree of Master of Science

in the
Bechir Hamdaoui
School of Electrical Engineering and Computer Science

November 2016

Declaration of Authorship

I, TINGZHI LI, declare that this thesis titled, ‘Energy-Aware Gossip Techniques for Wireless Broadcasting’ and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Imagination is more important than knowledge. Knowledge is limited. Imagination encircles the world. "

Albert Einstein

OREGON STATE UNIVERSITY

Abstract

Bechir Hamdaoui
School of Electrical Engineering and Computer Science

Master of Science

by Tingzhi Li

The current state of research on gossip techniques for wireless broadcasting is very limited because past research efforts have mostly focused on using gossip techniques for multicast communication. On the other hand, those research efforts that have focused on using gossip techniques for wireless broadcast communications ignore energy efficiency and network lifetime. With the emergence of Internet of Things (IoT) devices, known with their limited energy and processing resource capabilities, energy consumption is becoming more and more important to account for when designing wireless broadcasting protocols. In this thesis, we propose a new energy-aware broadcasting protocol for wireless adhoc networks. Specifically, the proposed protocol dynamically adapts the fanout parameter based on wireless nodes' remaining energy to prolong the lifetime of the network. Our simulation results show that our proposed energy-aware gossip protocol outperforms existing approaches by achieving fast message broadcasting times while extending the nodes' battery lifetime.

Acknowledgements

I would like to thank Dr. Bechir Hamdaoui for his advising. We also would like to thank Sherif Abelwahab for providing great suggestion and answering to my questions in many discussions.

Contents

List of Figures

List of Tables

Abbreviations

LAH	**L**ist **A**bbreviations **H**ere

Physical Constants

Speed of Light	c	$=$	$2.997\ 924\ 58 \times 10^{8}$ ms$^{-\mathrm{S}}$ (exact)

Symbols

a	distance	m
P	power	W (Js^{-1})
ω	angular frequency	$rads^{-1}$

For/Dedicated to/To my…

Chapter 1

Introduction

There is no doubt that the *Internet of Things (IoT)* is an innovative paradigm, [1] which is gaining popularity in our modern society. With the development of information technology, digital devices are getting smaller and yet more powerful. The basic ideal of IoT is that with "unique addressing schemes", various of *things* or *objects* such as smart phones, watches, thermostats, Radio-Frequency IDentification (RFID) tags, sensors are able to communicate, cooperate with each other to perform tasks [1].

Remote sensing is one of the promising services of IoT. With remote sensing, the users could retrieve collected data through the network instead of physically retrieving data. Remote sensing involves the search and selection of IoT devices to form a virtual sensor network. Afterwards, sensing task is sent to the virtual sensor network. The selected devices then perform sensing collaboratively and report the result back to the remote cloud agent.

There is one step during remote sensing process that I am particularly interested in, which is sending tasks to the virtual sensor network. There are two aspects of this area. One question is that what kind of network would IoT devices form? The other question is that what kind of protocol is efficient and robust for broadcasting the sensing task? As we know, IoT devices often have limited bandwidth and power. Their mobility further impacted the topology of the selected virtual sensor network. Due to the characteristics of IoT devices, ad-hoc network is often used to communicate among devices. Flooding was a simple algorithm to broadcast messages in wired network but was prove to be unsuited for wireless environment due to excessive overhead, media contention, and packets collision. Gossip technique instead is often used to quickly broadcast messages with lower overhead. Gossip technique is inspired by the form of gossip seen in social network. In a network, a node with a new message would randomly pick another node and gossip the message. The other node then would do the same thing. This is refer to as classic gossip technique. A variation of that would only randomly pick a node that is its neighbor. But regardless of how a node pick aother node or choose where to pick from, gossip technique could broadcast a new message in a timely and robust manner.

Over the years, researcher have been focusing on how to improve gossip technique's reliability while lowering overhead. Many proposed gossip schemes have been proposed. Some porposed to apply gossip probability on nodes. The idea behind that is that a message could be broadcasted successfully without every nodes' participation. In this scheme, a lower overhead could be achieved because a portion of the nodes participated in gossipping but a message is still being broadcasted. Some proposed a event counter based scheme to combat this issue. The gist of that shceme is that if a node overheard the same messages a times and $a > b$ were b is the threshold, it would not gossip its latest message this time. Some even went a step further, they tried to identify the dependency among a node and its neighbors and dynamically adjust gossip probability based on collected information.

