Spring 2013

Social-context based routing and security in delay tolerant networks

Roy A. Cabaniss

Department: Computer Science

SOCIAL-CONTEXT BASED ROUTING AND SECURITY IN DELAY TOLERANT NETWORKS

by

ROY ALAN CABANISS

A DISSERTATION

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

in Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

2013

Dr. Sanjay Madria, Advisor
Dr. Sriram Chellappan
Dr. Wei Jiang
Dr. Bruce M. McMillin
Dr. Jagannathan Sarangapani

PUBLICATION DISSERTATION OPTION

This dissertation consists of four articles prepared in the style required by the journals or conference proceedings in which they were published:

Pages 21 to 44, ”Dynamic Social Grouping Based Routing in a Mobile Ad-Hoc Network”, was published in the 18th IEEE International Conference on High Performance Computing (HiPC 2010), Dona Paula, Goa, India.

Pages 45 to 65, ”DSG-N^2: A group-based social routing algorithm”, was published in the IEEE Wireless Communications and Networking Conference (WCNC 2011), Cancun, Mexico.

Pages 66 to 105, ”Social Group Detection Based Routing in Mobile Ad Hoc Networks”, is under submission to the Wireless Networks Journal by Springer.

Pages 106 to 142, ”Three Point Encryption (3PE) - Secure Communications in Delay Tolerant Networks”, is under submission to the IEEE Transactions on Dependable and Secure Computing (TDSC).

Pages 143 through 163, ”Content Distribution in Delay-Tolerant Networks using Social Context”, is under submission to the 9th IEEE International Wireless Communications and Mobile Computing Conference (IWCMC 2013), Sardinia, Italy.

ABSTRACT

Delay Tolerant Networks (DTNs) were originally intended for interplanetary communications and have been applied to a series of difficult environments: wireless sensor networks, unmanned aerial vehicles, and short-range personal communications. There is a class of such environments in which nodes follow semi-predictable social patterns, such as wildlife tracking or personal devices. This work introduces a series of algorithms designed to identify the social patterns present in these environments and apply this data to difficult problems, such as efficient message routing and content distribution.

Security is also difficult in a mobile environment. This is especially the case in the event that a large portion of the network is unreliable, or simply unknown. As the network size increases nodes have difficulty in securely distributing keys, especially using low powered nodes with limited keyspace. A series of multi-party security algorithms were designed to securely transmit a message in the event that the sender does not have access to the destinations public key. Messages are routed through a series of nodes, each of which partially decrypts the message. By encrypting for several proxies, the message can only be intercepted if all those nodes have been compromised. Even a highly compromised network has increased security using this algorithm, with a trade-off of reduced delivery ratio and increased delivery time.

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank a few of the people that made this work possible. First, my advisor Sanjay Madria, whose support, advice, and encouragement guided my efforts in this field.

The faculty at the Missouri University of Science and Technology have been an invaluable aid, and I would like to thank them for their help in researching areas I didn't know existed before I came here. Special thanks go to my advisement board, Sriram Chellappan, Bruce McMillin, Jag Sarangapani, and Wei Jiang. Also, thanks to Sahra Sedigh and Daniel Tauritz.

I would like to thank my lab partners, whose endurance should be applauded. They have aided my research in many little ways, from open discussions, algorithm checking, and reviewing writing, and I am in their collective debt.

Finally, thanks go to my family, including my father Roy Franklin, my mother Candi, and my two sisters Amber and Rhiannon. Raising me was not a simple task, and I appreciate it.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

LIST OF ALGORITHMS

LIST OF TABLES

1. INTRODUCTION

When routing messages through a Delay Tolerant Network (DTN), efficient routing techniques utilize environmental information for efficiency and speed. In a social environment (e.g. human carried devices), nodes follow semi-predictable patterns based on the social context. It is possible augment message routing in a DTN by based message routes on this social behavior. Furthermore, security in a DTN is difficult to implement. Without access to a trusted third party capable of verifying a device or its key data, the trust between devices is limited to direct contact. While there are methods for verifying identity, ranging from timing analysis to mundane physical contact [1] [2], these methods only work in direct contact. Thus, the security information, especially the public key, can only be transmitted to a device while in direct contact. Even with these limitations, messages can be sent in a compromised network with high confidentiality.

Consider a global conference of researchers. Several of the researchers have met before and they tend to form groups that encounter each other frequently. An organizer gathering data from these users may identify the social groups formed, using the Dynamic Social Grouping algorithm to efficiently collect such information as participation surveys or menu selections. Another user with access to similar group information may want to send a message quickly and efficiently to another researcher at the conference. Rather than simply hold the message until they come into contact again, he can forward the message to a group the recipient is participating in by using the Dynamic Social Grouping - Node to Node algorithm. If the message is intended to be private, sending the message through the network is risky. As a new contact, the sender does not have the destination's public key, and requesting the key from nearby nodes is hazardous. While he could encrypt the message for and trust the proxy to encrypt the message for the final destination, the proxy then has access to the plaintext of the message. A more secure alternative is

to implement Three Point Encryption by sending the message through a series of mid-points.

The first paper introduced is the basis for Dynamic Social Grouping (DSG), a routing algorithm originally designed for wildlife data collection. Presented in Section 2.4.2, this protocol uses information about the social dynamics of the nodes to efficiently route collected information to a basestation. DSG begins with the assumption that the nodes are being carried by humans or animals that follow certain social patterns. The nodes consistently encounter members of the groups in which they participate. The nodes first identify the contact strength with other nodes and form small groups when two devices are strongly connected. These small groups identify potential group merges, creating larger groups that accurately reflect the environment. The protocol then uses this group data to identify which nodes and groups have a high probability of quickly delivering a message to a basestation. The algorithm was tested using real-world data collected from the IEEE Infocom conference, in which participants were asked to carry devices for 3 to 4 days to measure their contact patterns. The contact data was used by a simulation to reveal that using the group data could deliver messages very close to optimal levels at greatly reduced energy and bandwidth consumption, compared to Epidemic and Probabilistic routing.

With the previous algorithm demonstrating that using social data could improve the efficiency of node-to-basestation routing, the next step was to expand on the algorithm to implement node-to-node routing. The algorithm was modified for inexpensive human communications over a DTN. The DSG - Node to Node algorithm (DSG-N^2) presented in Section II identifies consistent contacts of a node and expands on the group merge logic. This paper expands upon the routing algorithm in order to allow users to send messages directly to other users, rather than to a basestation. This objective requires awareness of individual nodes' contact patterns, complicating the routing procedure significantly. To prevent multicollinearity and provide a better proof of concept, a longer dataset provided by the MIT Reality Project was used. The simulation results demonstrated that the node-

to-node routing problem could also be addressed using social group data with high efficiency.

Having confirmed the advantages of routing using social dynamics, advances on the routing algorithms were considered. When the social groups are formed and identified, alternate methods of using that information to route messages effectively were considered. Previously the Performance Based Probability was used. This based the nodes' estimated probability of delivering a message on their performance with previous messages. It is also possible to base probability on the contact patterns, tracking which nodes come into contact frequently. This method, called Contact Based Probability, is capable of delivering messages faster and more reliably, but at a cost of higher message traffic as messages follow indirect contacts to be delivered. Another method routes the message to the nearest group containing the destination, then floods the message among the group members. This method uses replication in parts of the network nearby the destination, as measured by contact frequency, to ensure rapid message delivery. These modifications to the social routing algorithms were presented in Section III.

Section IV addresses security in a DTN. Security is made more difficult by limited resources, intermittent communications, and easily monitored signals. While cryptography allows secure communications when given a known key, it is difficult for devices in a large-scale DTN to securely share keys with each other without direct contact. For this reason, the Three-Point Encryption algorithm was designed to secure messages without direct access to the destination's key. This algorithm is designed to function in a a compromised network. Messages are routed through a series of proxies. Each proxy partially decrypts the message, resulting in a message being compromised only if all proxies are compromised. Chaining and Fragmenting are the two applications of Three Point Encryption. The Chaining algorithm sends the message through the proxies in sequence, resulting in longer delivery times and higher security. However, the Fragmenting algorithm separates the message into distinct sections, sending each fragment of the key in parallel, allowing the final

node to decrypt the message only once a certain subset have been received. This results in a higher delivery ratio - But, if the adversary has compromised enough of the network, the adversary can read the messages as well, resulting in lower security. Both processes have trade-offs, but both work to improve security in a difficult environment.

Social context data is applicable to the problem of content distribution, as addressed in Section V. In a Mobile Ad hoc Network (MANET), nodes can attempt to dynamically distribute relevant files, such as advertising, media, or news articles. To conserve communication resources, nodes can be appointed as mobile data repositories, also called throwboxes. A node in such an environment can retrieve data from a repository, rather than the original data source. The optimal selection of nodes to act as repositories can be augmented by identifying the social groups in the environment, based on the supposition that socially similar nodes will have similar data interests. The Social Content Distribution (SCD) schema identifies social groups, uses the group data to identify which nodes are likely to request data items, and uses this information to position the mobile repositories.

2. LITERATURE REVIEW

2.1. PRELIMINARIES

The DTN is a networking architecture characterized by long delays in message transmissions [3]. These delays may be caused by disconnected networks, high disruption rate, or low bandwidth between devices. Also called Disruption Tolerant Networks, the Bundle Protocol is used by grouping a message into a single unit called a bundle. This bundle contains contains all of the information needed to deliver the message to its destination. These bundles are stored in a device's message buffer to be transmitted when the device either comes into contact with the destination or with another node that is more capable of delivering the message. This store-and-forward method allows a message to make incremental steps toward successful delivery, coming closer to delivery with each transmission. Different routing algorithms use different metrics to determine which devices are more capable of delivery; in general, each algorithm is designed for a specific environment of devices.

A subset of DTNs are Mobile Ad Hoc Networks (MANETs) (these terms are sometimes used interchangeably). These networks consist of several devices (a portion of which are mobile) and must organize themselves to deliver messages successfully [4]. The mobility patterns used by the nodes can vary from random mobility (oceanic probes), scheduled patterns (e.g bus schedules), bundling together (e.g. highway traffic), or more social patterns (e.g. human carried devices). Routing and content distribution algorithms can function more efficiently, in terms of time-to-deliver and battery consumption, by utilizing the mobility patterns the nodes follow.

2.2. ROUTING ALGORITHMS

2.2.1. Epidemic Routing. While Epidemic Routing is perhaps the simplest of the algorithms available, it nonetheless demonstrates both a very high delivery ratio and a low delay via brute force. Sometimes called Flooding, this algorithm takes advantage of every encounter, transmitting all messages to all available nodes. Messages are spread through the network, 'infecting' nodes at every opportunity (hence the original title). Provided messages never expire, they are guaranteed to eventually reach the destination [5].

The procedure begins when both $Node_A$ and $Node_B$ come into contact range. After establishing a connection, $Node_A$ transmits a list of message identifiers within its buffer to $Node_B$. A check is performed in $Node_B$, creating a list of messages it wants from $Node_A$. Afterwards, another check identifies the messages in $Node_B$'s buffer which are not held by $Node_A$. Both lists are transmitted to $Node_A$, so both nodes are aware of which messages they need to transmit. After transmitting the messages they transmit acknowledgements and message hashes to ensure all messages are received successfully.

A variation on Epidemic Routing is adding a hop count attribute to messages. This attribute indicates the maximum times a particular message will be transmitted, reducing itself at each stage. A hop count close to the expected distance between the sender and the receiver nodes can reduce the total energy expended in the network by ensuring messages are not retransmitted *ad infinitum*.

Epidemic Routing is generally considered a baseline of comparison for routing algorithms, partly due to ease of implementation and its high delivery ratio. However, implementing this algorithm in a realistic environment has certain difficulties. Sending messages at every opportunity often results in message collision. When several nodes in the same area attempt to transmit messages simultaneously the messages are garbled. Both acknowledgement and retransmission fails to resolve this issue. The algorithm also tends to overfill the message buffer very quickly. This is especially when using low-cost nodes such as those used by a wireless sensor network. Due to these limitations, the Epidemic algorithm works

best in low traffic, low connectivity environments with high powered devices, such as smart phones.

2.2.2. MaxProp. When Epidemic Routing overflows the buffer, it removes messages in a last-in-first-off manner. This removal is based on the assumption that messages which have been in the buffer longer have already been delivered and can therefore be removed without consequence. While generally true, this method ignores information about a node's access to the destination or if the message has already been delivered. The MaxProp algorithm is a variation on Epidemic Routing which prioritizes messages, affecting both the order of transmission and buffer removal[6].

A message's priority is based on both a given node's likelihood to deliver the message and the likelihood that other nodes will deliver the message. A given node's probability is estimated using *incremental averaging*. Upon contact, a node's probability to deliver that destination is increased by 1. All node probabilities are then normalized to sum to 1. This method allows nodes to track both recent and current contacts very well. Previous contacts, however, tend to be overridden regardless of how well they performed historically.

When two nodes encounter each other, they first deliver any messages that are intended for the other node. Nodes then exchange probability data, containing the likelihood of delivering a message to any given node. The next step is to share delivery acknowledgements; this step allows nodes to share information about which messages have already been received and can therefore be dropped from the buffer. At this stage, both nodes have all of the data they need for actual message transmission. Priority is given to any messages which have not traveled beyond a certain distance from the source node; this behavior forces messages to be flooded to nearby nodes, ensuring all messages are propagated. Finally, each node transmits all messages in the order of delivery probability.

These additions to the Epidemic Routing algorithm allow MaxProp to transmit messages while avoiding some of the redundant transmissions. By prioritizing between messages for transmissions and buffer removals, the algorithm routes more efficiently than base

Epidemic Routing. Overall, however, it has many of the same advantages and disadvantages as Epidemic Routing: high delivery ratio, low delivery time, very high energy consumption, signal noise, and buffer overflows that cripples the network as traffic increases.

2.2.3. AODV. The routing algorithm Ad hoc On-demand Distance Vector (AODV) is based on obtaining the path a message will follow before transmitting the actual message [7]. To send a message, the node first floods the network with path requests. Each node appends its identity to the path request before broadcasting it to other nodes within range. When the destination node receives the path request, it replies along the same route to the original source. The source can then use the path to transmit the message. As a result, the path requests flood the network while the message itself follows the shortest path to the destination, provided one exists.

This algorithm is primarily used in relatively static, highly connected networks. It does not need to store messages in a large buffer, allowing it to function using low-powered devices. Additionally, because the nodes only broadcast when a message is sent, there is no overhead and little energy is used use unless a message is actively being transmitted. The algorithm fails to function in an intermittently connected network; if nodes cannot deliver the message directly to the destination they will not forward the message to a node more capable of delivery. This is a useful, efficient algorithm, but one with severe limitations.

