A Theoretical Framework for Analysis of Communication Pathways

A Thesis

Submitted to the Faculty of Graduate Studies and Research

In Partial Fulfillment of the Requirements

For the Degree of

Master of Applied Science

In Software Systems Engineering

University of Regina

By

Mohammad Nikou Sefat

Regina, Saskatchewan

April, 2013

UNIVERSITY OF REGINA

FACULTY OF GRADUATE STUDIES AND RESEARCH

SUPERVISORY AND EXAMINING COMMITTEE

Mohammad Nikou Sefat, candidate for the degree of Master of Applied Science in Software Systems Engineering**,** has presented a thesis titled, ***A Theoretical Framework for Analysis of Communication Pathways,*** in an oral examination held on April 2, 2013. The following committee members have found the thesis acceptable in form and content, and that the candidate demonstrated satisfactory knowledge of the subject material.

External Examiner:	Dr. Michael J. Kozdron, Department of Mathematics & Statistics
Co-Supervisor:	Dr. Christine Chan, Software Systems Engineering
Co-Supervisor:	Dr. Craig Gelowitz, Software Systems Engineering
Committee Member:	Dr. Mohamed ElDarieby, Software Systems Engineering
Committee Member:	Dr. Paul Laforge, Electronic Systems Engineering
Chair of Defense:	Dr. Rodney A. Kelln, Faculty of Graduate Studies & Research

Abstract:

Networks such as Peer-to-Peer (P2P), ad hoc, wireless sensor, and other complex networks do not support optimal communication between different devices across their configurations. They require appropriate algorithms to distribute data because they may have unspecific structures with a large amount of nodes, energy constraints, channel restrictions, and delays. The desired communication algorithms for these networks are simple and robust in order to handle these unpredictable characteristics. For example, gossip algorithms are used as core messaging algorithms for many distributed applications in wireless ad-hoc networks, sensor networks, and peer-to-peer networks because the underlying networks have complex unstructured topologies. This thesis provides a theoretical analysis of the base-line gossip algorithm where the goal is to maximize the aggregate message throughput in all to-all communication nodes. For all-to-all communication, every node attempts to broadcast its messages to all other nodes. By establishing a connection between the gossip algorithm and percolation theory in random graphs, we are able to analytically derive the aggregate throughput as a function of the basic parameters of the gossip algorithm. We use this formulation to find the optimal points in the algorithm parameters to obtain the resulting maximum aggregate throughput. The results are derived for a simplified model of the gossip algorithm and for specific classes of random networks. These results shed light on the fundamental tradeoffs involved in the gossip protocol. These tradeoffs can be used as a guideline for optimizing the performance of these algorithms.

Acknowledgements:

I would like to express my sincere gratitude to my supervisors, Dr. Nima Sarshar and Dr. Christian Chan, for their supervision and guidance in every stage of this research and for providing me with full financial support while I was pursing my Master's degree at the University of Regina.

I have also been indebted in the preparation of this thesis to my co-supervisor, Dr. Craig Gelowitz, whose patience and kindness, as well as his academic experience, has been invaluable to me.

I gratefully acknowledge the Faculty of Graduate Studies and Research at the University of Regina for its financial support in the winter 2011 and 2012 semesters.

Table of Contents

List of Figures:

Chapter 1

INTRODUCTION

1.1 Motivation

The previous century has seen many improvements in communication networks over large distances with respect to reliability, quality, and performance. With these improvements came the creation of massive communication networks. The Internet is an example of such a communication network. Networks that have limited resources are typically engineered to be as efficient as possible.

There are networks that have been established for other purposes beyond static node to node communication. Wireless sensor networks, mobile ad hoc networks, and Peer to Peer (P2P) networks are examples of these networks. These next generation networks adjust themselves based on differing environmental conditions. The environmental conditions vary with the number of users and with the network topology, which is dynamic because users can join or leave the network at any time. Networks may also have constraints on energy resources for communication and computation, as is the case for wireless sensor networks.

In order for next generation networks to be as efficient as possible, we require new algorithms that can handle these constraints and dynamic changes. Efficient algorithms play an important role in a highly constrained environment. The algorithms should contain properties that allow them to efficiently operate in a constrained and dynamic network environment. If there are a massive number of nodes in the network, it is most likely that the global topology of the network is unavailable and nodes should be able to

operate with only local information. In addition, the complexity of the computation in which each node needs to process data and then communicate with other nodes should be as low as possible to save energy and time because they may only have limited energy resources. Finally, the algorithm should be robust against dynamic changes in the network, which is likely in mobile networks or other networks where links may be disconnected at any time. All of these properties have suggested the development of the gossip algorithm, which is explained in the next section, as an accepted solution for these types of networks.

1.2 Thesis Goal

Gossip algorithms are the core messaging algorithms in many networks that are decentralized and distributed like unstructured P2P networks, wireless sensor networks, and ad-hoc networks. Gossiping problems have been extensively studied in recent years [12], [8], [24], [5].

Gossiping has been used in wireless sensor networks and in Internet networks [13]. It has also been used for canonical signal processing tasks including distributed estimation, source localization, and compression [14]. It has been applied in Stochastic Optimization [49], to aggregate sensor readings in sensor networks [13], as a search and discovery mechanism in P2P networks [53], [65], and for ad-hoc routing [23]. Gossip algorithms can be explained as follows: a node v tries to distribute a message M to other nodes in the network. Randomly, the node chooses its neighbours and sends the message to the neighbouring nodes.

By applying the same randomized method, neighbouring nodes pass the message (or a changed version of the message) to its neighbours. The message is not sent back to the node where the message originated from. This procedure is repeated until the message cannot be further relayed or the time-to-live counter (the maximum number of allowed hops) of the message has expired. When global information with respect to the network topology is unavailable, this algorithm is a natural choice for broadcasting messages because it has a distributed and stateless characteristic. Some research has already been done to optimize distribution time for when the topology of a network is available or unavailable [33, 38].

Many baseline algorithms have already been designed, but the common feature of all of them is the randomized strategy for multicasting messages from a node to its neighbours. This thesis implements a theoretical analysis of the baseline gossip algorithm. It is assumed that every single node applying the gossip algorithm tries to send out messages at the rate of one message per unit time to the other nodes in the network. It is also assumed that the size of the message, L, is fixed. If M is one such message, every node will pass the message to each of its neighbours (if the node has not received the message before) with a constant probability $0 < p < 1$. At the end of this procedure, a subset of the nodes in the network will have received this message. After sending the first message, the procedure is reordered for subsequent messages. Because of the random characteristic of the gossip algorithm, the subset of nodes that receives the second message may not be the same as the subset of nodes that received the first message. The random probability p and the size of the messages are selected such that the average

throughput on each network link does not extend beyond the capacity of the links because the capacities of the network links are finite.

By designing a simple model for the gossip algorithm, we are motivated to answer the following questions: What is the relationship between L and p with the average aggregate throughput of the messages received by all nodes in the network? How can we optimize aggregate throughput by changing L and p? This thesis attempts to find the primary tradeoff between the number of the network links used for distributing each message and the number of the total messages L that can be sent through the networks. The tradeoff can be controlled by the broadcasting probability p. To realize this, we consider the two boundaries of p (p=1 and p=0). When $p=1$, all of the nodes will receive the broadcast message. This is called a "flooding algorithm" that was applied in the early Gnutella P2P networks for search. This has a huge network cost since every single message will have to pass over all of the links in the network. Therefore, many nodes may deliver duplicate messages and many links will carry **redundant** messages. This means the total number of redundant messages to distribute just one message through the network is $|E|-|V|+1$ (where $|E|$ is the number of links in the network and $|V|$ is the number of the nodes in the network). The message size L can be, at most, 1 if the capacity of all the links is equal to 1 unit per unit of time, and the aggregate throughput received by all the nodes would be exactly 1. If we decrease the probability p, the probability that all the nodes receive each message decreases. Therefore, the probability of delivering redundant messages also decreases. Simultaneously, the number of the nodes that deliver a specific message would also be reduced, and there is no guarantee that each node will receive all the messages even if they are connected to the network.

We can increase the number of packets, L, without violating capacity restrictions on the links because every link does not have to carry all of the messages. As explained in the chapter 3, we can increase the size of the messages to $1/p$ if all links' capacity is 1. The tradeoff is between the size of the messages that can be sent due to the capacity constraints and the average number of the nodes that receive each message. If p is considered too small, the average number of the nodes that deliver each message decreases such that in an extreme case, when $p=0$, there is no node that receives any message. If the aggregate throughput is a continuous function of p, at some point between the extreme cases, we should be able to find an optimal probability for maximizing aggregate throughputs.

1.3 Thesis Contribution

To solve the problem described in the previous section, we applied random graphs to a communication network. This section provides an overview of the thesis contributions.

- **Contribution to gossip algorithm**

  The gossip algorithm has been studied by other researchers. In previous works, the dimensions of the network are limited, and other algorithms concentrate on minimizing the transmission delays. In this research, we attempt to the minimize delay using the maximum capacity of the links in the network for maximum throughput. This thesis also studies the algorithm for an infinite number of users with an infinite number of messages.

- **Contribution to information theory**

  This thesis applies information theory to a random network with respect to the capacity of channels where the gossip algorithm is the main distribution strategy. This thesis shows that there is trade-off between throughput and capacity of the channels in the gossip algorithm and an optimal point can be determined.

- **Contribution to random graphs**

  This thesis applies random graphs and percolation theory to show a direct relationship between percolation theory and random broadcasting in a general random network. We find the properties of the random network by finding the general properties of percolated random graphs.

1.4 Related Works and Applications

1.4.1 Network structures in Virtual Environments

Virtual reality is an environment that allows users to communicate with each other in a virtual world. These virtual worlds attempt to simulate reality and events that can happen in the real world. They also include fantasy reality such as gaming networks. The improvement of data communication networks and the increased capabilities of personal computers have enabled the use of simulated environments to permit users to interact with each other. A simulated environment is executed by many computers that are connected together via a computer network like the Internet. The users located in different geographical regions are able to interact with each other through a virtual world.

Virtual environments are also used for other applications such as military applications where pilots and tank drivers can experience and practice being present in a virtual battle and destroy virtual enemies. Military modeling and simulations are active fields for designing immersive virtual environments. SIMNET (Simulator Network) [44] and DIS (Distributed Interactive Simulations) [28] are examples of such systems. There are many distributed virtual environments, such as NPSNET [39], PARADISE (Performance Architecture for Advanced Distributed Interactive Simulation Environments) [15], MASSIVE [22], Mimaze [34], DIVE (Distributed Interactive Virtual Environment) [7] [18], RING [19], and BRICKNET [57], that have been modeled and developed in different research fields. Although the main goal of all of these architectures is the presentation of algorithms for efficiency of development, such systems are currently used for different applications as their fundamental architectures. All of these systems are designed for limited users within local networks and the appropriate policies to send messages through the Internet are not sufficiently addressed.

Network gaming is a field where the virtual environment has had significant investment and has consistently improved. The network MMOG (Massively Multiplayer Online Games) attract hundreds of thousands of users into virtual worlds. As such, network games cause a significant share of data traffic on the Internet. According to recent predictions, the network games market will increase from a 4.5 billion dollar to a 13 billion dollar industry in 2012[60]. For example, WoW (World of Warcraft) currently has 10 million clients and a half million simultaneous players.

Virtual environments can be studied in several different aspects. One of the most important engineering aspects in a virtual environment is the interaction between its users. In a virtual network, information is exchanged between users for:

- **The description of objects and scenes**
- **The information of sound and images**
- **Update messages**

The description of objects and scenes is used when a new user or a new object is added to a virtual environment. For instance, when a user adds a new object to an environment, the description of that object is sent to all other users who interact with that object. Importing or exporting objects and users does not happen as often as other events, and, therefore, the related messages do not require short delays.

Information for sound and images has time constraints for transmission, and, as such, the transmission protocol RTP (real time transport protocol) is suggested for the immediate transmission of sound and images.

Another example of transmitted information between virtual environments for users are update messages. Each user in virtual world has an identity called an "avatar". Avatars change the virtual environment through input from the users. These changes are received by all other users who share the environment in order to experience the same simultaneous virtual environment. Update messages are created as the result of user interaction within virtual environments. For example, when a user moves an object, update messages include the characteristic point of the object and direction of its movement. Messages are sent to all of the other users in the virtual world, and the object representation is changed for these users based on the new description. To decrease the

number of update messages, information transmitted to users is selected and managed according to the categories of interest for each user. This procedure avoids sending all of the update messages to unnecessary locations.

Virtual environments rely on the correct operation of the entire system and updated interaction between distributed machines. This interaction requires a communication infrastructure. The Internet has enabled a public infrastructure that provides network services that have acceptable quality and reasonable cost for users. This has enabled virtual systems to be utilized on the public Internet instead of on only private, special infrastructures. For Internet virtual architectures, transmission of update messages is done using an overlay network. The network studied in this thesis is an overlay network. Figure 1.1 shows a triple layered architecture to exchange messages in a virtual environment.

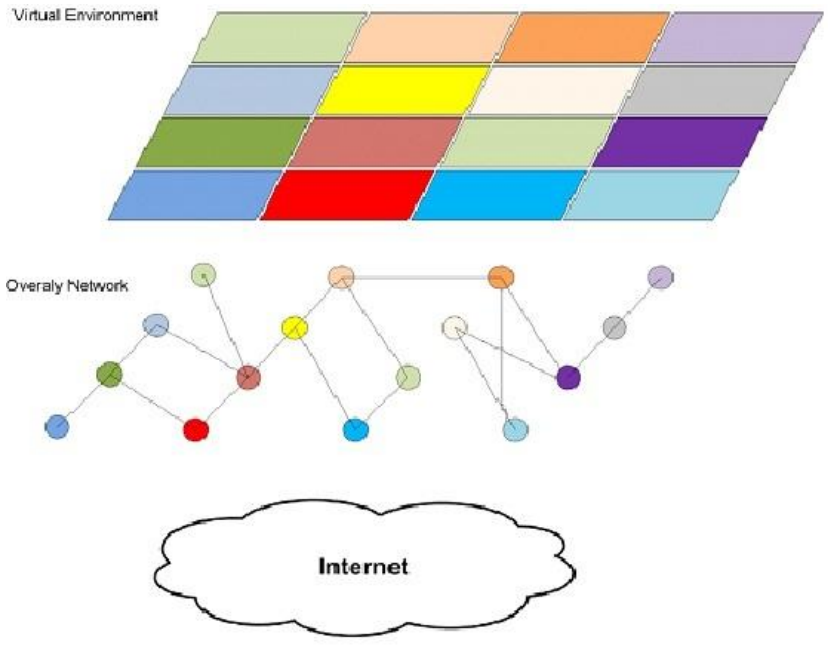


Figure 1.1: The structure of messages interchanges in a virtual network

There are many challenges to enabling extensive distributed systems on the Internet. For example, if the users want to share a virtual world from their home, all the information needed for this interaction needs to be transmitted through a home modem. Utilizing an immersive network that is layered on the Internet means the user's messages can be affected by a considerable amount of jitter and delay, which can affect the interactions. Users on the Internet may also have different software and hardware, and thus, the interactions need a system which is adaptable to all of these variations.

To exchange information in a virtual environment, three different architectures can be utilized:

- **Client-Server**
- **P2P**
- **Combination of the Client-Server and a P2P network**

For the Client-Server architecture, the description of a virtual environment and its parts is located at a unique node, which is the server. The server is responsible for receiving and processing all of the users' conditions, calculating any new conditions for virtual environment objects, and announcing these new conditions to all of the users. In this case, the management of all of the conditions is undertaken by the server directly. Due to the concentration of all of the information at the server, the problems of presenting and managing the information is not necessarily as complicated because the server determines what information to send about different objects in the virtual environment to all of the users. In the Client-Server architecture, increasing the number of users causes communication restrictions and increased processing at the server. Since the response is meant to be near simultaneous for all of the users, the problem of scalability plays an

important role. To solve this problem, combination architecture can be used where the virtual environment is partitioned into different regions and the management of each region is undertaken by each region's server. For these architectures, the main challenge is message transmission management when the users move between regions that are controlled by different servers. In addition, the problem of capacity constraints in each region creates a scalability problem with respect to increasing the size of the network and the number of users.

The expansion of the number of users in a virtual environment requires utilizing architectures that can support a large increase in the number of users. Because the Internet is a public communication network, it requires the design to consider the public architecture and its constraints, including control and storage information resources. In P2P architecture, each user can directly connect to other users. Therefore, instead of sending messages through a central server, which creates multiple channel problems and potentially requires large bandwidth, network users communicate with each other directly. Establishing a direct connection without passing through an intermediary like a central server increases scalability and is appropriate for users who need to exchange large amounts of information with each other. This thesis is based on a virtual environment with a large number of users and describes this in more detail in the following chapter.

1.4.2 Challenges in immersive virtual environments

One of the most important challenges in designing immersive virtual environments with a large number of users is to have proper switching between different

states in the environment. All of the users within a virtual environment should share the same experience.

A P2P network with a large number of nodes typically faces difficulty in searching for information and requires a referencing recognition system to find information categories (names, etc.). An immersive virtual environment uses the Internet to transmit information between distributed machines.

The more dynamic the changes in a virtual environment, the more information exchanged to update the conditions. Increasing the number of users also increases the need to exchange a larger amount of information. Since bandwidth is generally limited for each user, one of the challenges of designing a system is the management of bandwidth so that immediate interaction is not affected.

One constraint for immediate interaction is the delay of transmission. Delay optimization is one of the biggest problems in virtual environments executed on the Internet. Transmission delay is considered an inseparable part of all communication systems. Some of this delay is generated by the delay of signal spreading in communication channels in a public network like the Internet, but the largest part of this delay is caused by the waiting time of packets in transmission through routers. The delay caused by packet transmission is random and out of the control of virtual environment designers. However, there have been efforts to design virtual environment architectures in order to receive the best possible service from the Internet network.

1.4.3 Mobile Ad Hoc Networks

Wireless systems have been used since the 1980s, and we have witnessed the first through fourth generation of wireless systems. These systems are typically operated

based on a centrally controlled architecture through access points. Access points allow users to change their positions and remain connected to the network. However, due to the existence of fixed points in the network, there are some restrictions. There can be physical areas where there are no access points, and, thus, the wireless connection to the network is not available. Recent developments, including the invention of Bluetooth, introduced new wireless systems called Mobile Ad hoc Networks (MANET). The MANET, which is sometimes called 'short live,' can operate in the absence of a fixed structure. A MANET can be used in areas with fixed infrastructure. MANET is an automatic system that contains mobile nodes that are connected to each other by wireless links. Each node is utilized as an end-system and as a router for other nodes. In this network, each node can change its position while it is communicating with other nodes. This is generally referred to as an ad hoc network.

An ad hoc network is a network that consists of wireless hosts that can also be mobile. In this network, any premade infrastructure is not necessarily used. This means that in an ad hoc network, there is no defined infrastructure such as a base node, router, switch, or other typical network infrastructure components. There are only wireless nodes that can communicate with non-neighbour nodes through their neighbouring nodes. The IEEE 802.11 protocol is able to provide lower level facilities of ad hoc networks when there is no access point.

In Figure 1.2, an ad hoc structure is shown. The small circles are wireless nodes. The big circles show the effective range of each node. Any node located in the effective range can receive transmitted data by each respective node and recognize it from environmental noise. For convenience, the network is shown as an equivalent graph. The

edges of the graph show which nodes of each edge at a distance from each other can communicate directly. The nodes that are located within each other's range are connected in a graph representation of the network.

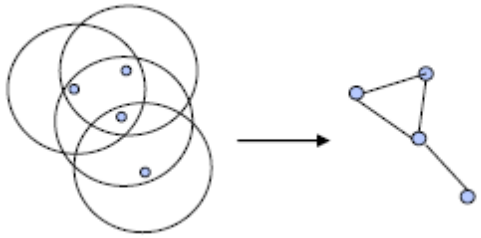


Figure 1.2: An ad hoc network structure

In ad hoc networks, the mobility of the nodes can cause changes in the path between any two nodes, which makes them different than traditional static networks. In spite of these issues, they are used in many applications because of the ease and relative quick implementation of this kind of network. The applications of such networks include laptop communications, public applications like vehicle communication, and military applications like navy communication.

1.4.4 Wireless Sensor Networks

Recent improvements in Micro-Electro-Mechanical-Systems, smart sensors, wireless communication, and digital electronics has enabled small, cheap, low-power sensor nodes that are able to connect through wireless networks [1][31][45][48]. A small sensor node contains three parts, a sensor, an information processor, and a wireless transmitter. Generally, wireless sensor networks include many nodes that measure parameters in which the data of all of the nodes are considered. All of the information for a given parameter is collected in one network node called the base station. The data from the nodes are processed so the actual measure of the parameter is calculated very accurately.