However, none of them focused directly on how to conserve energy while gossipping. Some might argure that the effort on lowering overhead is equivlent to conserving energy but (!!!need to think about this). As we are moving to an IoT and mobile devices dominated world, energy conservation become more and more important as to overall user experience or network survival time. In this thesis, I proposed a gossip technique that dynamically adjust gossip fan-out based on each node's remaining energy. The results showed that my proposed approach performed as well as constant gossip fan-out approach while still conserving significant amount of energy.

1.1 A Section

1.1.1 A Subsection

1.2 Another Section

Chapter 2

Related Work

(from 563 paper).

2.1 Virtual Sensor Networks

The ongoing technological progress further and further improves the computation, connectivity and sensing capabilities of various devices, sometimes mobile ones. [6] 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. [6] 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 **??** based on [6] 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. [6] Further advantages of VSNs are enhanced performance and better scalability.

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.

2.2 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. [7] 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. [7] Moreover, virtualization on top of IP is fixed to the network layer protocol and cannot deploy IP independent mechanisms. [7] 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.

2.3 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. [8] 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. [8] This virtualization algorithm will search and select appropriate sensors from the whole network to form the virtual network which then executes the sensing task.

The project goal is to evaluate gossip protocol performance in the wireless ad-hoc network since several papers[2][3] claimed that gossip protocol is an efficient, scalable, reliable message dissemination approach. I would like to implement gossip protocol in a wireless ad-hoc network in ns-3 and study its reliability, scalability and efficiency. Based on the progress of implementation gossip protocol in a wired peer-to-peer network in ns-3, my first step is to switch the network environment from wired peer-to-peer to wireless ad-hoc network. The wireless ad-hoc network setting are as following:

- WLAN Standard: IEEE 802.11b
- MAC layer: wifi ad-hoc mode
- Add non-QoS upper mac layer
- Modulation: DSSS
- Data Rate: 1Mbps
- RTS/CTS: On
- Receiver Gain: 0dB
- Delay Mode: Constant Speed Propagation Delay Mode
- Loss Mode: Friis Propagation Loss Mode
- Ipv4 address base: 10.1.0.0
- Ipv4 address netmask: 255.255.0.0

For the topologies that I used to evaluate gossip protocol, the number of nodes are 100, 250, 400, 550, 700, 850, and 1000. For each case, there are 100 random generated topology files defining the connectivity among those nodes. In the ad-hoc network, each node usually has multiple edges and we assume that there is no isolated node in the network. For the allocation of those nodes, I used random grip allocator in ns-3. The distance between two adjacent nodes is 5 meters. I assume that the connectivity remained regardless of the distance between two nodes. A simple example of 9 nodes random grip allocation would look like fig. **??**. Implementation of gossip protocol will be presented in section **??**.

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

- Average number of data packets sent per node
- Average hops per node needed to spread the message
- Maximum time needed until the message is spread

The average number of data packets sent per node is a key metrics that measure the efficieny of gossip protocol comparing to other popular multicast protocol like MAODV. It mostly emphysis on the efficiency or workload of sender's side. Theoretically, the average number of data packets sent per node would remain constant regradless of the scale of the network.

The average hops per node needed to spread the message is a metrics that indicates the efficiency on the receiver's side. It is a metrics that represents how many times the message is forwarded before the node received it. Generally, lower average hops per node is preferred.

The maximum time needed to spread the message could be used to evaluate the time complexity of gossip protocol. Baically, this metrics indicates how fast a message can be spred across the whole network.

In this project, randomness is shown in three different aspects: (1) network topologies are random generated. (2) The node that get the initial message is randomly chosen. (3) For each node during simulation, it randomly chooses neighbour to perform "gossipping."

After evaluation the performance of gossip protocol, we hope to verify the following assumptions.

- Time complexity of the gossip protocol is $O(\log N)$, where N is the number of nodes.
- Average number of data packets sent per node will remain constant regardless of network scale.

2.4 A Section

2.4.1 A Subsection

2.5 Another Section

Chapter 3

Energy-aware Gossip

The objective of gossip protocol is to broadcast messages in an efficient way by mimicing social activities when people spread rumors in office. It works as follows: when a node received a new message, it then will send it to multiple nodes in the network. The number of nodes it contacted is defined as the *fan-out* of gossip protocol. It is denoted as f. Each time when a node face the decision of sending a new message to another node, the probability of doing so is defined as p_{gossip}. In the rest of the thesis, I will refer to it as p_g. p_g is usually between 0 and 1. Once a node received a new message, the number of times it will contact other nodes is defined as *message live time*. It is denoted as T_l.