2.2.4. Probabilistic. As an alternative to sending either messages or path requests epidemically, the Probabilistic Routing algorithm was designed to forward messages only to nodes more capable of delivering them to a destination [8]. This algorithm was designed primarily for highly disruptive networks which cannot afford high message traffic, such as wireless sensor networks. When a node is carrying a message, its own probability of delivery slowly decreases. When the node encounters another with a higher probability the message is transferred, and the sending node increases its probability estimate. For example, when $Node_A$ and $Node_B$ encounter each other, both nodes compare their probability of delivery, σ_A and σ_B. If σ_B is greater, then $Node_A$ will transfer the message and increase σ_A.

The control variable α, between 0 and 1, determines how rapidly the probability estimates change based on new data. A high α is suitable for more static environments; a lower α reflects patterns shifting periodically, disregarding previous probability estimates. This is illustrated in Figure 2.1.

$$\sigma_A = \sigma_A * \alpha + \sigma_B * (1 - \alpha) \tag{1}$$

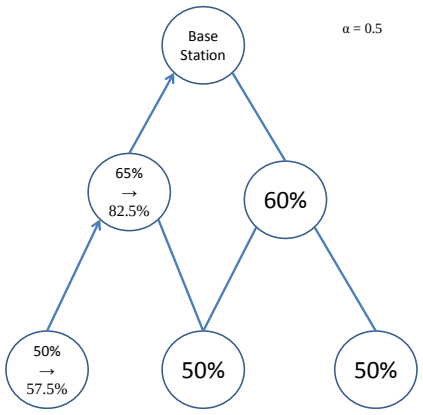


Figure 2.1. Successful Message Delivery

In addition, to reflect a node's inability to deliver messages the probability decays occasionally as a message times out, as shown in Figure 2.2. If a message is contained in a buffer over a certain period without being forwarded, the node is considered to be unable to deliver the message, and the probability estimates are reduced to reflect this inability. Note that a timeout (depending upon implementation) will not actually remove a message from the buffer, but it will lower the probability of the containing node.

$$\sigma_A = \sigma_A * \alpha \tag{2}$$

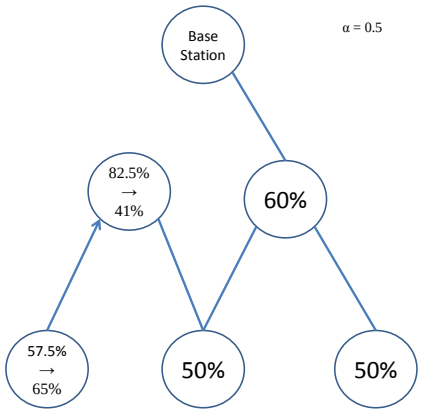


Figure 2.2. Message Timeout

Because the destination node has a probability of 100%, the node probability propagates throughout the network. The nodes within 1 hop of the destination will increase their probability as they deliver messages. Then the nodes 2 hops away will increase their probability when they transfer to the 1 hop neighbors, *ad infinitum*. This process continues until all nodes have an accurate measurement of their probability to deliver a message.

Although not designed specifically for mobile networks, the Probabilistic algorithm can function within them if the nodes follow repeated patterns. Because only a single copy of any given message exists, any node not actively involved in routing a message can remain idle which lowers energy costs. Because the algorithm has no load distribution the optimal nodes quickly lose battery life by routing other node's messages, shortening the network lifespan. A more significant disadvantage, however, is that the probability only changes when messages are sent. In a low traffic environment, the probability estimate (σ) will be wildly inaccurate. Efforts to improve this inaccuracy by sending a series of junk messages increases network costs considerably.

2.2.5. PRoPHET. A more proactive solution to the routing problem is the Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET)[9]. PRoPHET is similar to Probabilistic Routing in that nodes estimate their probability of de-

livering a message and then route messages based on these estimates, but the probability is based on contact frequency rather than upon previous message performance. Variable $\sigma_{A,B}$ indicates the probability that $Node_A$ can successfully deliver a message to $Node_B$. PRoPHET is based on three methods of adjusting the probability estimates: direct contact, indirect contact, and decay. Direct contact occurs whenever two nodes come to point to point contact. The control variable P_{init} determines how much the delivery probability $\sigma_{A,B}$ changes on each contact.

$$\sigma_{A,B} = \sigma_{A,B} + (1 - \sigma_{A,B}) * P_{init} \tag{3}$$

This method allows nodes to keep track of direct contacts. In addition, PRoPHET allows nodes to adjust their probability estimates based upon indirect contacts, indicating that $Node_A$ can successfully deliver a message to $Node_B$ by routing through another node. This increase, known as either transitive or indirect contact, occurs when $Node_A$ encounters $Node_B$. Both nodes increase the chance of delivery for all nodes the other has encountered. Similar to P_{init}, the control variable β determines the rate of increase.

$$\sigma_{A,C} = \sigma_{A,C} + (1 - \sigma_{A,C}) * \sigma_{A,B} * \sigma_{B,C} * \beta \tag{4}$$

Both direct and indirect contact can raise the probability of delivery. An accurate estimate of the delivery chance, however, must reduce itself whenever nodes do not encounter for long periods. For this reason, a probability decay function performs periodically. The rate of decay is based on the control variable γ, occuring periodically in every node.

$$\sigma_{A,B} = \sigma_{A,B} * \gamma^{time} \tag{5}$$

These three events occur continuously throughout the network lifetime, working to make the estimate $\sigma_{A,B}$ an accurate representation of the probability of delivery. When a node with a message encounters another node, it compares σ. If the encountered node has a higher chance of delivery, it forwards the message. The PRoPHET routing algorithm is ideal for disconnected networks following some patterns of contact and adapts well to changing layouts. PRoPHET has a notable weakness in that it maintains the probabilities whether there is any traffic or not. While this is a significant energy drain, it also allows messages to be consistently sent to new nodes.

2.2.6. SimBet. The SimBet routing algorithm [10] is a node-to-node routing algorithm that takes advantage of the social structure of the network. Nodes engage in 'conversations' when they meet, exchanging information about both data messages and current neighboring nodes. This information is used to determine 'betweenness' and 'similarity'.

Betweenness is a measure of how often a node lies on a path between otherwise unconnected nodes. Equation 6 shows the method for calculating the betweenness of $node_i$, where $g_{jk}(this)$ is the number of paths from $node_j$ to $node_k$ that include $node_{this}$, and g_{jk} is the total number of paths between $node_j$ and $node_k$.

$$Bet_{this} = \sum_{j=1}^{N} \sum_{k=1}^{j-1} \left(\frac{g_{jk}(this)}{g_{jk}} \right) \tag{6}$$

$$BetUtil_{this} = \frac{Bet_{this}}{Bet_{this} + Bet_{other}} \tag{7}$$

Similarity is simply the ratio of common neighbors between two nodes. To calculate similarity between $node_x$ and $node_y$, identify all nodes which are neighbors to either, as shown in Equation 8.

$$Sim(this, dest) = |N(this) \bigcap N(dest)| \tag{8}$$

$$SimUtil_{this}(dest) = \frac{Sim(this, dest)}{Sim(this, dest) + Sim(other, dest)} \tag{9}$$

Once both metrics are calculated, the system can route through nodes with higher values. Both metrics are locally determined; global knowledge of the network is not needed. The ratio of priority given to either metric is the control variable α and can affect the systems performance. When two nodes encounter, they determine their relative *SimBetUtil* (see eqn. 10), and all messages are forwarded to the node with the higher value.

$$SimBetUtil_i = \alpha SimUtil_i + (1-\alpha)BetUtil_i, \forall i \in DTN \tag{10}$$

A notable disadvantage of SimBet is the quantity of messages the aforementioned conversations send across the network. The amount of data which must be exchanged is a significant strain on the node's limited power supply. Additionally, relationships between nodes in SimBet are represented as either in-contact or not-in-contact. More specific information, such as percent of time in contact, would result in a better measure of a node's likelihood to forward a message to the destination.

2.3. SECURITY IN DTN

2.3.1. Public Key Encryption. Originally, encryption was performed using a shared secret (e.q., passwords, large numbers, or book titles) communicated by secure channels [11]. In order for Alice to communicate securely with Bob, they had to privately exchange a key. That key was the basis for encryption or decryption of all secure message traffic. When performing large scale communications, the distribution of these keys was always a security risk - difficult and dangerous. As mathematics progressed, the theory of asymmetric encryption was developed. These encryption techniques would allow a user with a publicly available key to encrypt messages for a destination, but access to that key could not decrypt the message. Such encryption schemes enabled large scale secure communications, as any user could securely send a message provided he had access to the public key.

The first public-key encryption algorithm was developed by James Ellis, Clifford Cocks, and Malcolm Williamson[11]. These cryptographers worked for the Government Communications Headquarters, a British intelligence agency dedicated to signal intelligence. Due to the secret nature of the organization, they did not disclose their discovery. This secrecy resulted in Whitfield Diffie and Martin Hellman independently researching and publishing the 'first' public key exchange algorithm: the Diffie-Hellman Key Exchange. Later, a series of MIT mathematicians, Ron Rivest, Adi Shamir, and Leonard Adleman, independently developed a version of Ellis' work that enabled public keys to be distributed openly for secure communication[11] [12]. Currently, several public key algorithms are available; the most commonly used are RSA, ElGamal, and Elliptical Curve Cryptography.

Public key infrastructures are considered secure in wired networks when nodes setting up to communicate securely can exchange keys. A series of trusted servers containing public keys are available. These servers are used if the destination computer or user is not immediately available. However, this infrastructure is difficult to implement in DTNs since messages take longer to deliver. Further, a public key exchange is vulnerable to Main in the Middle (MitM) attacks. When $Node_A$ sends its public key to $Node_B$, nothing exists to stop a malicious node along the way from replacing the key with their own. By controlling the key nodes use to communicate, the node can access any secure traffic. Techniques to prevent this are difficult to implement in a DTNs, requiring either precise timing or verification by several other nodes. While the public key infrastructure is a powerful tool, there are difficulties applying it to DTNs.

2.3.2. Threshold Encryption. Generally, encryption algorithms use a single key to encrypt a message, and a single key to decrypt a message. A class of encryption algorithms, known as Threshold Encryption, encrypts a message and then generates several decryption keys [13]. A subset of those keys must be used to decrypt the message.

Shamir's Secret Sharing key is a sample threshold encryption algorithm[14]. To encrypt a number m, a k-degree polynomial equation is plotted, where $k+1$ is the number of keys needed to decrypt the algorithm. For example, when the user wants for 4 keys to decrypt the message, the algorithm will generate $f(x) = a*x^3 + b*x^2 + c*x + d$. Coefficients a, b, and c, are randomly generated, but d is selected such that $f(1) = m$. Because the k-degree curve can be plotted correctly once $k+1$ points are known, points on the curve $(y, f(y))$ are treated as the threshold keys. Once an intended recipient has obtained any $k+1$ keys he can solve for a, b, c, and d, finally generating $f(1)$, the original message.

This encryption algorithm is very useful in unreliable networks, where keys can be lost due to either failed routing or message disruption. Because as many keys as desired can be generated, the algorithm can be scaled to match the reliability of the network. This feature makes reliable security possible in a DTN, as demonstrated in Section IV.

2.3.3. Onion Routing. A security system currently used in wired networks is the Onion Router, also known as Tor [15]. The purpose of this system is to privately and anonymously transmit messages to a destination. Neither the destination nor any monitoring nodes are allowed to know both the source and destination of the messages. The technique uses a series of proxies, public key and symmetric key encryption, and a series of publicly available servers.

When Alice wants to establish an anonymous connection using a Tor network, she first reviews the servers that contain a list of computers volunteering to act as proxies. The servers contains the IP addresses and public keys of the computers. Alice randomly selects 3 nodes, acquiring both their IP_A and their Key_A. The final node, $Proxy_C$, is the exit relay, which will have unmonitored traffic. Alice then generates a series of symmetric keys using the Advanced Encryption Standard (AES). AES is a powerful symmetric encryption algorithm, which the proxies will use to communicate between themselves. Alice then generates $AES_{A,B}$, the symmetric key which $Proxy_A$ and $Proxy_B$ will use to communicate, for all proxy pairs. With this data, Alice prepares the message to be sent through the

network. The message is composed of a series of contact information layers, each of which contains the address and encryption data of both the previous and the next step in the proxy chain. Each node, upon receiving the message, will remove its layer of encryption, establish a secure channel to the previous proxy, and send the remainder of the message to the next node.

When $Proxy_A$ receives the message, it has no difficulty removing the first layer of encryption. This action reveals the source of the message IP_{Alice}, the next step in the chain IP_B, and the secure keys to be used to send future messages up and down the chain. The remainder is still encrypted with Key_B and, thus, is sent on. (An important note: $Proxy_A$ cannot access any information about the chain beyond the two nodes it is between - it is all encrypted.) Later nodes will not have access to the source of the stream; without IP_{Alice}, they can only guess the source of the connection.

After each proxy has received the AES of the previous and the next proxy, $Proxy_C$ sends an encrypted acknowledgement message back up the chain of proxies - $AES_{BtoC}(ack)$. At each proxy, the AES encryption layer is removed and replaced with the next stages. The result is that messages can be sent securely, without any proxy knowing both the source and the destination of the chain. The final node, $Proxy_C$, is the link to the outside world, in the sense that it will make requests on Alice's behalf to insecure servers along open channels. Anyone monitoring $Proxy_C$ can identify its traffic but cannot link it to Alice.

Tor networks are secure in the sense that they preserve both the privacy of the source and the destination nodes. Already implemented in large scale, they are useful in monitored environments. There is ongoing research into timing attacks on the Tor network, which work by deliberately delaying Alice's traffic. If the delays are noticed on the exit node, a monitor can link the source and destination of traffic. These attacks, however, are difficult to implement [16].

While invaluable for wired communications, an Onion Router network is difficult to implement in a DTN. Without consistent access to the central server, a node has to generate its own list of device addresses and public key information, and an adversary can inject false data into these lists. To address these issues is the purpose of 3PE, detailed in Section IV.

2.3.4. EnPassant. Researchers Vakde et. al. [17] have detailed one solution to the Tor weaknesses as they apply to the DTNs. The purpose of the EnPassant algorithm is to disguise the source of a message by both encrypting it and forwarding it through the network. Each node is arbitrarily assigned to a pawn group, a set of nodes with a shared public-private key pair. Messages are encrypted with the public keys of these groups, then forwarded through any member of the selected groups. Unlike Tor, there is no access to a central server containing information on each device. However, nodes can reasonably contain a key for each group.

The process begins when the sending node selects which groups to use as proxies. The message is first encrypted and then sent on to a member of the first group - $Key_{Group1}(Key_{Group2}(Key_{Group3}(dest, msg)))$. The method of selecting a member of the pawn groups to send the message to was studied. The nodes can select either randomly for improved security, the nearest member for greater speed, or, if the nodes have enough information, the member closest to the original path from the source to the destination. Regardless of how the next member is chosen, a member of $Group_1$ should acquire the message. This member removes the $Group_1$ layer of encryption and forwards it to the next group. At the final stage, a member of $Group_3$ will acquire the message, and so has no information on the original sender. This node removes the final encryption layer, revealing both the actual message and the destination. It can now forward the message to the actual message destination.