For these networks, if one of the nodes does not transmit, it has minimal effect on the estimation. In wireless sensor networks, many sensor nodes are located close to where the parameter needs to be measured. The position of each sensor is not designed beforehand and, thus, enables simplicity for placing the sensors. The protocols applied in a sensor network should be self-adjustable and self-organizing. Because sensors have a processor embedded in them, it helps decrease the amount of information that needs to be transmitted as sensors can simply send the required data after the processing all of the primary data.

The wireless sensor network is similar to Mobile-Ad hoc-Networks (MANET) in several ways; however, the protocols used for ad hoc networks are not sufficient for wireless sensor networks. In ad hoc networks, the main issue in the designing of protocols is Quality of Service (QoS). In wireless sensor networks, the main issue is to limit sensor power consumption for battery life, and protocols that consume low levels of power are generally considered. The main differences between ad hoc networks and wireless sensor networks are:

- The number of nodes in wireless sensor networks is much larger than in ad hoc networks
- Sensors are generally located close together in wireless sensor networks.
- Sensors have greater potential to malfunction.
- The topology of wireless networks change more often.
- Wireless sensor networks typically establish network connection through a broadcast mechanism.
- Sensors have a limited size, power, computational capability and memory storage capability.

1.3.3.2 The Applications of Wireless Sensor Networks

Wireless sensor networks include various kinds of sensors like seismic, movement, heat, gas, stress, photonic, infra-red, pressure, etc. Sensor nodes can be applied to the continuous sensing of the environment, event detection, local sensing, and local sensing of actuators. We review some of the applications in the following section [1] [61].

Military applications

Wireless sensor networks can be used as communication systems, supervision systems, intelligent navigation systems, and military processing systems. Since sensor networks can be designed for being placed in close proximity for accessibility and low-cost, the destruction of one node in a network has minimal effect on the network's functionality than in traditional wireless networks. As such, these networks can be better utilized in battlefields. Sensors in these networks sometimes contain Global Positioning Systems (GPS) and are used for exact positioning on the battlefield. Sensors can be placed anywhere and do not require any pre-design of the network.

Environmental applications

Another application of sensor networks is the surveillance of the natural environment for fire recognition and measurement of environmental parameters like temperature, pressure, and humidity. For example, they can be used in agriculture fields, etc. Wireless sensor networks have also been applied to tracking the movement of birds and small animals, in geology, in aerology, and in pollution studies. In forestry, they are

used for fire detection and for determining the exact position of a fire before it gets out of control.

Health and medical applications

Wireless sensor networks are used to supervise disabled patients, for drug management in hospitals, and for tracking the movement of patients. In factories for medical or chemical materials production, applications of wireless networks include supervising the combination of materials in drugs where small errors can cause danger to human life and supervising working operators.

Wireless sensor networks also have applications in health care clinics. For example, physiological information can be collected at a distance. In [30] and [52], we can see some of the applications in medicine such as the study of the procedures for drugs effects on a patient's body and supervising a patient's condition without direct interaction with doctors.

1.5 Organization of the Thesis

In the next chapter, the different routing algorithms and different network structures are presented. The third chapter is about random graphs, both their concept and applications. Also it provides the proposed algorithms and discusses how they can be implemented using current networks. Chapter Four presents simulation results, and, finally, the last chapter provides a summary of the thesis findings, concluding remarks, and recommendations for future work.

Chapter 2

ROUTING ALGORITHMS IN DIFFERENT NETWORK STRUCTURES

2.1 Routing Architectures in an Immersive Virtual Environment with a Massive Number of Users

A virtual environment network with a large number of users should be able to provide interaction for approximately 1000 to 100,000 users. The virtual environment should allow all of the users to simultaneously share the experience. This means all of the virtual environment users should be able to experience near immediate interaction with each other. In addition, the system should be able to handle simultaneous interaction among its users. For users of a virtual environment, computational hardware and communication bandwidth is normally limited. The goal of designing a virtual environment is to provide the best sense of immediate interaction for users by applying proper technologies and methods. A virtual environment can be categorized in to three groups with respect to the architecture of the communication network.

Each of these architectures has its own advantages and disadvantages that are explained in the following section.

2.1.1 The Architecture of Networks in Virtual Environments

2.1.1.1 Client-Server Architecture

In this architecture, one or more servers are responsible for transmitting update messages between different nodes of the virtual environment. There are no direct connections between nodes, and the servers handle the transmission of all of the messages. If there are N-nodes in a virtual environment, each node provides changes to the environment at a rate of *P*. Nodes receive other node's message at a rate of *(N-1)P* from the server. Under these conditions, the server receives messages at a rate of *NP* and transmits messages at a rate of *N(N-1)P*. By increasing the number of users in the virtual environment, the required bandwidth of the server quickly becomes very large. Because of the real-time nature of a virtual network, this architecture is not easily scalable because the update messages should not experience delays.

2.1.1.2 P2P Structure

Considerable improvement in the field of personal computers with respect to computational power and a decrease in consumer price has allowed for the expansion of the number of virtual environment users. As such, a centralized solution requires more powerful and more expensive servers as well as access to greater bandwidth.

The goal of a P2P network is to use the existing resources of the network at its boundary on the users' end. The goals of P2P networks are to utilize distributed resources

like memory capacity, processing capability, and the information of other users. The utilization of distributed resources requires execution in an unstable environment because users and their resources may leave the network at any time.

The motivation of organizing P2P networks is to increase the nodes on the network and free up the large bandwidth requirements of a central node. This also increases computational capability, information storage space, and the relocating of information resources to edge nodes of the network to eliminate the weaknesses of the central server architecture. In addition, P2P architecture has flexibility and is resistant to errors because copies of information can be spread over multiple nodes. P2P networks have been applied to the following:

- Distributing files between the users: such as the Gnutella, Fast Track, and Napster networks.
- Extensive computational networks such as SETI@HOME network.
- Establishing a partnership environment such as the Hive, Groove and myJXTA networks.

The revolution of P2P networks for distributing files in 1999 started with Napster as a massive network of nodes sharing music files. In spite of the efforts of the RIAA (Recording Industry Association of America) and other organizations to curb P2P file sharing, P2P traffic is still a significant percentage of information transmitted over the Internet.

In the first generation of P2P networks for distributing files, when a user looks for music, the architecture is Client-Server, and, thus, users send their queries to a central server to find the location of a music file. It is only during the interchange of files that it utilizes the P2P network. To overcome legal restrictions of sharing files, the next

generation of P2P such as Gnutella [60] and Freenet [6] utilized full P2P architectures with no central server that could be taken down. In these architectures, the network nodes create an overlay network with each other that enables searching and the routing of information between nodes. Searching for information without a central server is the challenge that these networks overcame. In order to handle this challenge, Gnutella applies a flooding mechanism for enabling search queries to all the nodes of the network. Flooding search algorithms over-utilizes network resources in the underlying network and usually requires significant time. Searching queries are stopped at a specific depth by limiting the number of hops, and, as such, the search may not get the desired result.

Networks such as CAN [6], Pastry [16], Khademlia [43], Chord [59], and Tapestry [64] are considered the third generation of P2P networks. In Chord, CAN, and Pastry networks with *N* independent nodes, each node has a routing table with dimensions of $m(N)=o(\log(N))$(i.e. $\lim_{N\to\infty}\frac{m(N)}{\log(N)}=0$). These systems apply different methods to categorize information and protect them during the separation and joining of other nodes in the network. In these networks, their efficiency and ability to develop is similar. In these architectures the response to one query requires $o(\log(N))$ steps to complete. Delay is dependent on each step and the random distance of the nodes on the Internet network. This cannot be controlled directly, and, as such, this mechanism is not sufficient for applications that are sensitive to delays.

Another applied field of P2P networks is extensive computations where the resources of the users are utilized to do distributed computation. Large computational tasks that can be broken in to smaller parts can be distributed to different users so it can be done in a more reasonable time. Breakable computational problems that require massive amounts

of computation can be solved in less time and at less expense. For instance, the goal for SETI@HOME is to utilize distributed personal computer power to find extraterrestrial life signals. Another category of software which requires massive computations are simulation tasks. For example, FOLDING@HOME is software that enables many distributed nodes to simulate a sequence for folding protein.

Using P2P networks for applications that require immediate interaction is a recent field of research. Virtual networks are an example of this application. Silopsis [32] and QuON [3] are such kinds of virtual networks. P2P structures are used to organize a virtual environment and support environment development. In P2P networks, all nodes are considered equal and should be able to communicate with other nodes directly. There is no central node that the operation of other nodes is dependent on. In a P2P virtual environment, the environment information is distributed among all the nodes and no central description of the virtual network is kept at any one node. Therefore, each user's node is responsible for updating parts of objects for all the other user nodes. Convergence and synchronization between users is difficult because using a P2P network consists of control complexities, security management problems, and private information protection difficulties in an immersive system. However, P2P has many advantages that justify trying to overcome these challenges:

- Decreased hardware and network demands
- Independent of future expansion
- Decrease of delay for messages by routing through users
- Independent of the support of a central node

2.1.1.3 Combination Architecture

There are challenges in the management of P2P networks and in the development of P2P applications. They require powerful servers for a centrally controlled P2P system with many nodes. As such, the usage of architecture such as the combination of the two previous ones is unavoidable. To solve the saturation problem of a single server's computational power, many different servers are used instead of one single server. For instance, in [11], a number of servers are applied to the management of a game so that many users at any time can login to the virtual environment and find a server to connect to. This could be an unavoidable server with minimum network delay that it connects to. The users are not connected to other servers and just connect to their own server with Server-Client architecture and organize a multicast tree to interchange the information. The important point is that, in this architecture, there are problems like balancing packets between servers and minimizing delays [62].

2.1.1.4 Area of Interest Management

To establish presence in a virtual environment, each user should keep a local version of the current state of the virtual environment, and this local version should be updated immediately during any transitions within the virtual environment. Each user after each interaction should report the new final condition to all other users who are affected by the interaction. The simplest way is sending a copy of the new state (conditions) to all other users through multicasting. Each avatar that interacts with the environment needs to know only a small portion of the information. A multicasting strategy, in spite of its simplicity, requires sending and processing several unnecessary messages that causes many problems for the development of a virtual architecture. A

solution for sending updated messages instead of multicasting messages is to send only messages related to a user's avatar to the user. Area of interest management (AoIM) is the process used to recognize information categories related to avatars within a virtual environment. In this architecture, the environment is partitioned into smaller parts called virtual rooms. Virtual environment clients send their messages only to rooms that are located in their AoIM instead of sending messages to all clients. The main idea of AoIM is filtering of messages based on their contents such that the messages are not sent to nodes that do not need the information. AoIM can decrease required bandwidth significantly for interchanging information, especially in an architecture with a massive number of users [17][35].

The AoIM of an avatar is related to the realization of the avatar. Since avatars have limited sensation powers and speeds, this information is only a small part of the whole virtual environment information. Thus, AoIM can decrease the amount of information that is interchanged. However, the management and filtering of information can be challenging.

2.2 Routing Algorithms in Mobile Ad Hoc Networks

One of the most important efficiency issues for any network is the routing of packets and finding the optimal paths from any source to any destination. Routing in wired networks and wireless networks that have infrastructure and where accessible nodes are fixed needs special strategies and solutions. For ad hoc networks, this is much more difficult because the node positions are not fixed and are continually changing.

There is no complete or standard category for routing in ad hoc networks, and different articles note different categorizations. Routing schemes can be studied from different viewpoints. Some of them are described below:

Routing information storing

Routing protocols can be divided to two different categories according to how they store routing information: 1) Link State Routing (LSR) and 2) Distance Vector Routing (DVR). In LSR the routing information is transmitted in Link State Packet (LSP) form. Any changes in links cause creation of an LSP and then flooding the transmission of that to the whole network. Each node plots a virtual network topology from this information. It then computes all of the paths to the other nodes. The main problem of LSR is overloading. Since in ad hoc networks the network topology is changing dynamically, many LSPs are transmitted and can cause overloading in the network. In DVR, each node keeps a distance vector that includes three components. These components are the ID of all destinations, distance in respect of number of hops to a destination, and a next hop to each destination. Each node updates its DVRs periodically.

Updating

Routing information on network nodes should be updated to maintain knowledge about link conditions and network topology. From this viewpoint, the ad hoc protocols are divided in to two categories: 1) protocols that have periodic updates and 2) protocols that have event-driven updates. In protocols with periodic updating, routing information is spread across the network in time intervals. This makes for easier protocol operation and supports network mobility. This informs the nodes about the current state and topology of the network.

For event-driven updating, it is necessary that at least one event, such as a link disconnection or a new link connection, happens before updating the network. In some networks where the mobility of the nodes is frequent, the network topology changes so fast that many update packets would be transmitted and this utilizes bandwidth. One can use both of the schemes simultaneously, which is called a hybrid scheme. For instance, in DSDV algorithm, each node spread out its distance vector periodically and an update message is disseminated whenever a link is involved in a change.

Computation

In each protocol, a definite path is computed. Two kinds of computing schemes are suggested: 1) Centralized Computation and 2) Distributed Computation. In the first scheme, each node has the complete global information about the whole network topology. Whenever it is necessary, it can compute a path to a destination like in LSR. In a distributed computation scheme, each node only keeps local information about the network, and when a path needs to be computed, many nodes cooperate with each other, like in DVR.

Architecture

Routing protocols have a flat architecture or a hierarchical one. In flat architecture, all of the nodes are at the same level with respect to access to network information. This mechanism is good for small networks, but in large networks, the amount of routing information is increased, and it may take significant time to reach a far node. In a hierarchical architecture in large networks all of the nodes in the network are partitioned

into groups called clusters. By accumulating clusters, larger clusters are created and a hierarchical architecture, clusters are created. The organization of clusters in a network helps the mobility of the network. Only high level and stable information, like a super cluster, is spread out on the whole network to far distances, and overloading of traffic is decreased dramatically. In each cluster, nodes have the complete information about the network topology within their own clusters. If the destination is located in their cluster or another cluster they can use proactive or reactive routing or a combination of both.

In general, we can categorize routing protocols into two categories: 1) Topology-based protocols and 2) Location-based protocols.

This section reviewed topology-based protocols. The following sections review location-based protocols for wireless sensor networks, which are similar for other ad hoc networks. Most topology-based protocols are studied with respect to time and delays in the networks. This can be a constraint for capacity in large networks. From this viewpoint, routing protocols are proactive or on-demand.

2.2.1 Proactive Routing Protocols

The other names for this scheme are Pre-computed or a Table Driven mechanism. For this scheme, paths to all destinations are computed beforehand. In order to do this, nodes store all of the details about the network topology and the link conditions. To keep the information updated, the nodes update their information periodically or when there is a change in network topology. The advantage of this scheme is that when a source needs to transmit a packet to a destination, the path is already determined and available so there is no need to compute a path. This enables an increase in speed and a decrease in delay.

The disadvantage of this scheme is that although all the paths are computed ahead of time, some of them may never be used. The other problem with this protocol is that information distribution with respect to routing occupies a big part of limited bandwidth when the topology of the network changes quickly. The LSR and DVR schemes are from this category.

2.2.2 On-Demand Routing Protocols

The schemes that are considered on-demand have a slow policy for routing. For these methods, a path to a destination is not defined beforehand and, in comparison with proactive protocols, all of the updated paths are not kept by all of the nodes. Whenever there is a need for a path to send data, the path is created. Whenever a source needs to send data to a destination, a mechanism is used to discover the paths to a destination. This procedure is called route discovery. After finding paths, the source sends packets on these routes. Paths are valid until they become unavailable or there are other paths that are shorter. During data transmission, a path may be disconnected if the nodes on the path are moving or changing their conditions. In this case, the broken route should be constructed again. The procedure of broken path discovery and reconstructing is called route maintenance. The main advantage of these schemes is the bandwidth use of wireless ad hoc networks is more efficient because consuming bandwidth in data communication for routing is restricted only to those destinations where a node needs to send data. The on-demand mechanism also does not require spreading of information periodically by broadcasting or flooding. The problem of this scheme is that there is some delay at the

beginning of the transmission as the route is created by the route discovery mechanism. These schemes are also called reactive routing.

2.3 Routing Algorithms in Wireless Sensor Networks

2.3.1 The Important Factors in Wireless sensor Networks Design

The design of wireless sensor networks is affected by parameters such as fault tolerance, scalability, production cost, operating environment, sensor network topology, hardware constraints, transmission media, and energy consumption. Although none of the studies take a comprehensive look at all of these factors, the individual factors have been studied by many researchers. These factors are important because they are the basis of policies for designing algorithms and protocols in sensor networks. These parameters can also be used for comparison between different viewpoints. To design efficient protocols for wireless sensor networks, the following parameters are considered [1]:

Fault Tolerance

Protocols should be designed in such a way that the malfunction of one or more sensors in the network does not affect the network's operation. For example, the Medium Access

Control (MAC) and routing protocols should replace routes that are blocked during the malfunction of one or more sensors. To achieve this aim, multiple redundant layers are required.

Sensor nodes may be out of order because of a lack of energy or some other reason. The malfunctioning of a node should not have an effect on the whole network of sensors and the system should still be able to continue its operation. Fault tolerance in wireless sensor networks means the system continues its operation without stopping when faults occur. Reliability R_k(t) of a sensor node is modeled with Poisson distribution:

$$R_k(\mathrm{t}) = e^{-\lambda_k t},$$

where λ_k is the defection rate of a sensor. The reliability of a system is the probability that a system executes its duty cycle accurately within an interval of time if the system worked accurately from the beginning.

Scalability

Scalability for sensor protocols is the ability to handle from one to many sensors. Scalable protocols are written such that it is possible to handle the activity of a network with a large number of sensors. Several protocols, despite having several advantages, do not work with a large number of sensors because they impose high workloads on special nodes in the network.

Production Cost

Due to large amounts of sensors in each network, the cost of each sensor should be very low such that these networks can be operated in practice.

Hardware Constraint

In practice, each sensor should not occupy much physical space. Therefore, sensors are small and light and operate with low power. A sensor node includes four main parts: the sensor, the processor, the transceiver, and the power unit. Depending on the application, it may also include a location finding system and mobilizer.

The sensor includes two minor parts: a sensor and an Analog to Digital Converter (ADC). Generated analog signals by sensors are converted to digital signals by ADC and are transmitted to the processor. The processor contains memory that manages a procedure to follow for the final goal with the other nodes' cooperation. The transceiver is used to connect nodes in the network. An important part in sensor nodes is the power consumption, which may be supplied by solar cells. The diagram in Figure 2.1 shows the sensor node parts in a wireless sensor network.

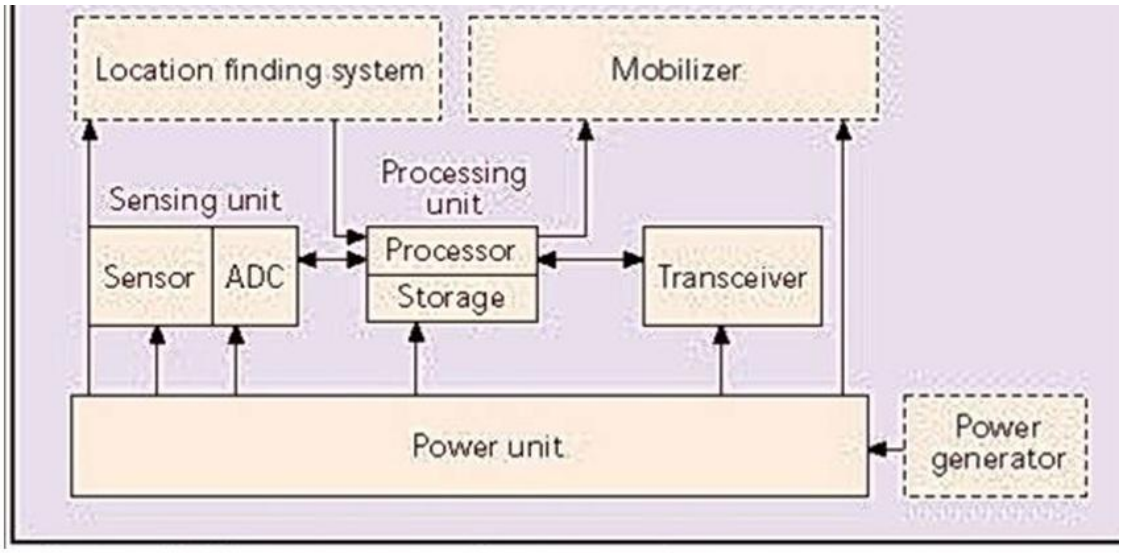


Figure 2.1: The sensor node parts in wireless sensor networks

The transceiver may contain optical or non-optical elements or may use Radio Frequency signals (RF). Radio connections require modulation, filtering, demodulation, and multiplexing circuits. Since the antennas of these nodes are very close to each other, there is a possibility that the signals of different nodes will experience interference.

To save energy in sensor nodes, efficient algorithms create short routing pathways. Bluetooth technology is not efficient for these networks because by turning on and off, they cause unnecessary energy consumption in sensor nodes.