3.1 Classcial Gossip

For classic gossip protocol in wired network sceniro, *probability of gossip* is set to be 1 and *fan-out* is usually set to be 1 or 2. *message live time* could vary depending on the requirement.

3.1.1 How it works

3.1.2 All Parameters

3.1.3 Different Approach

3.2 Proposed Adaptive Gossip

3.2.1 Intuition

3.3 Key Intuition of Our Approach

In order to save each node's energy and potentially extend the lifespan of the whole network. My idea is to use each node's remaining energy fraction (much like the battery indicator on your phone) as the X for the function of one of gossip protocol's parameter fanout. The more energy a node has, the bigger the fanout will be for it. Thus, compare to constant fanout protocol, which does not take nodes' energy into account, our apprach adaptive fanout will dynamically adjust fanout based on a node's remaining energy.

Why not adjust gossip probability based on remaining energy fraction?

Why not adjust other gossip protocol parameters based on remaining energy fraction?

(from 563 paper).

3.4 From gossip course paper

To achieve the goals stated in section **??**, I took four crucial steps.

- Extend the Internet Control Message Protocol (ICMP) to support transmitting three simple control messages needed for gossip protocol.
- Develop the gossip protocol application to be installed on network nodes.
- Set wireless ad-hoc network attributes and gossip protocol attributes, which has already be presented in section **??**.
- Import nodes connectivity information from topology files and export simulation results for performance evaluation.

3.4.1 ICMP Extension

ICMP stands for Internet Control Message Protocol. The most common use of ICMP is for error reporting [4]. 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 [5]. 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 3.1.

Table 3.1: ICMP Header Structure

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

Table 3.2 here presented some selected ICMP message types.

Table 3.2: 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

Since type 42 to 255 are reserved for further development, I decided to extend ICMP by defining type 42, 43, and 44 to represent ackownlegement packet, request packet, and data packet respectively. The detail is shown in table 3.3.

Table 3.3: 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 extension, we could further develop gossip protocol in ns-3.

3.4.2 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 network

```
[fontsize=\small]
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
```

FIGURE 3.1: The pseudo code of the gossip algorithm.

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 as mentioned above. The data packet contains the message from a certain source node that is expected to be disseminated to the whole network. The ACK packet is used to acknowledge to source node that destination node already received message before. The request packet is used for source node to request for 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 3.1. All nodes in the network have same behavior and they are independent to each other. When user would like to spread a message x to the whole network, the protocol randomly assign the message x to a node and run this algorithm. For the discussion, let's assume node 1 got the initial message x. Node 1 start with running state, and node 1 has a message x, so it will find

```
[fontsize=\small]
#Nodes
0
1
2
#Edges
(0, 1)
(1, 2)
```

Figure 3.2: A topology file of a linear topology with three nodes.

a random neighbor every 0.005s 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 above 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 x 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 control messages to it and it will send data packet or ACK packet back depends on what kind of packets it receives.

3.4.3 Gathering Simulation Data

To evaluate the performance of 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 represents the connections between nodes. As an example, the content of a simple topology file is shown in figure 3.2. A parser written in C++ is developed 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 with assigned Ipv4 address and stores the Ipv4 addresses of his neighbors.

For the simulation, we set the link rate for all connections to be 1Mbp. Also, all nodes are instructed to execute the gossip process of sending out data periodically every 5ms. The interval of requesting new data is set to 5s.

To allow the performance analysis, all nodes count the number of data packets they sent. All nodes would track how many hops the data message experienced before reaching them and they record the time when they received the data message as well. For every single simulation, we collect the information from the nodes and determine the average amount of data packets sent per node and average number of hops per node. Moreover, the information about how long it took for the message to reach the "last" node is also recorded.

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 **??**.

3.4.4 A Subsection

3.5 Another Section

Chapter 4

Implementation

Topology wise, nodes are randomly place in a square area which is proportional to the number of nodes. The equation to calculate the size of the square is as follows:

(Insert the equation here).

Every node's coordinates are used to calculate neighbors list for each node. In practice, the global access of this information is usually not easy to obtain. Thus, Hello packets are used to compile neighbors list for each node.

Because nodes are randomly placed in a square area, with fixed WiFi range there could be a case that each node has at least one neighbor but the network is separated into two subnets unconnected. [twoSepNet.png]

Therefore, I used an algorithm that uses depth-first search algorithm to determine whether all nodes get visited during the recursive search. If the algorithm sucessfully traverse all nodes, it is considered a complete graph which means the network is connected. If the algorithm yield a failure, this trial will be rejected and the simulation will move on to next trial.