While this technique has little impact on either the delivery ratio or the speed of messages, the question remains about how well it addresses Byzantine attacks. These attacks assume the adversary is capable of controlling a portion of the network, either by taking

over machines already in the network or by inserting devices controlled by the adversary. Because each member of a pawn group acquires access to the private key information of that group, if the message reaches a compromised node first the adversary can remove that group's layer of encryption, and forward it to another compromised member of another group. Because the first to receive the message has knowledge of the source of the message, and the final member is compromised, the adversary has access to both the source and the destination, violating anonymity. This ignores the possibility that the adversary will simply share compromised keys, allowing the first node to read the message without needing to forward anything. Although the algorithm can disguise the path from monitors, it does little to protect messages from partially compromised networks.

2.4. CONTENT DISTRIBUTION

2.4.1. R-P2P. The R-P2P system is designed to allow mobile repositories to request data items from other repositories. By maintaining a Distributed Hash Table (DHT), a repository can identify the location of a particular data item quickly and efficiently. This allows a node to find the location of a data item by referring to the nearest repository, improving the time to delivery. One contribution of this article was the introduction of the throwbox concept, in which a server designates other nodes in the network to store and distribute data. Distributing content in this manner improves delivery time, and efficient identification of the nodes to act as throwboxes improves it further.

2.4.2. OnMove. An earlier attempt to improve throwbox selection by using social context was the OnMove protocol. The system is designed to determine optimal placement with a series of metrics, including social similarity, meet frequency, and betweenness (among others). By assigning different weights to each metric the protocol can optimize itself for any environment. While the overview is promising, the complete protocol was never developed. The concept has been expanded on to introduce the Social Content Distribution schema.

3. BIBLIOGRAPHY

[1] F. Stajano and R. J. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *Proceedings of the 7th International Workshop on Security Protocols*, (London, UK, UK), pp. 172–194, Springer-Verlag, 2000.

[2] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong, "Talking to strangers: Authentication in ad-hoc wireless networks," in *NDSS*, 2002.

[3] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, (New York, NY, USA), pp. 27–34, ACM, 2003.

[4] I. Chlamtac, "Mobile ad hoc networking: imperatives and challenges," *Ad Hoc Networks*, vol. 1, pp. 13–64, July 2003.

[5] A. Vahdat and D. Becker, "Epidemic Routing for Partially-Connected Ad Hoc Networks," tech. rep., Duke University, Apr. 2000.

[6] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "MaxProp: Routing for vehicle-based disruption-tolerant networks," in *INFOCOM*, 2006.

[7] C. Perkins, E. Royer, and S. Das, "RFC 3561 Ad hoc On-Demand Distance Vector (AODV) Routing," tech. rep., 2003.

[8] Y. Wang and H. Wu, "Delay/fault-tolerant mobile sensor network (DFT-MSN): A new paradigm for pervasive information gathering," *IEEE Transactions on Mobile Computing*, vol. 6, no. 9, pp. 1021–1034, 2007.

[9] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, pp. 19–20, July 2003.

[10] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant MANETs," in *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '07, (New York, NY, USA), pp. 32–40, ACM, 2007.

[11] S. Singh, *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots, to Quantum Cryptography*. New York, NY, USA: Doubleday, 1st ed., 1999.

[12] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and

public-key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, 1978.

[13] I. B. Damgard and M. J. Jurik, "A length-flexible threshold cryptosystem with applications," in *In proceedings of ACISP'03, LNCS series*, vol. 2727, pp. 350–364, 2003.

[14] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[15] P. F. Syverson, M. G. Reed, and D. M. Goldschlag, "Private web browsing," *Journal of Computer Security*, vol. 5, no. 3, pp. 237–248, 1997.

[16] V. Shmatikov and M.-H. Wang, "Timing analysis in low-latency mix networks: Attacks and defenses," in *ESORICS*, pp. 18–33, 2006.

[17] G. Vakde, R. Bibikar, Z. Le, and M. Wright, "Enpassant: anonymous routing for disruption-tolerant networks with applications in assistive environments," *Security and Communication Networks*, vol. 4, no. 11, pp. 1243–1256, 2011.

PAPER

I. Dynamic Social Grouping Based Routing in a Mobile Ad-Hoc Network

Roy Cabaniss∗, Sanjay Madria∗, George Rush∗,
Abbey Trotta†, and Srinivasa S. Vulli∗
∗ Department of Computer Science,
Missouri University of Science and Technology, Rolla, Missouri 65401
† School of Computing and Engineering,
University of Missouri, Kansas City, Missouri 64110

The patterns of movement used by Mobile Ad-Hoc networks are application specific, in the sense that networks use nodes which travel in different paths. When these nodes are used in experiments involving social patterns, such as wildlife tracking, algorithms which detect and use these patterns can be used to improve routing efficiency. The intent of this paper is to introduce a routing algorithm which forms a series of social groups which accurately indicate a node's regular contact patterns while dynamically shifting to represent changes to the social environment. With the social groups formed, a probabilistic routing schema is used to effectively identify which social groups have consistent contact with the base station, and route accordingly. The algorithm can be implemented dynamically, in the sense that the nodes initially have no awareness of their environment, and works to reduce overhead and message traffic while maintaining high delivery ratio.

1. INTRODUCTION

Mobile Ad-hoc networks are a collection of computing devices connected through wireless communications, such as Bluetooth or wireless LAN. They are characterized by the mobility and dynamic nature of the devices, referred to as nodes. This mobility makes conventional routing algorithms ineffective or inapplicable, and to accommodate these environments new routing methods have been developed. These routing methods are generally examples of Delay Tolerant Routing, which holds copies of transmitted messages to transmit them to appropriate nodes, as opposed to traditional routing which broadcasts them immediately.

As a general rule, routing algorithms are more effective when they can rely on more information regarding the mobility patterns of nodes. Conventional probabilistic routing schema assumes consistent avenues of communication - nodes which interact with a set group of nodes in the past will do so again in the future [1]. Similarly, social routing assumes that nodes that are assigned to the same social network (classroom, project team) will regularly interact with members of that social group. To take advantage of the partial applicability of both social and probabilistic routing, a grouping method is studied which will form social groups based on contact patterns. With these groups identified, consistent routes to basestations are identified based on the delivery history of a group or node. We compare this algorithm, called Dynamic Social Grouping (DSG), with two well known algorithms referred as Epidemic routing [2] and Probabilistic routing [1], and results show that DSG performs better than both routing algorithms in terms of message delivery ratio and the time to deliver a message.

The objective of this algorithm is to reliably deliver messages from a group of sensor nodes to basestation nodes, as presented in Probabilistic Routing, using as few communications as possible. The basestations are immobile, and thus can use more reliable

communications to transmit the data to the end user, but the sensor nodes are attached to mobile social entities. This algorithm is designed to identify social groups without relying on outside information, based only on its contact patterns with other nodes. A given node may belong to several social groups and will attempt to merge together groups who share common members. Once the social groups are identified, routing occurs through these groups based on which group has more reliable access to the basestations, as determined by the reliability of previously delivered messages. Simulations with real-world data comparing DSG to Probabilistic and Epidemic routing algorithms show that the Dynamic Social Grouping algorithm is superior in terms of delivery ratio and message costs to the Probabilistic scheme in a social environment, and much less expensive than Epidemic routing while maintaining high delivery ratio and low transit time (refer to Section 4 for details). Applications of this technique include wildlife tracking or tracking human social behaviors, although it can improve any environment in which groups of regular contacts are formed.

2. BACKGROUND

Research in the area of Delay Tolerant Networks (DTN) has been receiving considerable attention in the last few years owing to their widespread occurrence in a variety of applications. Routing and data aggregation are of particular interest as they affect the performance of the network as a whole and also affect longevity of the sensor nodes considerably. A survey of routing techniques for DTNs is presented in [3].

2.1. EPIDEMIC ROUTING

Epidemic routing [2] was designed for partially connected networks, and its goal is to maximize the message delivery ratio while minimizing the time necessary to deliver a message. The main strategy is to pass messages along to each node encountered, in hopes of making a connection with parts of the network with low connectivity. In wireless sensor networks, this often results in excessive network traffic which reduces the life of the network. Also, minimizing time to message delivery is not as important in delay tolerant networks. That said that the delivery ratio of epidemic routing is considered the ideal delivery ratio possible when ignoring network life or node overflow.

2.2. PROBABILISTIC ROUTING

Introduced by Yu Wang, the Probabilistic Routing Schema [1] is based on individual nodes having a set probability of successfully delivering a message to a base station. This probability is based on the set of neighbors which the node regularly interacts with, and as such it is based on regular, if not social, patterns of movement. Nodes individually begin with a set delivery probability and transfer the messages they are carrying to neighbors with higher delivery probability, adjusting their own probability upward as they do so. If they are carrying a message when it times out their probability is reduced to reflect the node's

inability to transfer. This algorithm shows a marked improvement in terms of message transmission rate while maintaining a high delivery probability, but does not take advantage of any knowledge other than past contacts.

2.3. BUBBLE RAP

The Bubble-Rap grouping method [4] allocates nodes into social groups based on direct and indirect contacts. They distribute using a method called $k-cliques$, in which all fully connected groups of k members are considered a distinct social group and are then merged with all other $k-cliques$ which can be reached through $k-1$ nodes. This provides a very accurate representation of the social groups formed by a set of nodes. However, it relies on global knowledge of the nodes' contacts, and it must have them before the algorithm can group them. In this sense, k-grouping is neither dynamic nor distributed, which limits it's applicability to a MANET.

2.4. SOCIALCAST

Costa et al. described an application using social dynamics for a publish/subscribe schemata across a wireless sensor network. They detail a SocialCast model [5], in which nodes are assigned a Utility Value for each interest in which they participate which indicates their routing utility for that group. By separating the routing utility from group participation, the authors improve the lifespan and routing efficiency of the wireless nodes. The given algorithm is designed for use across a publish/subscribe model, as well as a wireless network (as opposed to a sensor network), but can still serve as a basis for further research, especially regarding the social dynamics described, such as the Caveman Model.

2.5. SIMBET

The SimBet routing algorithm [6] has a similar routing structure to DSG in that it takes advantage of Social Grouping and contact patterns to predict paths to node destina-

tions, improving delivery ratio and time. The algorithm calculates the Betweenness rating of a node, which is a measurement of the number of message routes which contain the node. By using this, as well as the Centrality of the node, the algorithm can route messages through to destination nodes with remarkable efficiency. A notable deficiency of this algorithm is it's reliance on near-complete knowledge of neighboring nodes, increasing traffic and memory use. It can, however, deliver messages from node to node, whereas the Dynamic Social Grouping algorithm is strictly node to base station.

3. PROPOSED ALGORITHM

The algorithm addresses two distinct subproblems. First, given an arbitrary collection of nodes, the network must identify cohesive social patterns, and identify them as groups, at the same time distributing this information throughout the network. This network organization can change to reflect changes to the network or to the social patterns of the nodes. Second, once the networks have been identified, a route must be identified to deliver messages to a base station. Delivery must minimize the number of message repetitions while ensuring a high percentage of messages delivered to the destination.

3.1. GROUPING

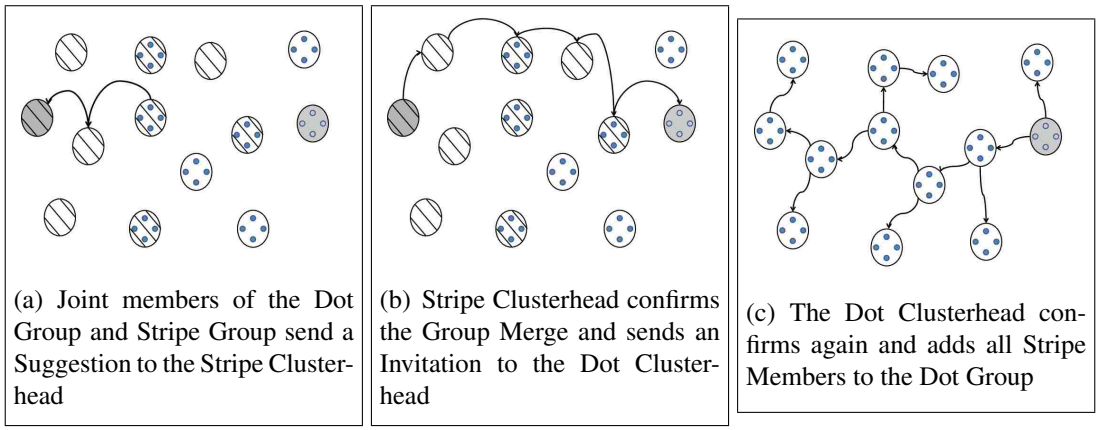
A major advantage this algorithm has over conventional mobile ad-hoc routing methods is the use of social groups to improve communication throughput. The task of identifying such groups, however, requires knowledge of the nodes which is not present at startup.

3.1.1. Contact Strength. The first task in identifying a social group is to calculate a metric for the contact frequency two nodes have with one another. The symbol $\lambda_{i,j}$ is used to represent the contact strength between two nodes. This is measured as a function of the time from the previous contact, where the symbol ϕ is used to determine how much the λ changes based on new data. Initially, $\lambda_{i,j}$ will be 0 between all nodes, but when nodes contact one another, it is recalculated as shown in Eqn 11.

$$\lambda_{A,B} = (1-\phi)\lambda_{A,B} + \frac{\phi}{time_{current} - time_{prev}} \tag{11}$$

3.1.2. Forming New Groups. When the $\lambda_{i,j}$ exceeds a certain threshold ψ, the nodes can identify as being members of the same group. Initially, all groups will have two members. The node with the higher delivery probability (as defined in section 3.2.1) be designated the group clusterhead - this node has the responsibility of maintaining the

group membership list and approving any changes to the group. The group will also have its probability set to the average of the two founding members (this will be expanded upon in section 3.2).



(a) Joint members of the Dot Group and Stripe Group send a Suggestion to the Stripe Clusterhead

(b) Stripe Clusterhead confirms the Group Merge and sends an Invitation to the Dot Clusterhead

(c) The Dot Clusterhead confirms again and adds all Stripe Members to the Dot Group

Figure 3.1. Three Stages of Merging Groups

3.1.3. Merging. Having formed two-node groups, the next step is to determine which of these groups can be merged with each other to form larger, more applicable social groups. Groups are merged when the similarity between the groups exceeds a threshold value τ. Similarity is defined as the number of common members divided by the total number of members in both groups. Each node checks through its list of groups periodically to see if the groups should be merged. If it finds two groups that qualify for a merger, it sends a suggestion message to the clusterhead of the smaller group, as shown in Figure 3.1(a). This message is distributed epidemically through that group until it arrives at the clusterhead, which contains the up-to-date member list of the group. Once the clusterhead receives the suggestion it compares it's member list with the member list for the larger group (contained within the suggestion message), confirms that the groups should be merged, and sends an invitation message to the clusterhead of the larger group (Figure 3.1(b)). The larger group's clusterhead repeats the confirmation. Once confirmed, the clusterhead updates the member

list to include the new members from the smaller group (Figure 3.1(c)). A Kill message is then sent to remove the smaller group. This process is detailed in Algorithm 1.