The processor and memory in wireless sensor networks are also important. For instance, a sensor node called Smart Dust Mote consists of an Atmel AVR processor with 8 KB capacity, 512 B EEPROM memory, and TinyOS is used as the operating system to control the processor.

In many applications, the sensor needs to know its own geographical location. Since the sensor is located in an area randomly, it needs to have information about its

location to be able to cooperate with the other nodes.Each node may have access to its own geographical location using the Global Positioning System (GPS). The sensor nodes equipped with GPS are tracked within, at most, five meters of accuracy. All of the nodes do not require GPS because some of them have their own geographical information programmed and can provide their information to the other nodes. Using GPS can be a problem for a network to have scalability and low cost.

Sensor Network Topology

Because it is probable that at any moment one or more sensors may stop their activity, the topology of these networks changes. The designed protocols should be capable of handling these changes in topology.

Operating environment

These networks may be used in different regions. For differing environmental conditions, special protocols may work more efficiently.

Transmission Media

Transmission Media can use RF signals, Infra-Red, or optical signals. In any of these cases, the desired frequency should be specified.

Energy Consumption

This factor is particularly important in designing protocols for wireless sensor networks. Each sensor consumes energy for measuring desired parameters, information transmission, and information processing. Most of the power consumption is related to

information transmission. As such, the design of sensor protocols tries to reduce transmitted information through local information processing.

2.3.2 Protocol Stacks

Protocol stacks for a base node for regular nodes are the same. They have three management planes including power consumption management, mobility management, and task management. The stack protocol is illustrated in the Figure 2.2.

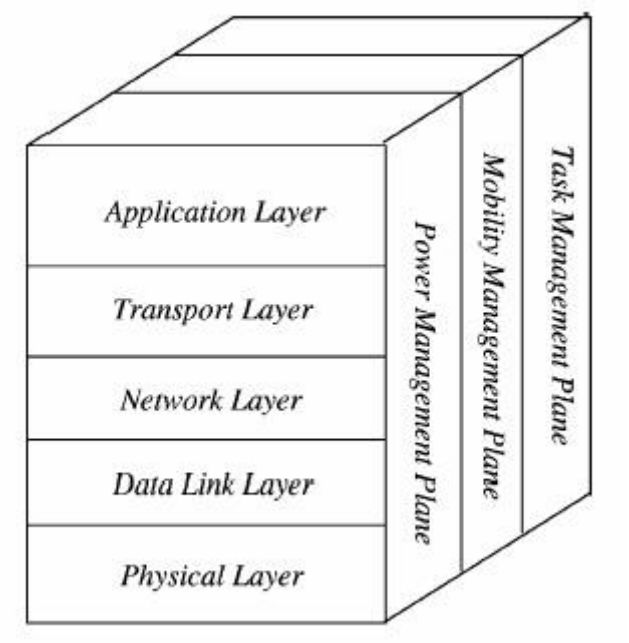


Figure 2.2: Protocols stack in wireless sensor networks

For the physical layer, designing low-cost, efficient hardware and low power modulation methods are still open problems for research. In the data link layer, designing MAC and finding the minimum sufficient energy for self-organization of nodes are subjects still being researched. For example, in [63], a MAC protocol called S-MAC is introduced and is more efficient in energy consumption compared to the other protocols.

One of the most famous simulators for wireless sensor networks is NCTUns, which is able to simulate algorithms and other characteristics of sensor networks. The following graph is a simulation done by this software on power consumption of sensor nodes that shows nodes in a state of transmitting, receiving, idleness, sleeping, sensing, and processing. As seen in the graph, transmitting requires more energy than processing which, thus, illustrates the importance of routing algorithms.

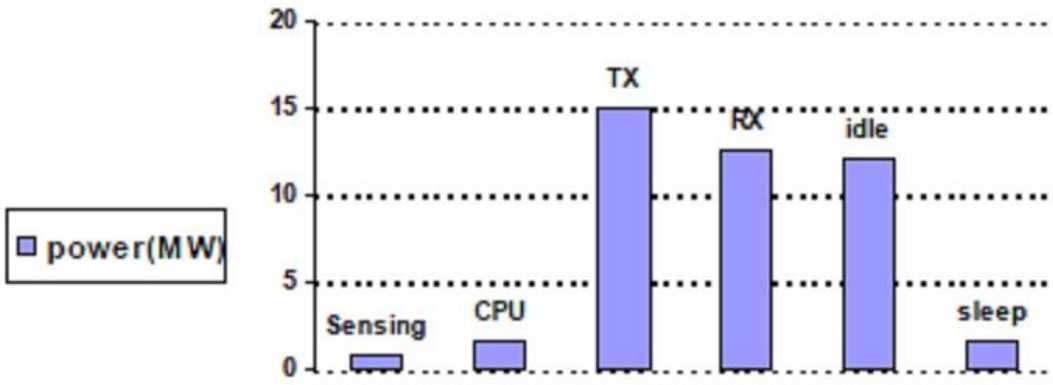


Figure 2.3: Energy consumption simulated in wireless sensor networks by NCTUns

A proper routing protocol for information transmission is required because a sensor's position is not defined beforehand. The protocol should be able to handle routing dynamically. An important consideration in the design of routing protocols is the energy consumption problem. Energy consumption in sensor nodes should be minimized such that the network can be more permanent. The protocols also consider that sensors located in specific positions may be lost earlier than others and, as such, it would not be able to supervise that part of the network. This problem may occur in such a way that the operation of the whole network would be stopped.

2.3.3 Routing Algorithms

Due to the differences between current wireless sensor networks and MANET networks, routing algorithms in these networks are a currently high-interest topic. These differences are:

- There is no possibility for general addressing in these networks so protocols like Internet Protocols (IP) are not applied in such networks.
- In almost all of the wireless network applications, it is required that data is routed to one base node from different regions (in contrast with other communication networks).
- Sensors are generally close to related phenomena for measurement and may generate the same data, so the amount of similar data is large in this sort of network.
- Sensors are restricted with respect to consuming energy for sending data, information processing, and memory capacity. Therefore, routing protocols should supervise these resources.

According to the above-mentioned differences, many algorithms are suggested to solve the routing problems for wireless sensor networks. In the design of these routing mechanisms, in addition to specific characteristics of a network, network applications and network requirements also should be considered. Most of the routing algorithms can be categorized in following groups:

Data Centric Protocols

These protocols are query-based and operate in such a way that enables the ability for repeated data not to be sent.

Hierarchical Protocols

For these protocols, the network is partitioned into minor parts called clusters, and in each the cluster, the cluster head is responsible for tasks like data combination and repeated data elimination.

Location-Based protocols

For these protocols, the information related to the sensor's position is applied to find an optimized route for information transmission.

Some protocols also operate based on a network-flow model or based on QoS (explained later). The following explain some parameters for routing protocols in wireless sensor networks:

Dynamic Network

The positions of most of the sensors are fixed in most networks. There are some networks where the sensors move. For mobile sensors, the problem of route stability becomes important. Each measured parameter can be static or dynamic. For example, in an application for target tracking and recognition, the network may be dynamic but for fire recognition in forests, the network may remain static. In a dynamic state, data is being sent to a base node from sensors continually, but in a static state, whenever a phenomenon occurs, it is reported to a base node.

Sensor establishment in networks

Sensor establishment in different networks is different and can have an effect on the efficiency of the entire network. In some applications, the sensors' positions are defined beforehand, but in other applications, they are not. In networks in which sensor positions are defined, the information is transmitted through routes that are defined beforehand, but in networks in which the sensor positions are not defined beforehand, sensors must do the routing procedure automatically.

Energy Constraint

For the design procedure of a network infrastructure, the route establishment is affected by the energy constraints. The transmission power is dependent on the distance from the receiver. Using multi-route methods can enable lower energy consumption. However, using this method can cause problems in topology management and access control of transmission media. Therefore, in networks where sensors are located randomly, applying the multi-route method is not possible.

Information Transmission Model

In wireless sensor network applications, information transmission media can be time-driven, event-driven, query-driven, or a combination of these. For time-driven models, each sensor sends its own information continually. For event-driven trans models mission, only if a predefined event happens in the network will it be reported. For query-driven models, an inquiry is spread in the network from the base node and arrives at the

desired sensors. These sensors also provide a response. Some networks use a combination of these models. The information transmission model decreases energy consumption and provides stability for established routes. For the application of fire recognition in a forest, the information transmission model is efficient and is event driven.

Data Aggregation

Sensors may generate repeated data, and by using the data combination method, we can decrease the amount of transmitted information. A different kind of method is explained in detail in [30]. Since information processing wastes less energy in comparison to information transmission, data combination seems to be popular because many protocols utilize this technique. More information can be achieved by signal processing techniques, and it is often called data fusion. For example, the beam-forming method is based on such a strategy. In the following section, protocols for wireless sensor networks where data combination is used will be explained.

2.3.3.1 Data-Centric Protocols

In many wireless sensor network applications, specifying a general ID for each sensor in a network is impossible. Lack of this ID with random sensor placement in a network causes difficulty in information gathering from a specific group of sensors. Therefore, data with high repetition is disseminated to such groups. To prevent wasting energy, methods like data combination are used. This causes routing to be data-centric instead of based on addresses.

In data-centric protocols, the base node sends its queries to desired regions and waits until data is returned from the sensors located in those regions. For this method, the

characteristic of a sensor is more important than its address. This characteristic includes position and those parameters that can be measured by the sensor. In the sections below, we discuss some important data-centric protocols.

Flooding and Gossiping

These two methods are classic mechanisms to transmit data in wireless sensor networks. For these protocols, there is no special design for the routing algorithm, which preserves the network topology. For the flooding algorithm, each sensor that receives data sends the data to all of its neighbours except the original sender, and this procedure continues until the message is received by the desired destination or a number of specific hops that the message travels.

The gossiping algorithm is a generalized flooding method. Each sensor receives data and sends that data to only one of its neighbours randomly. This procedure continues until the destination receives the message [24][1].

The above methods are very simple, but these methods have many problems. One occurs in the flooding method when redundant data is received by a sensor through two or more different routes as shown in Figure 2.4. The overlap of two or more sensors close to a desired region is another problem because they send the same information to a single sensor. The third problem is that the flooding algorithm is blind with respect to energy resources and bandwidth available in the network. In gossiping, an explosion of data does not occur because there is only one neighbour selected to transmit the data. A problem of this algorithm is in the delay experienced for receiving messages.

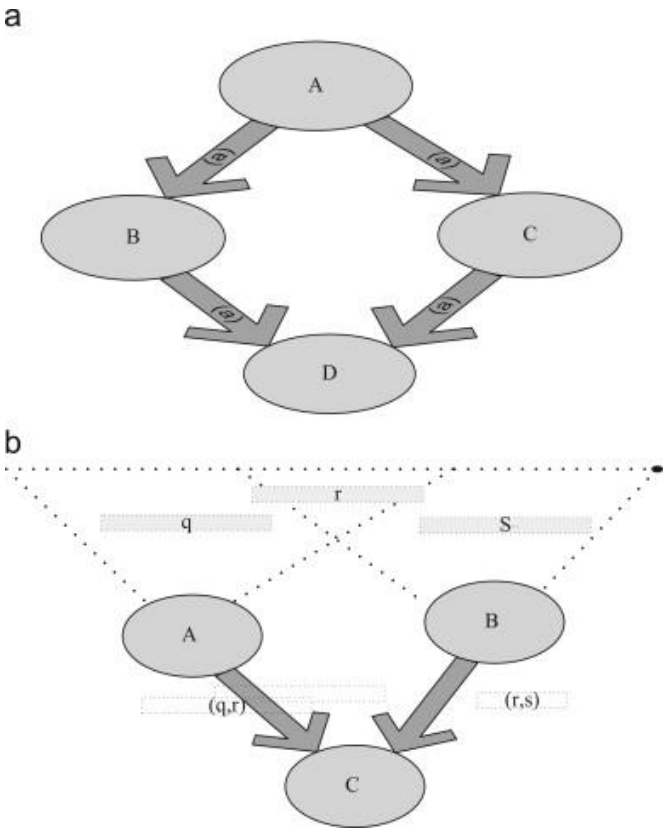


Figure 2.4: Flooding problems: a) Implosion problem b)Overlap problem.

Sensor Protocols for Information via Navigation

This is a new method for routing in wireless sensor networks. For this protocol, in addition to data messages, other messages also are used to help decrease the amount of information transmitted. Sensor Protocols for Information via Navigation (SPIN) uses a method that sends to all of its neighbours an advertisement (ADV) after receiving new data. Each neighbour that needs data returned sends a REQ (Request) to the sensor. When the sensor receives a REQ message, it sends data to those neighbours that sent the REQ. Through this strategy, the problem of repeated data as in the flooding algorithm does not exist, and, therefore, the efficiency increases. In Figure 2.5, the SPIN operation method is shown.

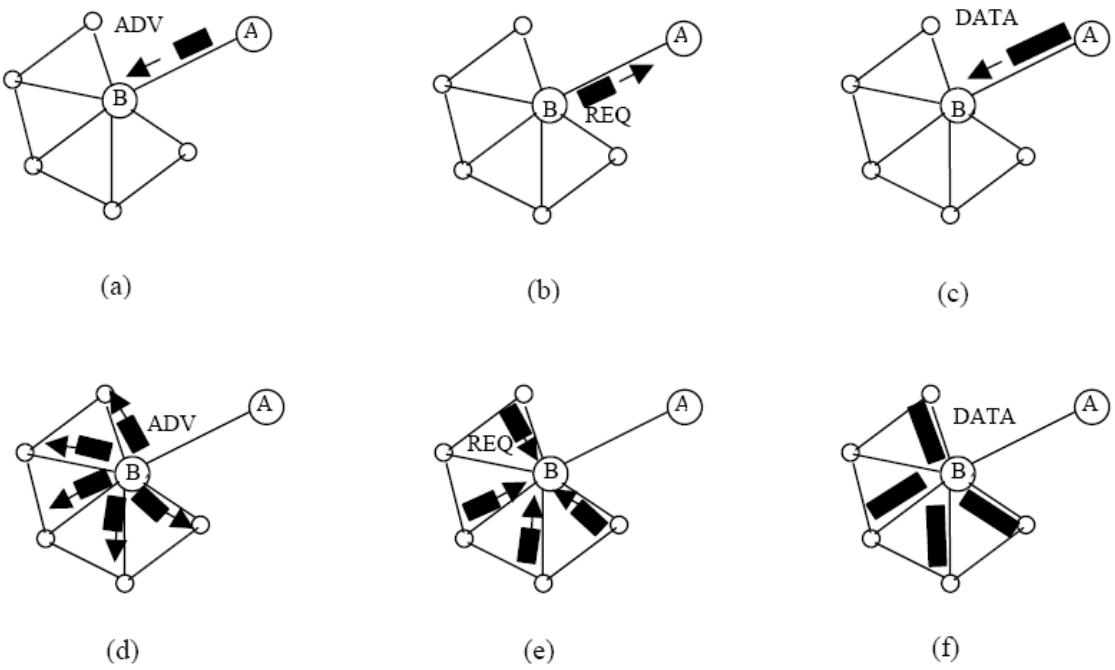


Figure 2.5: The operation strategy of SPIN protocol.

An advantage of the SPIN protocol is that, in this protocol, the changes in the network are done locally. In fact, each sensor needs to know only one of its neighbours. Energy consumption in SPIN is about 3.5 times less than flooding. Using ADV and REQ messages decreases the amount of repeated data by one half. A problem that exists in SPIN is that there is no guarantee data is transferred from one sensor to another sensor. For instance, if a sensor is located at a far distance from the base sensor and intermediate sensors are not interested in receiving the information, there is a possibility that the data will not arrive at the destination. Therefore, the SPIN protocol, where intrusion detection is important, is not a good choice because for these types of applications, guaranteed data transmission is important.

Directed Diffusion

This protocol is a fundamental protocol like the data-centric protocols. In this protocol, for data and queries, some attributes are defined which are called attribute-based naming. Instead of sending raw data, the attributes are sent. This protocol is designed such that whenever a new query is established, on-demand routing is started. The properties of a query can be the name of the required parameter to be measured, the data transmission period, the required transmission time, geographical location, etc. Each sensor that receives a query, keeps it in memory for the next usage. Sensors combine the data locally to decrease the amount of information that needs to be transmitted [29].

When a sensor receives a query, it sends it to its neighbours, and a gradient is formed between them. Gradients are a known return route that neighbouring sensors receive when the query is sent through them. By establishing gradients between transmitters and receivers, new paths are created. Among all of these new paths, only one path is selected as a reinforced path. This reinforced path is based on the information rate determined over all the paths. The reinforced path is selected if the information rate is higher than the other paths. In Figure 2.6, Directed Diffusion is shown.

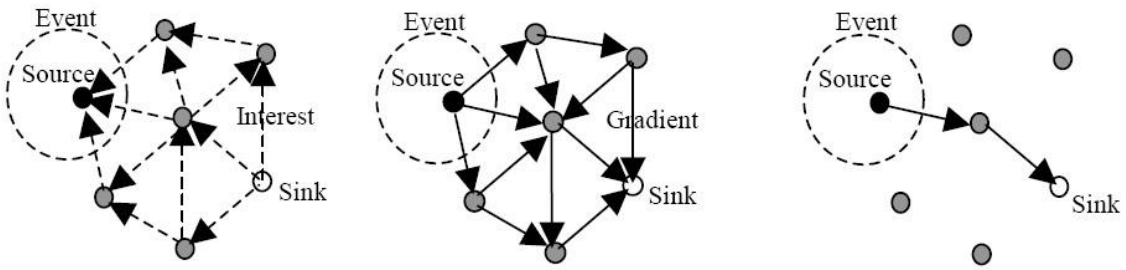


Figure 2.6: The operation strategy of Directed Diffusion protocol.

If the chosen path is not able to continue its activity, a new path is selected. New searching operations are run to find the best path between remaining paths. A path with the highest information rate is selected and replaces the previous one. Ganesan et al. in [20] suggested that instead of finding one path, multiple paths should be selected to replace a path as soon as the selected path is unavailable. Although this strategy requires more energy in the first step of finding a path when a selected path becomes unavailable, there is an energy saving since there is no need to find a new path.

SPIN and directed diffusion are different with respect to information transmission. In directed diffusion, whenever a base node requires information, it sends a query to the whole network. For SPIN, whenever a sensor extracts desired data,* it sends the data to a base node. Directed diffusion has several advantages because it is data-centric and all of the data transmission is done with neighbouring nodes. It does not require a special addressing mechanism. Each sensor can do information combination and storage. Information storage is a necessary ability to save energy and decrease delays. The network starts its activity only when a base node sends a query, and there is no need to preserve the network topology; therefore, this protocol is efficient with respect to energy consumption.

Even with these advantages, this protocol is not appropriate for all applications. For example, in applications where sensors need to send their data to a base node continually, using directed diffusion is not justifiable. This protocol is not appropriate for applications such as natural environment control. Attribute-base naming depends on the application and should be defined beforehand. Information processing and finding the correct receiver of the query uses energy and causes delays in the network.

Energy Aware Routing

In this method, a group of suboptimal paths is used in order to increase the network's lifetime. Paths are selected by a probability function dependent on the energy consumption in the paths. An important parameter in this protocol is the network survival time. Finding the path where the least energy is used causes all the sensors to utilize their energy. Instead of using the optimal path, some suboptimal paths are considered by using a probability function. One of them is selected and used for a time duration. This protocol consists of three phases:

1. **Setup Phase**

In the setup phase, flooding is used for paths to be recognized by the destination, and forwarding tables are created. During this operation, the energy cost function is calculated for each path. For instance, if a query is sent from sensor N_i to N_j , sensor N_j calculates the cost function in following way:

$$C_{N_j,N_i} = Cost_{N_i} + Metric_{N_j,N_i}$$

In this formula, the energy metric $Metric_{N_j,N_i}$ is a function of transmission and receiving cost of N_j and N_i and also the sensor energy along the path. The paths with the highest costs are not considered. Each sensor passes a probability to its neighbours from its forwarding table. The probability is the inverse of the cost and is obtained in the following way:

$$P_{N_j,N_i} = \frac{\frac{1}{C_{N_j,N_i}}}{\sum_{k \in FT_i} \frac{1}{C_{N_j,N_i}}}$$

N_j also calculates the average cost up to the destination with respect to the neighbours located in the forwarding table. The total cost of N_j is sent to the neighbours.

$$Cost_{N_j} = \sum_{i \in FT_j} P_{N_j,N_i} \; C_{N_j,N_i}$$

2. **Data Communication Phase**

Each sensor sends its own data to one of its neighbours in the forwarding table based on the specified probability.

3. Route Maintenance Phase

Periodically, data is sent by a local flooding mechanism in order to be sure that all paths are available.