In order to collect benchmark of adaptive fanout gossip protocol, one extra node is place in the area. Its functions are generate new packets, collect other nodes' received time of every packets, and store new packets generate time. A trimmed version of adaptive fanout gossip protocol is used to generate new packets. An UDP server application is installed on the node to collect other nodes' received time of every packets.

Adaptive fanout gossip protocol is installed on all other nodes in the network. Besides that, a UDP client application is also installed on them to send received time of every packet. The simulation stops when any one node's energy is depleted. After that, all the data collected is processed to generate our performance metrics.

4.1 A Section

4.1.1 A Subsection

4.2 Another Section

Chapter 5

Performance Evaluation

In order to evaluate my protocol, I choose to use a open-source simulation software called NS-3 (Network Simulator 3).

Introduction of ns3

The number of nodes, simulation time, maximum wifi range, initial battery energy, gossip interval, and solicit interval are all the parameters I could control in the simulation. The number of nodes is ranging from 10 to 200 with step 10. The simulation time is set to be large enough to ensure any node's battery will die before the simulation stops. The maximum wifi range is set to be 50 meters so that no one node is connected to all other nodes. The initial battery energy is set to be 1080 Joule (equivalent to 100mAh) for every node. The gossip interval is set to be 1 second meaning for any node every second the gossip process will wake up and check if it need to gossip a new packet to its neighbors. The solicit interval is set to be 5 seconds which means every 5 seconds, a node will randomly pick a neighbor and query for their latest received packet.

In summary, the simulation parameters are set to be as follows:

- Number of Nodes = 10, 20, 30, , 200
- Simulation time = large enough to ensure energy depletion happens first
- Maximum Wifi Range = 50m
- Initial Energy = 1080J
- Gossip Interval = 1.0s
- Solicit interval = 5.0s

The simulation will stop when any node's energy was depleted. For each scenario (e.g. n = 10), 100 independent trials were simulated in order to gather performance data. Any simulation parameters other than number of nodes stay the same through all simulation trials.

5.1 Performance Metrics

Life Span of the Network

Definition: The time when any node's battery die

Because the goal of this protocol is to broadcast any new packet, the network will lose the phsical ability whenever any node lost wireless connection to its neighbors due to energy depletion. Therefore, this metric indicates how long a network with size n could stay connected using a broadcast protocol.

Average Packet Broadcast Time

Let's think about broadcasting one packet from one node to (n-1) nodes, the packet broadcast time of this packet for this network is the difference between the time when last node received the packet and the time this packet was first sent.

Because for each independent trial, multiple packets will be sent. The packet broadcast time is calculated for each packet in the way described above. In the end, we average the times for each scenario (e.g. n = 10).

Need to develop a mathmatical equation for this

The average packet broadcast time indicates the time needed for a packet to reach every node in the network.

Protocol Overhead

There are three types of protocol packets for the protocol. The ack packet, solicit packet, and the payload packet. The ackownledgment packet is sent to the sender from receiver when a node received a payload that it already received before. The solicit packet is sent to query a random neighbor for its latest payload. For example, in one trial with n nodes, if it broadcasted m packets, then the average protocol overhead is defined as:

$overhead = ()(p1 + p2 + \ + p_n)/n)/m$

This metric indicated how many packets is needed for a node to facilitate broadcasting one packet.

Consumed Energy

The consumed energy per node per packet is defined as

energy = ((e1 + e2 + + en)/n)/m

This metric measures the amount of energy needed for a node to facilitate broadcasting one packet.

5.2 Results Analysis

Fanout of gossip protocol is defined as the number of neighbors a node contact each time when a new packet is received. It is one of the parameters that control the behavior of gossip protocol. In one scenario, fanout is set to be one. In another scenario, fanout is set to be five. These two scenarios 's trials are conducted to compare to the adaptive fanout scenario.

(from 563 paper).

The random generated topology files are the input of our simulations. There are 7 different cases where the number of nodes vary from 100 to 1000 , increasing with 150 nodes step. And for each case, we sampled 100 topology files to run the simulations.

First, I analyzed the average number of data packets each node sent, depicted in fig. **??**. 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. **??** that this value is ranging from 190 data packets per node to 280 data packets per node depending on the scale of the network. But considering the network scale, average number of data packets per node didn't increase proportionally as we can see in fig. **??**. This metrics actually decreased exponentially.