3.1.4. Dynamic Grouping. Due to changing social patterns, group membership needs to allow nodes to remove themselves from a group's membership, as shown in Algorithm 2. To reduce communication overhead and group fragmentation, however, the clusterhead has to know about the resignation immediately. When a node contacts the clusterhead it will calculate it's $Group\lambda_{A,Y}$, which is the average $\lambda_{A,B}$ between the node and all members of the group, to determine whether it should resign from the group. If this average λ is below the threshold ψ, it sends a resignation to the clusterhead and removes the group data. This method has the advantage of low overhead and communication, but nodes cannot leave the group until they contact the clusterhead, which can result in nodes maintaining group data for groups they don't participate in. Experiments which allowed nodes to periodically review their groups and resign through a message sent to any group member resulted in badly fragmented group data.

3.1.5. Group Versions. A constant issue when creating dynamic groups is reflecting changes to the group data to all members of that group. Data fragmentation occurs whenever two nodes, members of the same Social Group, have different data regarding that group's membership. To prevent this issue, all changes are controlled by the group's Clusterhead. This node, arbitrarily chosen, holds the group's master member list.

As movement patterns change or are revealed, there will be changes to a group's membership. The Clusterhead will increment the version value of the group as changes are made, including merges and nodes resigning from the group. Whenever two members of the same group meet, they compare the version number of their local copy of the group. The higher version number is the one whose information regarding the group came from the clusterhead more recently. This information is copied over to the less recent node, along with the version number. This process is detailed in Algorithm 3.

Algorithm 1 Merging Groups

Notation
$Group_Y$, $Group_Z$ - Any social groups in MANET
$Node_A$ - Member of both $Group_Y$ and $Group_Z$
$Node_B$ - Clusterhead of $Group_Y$, $Clusterhead_Y$
$Node_C$ - Clusterhead of $Group_Z$, $Clusterhead_Z$
τ - Threshold for Merging a Group

Trigger - Periodically in $Node_A$
for all $Group_Y, Group_Z$ of which $Node_A$ is a member **do**
  if $\frac{|Group_Y \bigcap Group_Z|}{|Group_Y \bigcup Group_Z|} > \tau$ **then**
    if $|Group_Y| > |Group_Z|$ **then**
      Send SUGGEST Message to all members of $Group_Z$
    else
      Send SUGGEST Message to all members of $Group_Y$
    end ifcannot determine size of graphic no boundingbox
  end if
end for

Trigger - $Node_A$ contacts $Node_B$
In $Node_A$...
for all $Group_Y$ which contain both $Node_A$ and $Node_B$ **do**
  if $Node_A$ has Control Messages for $Group_Y$ **then**
    $Node_A$ sends Control Messages to $Node_B$
  end if
end for

Nodes pass Control Messages epidemically to all Group Members

Trigger - $Node_B$ receives SUGGEST message
if $Node_B$ is $Clusterhead_Y$ **then**
  if $\frac{|Group_Y \bigcap Group_Z|}{|Group_Y \bigcup Group_Z|} > \tau$ **then**
    Send INVITE Message to all members of $Group_Z$
  end if
end if

The Clusterhead confirms the suggestion, and sends an Invite to the other group

Trigger - $Node_C$ receives INVITE message
if $Node_C$ is $Clusterhead_Z$ **then**
  if $\frac{|Group_Y \bigcap Group_Z|}{|Group_Y \bigcup Group_Z|} > \tau$ **then**
    $Group_Z = Group_Z \bigcup Group_Y$
    $Group_Y = \varnothing$
    Send KILL message to $Group_Y$
  end if
end if

If both Group Clusterheads approve, the smaller group is added to the larger, and then removed.

Algorithm 2 Dynamic Groups

Notation
$Node_A$ - Member of $Group_Y$
$Node_B$ - Clusterhead of $Group_Y$, $Clusterhead_Y$
$NodeList_Y$ - All nodes in $Group_Y$
$Group\lambda_{A,Y}$ - Contact Strength between $Node_A$ and $NodeList_Y$
ψ - Threshold for Forming a Group

Trigger - $Node_A$ contacts $Node_B$
In $Node_A$
if $Node_B$ is $Clusterhead_Y$ **then**
  $Group\lambda_Y = Average(\lambda_{A,Y})$
  if $Group\lambda_Y < \psi$ **then**
    Send RESIGN to $Node_B$
    Remove $Group_Y$ from $Node_B$
  end if
end if

Trigger - $Node_B$ receives RESIGN from $Node_A$
$NodeList_Y = NodeList_Y - Node_A$

Algorithm 3 Group Updates

Notation
$Version_{A,Y}$ - Version Number of $Group_Y$ kept by $Node_A$

Trigger - $Node_A$ contacts $Node_B$
In $Node_A$
for all $Group_Y$ which contains both $Node_A$ and $Node_B$ **do**
  Send $Version_{A,Y}$ to $Node_B$
  Receive $Version_{B,Y}$ from $Node_B$
  if $Version_{A,Y} > Version_{B,Y}$ **then**
    Send $Group_Y$ to $Node_B$
  else if $Version_{A,Y} > Version_{B,Y}$ **then**
    Receive $Group_Y$ from $Node_B$
    $Version_{A,Y} = Version_{B,Y}$
  end if
end for

3.2. ROUTING

Having organized the nodes into social groups, the algorithm can now use this information to route data to the base station. The Probabilistic method assigns a metric to each node to depict the node's chance of successfully delivering the message, and continually routes messages to higher performing nodes until the message reaches a destination. To improve on this the DSG identifies a similar metric to measure a group's ability to deliver a message to the base station. Both methods are described below.

3.2.1. Individual Probability. Nodes are initially assigned a default probability σ, while base stations are assigned a probability of 100%. When nodes encounter one another, they compare probabilities to determine routing (more on this in section 3.2.3). The member with the lower probability will forward all its messages to the other, and then it will update its σ to reflect the ability of the node to deliver either directly or indirectly to a base station. The result is that nodes with immediate access to a base station achieve a higher probability, and the probability cascades outwards through node contacts. The cascade rate is determined by a control variable α, which is similar to ϕ in that is controls how quickly the probability changes based on new data. For details, review Algorithm 4.

In addition to message transmission, the Individual Probability σ can also update to reflect inability to deliver a message. Messages log the time they were originally sent, and can use this to determine if they have been in the system too long. When these messages expire, all nodes which contain the message have their individual probability reduced to reflect their inability to reach a sink.

3.2.2. Group Probability. Each node calculates group probability, depicted as β, independently, based on the contact patterns of that specific node. This means that each node will have different values for the probability of the same group, but these values are based on the subset of the group which each node contacts. Nodes which exist in a common group and are not encountered are estimated using the current group probability. The exact method, described also in Algorithm 5, begins when $Node_A$ contacts $Node_B$ and both are

Algorithm 4 Calculating Individual Probability

Notation
$Node_A$, $Node_B$ - Nodes in MANET
σ_A - Individual Probability for $Node_A$
σ_B - Individual Probability for $Node_B$
$message_i$ - Message in MANET
$timeSent_i$ - Time $message_i$ was sent
$timeOut$ - Parameter determining max duration of messages
α - Control Parameter determining Probability decay ratio

Trigger - $Node_A$ transmits $message_i$ to $Node_B$
$\sigma_A = (1-\alpha)\sigma_A + \alpha\sigma_B$

Trigger - Periodical maintenance in $Node_A$
for all $message_i$ in $Node_A$ **do**
  if $time_{current} - timeSent_i > TimeOut$ **then**
    Remove $message_i$
    $\sigma_A = (1-\alpha)\sigma_A$
  end if
end for

in $Group_Y$. Since the σ for all other nodes in $Group_Y$ are unknown, they are assumed to be the current β_Y. It then calculates the average probability of all members of the group, based on previous β_Y, σ_A, and σ_B, as detailed in Algorithm 5.

Algorithm 5 Calculating Group Probability

Definition
β_Y - Group Probability for $Group_Y$

Trigger - $Node_A$ contacts $Node_B$
$Node_A$ sends σ_A to $Node_B$
$Node_A$ receives σ_B from $Node_B$
for all $Group_Y$ which contain $Node_B$ and $Node_A$ **do**
  $\beta_Y = \frac{\sigma_A+\sigma_B+\beta_Y\times(|Group_Y|-2)}{|Group_Y|}$
end for

3.2.3. Using Probabilities to Route. When two nodes contact each other, they independently determine their γ value. This is the maximum probability of all the groups they participate with and their individual probability. The node with the higher value is assumed to either have more consistent contact with the base station or to be a member of a group which has consistent contact. In either event, the node with the lower value transfers all messages to the node with the higher probability, then updates its individual probability based on successful delivery of a message. This algorithm is detailed in Algorithm 6.

Previously, the individual probability was adjusted by the individual probability, but as the system gains more information it can update based on the groups it contacts, rather than the individual nodes. For this reason, the individual probability (Algorithm 4) is adjusted to use the γ_B of the destination node, rather than the individual probability σ_B.

Algorithm 6 Routing Algorithm

Notation
β_A - List of all Group Probabilities in $Node_A$
γ_A - Max Group / Individual Probability of $Node_A$

Trigger - $Node_A$ contacts $Node_B$
In $Node_A$
$\gamma_A = max(\beta_A, \sigma_A)$
Transmit γ to $Node_B$
Receive γ_B from $Node_B$
if $\gamma_B > \gamma_A$ **then**
  Transmit messages to $Node_B$
  $\sigma_A = (1-\alpha)\sigma_a + \alpha\gamma_B$
else
  Receive messages from $Node_B$
end if

4. ANALYSIS

4.1. EXPERIMENTAL SETUP AND EVALUATION

To evaluate this algorithm and the impact of control variables a simulation was designed and implemented using MATLAB. Once the routing efficiency was determined, a power consumption comparison was performed in NesC, using a TOSSIM simulator modified with PowerTossim Z. For comparison, both simulations also ran Epidemic Routing and base Probabilistic Routing schema. Epidemic Routing is considered to be the ideal in terms of message time and delivery ratio, and is therefore used for comparison, while Probabilistic Routing is an improvement over Epidemic in terms of resources used. Other comparable algorithms were reviewed, but either use previous knowledge of the environment (such as Bubble Rap) or are targeted to solve different communication issues (Simbet performing node-to-node communication, while SocialCast performing publish/subscribe broadcasts). For these reasons, the Probabilistic and Epidemic schemas were considered as comparable algorithms. The control variables tested, along with ideal values identified, can be found in Table 4.1.

Table 4.1. Variable Reference Chart

Control Variables		Range Tested	Ideal Value
α	Probability Decay Rate	0.05 - 0.5	0.075
ψ	Group Formation Threshold	0.0001 - 0.005	0.004
τ	Group Merge Threshold	0.1 - 0.33	0.3
ϕ	Contact Decay Ratio	0.1 - 0.5	0.3
Ideal values tested by experimentation			
Variables			
λ	Contact Strength between nodes		
σ	Node's Probability of Delivery		
β	Group's Probability of Delivery		
γ	Maximum probability of Delivery		

4.2. SIMULATION DATA SOURCE

To accurately implement this algorithm, the node contacts must follow social patterns. Potential sources for this are either real-life tracking data or the results from a social prediction algorithm, such as the Caveman Model. For this reason the simulation used data obtained from an experiment conducted by the University of Cambridge at the 2005 Grand Hyatt Miami IEEE Infocom conference. Participants were asked to carry iMotes with them during the conference for 3 to 4 days to capture data on social interactions [7]. Although the experiment did not include base stations capable of collecting information, they did include static immobile nodes. For simulation purposes, these units were treated as data sinks.

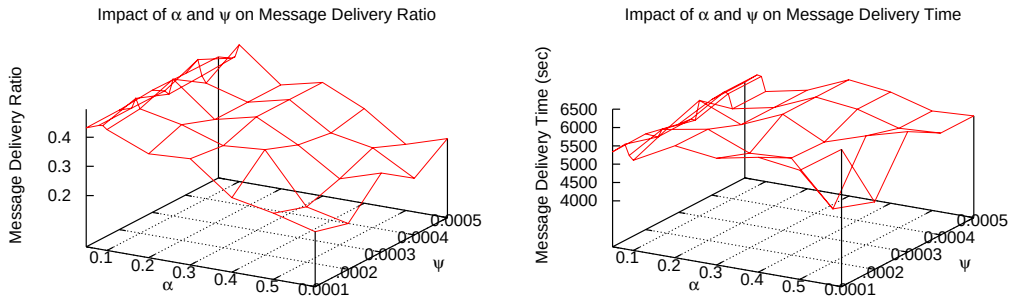
4.3. IMPACT OF α AND ψ

Based on the dataset used, the α setting is directly influenced by the dynamic nature of the contact patterns. In either long range simulations or simulations in which nodes consistently follow similar contact patterns, a lower value for α can result in more effective simulations. In contrast, a higher α results in the algorithm placing more weight in recent data. If the probability is more dynamic, it can more rapidly respond to changes in the layout.

The ψ setting influences the ease with which the nodes form social groups. In an environment in which nodes only encounter other nodes they are in a social group of, a lower value can result in the network being rapidly organized, whereas a higher ψ results in groups forming slowly, and only with regular contacts.

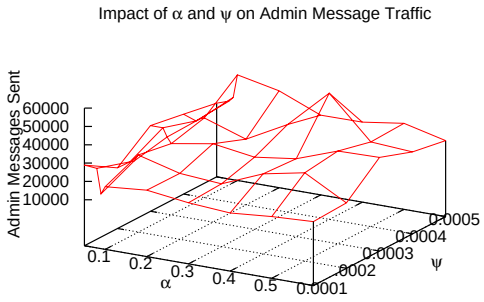
The results (as shown in Figure 4.1) indicate a peak Message Delivery Ratio as α decreases and ψ increases. The lower α seems to be due to the large amount of 'garbage' data - a node which successfully delivers to the basestation is quite likely just passing through, and will not likely be a good long-range contact. A lower α allows the algorithm to ignore

these incidental contacts and concentrate on nodes which regularly successfully deliver. Similarly, the ψ peak value is based on ignoring the large number of casual contacts. Because this dataset occurs at a conference, there are several regular contacts between nodes which are not members of the same group. To ignore this noise, a higher ψ setting is optimal.