This protocol has the same algorithm as directed diffusion with respect to finding paths from base nodes to sensors. In directed diffusion, only a path with the maximum data rate will be selected. For EAR, the path is selected based on probability functions. Simulation results show that EAR operates at *21.5%* of the energy consumption and increases network lifetime by *44%* more than directed diffusion. EAR, like directed diffusion, when the main path is unavailable, does not face problems because other paths have been determined in addition to the main path.

Gradient-Base Routing

This protocol is a modification of directed diffusion. When a base node query is spread out through the whole network, each sensor measures the required hops to reach the base node. In this way, it can obtain the minimum hops needed to reach the base node. These minimum numbers of hops are called the height of the sensor [55].

The neighbour sensor height is the path gradient between them. The sensor data is sent to a base node through the paths that have the least gradient. This scheme tries to reach the base node through the least number of hops.

To distribute traffic uniformly on the network, two techniques are applied: data combination and traffic spreading. Sensors that receive data from multiple paths do the data combination task. The traffic spreading techniques are as follows:

Stochastic Scheme

If there are two or more paths with the same gradient, path selection is random.

Energy-Based Scheme

When sensor energy is lower than a threshold, it increases its height to show to other sensors that they should not use this sensor to transmit the data as much as they can.

Steam-Based Scheme

This scheme attempts not to use paths that are currently being used for data transmission and creates new paths.

By applying the above schemes, traffic is spread through the whole network increasing the network's lifetime. These methods can also be applied in other protocols. Simulation results show that GBR in energy consumption is more economical than directed diffusion [55].

2.3.3.2 Hierarchical Protocols

As explained previously, scalability and network developability are important parameters for the design of wireless sensor networks. If the entire network load is on one specific path, the network may become congested and subsequently increase delay and decrease network efficiency. Because sensors are not able to directly communicate over far distances, using this scheme restricts the network developability. To increase the network capability for coverage of a greater area experiencing problems in quality of service, network categorization into clusters is suggested. The main aim of hierarchical protocols (based on clustering) is to apply an appropriate method to optimize energy usage of resources. This is done by utilizing multi-hop communication in the network and data combination within a cluster to decrease the amount of transmitted data. The Leach protocol was one of the first hierarchical protocols that was introduced for wireless sensor networks. Many other protocols have been designed based on the LEACH protocol. In the following sections, some hierarchical protocols are discussed:

Lower-Energy Adaptive Clustering Hierarchy

This protocol is one of the most well-known hierarchical protocols for wireless networks. In this protocol, time is divided into rounds. Each round is also divided to two phases. The first phase is called the set-up phase. The setup phase creates the clusters. The second phase is called the steady state phase and is related to the regular operation of the network [25][26]. The first phase is based on an adaptive probability function, where Clustering Hierarchies (CHs) are selected. The selection of CHs requires that each sensor node chooses a random number between zero and one. If the number is less than a specific threshold during a time interval, that node is selected as a CH.

This probability function is designed such that for a defined number of rounds, each sensor gets the CH only once so the energy consumption is spread across the whole network. After the setup phase of each round, CHs are selected. Each CH announces to the other nodes that it is selected as a CH and each node choses its own appropriate CH and announces itself to its selected CH. In this way, clusters are created. Each CH schedules for all of the sensors in its cluster and specifies a time slot for each sensor to prevent collision events between the sensors in the cluster. To prevent collision events between different clusters, the method of Direct Sequence Spread Spectrum (DSSS) is applied.

In the second phase, every sensor sends its data during its own time slot. The CH after receiving all of the sensors' information combines them all together and sends the information to the base node. The act of the CH combining all of the data in the cluster results in a saving on the transmitted data and the overall energy consumption.

One of the most important advantages of the LEACH algorithm is that the usage differences between different nodes are decreased compared with other protocols, and the network lifetime is increased by using dynamic clustering. However, the clusters created in the first round cause interval energy consumption, which is a disadvantage of the protocol. LEACH operates more efficiently when the sensors are in close proximity.

Power-Efficient Gathering in Sensor Information Systems (PEGASIS)

This method is an improved version of the LEACH protocol. In this scheme, instead of creating different clusters, a connective chain is established between all of the sensors. Each sensor communicates with its neighbours to send or receive data, and in the entire network, there is only one sensor selected as a data transmitter to the base node [37].

Threshold sensitive Energy Efficient sensor Network Protocol (TEEN)

This protocol is designed for situations where sudden changes in measured parameters are reported. For these applications, the network is not in an active state permanently, but when changes occur, it is reported to the base node. TEEN is a hierarchical protocol, but data centric protocol techniques also are applied. The operation of this protocol creates a cluster with sensors that are located close to each other. This strategy creates cluster levels up to the base node [57]. It is shown in Figure 2.7.

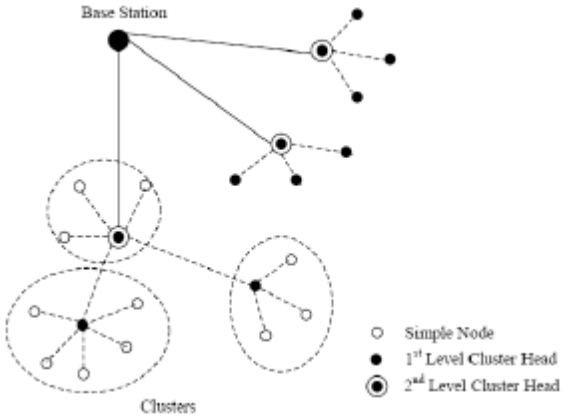


Figure 2.7: Clustering method in TEEN protocol.

Adaptive Threshold sensitive Energy Efficient sensor Network Protocol (APTEEN)

APTEEN is an adaptive version of TEEN wherein the network is able to receive information of the sensors periodically in addition to important events. For this protocol, after forming clusters, CHs send the desired parameters to the sensors that include threshold limits and the sensors' transmission schedule. In order to save energy, CHs combine all of the sensor information together and then sends it. APTEEN sends three different kinds of information through the network: 1) historical analysis of past data, 2) a one-time collection of network information for a particular moment, and 3) continuous supervision of the events that happen over a period of time [41].

2.3.3.3 Location-Based Protocols

Most of the routing protocols for wireless sensor networks require locational information of the sensors. In most cases, this information is applied by calculating the distance between the sensors in order to estimate the required energy to transmit data.

There is no basic addressing scheme in wireless sensor networks, and geographical information is helpful for using energy efficiently. For instance, if the sensors know about their locations, the base node inquiries can be sent only to the region where they are necessary so the amount of information required to be transmitted is decreased. Nevertheless, some of these protocols are designed for MANET and are also applied in wireless sensor networks. Other protocols are designed for ad hoc networks and are not appropriate for wireless sensor networks because, in their design, energy consumption is not considered.

2.3.3.4 Quality of Service Protocols

Most of the protocols relate only to the previous categories. There are only a few protocols that consider Quality of Service. For Quality of Service (QoS) protocols, the delays are considered the main issue. One such protocol is the Sequential Assignment Routing Protocol.

This protocol was the first scheme for routing in wireless sensor networks where QoS was considered. Two methods, SMAC and EAR, are suggested for this protocol [58].

Chapter 3

RANDOM GRAPH

3.1 Erdős- Rényi Random Graph

A random graph can be used to model different phenomena in communication networks. A random graph consists of a number of nodes and edges. The randomness property is based on the method for connecting the nodes by the edges. If the method has random characteristics, the final structure of graph is not deterministic and the final graph is a random graph. The concept of a random graph was first introduced by Paul Erdős and Alfréd Rényi in 1950[9] [47].

To construct a random graph, first consider two nodes randomly between all of the nodes of a graph. Then, specify one of the edges of the graph to connect those two nodes. Repeat this procedure until all of the edges of the graph connect any two nodes of the graph. If there are N nodes and K edges in the graph, in each step, we choose a pair of nodes from the N nodes then connect them by one of the K edges. Therefore, in K steps, all of the K edges connect a pair of nodes together. To find the probability distribution, the probability that an arbitrary node in the graph would be connected to k edges $\forall k \in N$ is found. The probability that there exists an edge between two nodes is p ($0< p< 1$), and the probability that a node would be connected to k nodes (have k edges for an arbitrary k) can be found.

The distribution of an edge between node v and another arbitrary node is the Bernoulli distribution with parameter p. The summation of independent and identically

distributed Bernoulli random variables has a binomial distribution. Therefore, the probability that node *v* has *k* edges would be a binomial distribution with parameter *p* and *(N-1)*. So, if x_i is a random variable and is defined as follows:

$$x_i = \begin{cases} 0 & if\ there\ is\ no\ edge\ between\ v\ and\ node\ v_i \\ 1 & if\ there\ is\ an\ edge\ between\ v\ and\ node\ v_i \end{cases}.$$

then it is equivalent to:

$$x_i = \begin{cases} 0 & with\ probability\ 1-p \\ 1 & with\ probability\ p \end{cases}.$$

Define a random variable *y* as a number of edges of node v. The probability distribution of *y* based on probability distribution of x_i, which is defined above:

$$y = \sum_{i=1}^{N-1} x_i \implies p_k = \frac{(N-1)!}{k!\,(N-k-1)!} p^k (1-p)^{N-k-1}$$

If the average number degree of nodes in graph is z, it is trivial to write $p=\frac{z}{(N-1)}$. Also, note that in this thesis the random graph considered has a huge number of nodes, i.e., $N\to\infty$; therefore, $\frac{z}{(N-1)} \approx \frac{z}{N}$. Replacing this amount instead of *p* and considering that *N* is very large, the binomial distribution tends to Poisson distribution with parameter *z* so the above equation becomes:

$$p_k = \frac{(N-1)!}{k!\,(N-k-1)!} p^k (1-p)^{N-k-1} \approx \frac{z^k e^{-z}}{k!}$$

Based on the definition for p_k, a generating function $G_0(\mathrm{x})$ for a unipartite graph can be defined. A unipartite graph is a graph in which each node can be connected to all other nodes with a positive probability. The generating function $G_0(\mathrm{x})$ is polynomial with coefficients p_k's for each x^k.

The degree of such a polynomial can be theoretically infinite when considering multiple edges between nodes.

$$G_0(\mathrm{x}) = p_0 + p_1 x^1 + p_2 x^2 + \cdots = \sum_{k=0}^{\infty} p_k x^k$$

Since p_k is the probability that a randomly chosen node has a degree k (k edges), then the distribution of p_k must be properly normalized, or:

$$G_0(1) = \sum_{k=0}^{\infty} p_k = 1.$$

Since $G_0(\mathrm{x})$ is a generating function, it is absolutely convergent on its domain, which is |x|<1. Therefore, there is no singularity in this region. This thesis concentrates on this domain for all calculations. One can derive many characteristics by working with generating functions. Some of these properties are mentioned here briefly. This thesis applies some of them in order to answer some questions about communication networks. The coefficient p_k can be obtained by getting the derivative from the generating function $G_0(\mathrm{x})$

$$p_k = \frac{1}{k!} \frac{d^k G_0}{dx^k} \Big|_{x=0}$$

The average degree of a random graph is one of the most important characteristics because the probability of having k edges is p_k and the average number of edges z, i.e., average degree $\sum_{k=0}^{\infty} k p_k$ is equal to:

$$z = \langle k \rangle = \sum_{k=1}^{\infty} k p_k = \frac{dG_0}{dx} \Big|_{x=1}.$$

Thus, it can be derived by the generating function. Generalizing this formula for finding higher moments in this distribution can be done by taking higher derivatives from the generating function. A summary of the formula for all moments is:

$$<k^n>=\sum_{k=0}^{\infty} k^n p_k = (x\frac{d}{dx})^n G_0(x)|_{x=1}.$$

The concept of $x(\frac{d}{dx})^n$ means that one takes the first derivative then multiplies that by *x* and repeats these procedures *n* times. If we have a generating function of N objects, the distribution of *m* independent realization of those objects can be found. If *m* nodes are selected randomly from a large graph with generating function $G_0(\mathrm{x})$, then the distribution of the sum of those nodes is ${G_0}^m(\mathrm{x})$. For example, suppose *m=2;* ${G_0}^m(\mathrm{x})$ would be:

$${G_0}^2(\mathrm{x}) = (\sum_{k=0}^{\infty} p_k x^k)^2 = \sum_{j=0}^{\infty}\sum_{k=0}^{\infty} p_j\, p_k x^{j+k}$$

$$= p_0p_0x^0 + (p_0p_1 + p_1p_0)x^1 + (p_0p_2 + p_1p_1 + p_2p_0)x^2 + (p_0p_3 + p_1p_2 + p_2p_1 + p_3p_0)x^3 + \cdots .$$

The coefficient of x^n in the above expression is the sum of all products $p_j p_k$ where $j+k=n$ so it will be the probability that the sum of the degrees of two nodes is *n*. The same can be said for higher powers of the generating function. There is another generating function $G_1(\mathrm{x})$ that is important in our calculation. It is the degree distribution of the degree of vertices that is reached by following a randomly selected edge. The probability of arrival at a special node with degree *k* is proportional to *k* since if a node has more edges, there is a higher chance to randomly reach that node after following a randomly chosen edge. Therefore, that node has a probability distribution degree proportional to kp_k. Since $G_1(1)$ =1 (which is true for all generating functions), the normalized distribution is:

$$G_1(x) = \frac{\sum_{k=1}^{\infty} kp_k x^k}{\sum_{k=1}^{\infty} kp_k} = x\frac{{G_0}'(x)}{{G_0}'(1)}$$

Beginning at a randomly selected node and choosing each of the edges of that node to arrive to the k closest neighbours, then the nodes reached have the degree distribution $G_1(x)$ of its other edges, i.e., all its edges except the one used to reach the node. Thus, $G_1(x)$ is less one power of *x* than $G_0(x)$ to allow for the edge used to enter the node. Therefore, the distribution of $G_1(x)$ of a random graph with average degree *z* is:

$$G_1(x) = \frac{G_0{}'(x)}{G_0{}'(1)} = \frac{1}{z} G_0{}'(x)$$

Note that the probability of any of these outgoing edges being connected to the original node is zero when the graph is very large since $\lim_{N\to\infty} N^{-1} = 0$.

The above conclusion can be generalized to calculate the generating function of the distribution for the second nearest neighbours and third nearest neighbours and so forth. For instance, the generating function of probability distribution of the second nearest neighbours for the original node can be calculated by:

$$\sum_{k=0}^{\infty} p_k {G_1}^k(x) = G_0(G_1(x))$$

The average number of second neighbours similarly is:

$$z_2 = \left[\frac{d}{dx} G_0\big(G_1(x)\big)\right]|_{x=1} = G_0{}'(1) G_1{}'(1) = G_0{}''(1)$$

The distribution of third closest neighbours is also calculated by $G_0\Big(G_1\big(G_1(x)\big)\Big)$, and this formula can be generalized for the generating function of the distribution of n^{th} nearest neighbours by calculating $G_0\big(G_1 \ldots \big(G_1(x)\big) \ldots\big)$ where $G_1(x)$ is repeated *(n-1)* times.

3.2 Component Size

In this section, connected components in random graphs and its distribution are discussed. The connected component of a graph is a maximal subset of nodes and edges where all of the nodes of the subsets are connected to each other by the edges of the subset. The subset also must be a maximum, meaning there are no other nodes that can be added to this subset and the new subset remains connected.

Let $H_1(x)$ be the generating function of the sizes of connected components which are arrived at by selecting a random edge and following it to one of its ends. By using the above definition, we can find the equation of the solution which results to find $H_1(x)$. The randomly chosen edge is connected to a node where its other outgoing edges are connected components with size distribution $H_1(x)$. The distribution of outgoing edges is $G_1(x)$ so the distribution of component size would be $G_1\big(H_1(x)\big)$, but according to the definition, it is also $H_1(x)$. Thus we have:

$$H_1(x) = xG_1\big(H_1(x)\big) \tag{3.1}$$

In the above formula, it is assumed that the probability q_k that the node reached from the random edge has k outgoing edges. Therefore, using the property of powers in the previous section, $H_1(x)$ must satisfy the following self-consistency equation, which is exactly the same as the previous one:

$$H_1(x) = xq_0 + xq_1\, H_1(x) + \, xq_2 {H_1}^2(x) + \cdots.$$

Note that if the number of the nodes is large enough, $(N \to \infty)$, the probability that there would be loop back to the original node would be $\lim_{N\to\infty} N^{-1} = 0$.

Let $H_0(x)$ be a generating function of the size of the whole component at the end of all edges connected to a randomly selected node. Therefore, the generating function for the size of the whole component $H_1(x)$ is:

$$H_0(x) = xG_0\big(H_1(x)\big) \tag{3.2}$$

Theoretically, by considering both equations (3.1) and (3.2), $H_1(x)$ and $H_0(x)$ can be found respectively, when $G_0(x)$ is given. By finding $H_0(x)$, there is a probability that a random node belongs to a component with arbitrary size s. It is enough to plug in a zero in the s^{th} derivative of $H_0(x)$.

$$H_0(\mathrm{x}) = p_0 + p_1x^1 + p_2x^2 + \cdots = \sum_{s=0}^{\infty} p_s x^s$$

$$p_s = \frac{1}{s!}\frac{d^s H_0}{dx^s}\Big|_{x=0}$$

Except in few special cases, in practice, it is very difficult to solve (3.1) since it is a non-algebraic equation. If we consider the Taylor expansion of $H_0(x)$, we can find the coefficient of x^s by iteration of (3.2) starting with $H_1 = 1$. $H_0(x)$ can be calculated up to finite order in finite time.

When the number of nodes in a graph is large ($N \to \infty$) , sometimes, a giant component of the graph appears, which is size N_G and is unbounded ($N_G \to \infty$). This giant component of the graph occupies part of graph S=$\lim_{N\to\infty} \frac{N_G}{N}$ and, as mentioned previously, the average degree of node $z = G_0{}'(1)$.

Similarly, when there is no giant component in the graph, we have the same property for the generating function of component size $H_0(x)$ (the average component <s>=$H_0{}'(1)$). In some circumstances, we can find the average component without having the exact expression of $H_0(x)$. From equation (3.2), we have:

$$< s >= H_0{}'(1) = 1 + G_0{}'(1)H_1{}'(1) = 1 + z_1 H_1{}'(1) \tag{3.3}$$

By taking the derivative from (3.1), we also have:

$$H_1{}'(1)=1+G_1{}'(1)H_1{}'(1) \Longrightarrow H_1{}'(1) = \frac{1}{1-G_1{}'(1)} \tag{3.4}$$

If $z_1 = z$ is the average degree in graph and z_2 is the average number of second neighbours, combining (3.3) and (3.4), we have:

$$< s >= 1 + \frac{{z_1}^2}{z_1 - z_2} \tag{3.5}$$

We see that in case $z_1 = z_2$, <s> diverges to infinity, and it is exactly the condition under which a giant component first appears in random graph. This point is called phase transition and is equivalent to the equation:

$G_1{}'(1) = 1.$

Instead of 1, we can replace $G_1(1)$. Then, by expanding both sides of equation, we have:

$$\sum_{k=0}^{\infty} k(k-2)p_k = 0 \tag{3.6}$$

which is the same condition at the transition phase. Note that when $\sum_{k=0}^{\infty} k(k-2)p_k > 0$ indicates the number of nodes with higher degree than 2, is quite large, which results in the existence of a giant component in the graph. Conversely, when$\sum_{k=0}^{\infty} k(k-2)p_k < 0$, the number of nodes with degree less than 3, the lower the number of components with a large size, which results in the disappearance of the giant component. When a giant component appears, (3.1) and (3.2) are still valid with little change in the definition of

$H_0(x)$, which is the generating function of the size of the components except for the giant component. Since the size of giant component is infinite in theory, $H_0(x)$ does not include the giant component in the probability distribution. This means that $H_0(1)$ is no longer equal to one, and it equals the percentage of the component where their sizes are finite. Therefore, the remaining part belongs to a component of which its size is infinite (giant component). Thus, returning to equations (3.1) and (3.2), if we assume $H_1(1)= u$, (3.1) changes to:

$$u = G_1(u) \tag{3.7}$$

and (3.2) gives the size of giant component *S*:

$$S = 1 - G_0(u) \tag{3.8}$$

The last equation for <*s*> will also be changed to a generalized one in the following way when a giant component exists in the random graph:

$$< s >= \frac{H_0{}'(1)}{H_0(1)} = \frac{1}{H_0(1)}\left[G_0\big(H_1(1)\big) + \frac{G_0{}'\big(H_1(1)\big)G_1\big(H_1(1)\big)}{1 - G_1{}'\big(H_1(1)\big)}\right]$$

$$= 1 + \frac{zu^2}{(1-S)(1-G_1{}'(u))} \tag{3.9}$$

The last equation is the same as (3.1) when *S=0* and *u=1*.

3.3 Percolation Theory on Random Graphs

One of the most interesting properties of a graph is its connectivity. In a random graph that is based on a distribution function, there is a chance for connectivity of nodes within the graph. The graph can also be made up of sub-graphs which satisfy the connectivity property. The probability distribution of size for each graph is a function of the probability distribution of degree.