Since when I collected the different data (average ICMP messages per node) from wired peer-to-peer network environment, here I could not perform a fair comparasion between these two evnironment. But as you can see in fig. **??**, the average ICMP messages per node is significantly lower than in the wireless ad-hoc network. I believe the reason behind this is the shared medium for wireless communication. With shared medium, collision is prone to occur thus RTS/CTS plays an important role during the whole communicating process. Thus higher average data packets sent per node is expected.

Second, the average number of hops per node is analyzed. This is different from what we collected in the P2P environment which is maximum number of hops. Thus fair comparison between these two environment can not be performed here. In the wireless ad-hoc network, as we can see in fig. **??**, the average number of hops per node mostly concentrated around 2.3 hops regardless of the growing number of nodes. For wired P2P network, the maximum hops is around 16.5. 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 for both network environment is that the number of hops either average or maximum are more or less constant. But why is that the average hops per node could remaine 2 to 3 in a wireless ad-hoc network? My explaination is that since the topology of the network is almost a complete graph as you can see in fig. **??** showing a simple 10 nodes case, with gossip interval 5ms and request interval 5s, before any node send out request packets, chances are the starting node already gossipped with most of the node in the network result to a low average hops per node.

Moreover, figure **??** illustrates the time needed to spread the message across the whole network. For the P2P network environment, the average time needed to spread the message is found to be more or less constant and slightly less than 15s. Due to the huge difference in the gossip-interval-time (5ms) and solicit-interval-time (5s), 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. However, when we compare this result with ad-hoc network simulation result, the different between them is significant. For the latter case, the spread time starts around 190s and grows almost linearly with the number of nodes in the network. In term of absolute maginitude, wireless network perform much worse than P2P network. However, this result is total within our expectation since wireless communication often encounter collision problem and thus result in longer spread time.

Section **??** proposes further work which can be done to gain a more thorough evaluation.

5.3 A Section

5.3.1 A Subsection

5.4 Another Section

Chapter 6

Conclusions and Future Work

(from 563 paper).

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 disseminate message across a ad-hoc network has been implemented and evaluated. We proposed several performance metrics to evaluate this our proposed approach.

We have implemented the gossip protocol in a wireless ad-hoc network environment and run the simulation in ns-3.

(!!!!update)The result from ns-3 shows that the average number of data packets sent per node almost remained constant when the network scale grows. Lastly, our results imply that the time until all nodes have received the data will grow linearly in a wireless ad-hoc network environment. Unfortunately, we could not verify that the time complexity of the algorithm is $O(\log n)$ as outlined in [2].

In the future work, I would like to investigate how request interval time and gossip interval time would impact the performance of gossip protocol. In my estimation, if we increase the gossip interval time and decrease the request interval time, average number of hops per node would increase. Spread time would potentially decrease and average data packets per node would not be affected.

Another direction would be implementing smart routing protocol. 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, the program 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 writing the code to generate more continous topology files in term of number of nodes so that we could better investigate the characteristic of gossip protocol.

Due to time limitations we could not address the huge spectrum of possible analysis. 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 only need for initialization.

6.1 A Section

6.1.1 A Subsection

Chapter 7

A chapter

7.1 A Section

7.1.1 A Subsection

7.2 Another Section

Chapter 8

A chapter

8.1 A Section

8.1.1 A Subsection

Appendix A

An Appendix

Bibliography

[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[2] Kate Jenkins, Ken Hopkinson, and Ken Birman. A gossip protocol for subgroup multicast. In *Distributed Computing Systems Workshop, 2001 International Conference on*, pages 25–30. IEEE, 2001.

[3] Ranveer Chandra, Venugopalan Ramasubramanian, and Kenneth P Birman. Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks. In *Distributed Computing Systems, 2001. 21st International Conference on.*, pages 275–283. IEEE, 2001.

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

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

[6] Anura P Jayasumana, Qi Han, and Tissa H Illangasekare. Virtual sensor networks-a resource efficient approach for concurrent applications. In *Information Technology, 2007. ITNG'07. Fourth International Conference on*, pages 111–115. IEEE, 2007.

[7] NM Mosharaf Kabir Chowdhury and Raouf Boutaba. A survey of network virtualization. *Computer Networks*, 54(5):862–876, 2010.

[8] Sherif Abdelwahab, Bechir Hamdaoui, and Mohsen Guizani. Cloud-assisted remote sensor network virtualization for distributed consensus estimation. *arXiv preprint arXiv:1501.03547*, 2015.