(a) Message Delivery Ratio

(b) Message Delivery Time



(c) Admin Message Traffic

Figure 4.1. Impact of α and ψ

4.4. IMPACT OF τ AND ϕ

The control variable τ represents the level of similarity necessary for groups to be merged. A higher value for τ results in fewer groups being merged and keeps the group

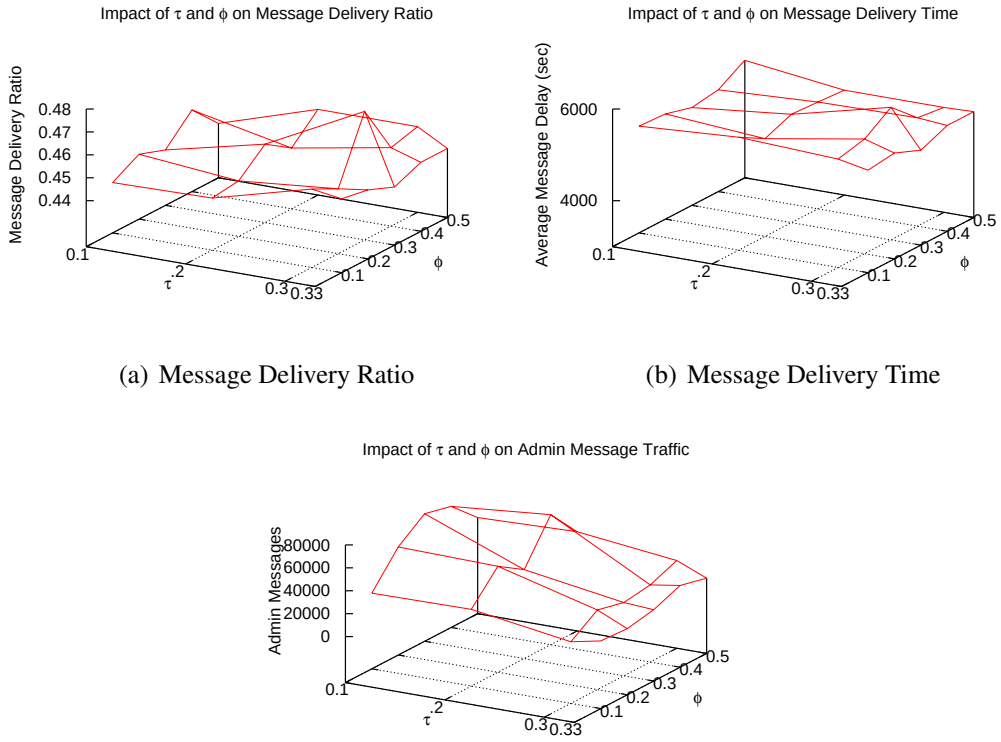
size small. This does not influence the number of groups being created; a higher value of τ will result in several smaller groups. Due to the ratio needed to ensure any group merges occur, the maximum value for τ is $\frac{1}{3}$. If it is higher, the initial 2-member groups cannot have enough common members to merge.

The contact deterioration ϕ determines how quickly the contact strength changes over time. A higher value for ϕ results in the contact strength changing more rapidly. Higher values would be used in more dynamic environment, so the nodes' contact strength are more influenced by recent events than historical data. This value is closely linked to the ψ value given earlier; more rapidly changing contact strength would result in needing a lower value of ψ to successfully form groups.

The results, shown in Figure 4.2, indicate that the algorithm functions best when the τ value is nearly, but not quite, at the maximum value. A τ of 0.3 allows groups to merge regularly, reflecting larger groups while still ignoring the casual group affiliations. Similarly, the ϕ results show peak functionality at the value 0.3, although this may be due to the ψ value already inserted. These metrics show the largest increase of delivery ratio, but also increase the average message delivery time (Figure 4.2(b)). That is due to this time only reflecting delivered messages – by increasing delivered messages, the system includes several messages which were ignored.

4.5. COMPARISON OF ROUTING ALGORITHMS

A review of the results (Figure 4.3) shows that the DSG Routing Scheme performs better than the Probabilistic Scheme in all metrics, and is comparable to the Epidemic in terms of both the delivery ratio and delivery time while having considerably less message traffic. Some of this is due to the applicability of the dataset – the social environment provided at the conference provided an area ideal for forming short-term dynamic groups. In contrast, the Probabilistic Schema had difficulty adjusting to the social groups that formed, and took too long to cascade the probabilities that would result from successful delivery.



(a) Message Delivery Ratio

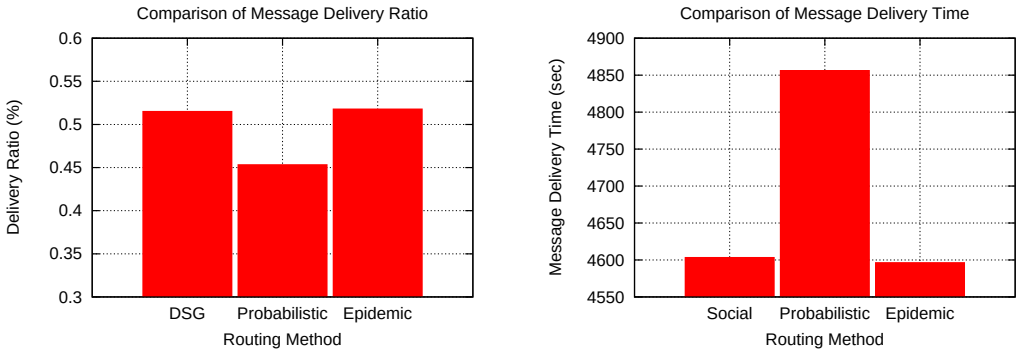
(b) Message Delivery Time

(c) Admin Message Traffic

Figure 4.2. Impact of τ and ϕ

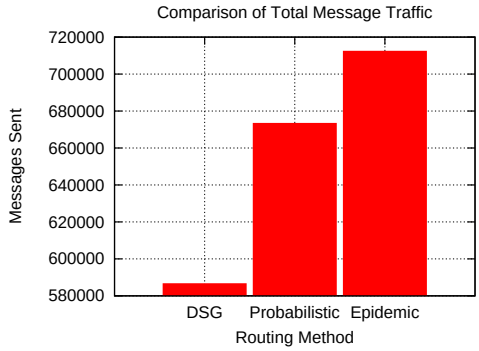
The results indicate that DSG is comparable to the ideal routing, with considerably less cost.

A cost comparison of the algorithms was performed in the TOSSIM simulation, using the PowerTOSSIMZ [8] model, as shown in Figure 4.4. The baseline shown in the graph shows the simulation's power comsumption when no routing occurs, to be used as a comparison point. We use this to find that the Dynamic Social Grouping's cost is 16% of Epidemic's cost, and 18% of the base Probabilistic algorithm's cost. This includes the overhead from group management such as group merge invitations, suggestions, and membership updates, but not standard maintenance such as neighbor discovery or message generation.



(a) Message Delivery Ratio

(b) Message Delivery Time



(c) Total Message Traffic

Figure 4.3. Comparison of DSG Routing, Probabilistic Routing, and Epidemic Routing

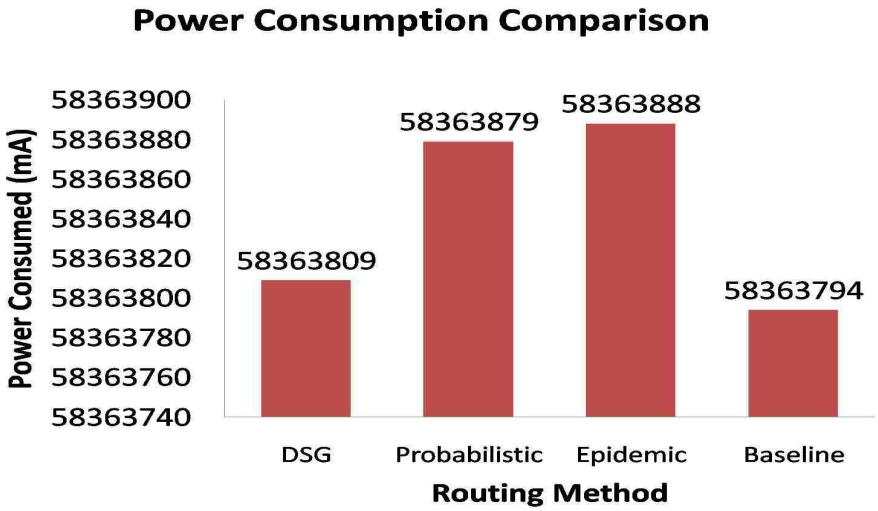
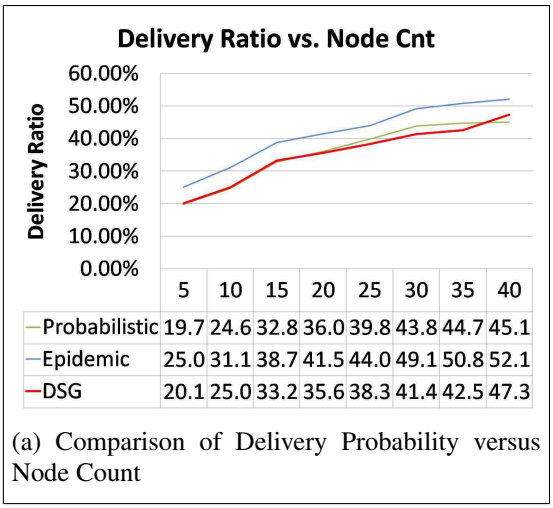
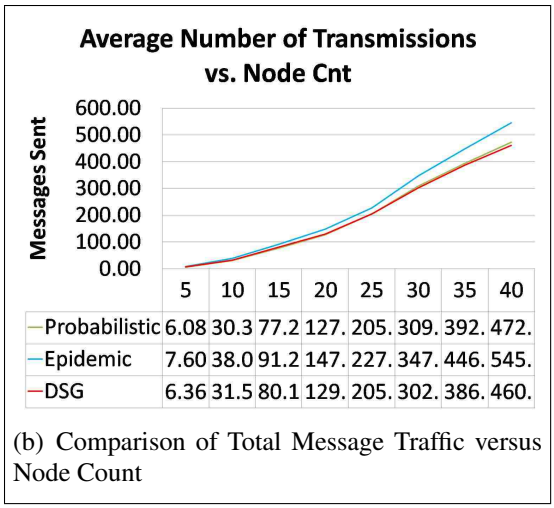


Figure 4.4. Comparison of DSG, Probabilistic, and Epidemic Power Consumption

When considering a routing algorithm for large-scale deployment, there is always concern by how well the algorithm scales to a higher number of nodes. For this reason, scalability experiments were performed to find the performance of the DSG algorithm with differing number of nodes. Because the experiment was performed using a real-world dataset, it is impossible to add additional nodes to the experiment, but relatively simple to randomly remove nodes from consideration. The results show that the ratios from the three algorithms tend to increase as the number of nodes increases, improving connectivity, but the DSG algorithm doesn't show a marked improvement until the full 40 nodes are in use (Figure 4.5(a)). This is due the same control variables being used through the entire experiment set, from 5 to 40 nodes. As indicated earlier, the algorithm is sensitive to the control variables used during execution (detailed in Table 4.1) - by using these variables despite the different environments, performance fluctuates wildly. Nonetheless, the costs of the system remains consistently lower than the alternatives (Figure 4.5(b)). This is due to the nature of the administration traffic. As the number of nodes decreases, reducing node connectivity, the administration overhead is reduced. Fewer groups are being formed, merged, or updated. As the connectivity increases, it improves routing performance by enough to offset the overhead.



(a) Comparison of Delivery Probability versus Node Count



(b) Comparison of Total Message Traffic versus Node Count

Figure 4.5. Scalability Experiments

5. CONCLUSIONS

The Dynamic Social Grouping (DSG) algorithm has been shown to provide a significant increase in efficiency over probabilistic routing and epidemic routing. While maintaining a lower overhead than either epidemic or probabilistic routing, DSG has managed to achieve as high a message delivery ratio and as low a delivery time as can be expected. This will lead to better data aggregation and longer battery life, both of which are primary goals in modern wireless sensor networks.

There remains further research to consider the implications of alternative merge methods. The current method using a simple ratio depicting commonality may not be optimal, and experiments in alternative methods may provide further improvement. It may also be improved by introducing a load-balancing method to equalize the drain on commonly used nodes, further improving the network lifespan. Load balancing techniques can also change the clusterhead, to spread energy drain among the group. A final improvement to consider is the expansion of this algorithm to include point-to-point communication, expanding its impact to include a wider range of applications.

6. BIBLIOGRAPHY

[1] Y. Wang and H. Wu, "Delay/fault-tolerant mobile sensor network (dft-msn): A new paradigm for pervasive information gathering," *IEEE Transactions on Mobile Computing*, vol. 6, no. 9, pp. 1021–1034, 2007.

[2] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," tech. rep., Duke University, 2000.

[3] Z. Feng and K.-W. Chin, "A survey of delay tolerant networks routing protocols," *CoRR*, vol. abs/1210.0965, 2012.

[4] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay-tolerant networks," *Mobile Computing, IEEE Transactions on*, vol. 10, pp. 1576 –1589, nov. 2011.

[5] P. Costa, C. Mascolo, M. Musolesi, and G. Picco, "Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks," *Selected Areas in Communications, IEEE Journal on*, vol. 26, pp. 748–760, June 2008.

[6] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant MANETs," in *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '07, (New York, NY, USA), pp. 32–40, ACM, 2007.

[7] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD data set cambridge/haggle (v. 2009-05-29)." Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle, May 2009.

[8] E. Perla, A. O. Catháin, R. S. Carbajo, M. Huggard, and C. Mc Goldrick, "Powertossim z: realistic energy modelling for wireless sensor network environments," in *PM2HW2N '08: Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, (New York, NY, USA), pp. 35–42, ACM, 2008.

II. DSG-N^2: A Group-Based Social Routing Algorithm

Roy Cabaniss∗, James M. Bridges∗, Andrew Wilson†, and Sanjay Madria∗

∗ Department of Computer Science,

Missouri University of Science and Technology, Rolla, Missouri 65401

†Department of Computer Science,

Truman State University, Kirksville, Missouri 63501

Devices in a mobile ad-hoc environment can follow different movement patterns based on the application environment. Some environments, such as mass transit systems, follow regular and predictable patterns. Others, such as an aerial monitoring network, generally follow random paths. Optimal routing schemes tend to take advantage of information regarding movement patterns available in social interaction domains. In a social environment like wildlife tracking or monitoring socio-human interactions, the devices and/or users will follow regular contact habits, tending to encounter social groups in which they participate. In this paper, by dynamically identifying these groups, the patterns are used to speed routing through a social environment. When social groups are formed, a probability based scheme is used to route messages to devices efficiently. This algorithm can be implemented *ad null*, meaning the devices have no information of their environment, and works to reduce overhead, message traffic, and delivery time while maintaining a high delivery ratio.

1. INTRODUCTION

Mobile Ad-hoc Networks (MANETs) are a collection of computing devices connected through wireless communications, such as Bluetooth or wireless LAN. Each node in a MANET can move freely throughout the network. In environments with unpredictable and changing motions, such as wildlife sensors, traditional routing algorithms are ineffective or inapplicable. In order to address this issue new routing methods have been developed. These new routing methods are often examples of Delay Tolerant Routing, in which nodes are able to retain copies of messages and forward them at a more opportune time.