If one consider Erdős-Rényi graph, the deletion of edges by probability P_c (percolation by P_c) means the average size of connected sub-graphs is decreased. If the number of the nodes in the random graph is increased when a degree distribution is a defined function, the average size of the connected sub-graphs increases. In addition, there exist some connected sub-graphs where the number of nodes goes to infinity if the number of the nodes in the graph goes to infinity. Percolation and the increasing of the number of the nodes of the graph to infinity act inversely on the connectivity of the graph, and there is a threshold for percolation probability P_c in which, above this amount, there is one connected sub-graph as a giant cluster where the number of its nodes is going to infinity. The percolation theory indicates there is a critical probability P_c such that below a threshold, each connected sub-graph has a finite size but the main graph has an infinite size. After percolation in a random graph, the average degree of the graph is changed, and, in fact, the function of degree distribution is compressed more as it is multiplied by P_c.

The new function of the degree distribution can be obtained from the original one and other characteristics of the new graph can be obtained by the new degree distribution

function. Applying this theory in a communication network shows the existence of a critical probability P_c such that below P_c the network is composed of isolated clusters but above P_c, a giant cluster spans the entire network.

Bond percolation with probability p on a graph $G=<V,E>$ is defined as the following random process: For every link e∈ E delete the link with probability p and retain it with probability $(1-p)$ randomly and independently from all other links. The percolation process is, therefore, a trimming of the edges of a graph. The percolation process may fragment otherwise connected networks into smaller sub-graphs. In fact, if $p=1$, then all the nodes will become isolated. For most large graphs, including random graphs, there exists a precise percolation probability P_c such that after percolation with any probability $p<P_c$ the graph will not have a giant connected component. The giant connected component will exist, however, for all $1 \geq p > P_c$. This critical probability P_c is called the percolation threshold.

The percolation threshold is particularly easy to calculate for random graphs. Note that after percolation with probability p on a random graph with average degree z, one is left with another random graph with average degree $z'=pz$. Therefore, the relative size of the largest connected component is S, which is the function of average degree z.

3.4 Mathematical Formulation of the Problem

In this section, a mathematical model is designed according to the optimization problem. The input to the problem is an undirected graph $G=< V, E >$. For simplicity, we assume that all links have equal, normalized capacity *1* (one unit of data on average),

and one message per unit of time can be communicated over each link without error. The communication model is an all-to-all broadcast model where each node *v* has an infinite sequence of messages $M_1^v, M_2^v, \ldots$; each of fixed size, L that it needs to broadcast to all other nodes using the gossip algorithm. $N=|V|$ is the total number of nodes in the network. Since the network is shared by all nodes, we assume that each node starts broadcasting a message every N units of time using the gossip algorithm. Suppose that node *v* broadcasts the i^{th} message M_i^v, and let $V_{i_k}^v$ be the set of all nodes that receive the k^{th}bits of i^{th} message (mathematically correct notation would be V_{ik}^v but we prefer to denote $V_{i_k}^v$ because in this notation we can interpret that first we care message *i* and after that bit *k*) then, for each node *t* ($t \neq v$), when node *v* broadcast its i^{th} message, the number of the bits received by node *t* is:

$$L_i\ (v,t) = \sum_{k=1}^{L} I(t \in V_{i_k}^v)$$

where I(·) is the indicator function. After node *v* broadcasts *n* messages, the total bits received by node *t* are:

$$\sum_{i=1}^{n} L_i\ (v,t)$$

Therefore, the total bits received by all nodes are:

$$\sum_{t \in V-v} \sum_{i=1}^{n} L_i\ (v,t) =$$

$$\sum_{t \in V-v} \sum_{k=1}^{L} \sum_{i=1}^{n} I(t \in V_{i_k}^v)$$

Since there are (*N-1*) nodes as receivers, the average number of bits received by each node per broadcasting message for all *n* messages from node *v* will be:

$$\frac{1}{N-1}\sum_{t\in V-v}\sum_{k=1}^{L}\sum_{i=1}^{n}\frac{I(t\in V_{i_k}^{v})}{n} \tag{3.10}$$

This procedure is repeated for all $v \in V$ so the overall average aggregate throughput Γ will be:

$$\lim_{n\to\infty}\frac{1}{N}\sum_{v\in V}\frac{1}{N-1}\sum_{t\in V-v}\sum_{k=1}^{L}\sum_{i=1}^{n}\frac{I(t\in V_{i_k}^{v})}{n} \tag{3.11}$$

This limit exists. The next section will show the calculation of the probability that two arbitrary nodes communicate with each other when this algorithm is repeated one time; this probability is calculated using equation (3.11). Similarly, let $E_{i_k}^{v} \subset E$ be the set of all edges that carry the k^{th} bit of M_i^v. The average traffic on each edge e during broadcasting each bit is $I(e \in E_{i_k}^{v})$, so the average traffic for sending one message, which contains L bit, is $C_i(v,e)= \sum_{k=1}^{L} \mathrm{I}(\mathrm{e} \in \mathrm{E}_{i_k}^{v})$. This is because the capacity of each edge is 1 message per unit of time if the node *v* broadcasts n sequence of messages M_i^v:

$$\Upsilon(e) = \sum_{i=1}^{n}\frac{C_i(v,e)}{n} \leq 1 \tag{3.12}$$

Therefore, the summation above should be bounded by 1 when $n \to \infty$. Time is not considered for message distribution in this problem because we are considering an infinite sequence of messages where $n \to \infty$. There is no need to consider time because of the following reason: As long as the capacity restriction for each link is satisfied, the messages can be broadcasted. Also, note that *U(v; n),* Γ, and $\Upsilon(e)$ are all random variables that will converge to their average values as n goes to infinity. As such, their values depend only on the two algorithm parameters of *p* and *L* and the topology of the graph *G*. In order to solve the optimization problem:

$$\begin{aligned} &\text{maximize } \Gamma \\ &Subject\ to: \\ &\forall v \in V, \forall e \in E: \Upsilon(e) \leq 1 \\ &1 \geq p \geq 0, L \geq 0, \end{aligned} \tag{3.13}$$

Start with the following simple lemma:

Lemma 2.1:

If $L \leq 1/p$, then $\Upsilon(e) \leq 1, \forall v \in V$ and $\forall e \in E$.

Proof:

Take any edge *e*. For every message, there are two cases: either one of the two ends of this link receives a message, in which case the message will be passed over this link with probability *p*, or none of the ends of this link receives the message, in which case the probability that the link carries the message will be zero. Therefore, the overall probability that the link carries the message is, at most, *p*. Plugging this into the equation for $\Upsilon(e)$, the average load on each link will be, at most, *pL*, which proves the lemma. ■

Lemma 2.2:

Let *deg*(*v*) be the degree of a node *v*. Then:

$$\Gamma \leq \sum_{v\in V, t\in V, v\neq t} \min\{deg(v), deg(t)\}$$

Proof:

Γ is equal to the average throughput of the communication between all pairs of the nodes in the network. The throughput of the communication between two pairs of nodes cannot exceed the max-flow between the two. The value of the max-flow between a pair of nodes *v* and *t* is upper bounded by the capacity of any cut that separates the two nodes. The set of all the edges connected to *v* defines a cut whose value is *deg*(*v*). Similarly, the set of edges connected to *v′* defines a cut with value *deg*(*t*), which establishes the lemma. ■

If *x* and *y* are degrees of two arbitrary nodes in an Erdős-Rényi graph and if *E(x)=E(y)=k*, then calculate *E(min(x,y))*, which is the maximum throughput that can be transmitted from one node to the other one according to lemma 2. Because *x* and *y* are two independent and identically distributed random variables:

$$z = \min(x, y) \Rightarrow$$

$$p(z \geq z_0) = p(\min(x, y) \geq z_0) = p(x \geq z_0)p(y \geq z_0) = (p(x \geq z_0))^2$$

For any non-negative integer random variable *z*, there is the following relationship:

$$E(z) = \sum_{z_0=0}^{\infty} p(z \geq z_0) \Rightarrow$$

$$E(\min(x, y)) = E(z) = \sum_{z_0=0}^{\infty} p(z \geq z_0) = \sum_{z_0=0}^{\infty} (p(x \geq z_0))^2 \approx \sum_{d=z_0+1}^{\infty} (\frac{k^d e^{-k}}{d!})^2$$

One can calculate the last part numerically for different values of *k*.

3.5 Optimal Gossip Parameters for Random Graphs

The optimal value of p depends solely on the network topology. It can be shown how the optimal *p* and the corresponding Γ can be found for a class of random graphs. What follows is a description of what random graphs are.

3.6 The Connection between Random Broadcast and Percolation

There is a close connection between random broadcast in gossip and the percolation process. Consider the network as a graph *G=<V,E>* with average degree *z.* When a node *v* starts broadcasting its message, it chooses each of its edges by probability *p* and then sends its own bit on those edges to its neighbours. Its neighbours repeat the same procedure except that they do not send the bit back to its origin. This distribution method repeats for all the nodes that receive this bit. It is of interest to maximize the number of nodes that receive the bit using fewer edges for this purpose due to the edge capacity restriction, which is stated by (3) to get the maximum throughput on average. If, before broadcasting each bit of node *v*, the edges of the graph are percolated with probability (*1-p*), a graph G_p with average degree *pz* is generated. Then, node *v* starts sending its bit to its neighbours on all of the remaining edges, and each node receiving the bit repeats the same strategy to distribute it except for sending it back to the node from where it originated. At the end of the distribution, all nodes in the connected component of the sub-graph to which node *v* belongs will receive the bit of node *v*. After random broadcast

with probability *p* starting from a node *v*, a set of nodes $V_{i_k}^{v}$ will receive that particular bit. Now consider a percolation process with the probability *(1-p)* and take the connected component after percolation to which the node *v* belongs. Statistically, this connected component is exactly equivalent to the set $V_{i_k}^{v}$. When node *v* broadcasts each bit of its message, it uses its links with probability *p*, which is like percolating its edge with probability *(1-p).* Then, each neighbour receiving the message repeats the same procedure but do not consider the link on which they received that bit. The process goes on until all the nodes percolate all of their links except the one on which they received the bit. There is a case that one edge may be percolated two times with probability *p* when one node does not know its neighbour already has the bit it is going to send. For example, the edge between them was percolated by its neighbour with probability *p* and was not selected to be one of the edges that carries the bit, so the neighbour did not send that bit to the node. The edge between them will be considered percolated twice by probability *p*. If the first time it is percolated but the second time it is not percolated and is selected to carry that bit, we can ignore the first percolation since it is selected to carry that bit the second time and it does not affect the throughput and traffic on that edge (the capacity restriction is not affected).

If the number of nodes is large ($N \to \infty$) and *(1-p)* $<p_c$, then we have a unique giant connected component with size $\lambda\ (pz)N$ where *z* is the average degree of the graph before percolation, the size of other component s is, at most, $\Theta(\log N)$, and the number of such components is, at most, $\Theta(\log N)$ [9]. So, the probability that two random nodes will be located in same component is:

$$\lim_{N\to\infty}(\lambda^2(pz)+\sum_{v\in s,s\in G_p}\left(\frac{\Theta(\log N)}{N}\right)^2)$$

$$\leq \lim_{N\to\infty}(\lambda^2(pz)+\Theta(\log N)\left(\frac{\Theta(\log N)}{N}\right)^2)=\lambda^2(pz)$$

But,

$$\lambda^2(pz)\leq\lambda^2(pz)+\sum_{v\in s,s\in G_p}\left(\frac{\Theta(\log N)}{N}\right)^2$$

So,

$$\lim_{N\to\infty}(\lambda^2(pz)+\sum_{v\in s,s\in G_p}\left(\frac{\Theta(\log N)}{N}\right)^2)=\lambda^2(pz)$$

when $N\to\infty$, and, therefore, we should just consider the case when node *v* belongs to the giant component.

If we consider (3.11) for Γ,

$$\lim_{n\to\infty}\sum_{i=1}^{n}\frac{I(t\in V_{i_k}^v)}{n}$$

is exactly the probability that two nodes will be located in same connected component after percolation. Due to the capacity restriction, *L=1/p* according to *Lemma 1*, which means this broadcasting method is done for all *1/p* bits of message M_i^v started by node *v*. In a percolated graph model, each edge exits with probability *p* so that it carries the bit if it is in the component containing node *v*. Therefore, we should maximize $\lambda^2(pz)/p$ for each message with size *1/p*. Similarly, $\lambda^2(pz)/p$ is equivalent to:

$$\lim_{n\to\infty} \sum_{k=1}^{L} \sum_{i=1}^{n} \frac{I\left(t \in V_{i_k}^{v}\right)}{n}$$

in (3-11).

Note that Γ is a random variable that converges to its average value $\lambda^2(pz)/p$. All other summation parts in (3-11) take the average of $\lambda^2(pz)/p$ for different nodes in the network, which will be, again, (3-11) since $\lambda^2(pz)/p$ does not depend on particular node *v*.

Borrowing the results in previous sections to calculate the objective function Γ as a function of the broadcast probability *p*, one can find the size of the largest connected component receiving a message; when the broadcast probability is *p*

$$\lambda\,(pz) = 1 - e^{-pz\lambda\,(pz)} \tag{3.14}$$

Now, a message sent by a node v is received by another node *t* only if both nodes are part of the same connected component. Since the size of all other connected components is negligible compared to the size of the giant connected component, we can simply confine our averaging to those nodes that are part of the giant connected component. The probability that both transmitter and receiver are in a giant cluster is, therefore, $\lambda^2(pz)$.

Since the size of each message is *L=1/p*, the average aggregate throughput will be equal to:

$$\Gamma = \lambda^2(pz)/p$$

The goal is, therefore, to maximize Γ, subject to (3.14). If we consider λ as a function of *p*, solving $\frac{d\Gamma}{dp}$=0 with (3.14):

$$g = \frac{\lambda^2}{p} \Longrightarrow \frac{dg}{dp} = 0 \Longrightarrow \frac{\left(2\lambda \frac{d\lambda}{dp}\right) p - \lambda^2}{p^2} = 0$$

$$\Longrightarrow \frac{d\lambda}{dp} = \frac{\lambda}{2p} \tag{3.15}$$

The derivative with respect of λ from (3.14) is:

$$\frac{d\lambda}{dp} = z\left(\lambda + p\frac{d\lambda}{dp}\right) e^{-pz\lambda\ (pz)} \tag{3.16}$$

Using (3.15) in above equation:

$$\frac{\lambda}{2p} = z\left(\lambda + p\frac{d\lambda}{dp}\right) e^{-pz\lambda\ (pz)} \Longrightarrow \frac{1}{3(pz)} = e^{-pz\lambda\ (pz)} \tag{3.17}$$

changing variable *(pz)=u* and considering (3.14) and (3.17) simultaneously:

$$\frac{1}{3u} = e^{-u\lambda} \& \lambda = 1 - e^{-u\lambda} \Longrightarrow \lambda = 1 - \frac{1}{3u} \tag{3.18}$$

$$\frac{1}{3u} = e^{-u\lambda} \Longrightarrow \lambda = \frac{\ln 3u}{u} \tag{3.19}$$

Comparing (3.18) and (3.19) results in:

$$1 - \frac{1}{3u} = \frac{\ln 3u}{u} \tag{3.20}$$

Solving this equation numerically obtains λ_1 = 0, *pz = u = 0.333* and λ_2 = 0.851, *pz = u = 2.2371*. For λ_1= 0, we do not have a giant cluster and the throughput would be zero. For λ_2 =0.851, we would have maximum throughput where $\frac{\lambda^2}{p}$ =*0.3237z*.

3.7 Exponential Random Graph

A random graph is a model for creating graphs devised by Paul Erdős in the 1950s [47] [8]. An exponential random graph is one of the most applicable random graphs first introduced by Holland and Leinhardt [27]. Since then, it has been used for modeling different phenomena extensively, particularly in social and biological networks [21][53][54][56]. While a random graph is a random structure, one can find the following properties for these graphs that hold true if the size of the graph is large [47]:

Let p_k be the probability that one node has *k* edges in the graph so that

$$p_k = \left(1 - e^{\frac{-1}{\kappa}}\right) e^{\frac{-k}{\kappa}} \tag{3.21}$$

where $\kappa > 0$ is a constant. Therefore, the degree distribution of the nodes in an exponential random graph generating function is:

$$G_0(x) = \sum_{k=0}^{\infty} p_k x^k = \left(1 - e^{\frac{-1}{\kappa}}\right) \sum_{k=0}^{\infty} e^{\frac{-k}{\kappa}} x^k = \frac{1 - e^{\frac{-1}{\kappa}}}{1 - x e^{\frac{-1}{\kappa}}} \tag{3.22}$$

Choosing a random edge and then following that to reach a vertex, the distribution of the remaining outgoing edge of that vertex will be $G_1(x) = \frac{G_0{}'(x)}{G_0{}'(1)}$; therefore:

$$G_1(x) = \left(\frac{1 - e^{\frac{-1}{\kappa}}}{1 - x e^{\frac{-1}{\kappa}}}\right)^2 \tag{3.23}$$

The average degree of the nodes is obtained by:

$$z = G_0{}'(1) = \frac{e^{\frac{-1}{\kappa}}}{\left(1 - e^{\frac{-1}{\kappa}}\right)} \tag{3.24}$$

Random graphs are often used to analyze the performance of graph algorithms due to their simplicity and analytical tractability. While most real world graphs cannot be as closely modeled as random graphs, the analysis of the performance of protocols on random graphs provides useful insight into the suitability of the algorithms in question, as suggested by the results in this thesis.

An important structural property of a graph is its connectivity. Each graph has a largest connected component that may not contain all of the nodes if the graph is not completely connected.

Let N_G be the size of the largest connected component in a graph *G*. Random graphs with any finite average degrees are not connected. One of the most famous results of the random graph theory is the following [32]: Take a family of random graphs with *N* nodes and with average degree $z>1$; the relative size of the largest connected component N_G/N converges to a constant $S>0$ depending only on *z* as *N* goes to infinity. For any $z<1$, the relative size of the largest connected component converges to zero as *N* goes to infinity. It can also be shown that for $z>1$, the size of the second largest component is exponentially smaller than the size of the largest connected component, as *N* goes to infinity. In other words, random graphs have, at most, one giant connected component that appears when $z>1$ and disappears when $z<1$. This abrupt jump is called a phase transition. These results and the functional form of *S* are at the heart of our results in this thesis.

3.7.1 Broadcasting in network with Exponential model

We can then borrow the results in percolation theory in section (3.2) to calculate the objective function Γ as a function of the broadcast probability *p*. In particular, we can find the size of the largest connected component receiving a message when the broadcast probability is *p*. One can repeat the method as in the previous section for exponential graphs:

$$S = 1 - G_0(u) \tag{3.25}$$

where u satisfies the equation below:

$$u = G_1(u) \tag{3.26}$$

By using (3.21) and (3.22), after some algebra and considering $u \leq 1$, we have:

$$u = \frac{2 - e^{\frac{-1}{\kappa}} - \sqrt{e^{\frac{-1}{\kappa}}\left(4 - 3e^{\frac{-1}{\kappa}}\right)}}{2e^{\frac{-1}{\kappa}}} \tag{3.27}$$

Replacing this amount in (3.25) and using (3.21) we have:

$$S = 1 - \sqrt{\frac{2 - e^{\frac{-1}{\kappa}} - \sqrt{e^{\frac{-1}{\kappa}}\left(4 - 3e^{\frac{-1}{\kappa}}\right)}}{2e^{\frac{-1}{\kappa}}}} . \tag{3.28}$$

(3.24) gives

$$e^{\frac{-1}{\kappa}} = \frac{z}{z+1}.$$

So by replacing $\frac{z}{z+1}$ in (3.28):

$$S = 1 - \frac{2}{z + \sqrt{z(z+4)}} \tag{3.29}$$

Since after percolation $z'=pz$, replace pz in (3.29):

$$S(pz) = 1 - \frac{2}{pz + \sqrt{pz(pz+4)}}. \tag{3.30}$$

Now a bit sent by a node v is received by another node v' only if both nodes are part of the same connected component. Since the size of all other connected components is negligible compared to the size of the giant connected component, this confines the average to those nodes that are part of the giant connected component. The probability that both transmitter and receiver are in the giant cluster is $S^2(pz)$. Since the size of each message is $L=1/p$, the average aggregate throughput will be equal to:

$$\Gamma = \frac{S^2(pz)}{p} \tag{3.31}$$

Our goal is, therefore, to maximize Γ, subject to (3.30).