As a rule, routing algorithms are more effective when they rely on information regarding the contact patterns of nodes. Conventional probability based routing scheme assumes consistent avenues of communication; nodes which interacted with a set group of nodes in the past will do so again in the future [1]. Similarly, social routing assumes that nodes assigned to the same social network (classroom, project team, department faculty, etc.) will regularly interact with members of that social group. To take advantage of the partial applicability of both social and probability based routing, our grouping method is designed form social groups based on contact patterns. Once these groups are identified, consistent routes to nodes are recognized based on the delivery history of a group or node. The new algorithm, called Dynamic Social Grouping: Node-to-Node (DSG-N^2), was based on a previous algorithm [2] and expanded to include node-to-node capability and improve the grouping and routing efficiency. The new algorithm was them compared to two well known algorithms referred to as Epidemic routing [3] and SimBet routing [4]. Results show that DSG-N^2 performs better than both routing algorithms in terms of total message traffic and delivery time.

The purpose of DSG-N^2 is to reliably deliver messages from a node to another using as few communications as possible. In this network scenario every node is capable

of movement, due to the host being mobile, and contact between each node is not always reliable or consistent. This algorithm is designed to identify social groups based solely on contact patterns between nodes without relying on full knowledge of the network. A node that is part of multiple identified social groups can suggest that two groups be merged based on common members and delay between groups. Routing occurs through groups or nodes that have a high probability of delivering the message, based on previous message deliveries, to the intended destination node. Simulations with real-world data comparing DSG-N^2 to Epidemic and SimBet routing algorithms show that our Dynamic Social Grouping algorithm is superior in terms of delivery ratio and message costs to both (see Section 4 for details). Applications of DSG-N^2 include inventory tracking or tracking human interactions, although it can improve any environment in which groups of regular contacts are formed.

2. BACKGROUND

Delay Tolerant Networks (DTN) have received considerable attention from researchers in recent years due to their widespread occurrences in a variety of applications. Routing and data aggregation are of particular interest as they considerably affect the longevity of the sensor nodes and the performance of the network as a whole. A survey of routing techniques for DTNs is presented in [5].

2.1. EPIDEMIC ROUTING

Epidemic routing [3] is designed to route messages through partially connected networks in a semi-probabilistic fashion. The routing is done through copying and forwarding packets to other parts of the network in order to reach more sparse sections. When two nodes meet any unshared messages are transmitted, and the message's hop count is incremented. A message is passed until its hop count has reached the max. Through this scheme, Epidemic routing is able to maximize the amount of messages successfully delivered without taking into account delivery time. This creates a large message overhead, redundantly transmitting messages throughout the network.

2.2. PROBABILISTIC ROUTING

This scheme is based on individual nodes tracking their probability of delivering a message successfully to a base station. The version introduced by Yu Wang functions by individual nodes tracking their performance based on individual messages [1]. Each node is assigned a probability estimate of successfully delivering a message to the basestation node, and increases this estimate with successful delivery to nodes with higher probability, or decreases as messages expire. The probability is therefore based on the contacts of nodes, and as such works well when nodes follow regular, if not necessarily social, contact

patterns. The algorithm is therefore an effective routing system, but it only takes advantage of any data other than past performance.

2.3. PROPHET

A similar routing scheme currently implemented is Probabilistic ROuting Protocol using History of Encounters and Transitivity [6], introduced initially to allow efficient communications in a sparse environment. Similar in many way to the Probabilistic method in Section 2.2, this method allows individual nodes to track their contact patterns to other nodes. Unlike the previous method, it is based not on message performance, but on the contacts themselves; the nodes will adjust their probability when they meet, whether any messages are transmitted or not. This method is also an effective algorithm in a sparse network, but also limits the data it applies to routing decisions.

2.4. BUBBLE RAP

The Bubble-Rap grouping method [7] allocates nodes into social groups based on direct and indirect contacts. They distribute using a method called $k-cliques$, in which all fully connected groups of k members are considered a distinct social group and are then merged with all other $k-cliques$ which can be reached through $k-1$ nodes. This provides an accurate representation of the social groups formed by a set of nodes. However, it relies on global knowledge of the nodes' contacts, and it must have them before the algorithm can create groups. In this sense, k-grouping is neither dynamic nor distributed, which limits it's applicability to a MANET.

2.5. SIMBET

The SimBet routing algorithm [4] is similar to DSG-N^2 in that it can deliver messages node to node using social patterns. Nodes engage in 'conversations' when they meet and exchange information about data messages and current neighbors. This information

is used to determine ‘betweenness’ and ‘similarity’. Betweenness is a measure of how often a node lies on a path between otherwise unconnected nodes. Similarity is a count of common neighbors between two nodes. These values can accurately predict which node is more likely to successfully forward a message to the destination. Both betweenness and similarity are locally determined; global knowledge of the network is not needed. A notable disadvantage of SimBet is the amount of messages the aforementioned conversations send across the network, resulting in significant message traffic and strain on the node’s limited power supply. Also, relationships between nodes in SimBet are represented as either in contact or not in contact. More specific and informative bookkeeping, such as percent of time in contact, would result in a better measure of a node‘s likelihood to forward a message to the destination.

2.6. SIMBETAGE

Link et al. provides an improvement to the SimBet routing protocol to deal with more dynamic networks [8]. Using SimBet as a baseline, SimBetAge improves upon it by adding a ‘freshness’ value, which dictates how often a node A has contacted node B and changes based on an encounter event or time event. On an encounter the freshness value will increase by a logistic growth function and on a time step event the freshness value will decrease in proportion to a exponential decay function. Since this algorithm takes into consideration temporal changes in the network it allows for more reliable and direct routing to occur.

3. PROPOSED ALGORITHM

DSG-N^2 addresses two distinct subproblems. First, given an arbitrary collection of nodes, the network must recognize cohesive social patterns and identify them as groups, at the same time distributing this information throughout the network. This network organization can adjust to reflect changes to the network or to the social patterns of the nodes. Second, once the groups have been recognized, a route must be identified to deliver messages to other nodes. The algorithm must minimize the number of message repetitions while ensuring a high delivery ratio.

3.1. GROUPING

A major advantage this algorithm has over conventional mobile ad-hoc routing methods is the use of social groups to improve communication throughput. The task of identifying such groups, however, requires knowledge of the nodes that is not present at startup.

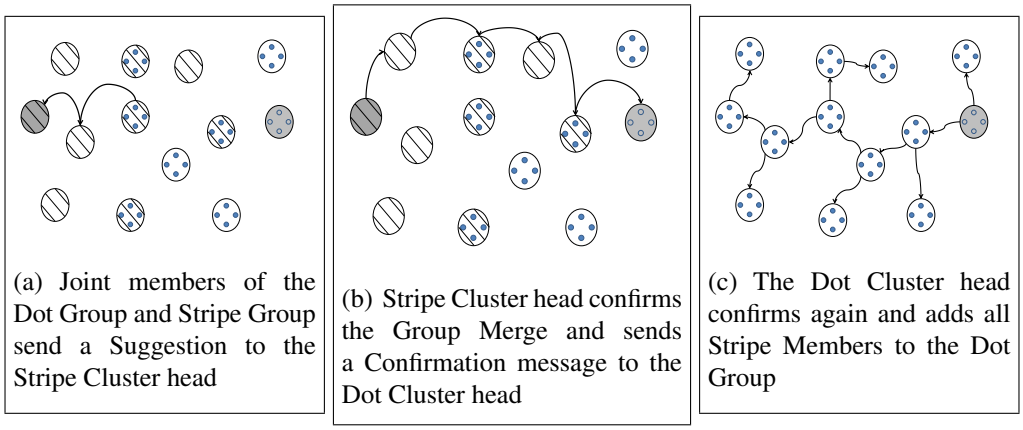
3.1.1. Contact Strength. The first task in identifying a social group is to calculate a metric for the contact frequency two nodes have with one another. The symbol $\lambda_{i,j}$ is used to represent the contact strength between two nodes. This is measured as a function of the duration for which two nodes, i and j, were in contact and the time that they were not in contact. The symbol α is used to determine how much the λ changes based on new data, mimicking a Markov process. Initially, $\lambda_{i,j}$ will be 0 between all nodes, but when nodes contact one another, it is recalculated as shown in Eqn 1.

$$\lambda_{A,B} = (1-\alpha)\lambda_{A,B} + \alpha\frac{time_{contact}}{time_{contact} + time_{nocontact}} \tag{1}$$

3.1.2. Forming New Groups. When the $\lambda_{i,j}$ exceeds or is equal to a threshold ψ, the nodes identify themselves as being members of the same group. Initially, all groups will have two members and the cluster head of the group is arbitrarily chosen. The cluster

head has the power to confirm an invitation for a merge, merge two groups, and send out kill group messages (see section 3.1.4 for details). The group will also have a probability assigned to it based on a weighted probability average (more information on this in Section 3.2).

3.1.3. Merging. Nodes regularly review their group listing for potential merges. When a node determines two groups should merge with one another, a suggestion message is sent to the cluster head of a group as shown in Figure 3.1(a). The criteria for a suggestion of a group merge is that the ratio of common members over total unique members is greater than τ. Once the cluster head receives the suggestion, it uses an up-to-date member list to determine if the two groups are fit for merging with the same criteria as the joint member. The cluster head will then send out an invitation message to the other cluster head epidemically as seen in Figure 3.1(b). Once the other cluster head confirms the ratio it measures the delay of the invitation message. If the delay is greater than one day, the merge is refused; otherwise the cluster head sends out a kill message to remove the other group and adds that group's member list to its own. This process is detailed in Algorithm 7.



(a) Joint members of the Dot Group and Stripe Group send a Suggestion to the Stripe Cluster head

(b) Stripe Cluster head confirms the Group Merge and sends a Confirmation message to the Dot Cluster head

(c) The Dot Cluster head confirms again and adds all Stripe Members to the Dot Group

Figure 3.1. Three Stages of Merging Groups (figures provided by [2])

Algorithm 7 Merging Groups

Notation
$Group_Y$, $Group_Z$ - Any social groups in MANET
$Node_A$ - Member of both $Group_Y$ and $Group_Z$
$Node_B$ - Cluster head of $Group_Y$, $Clusterhead_Y$
$Node_C$ - Cluster head of $Group_Z$, $Clusterhead_Z$
τ - Threshold for Merging a Group
MTT - Max Transit Time

Trigger - Periodically in $Node_A$
for all $Group_Y, Group_Z$ of which $Node_A$ is a member **do**
  if $\frac{|Group_Y \bigcap Group_Z|}{|Group_Y \bigcup Group_Z|} > \tau$ **then**
    Send SUGGEST Message to arbitrary Group ($Group_Y$ or $Group_Z$)
  end if
end for

Trigger - $Node_A$ contacts $Node_B$
In $Node_A$...
for all $Group_Y$ which contain both $Node_A$ and $Node_B$ **do**
  if $Node_A$ has Control Messages for $Group_Y$ **then**
    $Node_A$ sends Control Messages to $Node_B$
  end if
end for

Nodes pass Control Messages epidemically to all Group Members

Trigger - $Node_B$ receives SUGGEST message
if $Node_B$ is $Clusterhead_Y$ **then**
  if $\frac{|Group_Y \bigcap Group_Z|}{|Group_Y \bigcup Group_Z|} > \tau$ and transit time $< MTT$ **then**
    Send CONFIRMATION Message to all members of $Group_Z$
  end if
end if

The Cluster head confirms the suggestion, and sends a Confirmation to the other group

Trigger - $Node_C$ receives CONFIRMATION message
if $Node_C$ is $Clusterhead_Z$ **then**
  if $\frac{|Group_Y \bigcap Group_Z|}{|Group_Y \bigcup Group_Z|} > \tau$ and transit time $< MTT$ **then**
    $Group_Z = Group_Z \bigcup Group_Y$
    $Group_Y = \varnothing$
    Send KILL message to $Group_Y$
  end if
end if

If both Group Cluster heads approve, the arbitrary group is added to the other, and then removed.

3.1.4. Dynamic Grouping. Sometimes a node becomes alienated from a group because of movement and the contact frequency ratio does not meet the group's required value of ψ. In this case a node needs to be removed from that group by sending a resign message to the cluster head, as shown in Algorithm 8. In order to reduce group fragmentation the cluster head needs to know about the resignation immediately. Periodically a node will check to see whether or not it still belongs in a group. When this is done, the node will calculate its $Group\lambda_{A,Y}$, which is the average $\lambda_{A,B}$ between the node and all members of the group, and compare that value to ψ. If the average is less than ψ the node will send a message to the cluster head that states that the node is no longer part of the cluster head's group. This message removes all information regarding that node from the group, and the rest of the group must be updated with new group information. Until the update has propagated through the group, the group data is considered fragmented. This process is more thoroughly explained in Section 3.2.

Algorithm 8 Dynamic Groups

Notation
$Node_A$ - Member of $Group_Y$
$Node_B$ - Cluster head of $Group_Y$, $Clusterhead_Y$
$NodeList_Y$ - All nodes in $Group_Y$
$Group\lambda_{A,Y}$ - Average Contact Strength between $Node_A$ and $NodeList_Y$
ψ - Threshold for Forming a Group

Trigger - $Node_A$ checks it's groups
In $Node_A$
$Group\lambda_Y = Average(\lambda_{A,Y})$
if $Group\lambda_Y < \psi$ **then**
  Send RESIGN to $Node_B$
end if

Trigger - $Node_B$ receives RESIGN from $Node_A$
$NodeList_Y = NodeList_Y - Node_A$
Update all nodes in $Group_Y$ of removal and $Node_A$

3.1.5. Group Versions. A major concern when dealing with multiple nodes in a Delay Tolerant Network is keeping group information up to date and decreasing update overhead. For instance, assume two nodes of the same social group meet and they wish to exchange information about the groups they have in common. There would be a problem if there was no way to track which node has the most up-to-date member list and other properties specific to the group.

To solve this problem group versions are introduced. Every time contact patterns change (social patterns) there is a possibility of group membership changing or a group being removed altogether. If the group information is altered in any way, the cluster head, which has control of group membership, will update the group information and increment a version number. A higher version number indicates more recent information. If a node encounters another node that has a higher version number then the information is transferred. This process is further detailed in Algorithm 9

Algorithm 9 Group Updates

Notation
$Version_{A,Y}$ - Version Number of $Group_Y$ kept by $Node_A$

Trigger - $Node_A$ contacts $Node_B$
In $Node_A$
for all $Group_Y$ which contains both $Node_A$ and $Node_B$ **do**
  Send $Version_{A,Y}$ to $Node_B$
  Receive $Version_{B,Y}$ from $Node_B$
  if $Version_{A,Y} > Version_{B,Y}$ **then**
    Send $Group_Y$ to $Node_B$
  else if $Version_{A,Y} < Version_{B,Y}$ **then**
    Receive $Group_Y$ from $Node_A$
  else if $Version_{A,Y} = Version_{B,Y}$ **then**
    Do nothing
  end if
end for

3.2. ROUTING

Now that the nodes have been organized into groups this information can be used to route data to other nodes. Similar to Probabilistic Routing, each node is assigned a vector of probability values based on the chance that a message is successfully delivered to another node. In order to transfer a message throughout the network a node will forward a message to other nodes that have higher probability of delivering the message to the intended node. DSG-N^2 improves upon the Probabilistic model by incorporating group probabilities. Using group dynamics increases system awareness, in turn improving the network's delivery ratio.

3.2.1. Individual Probability. Nodes are initially assigned a default probability σ. When nodes encounter one another, they compare probabilities to determine routing (more on this in section 3.2.3). The node with lower probability for a given destination will forward all messages for that destination to the node with the higher probability. When a node successfully forwards a message, its probability to the destination is updated. The rate at which the probability changes is controlled by ϕ, similar to α's relation to contact strength. For details, review Algorithm 10.

In addition to message transmission, individual probability can be updated to reflect the inability to deliver a message. Messages eventually timeout in a network. When this happens a node's probability is decreased. This reflects its inability to transfer a message to the destination node.

3.2.2. Group Probability. Each node calculates vector of group probabilities, denoted as $\vec{\beta}$, independently based on the contact patterns of that specific node. Group probability differs among members, but this allows nodes to take into account only neighbors in the group. Nodes that exist in a common group and are not encountered are estimated using the current group probability. The procedure begins when $Node_A$ contacts $Node_B$ and both nodes are in $Group_Y$. The algorithm is a weighted average of the individual probabilities, with the weight determined by the contact strength. Since only the encountered

Algorithm 10 Calculating Individual Probability

Notation
$Node_A$, $Node_B$, $Node_C$ - Nodes in MANET
σ_A - Individual Probability for $Node_A$
σ_B - Individual Probability for $Node_B$
$message_i$ - Message in MANET
$timeSent_i$ - Time $message_i$ was sent
TTL - Parameter determining max duration of messages (time to live)
ϕ - Control Parameter determining Probability decay ratio

Trigger - $Node_A$ transmits $message_i$ to $Node_B$
if $Node_A$ can forward $message_i$ to $Node_B$ for $Node_C$ **then**
  $\sigma_{A,C} = (1-\phi)\sigma_{A,C} + \phi\sigma_{B,C}$
else
  $\sigma_{A,C} = (1-\phi)\sigma_{A,C}$
end if

Trigger - Periodical maintenance in $Node_A$
for all $message_i$ in $Node_A$ **do**
  if $time_{current} - timeSent_i > TTL$ **then**
    Remove $message_i$
    $\sigma_{A,C} = (1-\phi)\sigma_{A,C}$
  end if
end for

node's probability is known, the σ of other group members is estimated as the current group probability. This is simplified and shown in Algorithm 11.

3.2.3. Using Probabilities to Route. When two nodes contact each other, they independently determine their γ value. This is the joint probability of all the groups they participate with and their individual probability. The node with the higher value is assumed to either have more consistent contact with the destination node or to be a member of a group which has consistent contact. In either event, the node with the lower value transfers all messages to the node with the higher probability, then updates its individual probability based on successful delivery of a message. This algorithm is detailed in Algorithm 12.

When adjusting the individual probability of a sending node, the system initially uses the probability of the receiver. As time elapses, nodes gain more information regarding

Algorithm 11 Calculating Group Probability

Definition
β_Y - Group Probability for $Group_Y$
$Node_A$, $Node_B$ - Nodes in $Group_Y$

Trigger - $Node_A$ contacts $Node_B$ in a group of size 2
$Node_A$ sends σ_A to $Node_B$
$Node_A$ receives σ_B from $Node_B$
for all $Group_Y$ which contain $Node_B$ and $Node_A$ **do**
$\beta_Y = \frac{\vec{\sigma_A}+\vec{\sigma_B}\lambda_{A,B}}{1+\lambda_{A,B}}$
end for

Trigger - $Node_A$ contacts $Node_B$ in a group of size > 2
$Node_A$ sends σ_A to $Node_B$
$Node_A$ receives σ_B from $Node_B$
for all $Group_Y$ which contain $Node_B$ and $Node_A$ **do**
$\vec{\beta_Y} = \frac{\vec{\beta_Y}(\sum\lambda_{A,*}-\lambda_{A,B})+\lambda_{A,B}\vec{\sigma_B}}{\sum\lambda_{A,*}}$
end for

their social structure, resulting in a joint probability which can differ from its σ value. For this reason, the individual probability (Algorithm 10) is adjusted by the γ of the destination node, rather than the individual probability.

Algorithm 12 Routing Algorithm

Notation
β_A - List of all Group Probabilities in $Node_A$
$jointIndProb_A$ - Group / Individual Probability of $Node_A$

Trigger - $Node_A$ contacts $Node_B$
In $Node_A$ $jointIndProb_A = (1 - \sigma_{A,B})$
for all Groups $Node_A$ is a member of **do**
  $jointIndProb* = (1 - Group_Y)$
end for
$jointIndProb = (1 - jointIndProb)$
Transmit $jointIndProb$ to $Node_B$
Receive $jointIndProb_B$ from $Node_B$
if $jointIndProb_B > jointIndProb_A$ **then**
  Transmit messages to $Node_B$
  $\sigma_{A,B} = (1 - \phi)\sigma_a + \phi jointIndProb_B$
else
  Receive messages from $Node_B$
end if

4. ANALYSIS

4.1. EXPERIMENTAL SETUP AND EVALUATION

DSG-N^2 was first implemented in MATLAB to evaluate the control parameters and as a proof of concept. After optimal values were determined, the algorithm was implemented using TOSSIM and a virtual hardware simulation was performed. Data was output from the simulation depicting the time messages were generated, received, and the packet size of messages sent at each node to determine power consumption, average delivery time and delivery ratio. The SimBet and Epidemic Routing Algorithms were also implemented in the hardware simulation, and the results compared to DSG-N^2. Other routing algorithms were reviewed, but either use previous knowledge of the environment (Bubble Rap [7]) or solve different communication issues (Probabilistic [1] performs node-to-basestation communications). For these reasons, the SimBet and Epidemic schemes were considered as comparable algorithms. The control variables tested, along with ideal values identified, can be found in Table 4.1.

Table 4.1. Variable Reference Chart

Control Variables		Range Tested	Ideal Value
ϕ	Probability Decay Rate	0.1 - 0.7	0.6
ψ	Group Formation Threshold	0.1 - 0.7	0.3
α	Contact Decay Ratio	0.1 - 0.7	0.4
Ideal values tested by experimentation			
Variables			
λ	Contact Strength between nodes		
σ	Node's Probability of Delivery		
β	Group's Probability of Delivery		

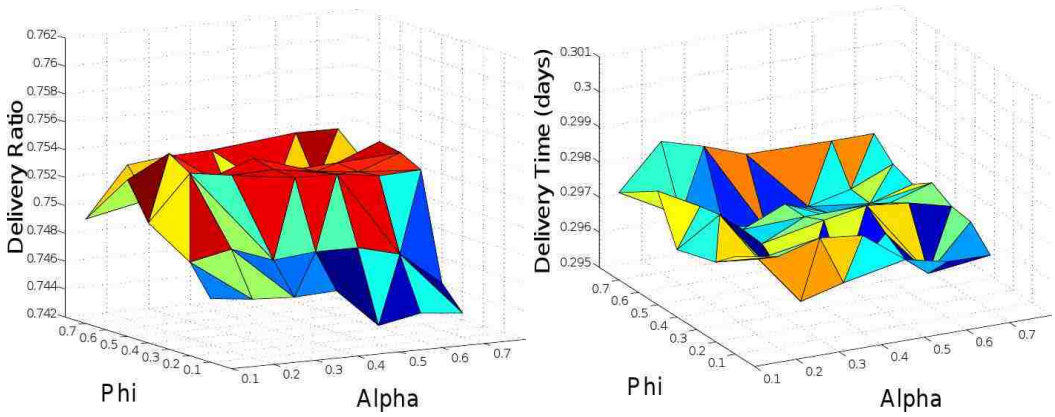
4.2. SIMULATION DATA SOURCE

In order to test our algorithm in a simulated environment a record of social interactions was needed. The data used for these simulations originated from the MIT Reality Dataset [9]. The study tracked 100 nodes (people) with cell phones and Bluetooth devices over the course of nine months. A set of Symbian based phones was programmed to measure contact data and messages, and participants were asked to carry and use them. Multiple social patterns developed between participants, which resulted in an ideal data set for testing social algorithms.

4.3. IMPACT OF α, ϕ, AND ψ

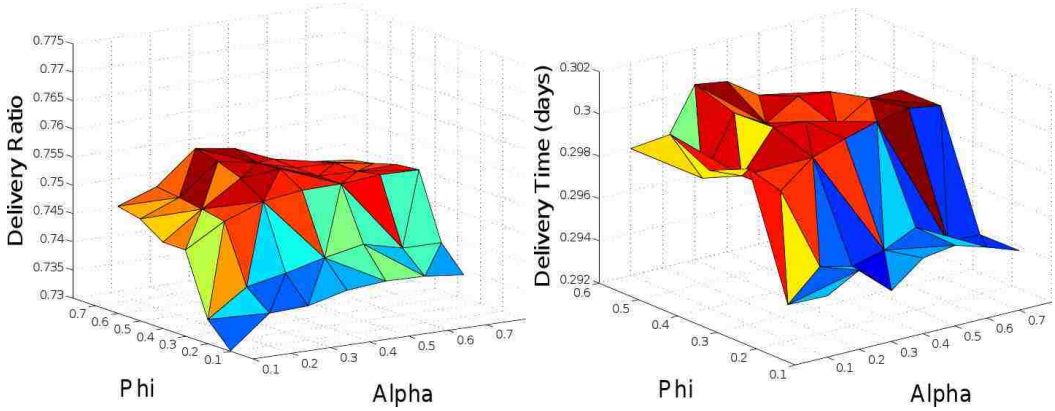
Based on the simulations (Figure 4.1), when ψ is low the value of α will have no effect on the delivery ratios, since any node to node contact will result in a new group. These groups do not accurately reflect social groups formed by nodes, lowering the delivery ratio. As ψ is increased (Figure 4.2), α will start to behave differently and make the groups more relevant. Based on current results, no matter the ψ the best ϕs seem to be midrange. These ϕs seem to hold probabilities between nodes at a relevant level. It appears as if upper and lower bound values of ϕ holds on to or ignores probability history, respectively.

Regarding delivery time, a low ψ will yield evenly distributed results. This is due to groups being easy to form and nodes transmitting messages liberally rather than finding efficient delivery routes. When ψ is increased and ϕ is low, meaning probability history is ignored, at any α the delivery time will be lower. The delivery ratio takes a hit due to contact history being ignored, but the average transmission time is better.



(a) Message Delivery Ratio (b) Message Delivery Time

Figure 4.1. Impact of α and ϕ with $\psi = .1$

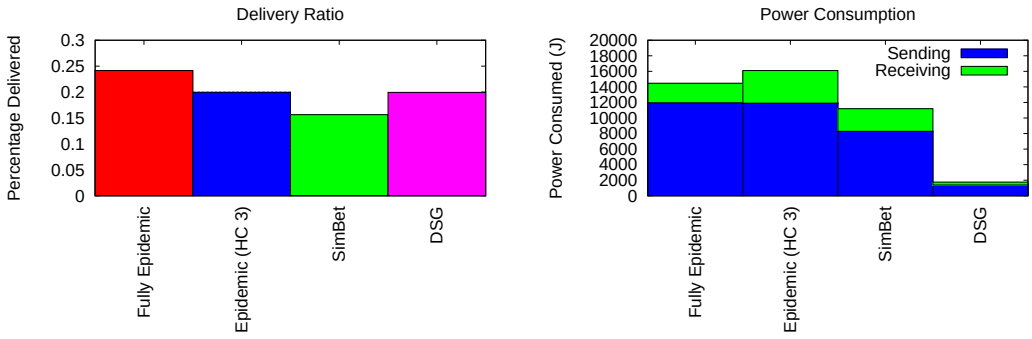


(a) Message Delivery Ratio (b) Message Delivery Time

Figure 4.2. Impact of α and ϕ with $\psi = .2$

4.4. COMPARISON OF ROUTING ALGORITHMS

When simulating the Epidemic routing protocol, there were two cases which were tested, based on propagation distance. The baseline version limited the hopcount to three, while the full execution had an unlimited hopcount where messages only ceased when they expired. SimBet was executed with default control parameters, and DSG-N^2 with control parameters of $\alpha = 0.4$, $\phi = 0.6$, and $\psi = 0.3$, based on optimal values found in MATLAB simulations 4. As seen in Figure 4.3(a), DSG-N^2 performed much better than both SimBet and Epidemic Routing. Epidemic underperformed in the TOSSIM environment, due primarily to realistic implementation of limiting factors such as buffer sizes, message collision, and limited time to transmit messages. Additionally, the power consumption was far less using the Dynamic Social Grouping algorithm (Figure 4.3(b)). Similarly, SimBet did not perform as well in regards to either delivery ratio or power consumption, due primarily to the disconnected nature of the environment. SimBet would perform optimally in a static or strongly connected environment, which does not apply to the real-world dataset used.



(a) Message Delivery Ratio

(b) Power

Figure 4.3. Comparison of DSG-N^2 Routing, SimBet, and Epidemic Routing

5. CONCLUSION

Dynamic Social Grouping with Node-to-Node transmissions (DSG-N^2) performs better than SimBet and Epidemic routing in both power consumption and delivery ratio for the dataset specified by identifying the regular contact groups and routing accordingly. Using the additional information yields considerable benefits for message delivery in a social environment.

Future improvements include optimizing the grouping metric even further, meaning not just based on time, but some other applicable metric. Adding load balancing logic to move the cluster head around in the group to the most optimal position is another possible improvement. Different environments can be explored, such as tool sets, datasets, or devices for DSG-N^2. A final improvement would be to make DSG-N^2 transparent to developers, easing implementation in a real world environment.

6. BIBLIOGRAPHY

[1] Y. Wang and H. Wu, "Delay/fault-tolerant mobile sensor network (dft-msn): A new paradigm for pervasive information gathering," *IEEE Transactions on Mobile Computing*, vol. 6, no. 9, pp. 1021–1034, 2007.

[2] R. Cabaniss, S. Madria, G. Rush, A. Trotta, and S. S. Vulli, "Dynamic social grouping based routing in a mobile ad-hoc network," *Mobile Data Management, IEEE International Conference on*, vol. 0, pp. 295–296, 2010.

[3] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Duke University, Tech. Rep., 2000.

[4] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant MANETs," in *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '07. New York, NY, USA: ACM, 2007, pp. 32–40. [Online]. Available: http://doi.acm.org/10.1145/1288107.1288113

[5] E. P. Jones and P. A. Ward, "Routing strategies for delay-tolerant networks," *Submitted to ACM Computer Communication Review (CCR)*, 2006.