Considering S as a function of p, solving $\frac{d\Gamma}{dp}$=0 in (3.31):

$$\frac{d\Gamma}{dp} = 0 \Longrightarrow \frac{\left(2S\frac{dS}{dp}\right)p - S^2}{p^2} = 0 \Longrightarrow 2p\frac{dS}{dp} = S \tag{3.32}$$

The derivative, with respect to S from (3.30), after some algebra:

$$\frac{dS}{dp} = \frac{2z(2 + pz + \sqrt{pz(pz+4)})}{\sqrt{pz(pz+4)}} \tag{3.33}$$

Using (3.33) and (3.30) in (3.32):

$$2p\frac{2z(2 + pz + \sqrt{pz(pz+4)})}{\sqrt{pz(pz+4)}} = 1 - \frac{2}{pz + \sqrt{pz(pz+4)}} \tag{3.34}$$

Thus, solving (3.34) with respect to pz and considering $pz>0$:

$$pz = 1 + \frac{6}{3+\sqrt{17}} \approx 1.8423 \Longrightarrow S \approx .6069 \;\&\; \Gamma \approx .2017z$$

Chapter 4

SIMULATION RESULTS

4.1 Erdős-Rényi Random Graph

This thesis simulates the network model for several random graphs with various degree distributions. Graph sizes of N=100 nodes in Matlab (see Appendix A) are used. Each node sends a total of 50 messages that are received by the other nodes in the graph. For each value of average degree, the aggregate throughput is then calculated for a range of values of the broadcast probability *p*. The experiments for *z=k=2, 3, 4, 5, 6, 12*, and *25* are shown in the following pages. As the theoretical calculations suggest, for any *z =k > 2*, there exists an optimal *p* for which the aggregate throughput is optimized.

To simulate the Erdős-Rényi graph with average degree z, it is enough to consider a $N \times N$ matrix as an adjacent matrix related to the network graph. Each element of the matrix has a value of one with the probability p where $p = \frac{z}{N}$; otherwise, it is zero. Figure 4.1 shows such matrices for different graphs when N is four.

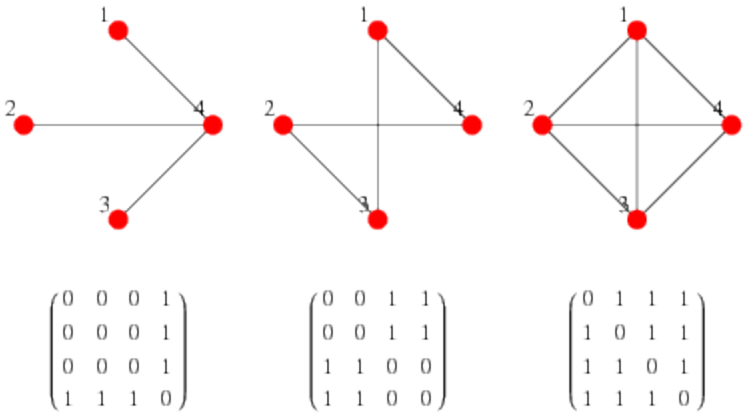


Figure 4.1: Adjacent Matrices related to different graphs when N=4.

When we create the adjacent matrix of the graph, we can create the graph of the network. In this model, the repeated bits that may be received by the nodes are not considered:

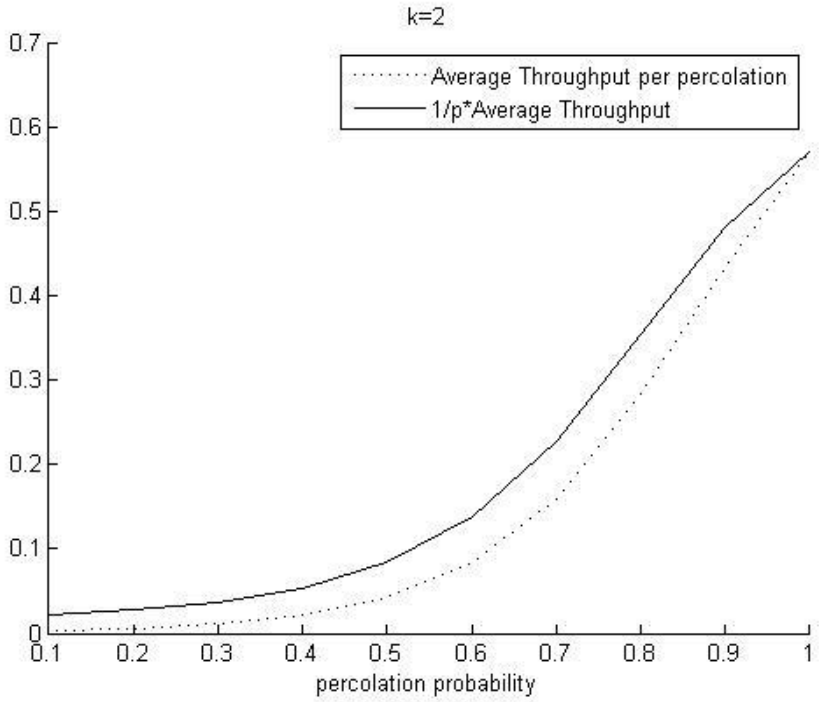


Figure 4.2: Gossip message throughput as a function of gossip probability: for average degree z=k= 2. In this case, there is no maximum point for p<1.

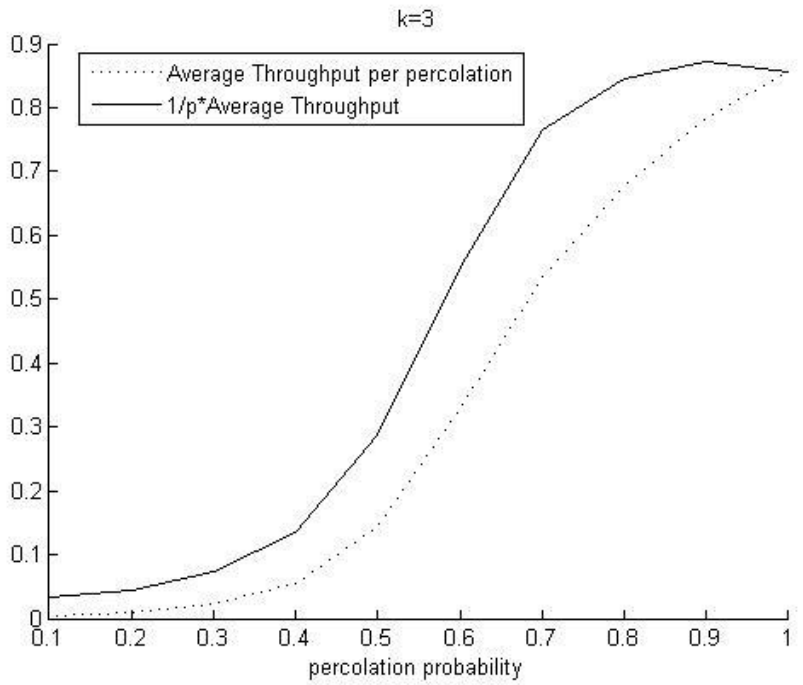


Figure 4.3: Gossip message throughput as a function of gossip probability: for average degree z= k=3, the gossip algorithm is optimized at a probability 0<p*≈. 9<1.

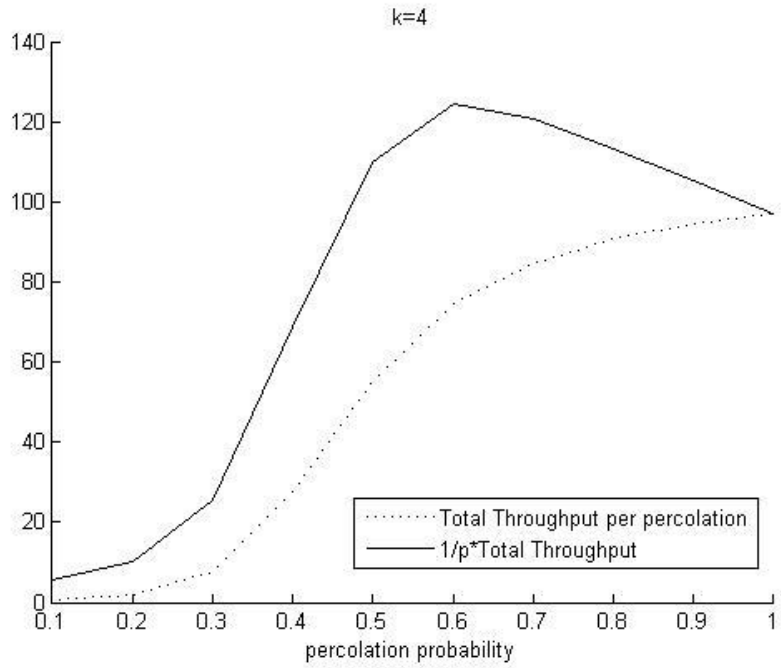


Figure 4.4: Gossip message throughput as a function of gossip probability: for average degree z= k=4, the gossip algorithm is optimized at a probability 0<p*≈. 6<1.

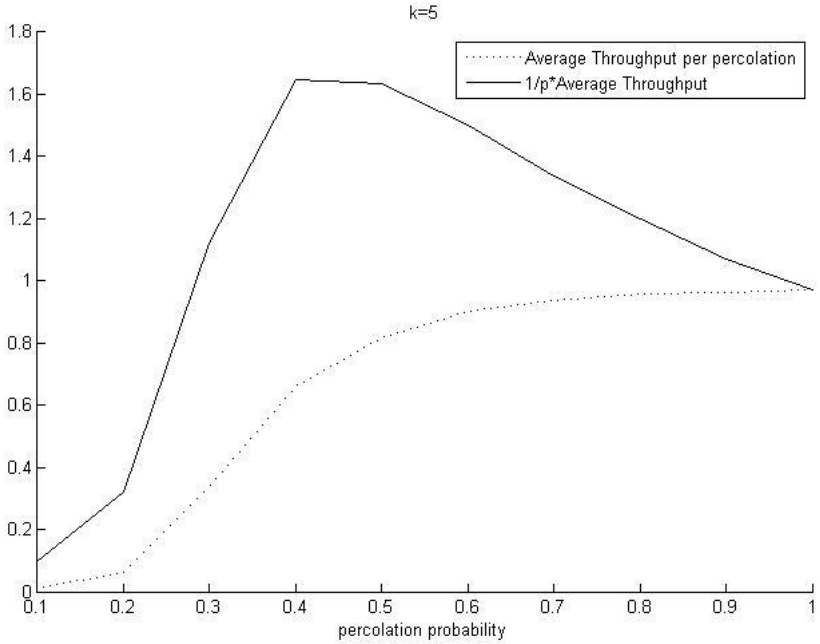


Figure 4.5: Gossip message throughput as a function of gossip probability: for average degree z= k=5, the gossip algorithm is optimized at a probability $0<p^*\approx .4<1$.

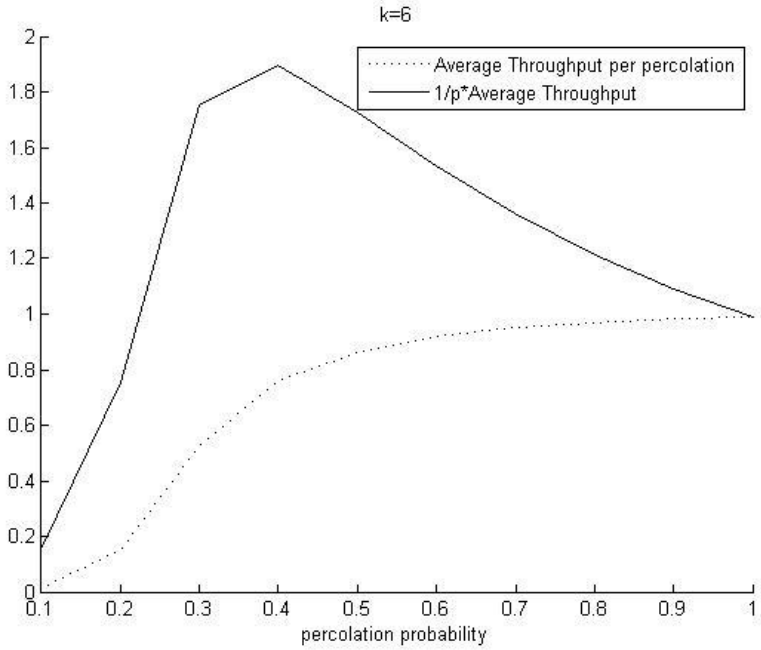


Figure 4.6: Gossip message throughput as a function of gossip probability: for average degree z= k=6, the gossip algorithm is optimized at a probability $0<p^*\approx .4<1$.

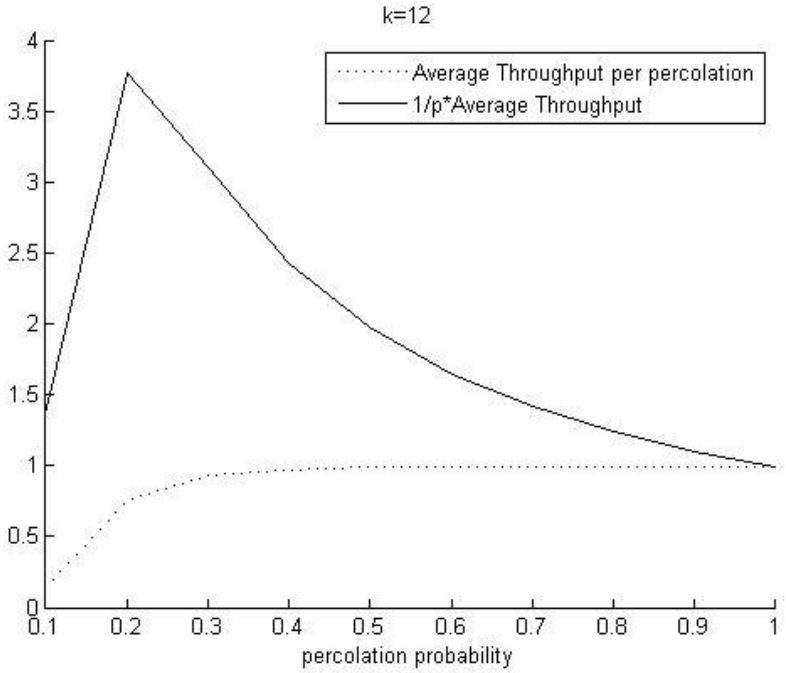


Figure 4.7: Gossip message throughput as a function of gossip probability: for average degree z= k=12, the gossip algorithm is optimized at a probability 0<p*≈. 2<1.

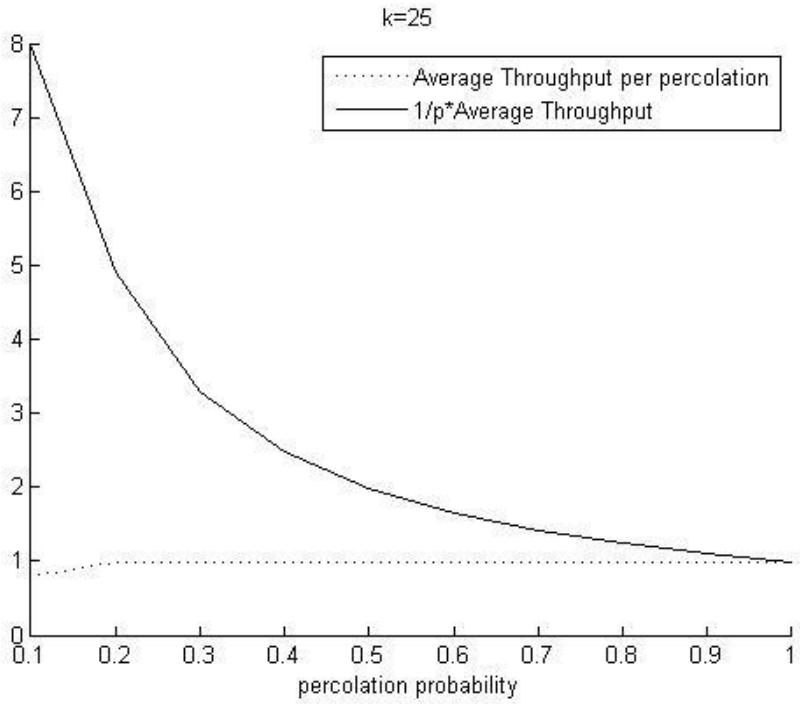


Figure 4.8: Gossip message throughput as a function of gossip probability: for average degree z= k=25, the gossip algorithm is optimized at a probability 0<p*≈. 1<1.

The next two figures show the value of the optimal aggregate throughput for different values of average degree z=k, which is found both analytically and through simulation. The simulations clearly validate the calculations. We use Lemma 2 to calculate an upper bound on Γ for each value of z (calculated using Lemma 2). This upper bound is also depicted in the next two figures.

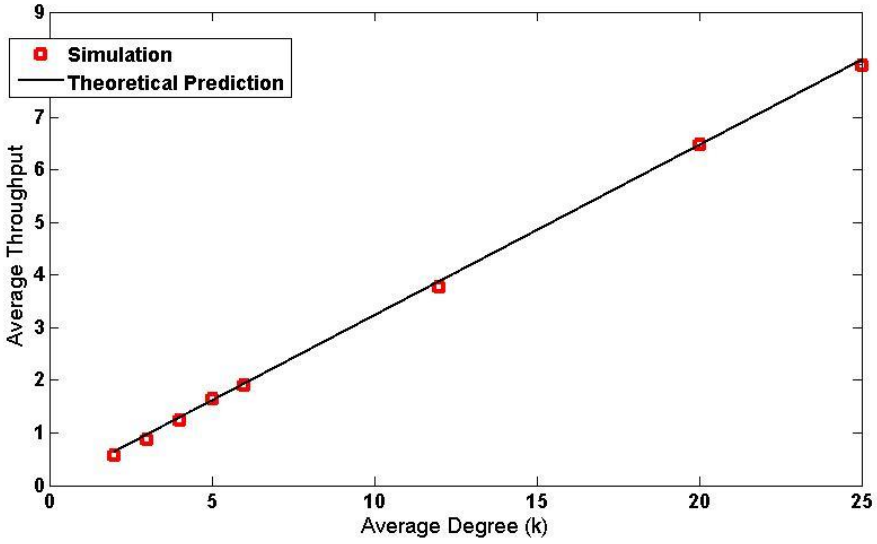


Figure 4.9: The comparison between maximum 1/p* throughputs derived by simulations and theoretical prediction.

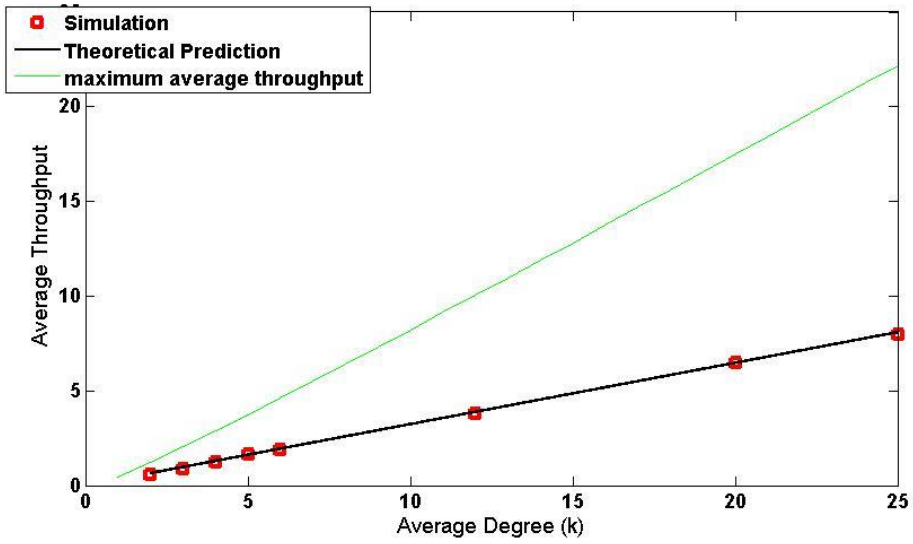


Figure 4.10: The comparison between maximum 1/p*throughputs derived by simulations and theoretical prediction and upper limit derived by Lemma 2.

4.2 Exponential Random Graphs

The same simulation was repeated for the exponential random graph. To understand the effect of the number of nodes on the final results, simulations for different numbers of nodes N= 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, and 10,000 were performed. All of the simulations for averages z=5, 10, and 15 were also performed. The following figures show all of the different average changes in the number of nodes, and they do not affect the optimal probability amount and the optimal throughput.

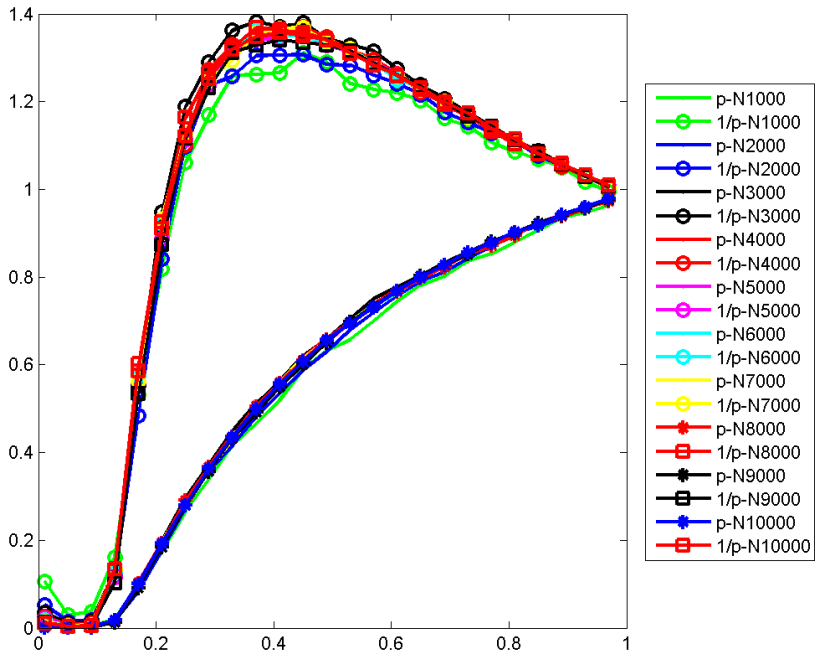


Figure 4.11: Gossip message throughput as a function of gossip probability: for average degree z=5 for different numbers of nodes.

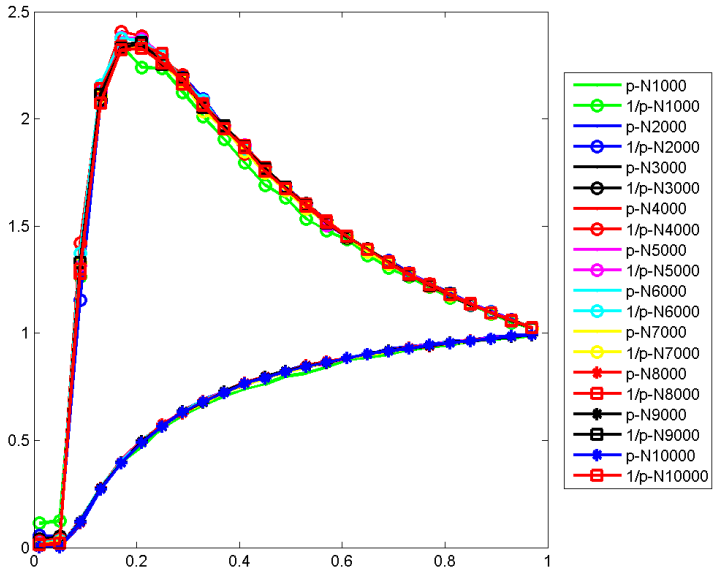


Figure 4.12: Gossip message throughput as a function of gossip probability: for average degree z=10 for different numbers of nodes.

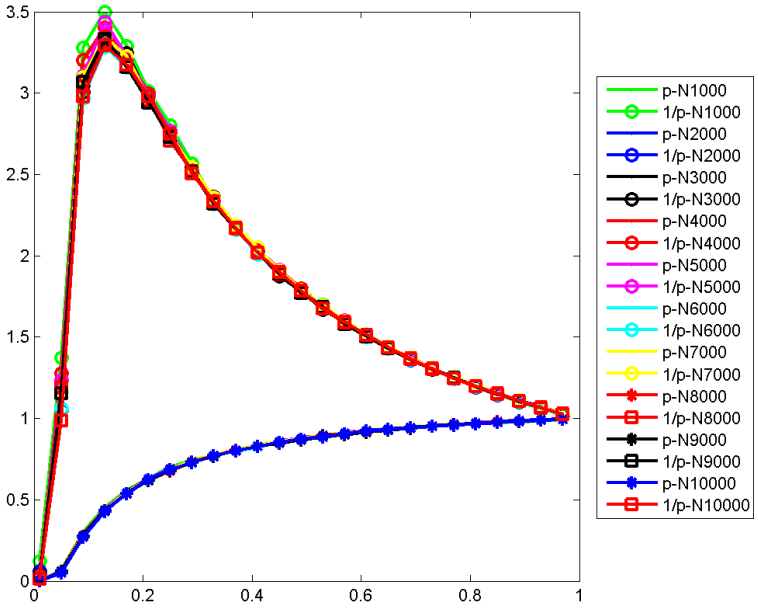
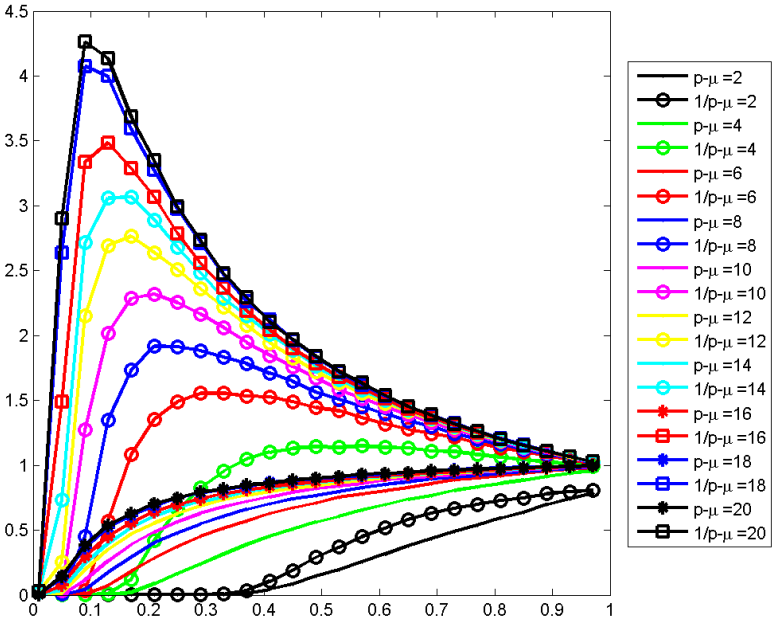


Figure 4.13: Gossip message throughput as a function of gossip probability: for average degree z=15 for different numbers of nodes.



1. **Figure 4.14: Gossip message throughput as a function of gossip probability: for different average degree and for N=5000.**

Focusing on the simulation for N=5000 nodes and for different average degree z, the simulation behaves exactly like the Erdős-Rényi graph in that the optimal probability p^* is decreasing by increasing the average degree. The average degrees in the simulation were z=2, 4, 6, 8, 10, 12, 14, 16, 18, and 20. In the following four figures, the throughput is shown for z=8, 12, 18, and 20.

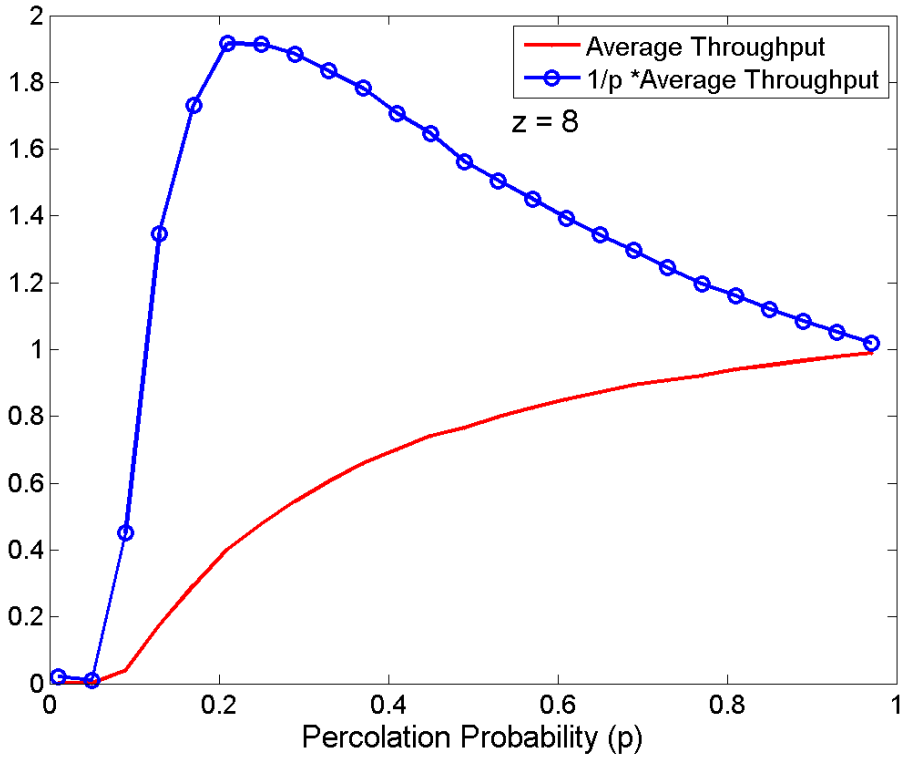


Figure 15.14: Gossip message throughput as a function of gossip probability: for average degree z=8 and for N=5000.

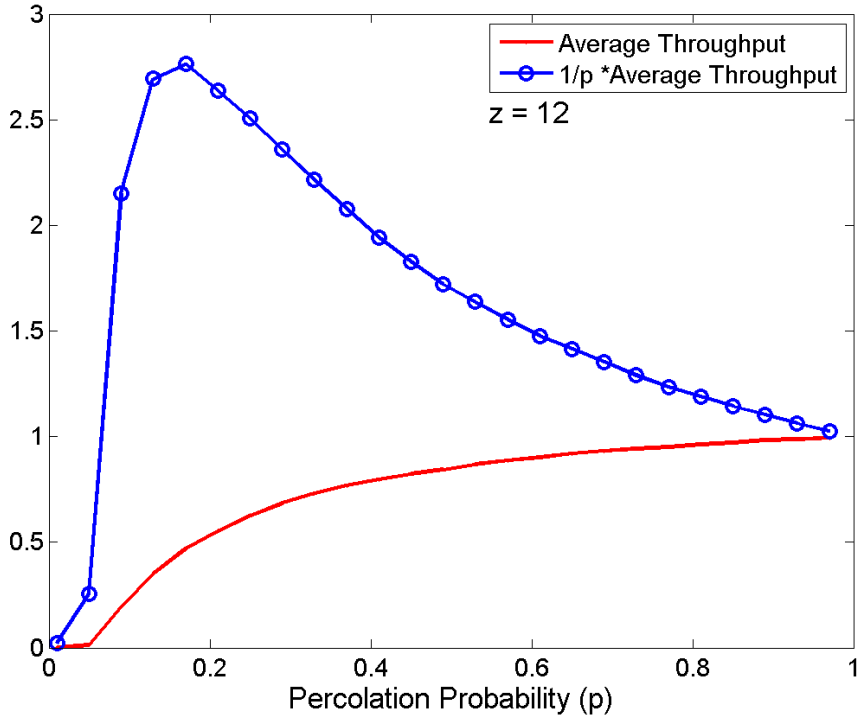


Figure 4.16: Gossip message throughput as a function of gossip probability: for average degree z=12 and for N=5000.

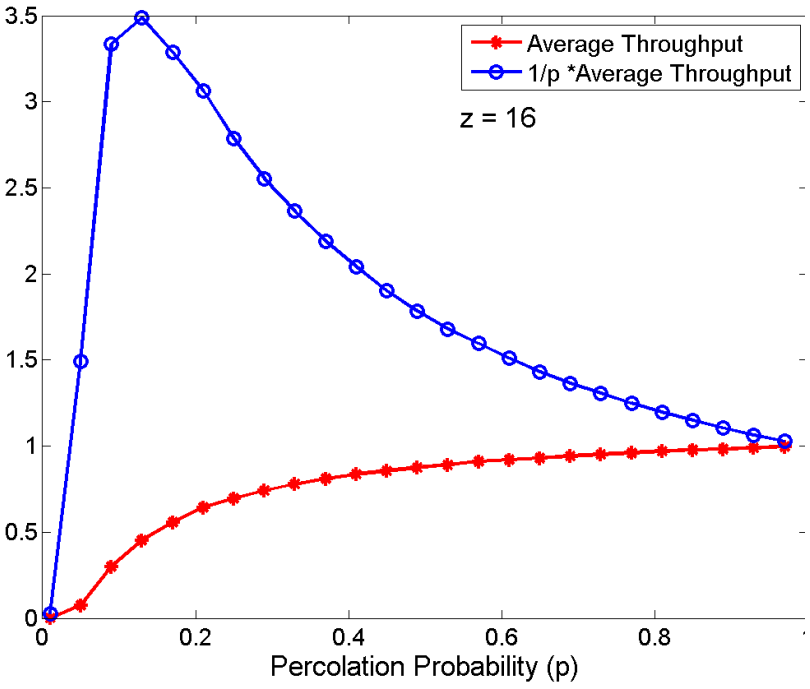


Figure 4.17: Gossip message throughput as a function of gossip probability: for average degree z=16 and for N=5000.

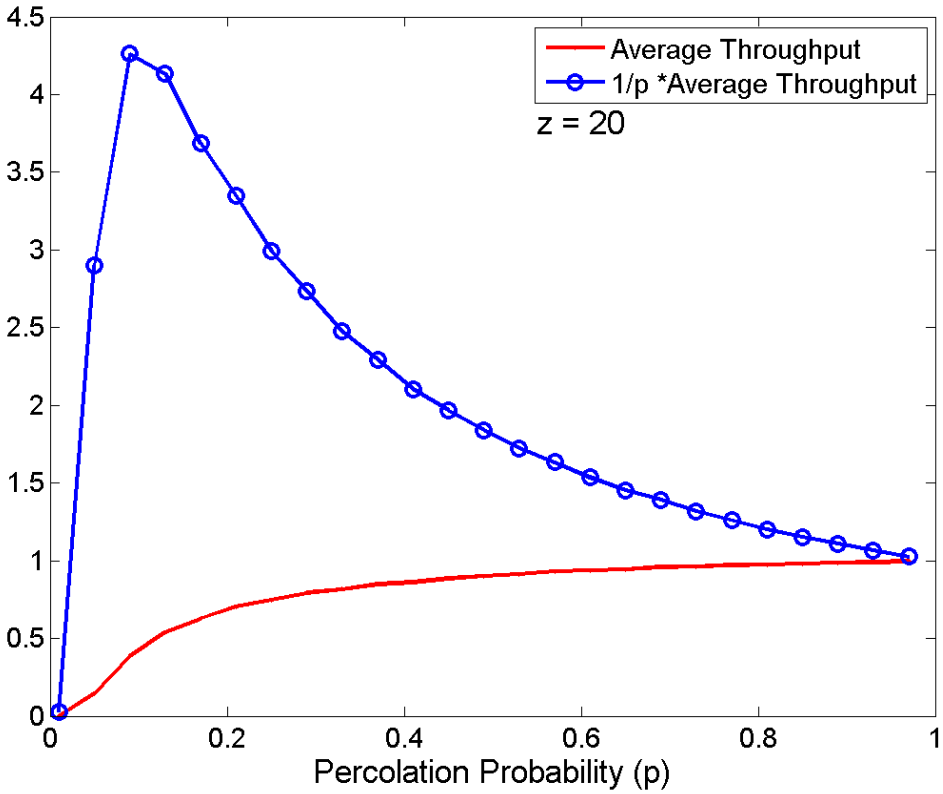


Figure 4.18: Gossip message throughput as a function of gossip probability: for average degree z=20 and for N=5000.

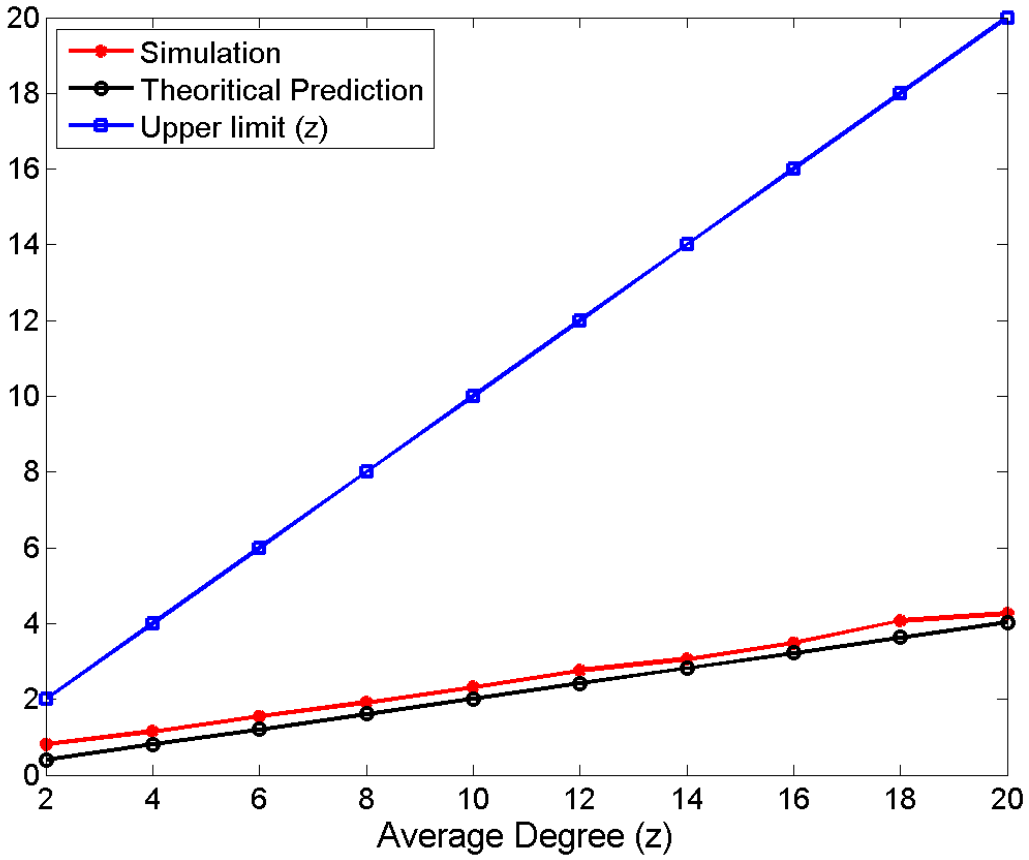


Figure 4.19: The comparison between maximum 1/p*throughputs derived by simulations and theoretical prediction and upper limit derived by Lemma 2.

Chapter 5

SUMMARY AND FUTURE DIRECTIONS

As discussed in the first and second chapters, the next generation networks suffer from specific environmental restriction problems that can be solved by efficient algorithms. However, all of the algorithms that have been studied in different networks have their own constraints for different conditions.

This thesis analyzed a theoretical framework for the gossip algorithm in order to maximize an aggregate message throughput in an all-to-all communication model. In this framework design, all of the nodes try to broadcast messages to all other nodes using the gossip algorithm. Examples of this model include sensor data aggregation in sensor networks and search in P2P networks. We found that the aggregate throughput is optimized by a careful choice of the broadcast probability and message size. This is due to a fundamental tradeoff: if the broadcast probability p is large, a larger set of nodes will receive a given message. However, the links will carry a large number of redundant messages. On the other hand, if the broadcast probability is too small, the gossip messages will “die down" quickly. The results in this thesis show that an optimal broadcast probability exists. This optimal point is the result of a balance between the two forces which are broadcast probability and capacity restriction.

Lack of global information about the network in each node presents big challenges for algorithms to operate in an efficient way. This algorithm does not require global information about the network; only a probability distribution is required.

Energy resource constraints are problems for future networks and can be managed well using this algorithm because algorithm computation is lower, which saves energy. In addition, each sensor does not need to send data to all of its neighbours and, therefore, the algorithm saves energy in this way as well.

The main assumption of the algorithm is that the network has a large number of users. The algorithm works with a degree distribution function, and it is robust with respect to changing users because mobility, including connection and disconnection of the nodes, does not affect the degree distribution, on average, which is a reasonable assumption in a large network.

Although the results are derived for a simplified model of the gossip algorithm, and for specific classes of random networks, they still shed light on the fundamental tradeoffs involved in the gossip protocol and can be used as guidelines for optimizing the performance of these algorithms. Future work includes a generalization of the analytical developments with respect to random graphs and arbitrary degree distributions [47] where links also have random capacities. This research could also be extended to more complex gossip algorithms beyond the stateless random broadcast algorithm studied in this thesis. For this framework, a constant answer was found as a broadcasting probability for each node independent of the situation. For future work, if an answer can be a function of local information, such as node degree, the final answer can be more generalized and, therefore, could provide a better output result.

According to the simulations, the algorithm may have an even better lower bound if the numbers of users are not large, but more mathematical proof is required.

REFERENCES:

[1] I. F. Akyildiz, W. Su, Y. S. Subramanian, E. Cayirci., "Wireless sensor networks: a survey", *Journal of Computer Networks*, Vol. 38, March 2002, pp. 393-422.

[2] R. Albert and A.-L. Barab´asi, "Statistical mechanics of complex networks," Reviews of Modern Physics, vol. 74, pp. 47–97, Jan2002.

[3] H. Backhaus, S. Krause, " QuON – a Quad-Tree Based Overlay Protocol for Distributed Virtual Worlds", Proc. Massively Multiuser Virtual Environments (MMVE '09), 2009.

[4] H. Balakrishnan H., M.F. Kaashoek, D. Karger, R. Morris and I. Stoica, "Looking up data in P2P systems," Communications of the ACM, vol. 46 no.2, 2003.

[5] A. I. R. Bar-Yehuda, O. Goldreich, "On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization,," Computer and System Sciences, vol. 45, pp. 104 – 126, 1992.

[6] J.L. Bentley, "Multidimensional binary search trees in database applications," IEEE Trans. on Software Engineering, vol. 5, no. 4, 1979.

[7] C. Carlsson and O. Hagsand, "DIVE - A Multi-user virtual reality system,". Proc. of the Virtual Reality Annual International Symposium, 1993.

[8] I. Chlamtac and S. Kutten, "On broadcasting in radio networks problem analysis and protocol design," IEEE Transactions on Communications, vol. 33, pp. 1240 – 1246, 1985.

[9] F. Chung and L. Lu, "Connected components in random graphs with given expected degree sequences," Annals of Combinatorics, vol. 6, pp. 125 – 145, 2002.

[10] I. Clarke, O. Sandberg, B. Wiley and T.W. Hong, “Freenet: A Distributed Anonymous Information Storage and Retrieval System,” Lecture Notes in Computer Science, 2000, pp. 44-46.

[11] E. Cronin, B. Filstrup, A. R. Kurc, and S. Jamin, “An efficient synchronization mechanism for mirrored game architectures,” Proc. NetGames ’02, 2002, pp. 67–73.

[12] A. Czumaj and W. Rytter, “Broadcasting algorithms in radio networks with unknown topology,” in Proceeding of the Annual IEEE Symposium on Foundation of Computer Science, 2003, pp.492–501.

[13] A. Dimakis, A. Sarwate, and M. Wainwright, “Geographic gossip: efficient aggregation for sensor networks,” in Proceedings of the Fifth International Conference on Information Processing in Sensor Networks, 2006, pp. 69–76.

[14] A. Dimakis, S. Kar, J. Moura, M. Rabbat, and A.Scaglione, “Gossip algorithms for distributed signal processing,” Proceedings of the IEEE, vol. 98, pp. 1847–1864, Nov 2010.

[15] Distributed Systems Group, The PARADISE Project. 2004, Stanford University: Stanford, CA. http://wwwdsg.stanford.edu/paradise.html

[16] P. Druschel and A. Rowstron, “Pastry: Scalable, distributed object location and routing for large-scale peer-to peer systems,” Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), 2001.

[17] L. Fan, H. Taylor, and P.W. Trinder, “Design Issues for Peer-to-Peer Massively Multiplayer Online Games,” Proc. Massively Multiuser Virtual Environments (MMVE '09), 2009.

[18] E. Frecon and M. Stenius, “DIVE – a scalable network architecture for distributed

virtual environments," Distributed Systems Engineering Journal (Special issue on Distributed Virtual Environments), vol. 5, 1998.

[19] T.A. Funkhouser, "RING: A Client-Server System for Multi-User Virtual Environments," ACM SIGGRAPH Special Issue on Symposium on Interactive 3D Graphics, 1995.

[20] D. Ganesan , R. Govindan, S. Shenker and D. Estrin, "Highly Resilient, Energy Efficient Multipath Routing in wireless Sensor Networks", *ACM Mobile Computing and Communications Review*, vol. 1, no. 2, 2002, pp. 11-25.

[21] S. M. Goodreau, "Advances in exponential random graph (p*) models applied to a large social network," Social Networks, vol. 29, pp. 231–248, 2007.

[22] C. Greenhalgh and S. Benford, "MASSIVE: a collaborative virtual environment for teleconferencing," ACM Trans. on Computer-Human Interaction (TOCHI), 1995, pp. 239-261.

[23] Z. Haas, J. Halpern, and L. Li, "Gossip-based ad hoc routing," in Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, 2002, pp. 1707–1716.

[24] S. Hedetniemi, S. Hedetniemi, and A. Liestman., "A survey of gossiping and broadcasting in communication networks,"Networks, vol. 18, pp. 319 – 349, 1988.

[25] W. R. Heinzelman, A. P. Chandrakasan and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks", *Proc. of the 33rd IEEE International Conference on System Sciences*, Honolulu, USA, Jan. 2000, pp. 1–10.

[26] W. R. Heinzelman, A. P. Chandrakasan and H. Balakrishnan, "An Application-

Specific Protocol Architecture for Wireless Microsensor Networks", *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, Oct. 2002, pp. 660-670.

[27] P. W. Holland and S. Leinhardt, "An exponential family of probability distributions for directed graphs," Journal of American Statistical Association, vol. 76, pp. 33 – 50, 1981.

[28] IEEE Computer Society, IEEE Std 1278.1a-1998: IEEE standard for distributed interactive simulation - application protocols. 1998: New York.

[29] C. Intanagonwiwat, R. Govindan, D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks", *Proc. of the ACM MobiCom'00*, Boston, USA, 2000, pp. 56–67.

[30] R. Jafari, A. Encarnacao, A. Zahoory, F. Dabiri, H. Noshadi, M. Sarrafzadeh, "Wireless Sensor Networks For Health Monitoring", *Proc. of The Second ACM/IEEE International Conference on Mobile and Ubiquitous Systems*, San Diego, USA, Jul. 2005, pp. 479-481.

[31] R. H.Katz, J. M. Kahn and K. S. J. Pister, "Mobile Networking for Smart Dust", *Proc. of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, Seattle, USA, Aug. 1999, pp. 350-355.

[32] J. Keller and G. Simon, "Solipsis: A massively multi-participant virtual world," Proc Int. Conf. Parallel and Distributed Techniques and Applications, 2003, pp. 262–268.

[33] D. Kowalski and A. Pelc, "Optimal deterministic broadcasting in known topology radio networks," Distributed Computation"., vol. 19, pp. 185 – 195, 2007

[34] E. Lety, L. Gautier, and C. Diot. Mimaze, “A 3D multi-player game on the internet,” Proc. of the 4th International Conf on Virtual System and Multimedia, vol. 1, 1998, pp. 84–89.

[35] S. Krause, “A Case for Mutual Notification: A Survey of P2P Protocols for Massively Multiplayer Online Games,” in Proceedings of NetGames 2008 Network and Systems Support for Games, 2008, pp. 28–33.

[36] B. Krishanamachari, D. Estrin and S. Wicker, “The Impact of Data Aggregation in Wireless Sensor Networks”, *Proc. of the International Workshop of Distributed Event Based Systems (DEBS)*, Vienna, Austria, Jul. 2002, pp. 575-578.

[37] S. Lindsey and C. S. Raghavendra, “PEGASIS: Power Efficient GAthering in Sensor Information Systems," *Proc. of the IEEE Aerospace Conference*, Big Sky, USA, March 2002, pp. 1125-1130.

[38] D. Liu and M. Prabhakaran, “On randomized broadcasting and gossiping in radio networks,” Lecture Notes in Computer Science, vol. 2387, pp. 340 – 349, 2002.

[39] M.R. Macedonia, M. J. Zyda , D. R. Pratt , P. T. Barham and S. Zeswitz, “NPSNET: A Network Software Architecture for Large Scale Virtual Environments,”Presence, vol. 3, no. 4, 1994.

[40] A. Manjeshwar and D. P. Agarwal, “TEEN: A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks”, *Proc. of the IEEE IPDPS,* San Francisco, USA, Apr. 2001, pp 23-26.

[41] A. Manjeshwar and D. P. Agarwal, “APTEEN: A Hybrid Protocol for Efficient

Routing and Comprehensive Information Retrieval in Wireless Sensor Networks," *Proc. of the IEEE IPDPS*, Fort Lauderdale, USA, Apr. 2002, pp.195-202.

[42]Market Forcast : DFC Intelligence : www.dfcint.com

[43] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," Revised Papers from the First International Workshop on Peer-to-Peer Systems, 2002, pp.53-65.

[44] D.C. Miller and J.A. Thorpe, "SIMNET: The Advent of Simulator Networking," Proc. of the IEEE special Issue on Distributed Interactive Simulation, vol 83, no. 8, 1995, pp. 1114-1123

[45] R.Min, M. Bhardwaj, S. Cho, E. Shih, A. Sinha, A. Wang, and A. Chandrakasan, "Low Power Wireless Sensor Networks", *Proc. of Internation Conference on VLSI Design*, Bangalore, India, Jan. 2001, pp. 205-210.

[46] B. Moon, H.V. Jagadish, C. Faloutsos, J.H. Saltz, "Analysis of the Clustering Properties of the Hilbert Space-Filling Curve," IEEE Trans. Knowledge and Data Engineering,2001, pp. 124—141.

[47] M. Newman, S. Strogatz, and D. Watts, "Random graphs with arbitrary degree distributions and their applications," Physical Review E, vol. 64, pp. 026 118–1 – 026 118–17, 2001.

[48] J. M. Rabaey, M. J. Ammer, J. L. da Silva, D. Patel and S. Roundry, "PicoRadio supports ad hoc ultra-low power wireless networking", *IEEE Computer Magazine*, Vol. 33, Jul. 2000, pp. 42-48.

[49] S. S. Ram, A. Nedic, and V. Veeravalli, “Asynchronous gossip algorithms for stochastic optimization,” in Decision and Control, 2009. ICDC 2009. IEEE International Conference, Dec 2009, pp. 3582–3586.

[50] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, “ A scalable content addressable network,” Proc. of ACM SIGCOMM, 2001,pp. 161–172.

[51] G. Robins, P. W. T. Snijdres, M. Handcock, and P. Pattison, “Recent developments in exponential random graph (p*) models for social networks,” Social Networks, vol. 29, pp. 192–215, 2007.

[52] M., F. Sarrafzadeh, Dabiri, R. Jafari, T. Massey and A. Nahapetian, “Low Power Light-weight Embedded Systems (Tutorial)”, *Proc. of the International Symposium on Low Power Electronics and Design (ISLPED’06)*, Tegernsee, Germany, Oct. 2006, pp. 207-212.

[53] N. Sarshar, P. Boykin, and V. Roychowdhury, “Percolation search in power law networks: making unstructured peer-to-peer networks scalable,” in Proceedings of the Fourth International Conference on Peer-to-Peer Computing, Aug 2004, pp. 2–9.

[54] Z. M. Saul and V. Filkov, “Exploring biological network structure using exponential random graph models,” Bioinformatics, vol. 23, pp. 2604 – 2611, 2007.

[55] C. Schurgers and M. B. Srivastava, “Energy efficient routing in wireless sensor networks”, *Proc. of the IEEE Military Communications Conference (MILCOM)*, McLean, USA, 2001, pp. 357-361.

[56] S. H. L. Simpson and P. J. Laurienti, “Exponential random graph modeling for complex brain networks,” PLoS One, vol. 6, p. e20039, May 2011.

[57] G. Singh, G., L. Serra, W. Png, A. Wong and H. Ng, " BrickNet: Sharing Object Behaviors on the Net," Proc. of the Virtual Reality Annual International Symposium (VRAIS'95), 1995, pp. 19-25.

[58] K. Sohrabi, B. Manriquez and G. Pottie, "Near-ground wideband channel measurements", *Proc. of the IEEE Vehicular Technology Conference*, New York, USA, 1999, pp. 571-574.

[59] I. Stoica, R. Morris, D. Karger, MF. Kaashoek and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," Proc. of the ACM SIGCOMM, 2001, pp. 149 – 160.

[60] I.J. Taylor, and A.B. Harrison, "Gnutella," Springer Computer Communications and Networks, From P2P and Grids to Services on the Web, 2009, pp. 181-196.

[61] S. Tilak, N. B. Abu-Ghazaleh and W. B. Heinzelman, "A Taxonomy of Wireless Microsensor Network Models", *ACM Mobile Computing and Communications Review*, Apr. 2002, pp. 28-36.

[62] S. Yamamoto, Y. Murata, K. Yasumoto, and M. Ito, "A distributed event delivery method with load balancing for mmorpg," Proc. 4th ACM SIGCOMM workshop on Network and system support for games(NETGAMES), 2005, pp. 1-8.

[63]W. Ye , J. Heidemann and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks", *Proc. of the IEEE INFOCOM 2002*, New York, USA, Jun. 2002, pp. 1567-1576.

[64] B. Zhao, J. Kubiatowicz and A. Joseph, "Tapestry: An infrastructure for faulttolerant wide-area location and routing," Tech. Rep. UCB/CSD-01-1141, University of California at Berkeley 2001.

[65] R. Zhou and K. Hwang, “Gossip-based reputation aggregation for unstructured peer-to-peer networks,” in Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International, Mar 2007, pp. 1–10.

H. S. Wilf, Generatingfunctionology, 2nd Edition, Academic Press, London (1994).

M. Molloy and B. Reed, “A critical point for random graphs with a given degree sequence,” Random Structures and Algorithms 6, 161–179 (1995).

[40] M. Molloy and B. Reed, “The size of the giant component of a random graph with a given degree sequence,” Combinatorics, Probability and Computing 7, 295–305(1998).

A.-L. Barab´asi and R. Albert, “Emergence of scaling in random networks,” Science 286, 509–512 (1999).

[23] R. Albert, H. Jeong, and A.-L. Barab´asi, “Diameter of the world-wide web,” Nature 401, 130–131 (1999).

[13] W. Aiello, F. Chung, and L. Lu, “A random graph model for massive graphs,” in Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (2000).

[14] L. A. N. Amaral, A. Scala, M. Barth´el´emy, and H. E. Stanley, “Classes of small-world networks,” Proc. Natl Acad. Sci. USA 97, 11149–11152 (2000).

[36] M. E. J. Newman, “The structure of scientific collaboration networks,” Proc. Natl. Acad. Sci. USA 98, 409–415
(2001)

APPENDIX A

The program designs an Erdős- Rényi graph with N=100 and p=.03. Then, it broadcasts a packet with length 1/Pc through the graphs from all the nodes and takes the average from throughputs and plots that with respect to Pc.

```
g = sparse([],[],true,100,100);
%for p=.1:.1:.4
p=.03;
   for i=1:1:100
    for j=(i+1):1:100
        if (rand(1)<=p)
           g(i,j)=1;
        end;
    end;
   end;
    g=(g+g');
      tg=biograph(g);
      for u=1:numel(tg.nodes)
             tg.Nodes(u).Color= [1,1,1];

       end;
   Pc=.1;
      r=0;

        for z=1:1:10
            for k=1:numel(tg.nodes)
          %    B=zeros(numel(tg.nodes),numel(tg.nodes));
         for reptition=1:1:10
         A=zeros(numel(tg.nodes),numel(tg.nodes));
        tm=g;
        tg=biograph(g);
         for u=1:numel(tg.nodes)
              tg.Nodes(u).Color= [1,1,1];

         end;

        tg.Nodes(k).Color= [1,0,0];
         t=1;
         while (t>0)
         for j=1:numel(tg.nodes)
             if tg.Nodes(j).Color== [1,0,0]
               for i=1:numel(tg.nodes)

                 if (tg.Nodes(i).Color==[1,1,1])
                     if (tm(i,j)==1)&&(rand(1)<=Pc)&&(A(i,j)==0)
                      tg.Nodes(i).Color=[1,0,0];
```

```
                t=t+1;

                A(j,i)=1;
                A(i,j)=1;
                % if B(k,i)==0
                    r=r+1;
                 %    B(k,i)=1;
                % end;
                  end;
               end;
              end;
                tg.Nodes(j).Color=[0,1,0];
                t=t-1;
            end;
         end;
         end;

         end;
          end;
        end;
          plot(Pc,r/(10*numel(tg.nodes)),'k:s');
          hold on


%end;
 Pc=.2;
      r=0;

       for z=1:1:10
            for k=1:numel(tg.nodes)
              % B=zeros(numel(tg.nodes),numel(tg.nodes));
         for reptition=1:1:5
        A=zeros(numel(tg.nodes),numel(tg.nodes));
       tm=g;
       tg=biograph(g);
        for u=1:numel(tg.nodes)
             tg.Nodes(u).Color= [1,1,1];

         end;

       tg.Nodes(k).Color= [1,0,0];
        t=1;
        while (t>0)
        for j=1:numel(tg.nodes)
            if tg.Nodes(j).Color== [1,0,0]
              for i=1:numel(tg.nodes)

                if (tg.Nodes(i).Color==[1,1,1])
                    if (tm(i,j)==1)&&(rand(1)<=Pc)&&(A(i,j)==0)
                     tg.Nodes(i).Color=[1,0,0];
                  t=t+1;

                  A(j,i)=1;
                  A(i,j)=1;
                 %  if B(k,i)==0
```

```
                    r=r+1;
                 %    B(k,i)=1;
                % end;
                  end;
             end;
            end;
              tg.Nodes(j).Color=[0,1,0];
              t=t-1;
          end;
     end;
     end;

     end;
      end;
    end;
      plot(Pc,r/(10*numel(tg.nodes)),'k:s');
      hold on


for Pc=.3:.1:.4
  r=0;

    for z=1:1:10
         for k=1:numel(tg.nodes)
             % B=zeros(numel(tg.nodes),numel(tg.nodes));
      for reptition=1:1:3
     A=zeros(numel(tg.nodes),numel(tg.nodes));
    tm=g;
    tg=biograph(g);
     for u=1:numel(tg.nodes)
          tg.Nodes(u).Color= [1,1,1];

      end;

    tg.Nodes(k).Color= [1,0,0];
     t=1;
     while (t>0)
     for j=1:numel(tg.nodes)
         if tg.Nodes(j).Color== [1,0,0]
           for i=1:numel(tg.nodes)

             if (tg.Nodes(i).Color==[1,1,1])
                 if (tm(i,j)==1)&&(rand(1)<=Pc)&&(A(i,j)==0)
                  tg.Nodes(i).Color=[1,0,0];
               t=t+1;

               A(j,i)=1;
               A(i,j)=1;
              %  if B(k,i)==0
                    r=r+1;
                 %    B(k,i)=1;
                 %end;
                  end;
           end;
          end;
```

```
          tg.Nodes(j).Color=[0,1,0];
          t=t-1;
      end;
  end;
  end;

  end;
   end;
 end;
   plot(Pc,r/(10*Pc*3*numel(tg.nodes)),'k:s');
   hold on
 end;
 %end;

  for Pc=.5:.1:.7
r=0;

  for z=1:1:10
      for k=1:numel(tg.nodes)
       %  B=zeros(numel(tg.nodes),numel(tg.nodes));
   for reptition=1:1:2
  A=zeros(numel(tg.nodes),numel(tg.nodes));
 tm=g;
 tg=biograph(g);
  for u=1:numel(tg.nodes)
       tg.Nodes(u).Color= [1,1,1];

   end;

 tg.Nodes(k).Color= [1,0,0];
  t=1;
  while (t>0)
  for j=1:numel(tg.nodes)
      if tg.Nodes(j).Color== [1,0,0]
        for i=1:numel(tg.nodes)

          if (tg.Nodes(i).Color==[1,1,1])
              if (tm(i,j)==1)&&(rand(1)<=Pc)&&(A(i,j)==0)
               tg.Nodes(i).Color=[1,0,0];
            t=t+1;

            A(j,i)=1;
            A(i,j)=1;
             %if B(k,i)==0
                r=r+1;
             %    B(k,i)=1;
             %end;
              end;
         end;
        end;
          tg.Nodes(j).Color=[0,1,0];
          t=t-1;
      end;
  end;
  end;
```

```
        end;
         end;
       end;
         plot(Pc,r/(10*Pc*2*numel(tg.nodes)),'k:s');
         hold on
       end;
       %end;
hold on;
for Pc=.8:.1:1
     r=0;

       for z=1:1:10
            for k=1:numel(tg.nodes)
          %   B=zeros(numel(tg.nodes),numel(tg.nodes));
         for reptition=1:1:2
        A=zeros(numel(tg.nodes),numel(tg.nodes));
       tm=g;
       tg=biograph(g);
        for u=1:numel(tg.nodes)
             tg.Nodes(u).Color= [1,1,1];

         end;

       tg.Nodes(k).Color= [1,0,0];
        t=1;
        while (t>0)
        for j=1:numel(tg.nodes)
            if tg.Nodes(j).Color== [1,0,0]
              for i=1:numel(tg.nodes)

                if (tg.Nodes(i).Color==[1,1,1])
                    if (tm(i,j)==1)&&(rand(1)<=Pc)&&(A(i,j)==0)
                     tg.Nodes(i).Color=[1,0,0];
                  t=t+1;
                  A(j,i)=1;
                  A(i,j)=1;
                  % if B(k,i)==0
                      r=r+1;
                  %     B(k,i)=1;
                 %  end;
                    end;
               end;
              end;
                tg.Nodes(j).Color=[0,1,0];
                t=t-1;
            end;
        end;
        end;

        end;
         end;
      end;
         plot(Pc,r/(10*2*Pc*numel(tg.nodes)),'k:s');
         hold on
       end;
```

```
%end;
```