[6] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," in *SIGMOBILE Mobile Computing and Communication Review*, 2004, p. 2003.

[7] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay-tolerant networks," *Mobile Computing, IEEE Transactions on*, vol. 10, no. 11, pp. 1576 –1589, nov. 2011.

[8] J. A. Bitsch Link, N. Viol, A. Goliath, and K. Wehrle, "Simbetage: utilizing temporal changes in social networks for pocket switched networks," in *U-NET '09: Proceedings of the 1st ACM workshop on User-provided networking: challenges and opportunities*. New York, NY, USA: ACM, 2009, pp. 13–18.

[9] N. Eagle, A. Pentland, and D. Lazer, "Inferring social network structure using mobile phone data," Downloaded from http://reality.media.mit.edu/download.php, pp. 15 274 – 15 278, 2009.

III. Social Group Detection Based Routing in Mobile Ad Hoc Networks

Roy Cabaniss, Srinivasa S. Vulli, and Sanjay Madria

Department of Computer Science,

Missouri University of Science and Technology, Rolla, Missouri 65401

When developing and implementing a Mobile Ad Hoc Network, a key characteristic of the network topology is the mobility pattern of the nodes. Based on the application, nodes can follow semi-predictable patterns, such as the routes followed by Vehicular Ad Hoc Networks, or the more strict schedules followed by aerial reconnaissance . Optimal routing schemes tend to take advantage of any information regarding these contact patterns. In social environments, such as wildlife tracking or sending messages between humans, the devices and/or users will follow regular contact habits, tending to encounter social groups in which they participate. By dynamically identifying these groups, the patterns are used to optimize routing through a social environment. One of two algorithms can be used to route messages based on these social groups. Dynamic Social Grouping (DSG), used to route messages strictly from a node to a basestation, is ideal for gathering sensor data and updating a shared data cache. In contrast, Dynamic Social Grouping - Node to Node (DSG-N^2) is used to route messages between nodes, generally conventional communications. Both of these algorithms can be implemented *ad null*, meaning the devices initially have no information about their environment, and they work to reduce overhead, message traffic, and delivery time while maintaining a high delivery ratio.

1. INTRODUCTION

Mobile Ad-Hoc Networks (MANETs) are a collection of mobile computing devices connected through wireless communications, such as Bluetooth or wireless LAN. Each node in a MANET can move freely throughout the network, generally following the mobility patterns based on the type of device[1] [2]. For sensors in environments with unpredictable and changing motions, such as wildlife sensors [3], traditional routing algorithms are ineffective or inapplicable. In order to address this issue new routing methods have been developed. These methods are often examples of Delay-Tolerant Networks (DTN) [4], in which nodes are able to retain copies of messages and forward them at a more opportune time. Originally designed to enable interplanetary communications, a DTN functions by grouping messages into a series of bundles intended for the same destination and routing each bundle as a single unit rather than as independent packets. The store-and-forward method enables a message to be delivered even when no immediate path to the destination exists. A device that is carrying a message will store it in the buffer until a path becomes available. The routing algorithm works by passing the message through nodes capable of forwarding the message to the destination. The question of which node is more capable of this forwarding is the main difference between the various routing algorithms.

As a rule, routing algorithms are more effective when they rely on information regarding the contact patterns of nodes. Conventional probability-based routing schemes assume consistent avenues of communication; nodes that interacted with a set group of nodes in the past will do so again in the future [5]. Similarly, social routing assumes that nodes assigned to the same social network (classroom, project team, department faculty, etc.) will regularly interact with members of that social group [6]. To take advantage of the partial applicability of both social and probability-based routing, our grouping method is designed form social groups based on contact patterns. Once these groups are identi-

fied, consistent routes to nodes are recognized based on the delivery history of a group or node.

The purpose of a routing algorithm is to deliver messages reliably and efficiently, which is difficult in networks in which nodes are capable of movement, and contact between each node is not always reliable or consistent. Both the Dynamic Social Grouping (DSG) and DSG - Node to Node (DSG-N^2) begin by identifying social groups based solely on contact patterns between nodes without relying on full knowledge of the network. Groups are formed based on the assumption that frequent and regular contact occurs within a group; there is no context regarding the group's purpose, nor is there pre-identification of groups. Once the groups are identified, routing occurs based on the type of message sent. If messages are sent in a node-to-base-station pattern, such as for wildlife sensor collection or data caching, the DSG algorithm routes messages through consistent nodes and groups. If messages are sent to individual nodes, such as for conventional message passing, the DSG-N^2 is used. Simulations with real-world data comparing Epidemic and Probabilistic algorithms were performed, showing that the Dynamic Social Grouping algorithm is superior in terms of its delivery ratio and message costs (see Section 4 for details). Similarly, DSG-N^2 was then compared to Epidemic routing [7] and SimBet routing [8], the results of which comparisons are shown in Section 5. These results show that DSG-N^2 performs better than both routing algorithms in terms of total message traffic and delivery time.

2. BACKGROUND

Delay-Tolerant Networks (DTNs) have received considerable attention from researchers in recent years due to their widespread occurrence in a variety of applications. Routing and data aggregation are of particular interest as they considerably affect the longevity of the sensor nodes and the performance of the network as a whole. A survey of routing techniques for DTNs is presented in [9], and a brief comparison can be found in Table 2.1.

2.1. EPIDEMIC ROUTING

Epidemic routing [7] is designed to route messages through partially connected networks in a semi-probabilistic fashion. Routing is accomplished by copying and forwarding packets to other parts of the network in order to reach more sparse sections. When two nodes meet, any unshared messages are transmitted, and the message's hop count is incremented. A message is passed along until its hop count has reached the max. Through this scheme, Epidemic routing is able to maximize the amount of messages successfully delivered but without taking into account delivery time. This creates a large message overhead, redundantly transmitting messages throughout the network.

2.2. MAXPROP

This variation on a flood based protocol was developed by Burgess et al [10] to prioritize the transmission of messages. By adding an intelligent priority to messages, they can be spread throughout the network such that messages are transmitted more often to their destination, and dropped from the buffer when the node either will not likely deliver the messages or another node is more likely. The priority is based on the individual node's chance of delivery (as based on past performance), other node's chance of delivery, and

how far the message has already spread. This algorithm has higher performance than a basic implementation of epidemic, although there is significantly greater overhead from a node gathering data on the network layout.

2.3. PROBABILISTIC ROUTING

This scheme is based on individual nodes tracking their probability of delivering a message successfully to a base station. The version introduced by Yu Wang functions by individual nodes tracking their performance based on individual messages [5]. Each node is assigned a probability estimate of successfully delivering a message to the base-station node; this estimate increases with successful delivery to nodes with higher probabilities decreases as messages expire. The probability is therefore based on the contacts of nodes, and as such works well when nodes follow regular, if not necessarily social, contact patterns. The algorithm is therefore an effective routing system, but it does not take advantage of past performance data.

2.4. PROPHET

A similar routing scheme currently implemented is Probabilistic ROuting Protocol using History of Encounters and Transitivity [11], introduced initially to allow efficient communications in a sparse environment. Similar in many ways to the Probabilistic method in Section 2.3, this method allows individual nodes to track their contact patterns to other nodes. Unlike the previous method, it is based not on message performance, but on the contacts themselves; the nodes will adjust their probability when they meet, whether or not any messages are transmitted. This method, while also an effective algorithm in a sparse network, limits the data it applies to routing decisions.

2.5. BUBBLE RAP

The Bubble-Rap grouping method [12] allocates nodes into social groups based on direct and indirect contacts. These contacts are distributed using a method called $k-cliques$, in which all fully-connected groups of k members are considered a distinct social group and are then merged with all other $k-cliques$ that can be reached through $k-1$ nodes. This provides an accurate representation of the social groups formed by a set of nodes. However, it relies on global knowledge of the nodes' contacts, which must be known before the algorithm can create groups. In this sense, k-grouping is neither dynamic nor distributed, which limits its applicability to a MANET.

2.6. SOCIALCAST

Costa et al. have described an application using social dynamics for a publish/subscribe schemata across a wireless sensor network, detailing a SocialCast model [13] in which nodes are assigned a 'Utility Value' for each interest in which they participate, which indicates their routing utility for that group. By separating the routing utility from group participation, the authors improve the lifespan and routing efficiency of the wireless nodes. The given algorithm is designed for use across a publish/subscribe model, as well as a wireless network (as opposed to a sensor network), but it can still serve as a basis for further research, especially regarding the social dynamics described, such as the Caveman Model.

2.7. SIMBET

The SimBet routing algorithm [8] is similar to DSG-N^2 in that it can deliver messages node to node using social patterns. Nodes engage in 'conversations' when they meet and exchange information about data messages and current neighbors. This information is used to determine 'betweenness' and 'similarity'. Betweenness is a measure of how often a node lies on a path between otherwise unconnected nodes. Similarity is a count of

common neighbors between two nodes. These values can accurately predict which node is more likely to successfully forward a message to the destination. Both betweenness and similarity are locally determined; global knowledge of the network is not needed. A notable disadvantage of SimBet is the number of messages the aforementioned conversations send across the network, resulting in significant message traffic and strain on the node's limited power supply. Also, relationships between nodes in SimBet are represented as either in contact or not in contact. More specific and informative data, such as the percentage of time in contact, would result in a better measure of a node's likelihood of forwarding a message to the destination.

2.8. SIMBETAGE

Link et al. have improved the SimBet routing protocol to handle more dynamic networks [14]. SimBetAge improves upon the baseline established by SimBet by adding a 'freshness' value, which dictates how often node A contacts node B and changes based on an encounter event or a time event. During an encounter, the freshness value will increase by a logistic growth function, and during a time step event, the freshness value will decrease in proportion to an exponential decay function. This algorithm considers temporal changes in the network, thereby allowing more reliable and direct routing to occur.

2.9. DISTRIBUTED CLUSTERING

A method based on identifying mobile clusters was researched by Dang et al [15]. This method uses exponentially weighted moving averages (ENWA) to track how similar the mobility patterns of two nodes are, under the assumption that two nodes moving in similar patterns are part of the same cluster. When the contact probability between two nodes exceeds a certain threshold, a new cluster is formed, while if a node's contact probability with all group members is high enough, it joins the cluster. Routing is based on a series of identified gateway nodes, which act as liaisons between two clusters. The routing

algorithm varies on the relationship between the nodes; two nodes of the same group will simply hold the message until direct contact is established, while neighbouring groups will route through a 'gateway' node, dedicated to transmitting messages to nearby groups. This algorithm is similar to DSG in many respects, with the main difference being a node's relation to the group. While routing in Distributed Clustering is based almost entirely on group membership, DSG integrates a node's independent routing ability with that of groups.

Table 2.1. Routing Algorithm Comparison

Routing Algorithm		Proactive / Reactive	Uses Social Data	Replication
Epidemic	Forward messages to all available nodes	Reactive	No	Yes
MaxProp	Message priority for epidemic transmission / dropping	Proactive	No	Yes
PRoPHET	Monitor node contact to measure delivery probability	Proactive	No	No
Probabilistic	Record message performance to measure delivery probability	Reactive	No	No
SimBet	Identify central & similar nodes	Proactive	Yes	No
SimBetAge	Centrality & similarity values decay	Proactive	Yes	No
Distributed Clustering	Identify clusters of nodes based on common mobility, route through clusters	Proactive	Yes	No
Dynamic Social Grouping	Identify social groups of nodes, route via group & individual probability	Proactive	Yes	Yes

3. PROPOSED ALGORITHM

Both DSG and DSG-N^2 address two distinct subproblems. First, given an arbitrary collection of nodes, the network must recognize cohesive social patterns and identify them as groups, while simultaneously distributing this information throughout the network. This network organization can adjust to reflect changes to the network or to the social patterns of the nodes. Second, once the groups have been recognized, a route must be identified to deliver messages to other nodes. The algorithm must minimize the number of message repetitions while ensuring a high delivery ratio.

3.1. GROUPING

A major advantage that this algorithm has over conventional mobile ad hoc routing methods is the use of social groups to improve communication throughput. The task of identifying such groups, however, requires knowledge of the nodes that is not present at startup.

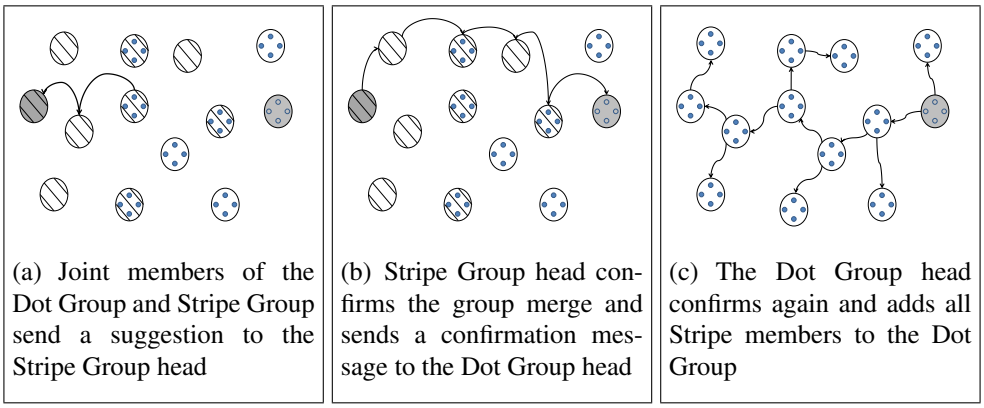
3.1.1. Contact Strength. The first task in identifying a social group is to calculate a metric for the contact frequency between two nodes. The symbol $\lambda_{i,j}$, used to represent the contact strength between two nodes, is measured as a function of the duration for which two nodes, i and j, were in contact and not in contact. The symbol α is used to determine the extent to which λ changes based on new data, mimicking a Markov process. Initially, $\lambda_{i,j}$ will be 0 between all nodes, but when nodes contact one another, it is recalculated as shown in Eqn 1.

$$\lambda_{A,B} = (1-\alpha)\lambda_{A,B} + \alpha \frac{time_{contact}}{time_{contact} + time_{nocontact}} \tag{1}$$

3.1.2. Forming New Groups. When $\lambda_{i,j}$ exceeds or is equal to a threshold ψ, the nodes identify themselves as members of the same group. Initially, all groups have two

members, and the head of the group is chosen arbitrarily. The group head has the power to confirm an invitation for a merge, merge two groups, and send out kill group messages (see section 3.1.4 for details). The group will also have a probability assigned to it based on a weighted probability average (see Section 3.2 for additional information).

3.1.3. Merging. Groups grow larger when two existing groups with shared members merge. The nodes regularly review their group listings for potential merges. When the number of common members shared by two groups is greater than the merge threshold τ, a suggestion message is sent to the head of a group, as shown in Figure 3.1(a). Once the group head receives the suggestion, it uses an up-to-date member list to determine if the two groups are fit for merging with the same criteria as the joint member. The group head will then send out an invitation message to the other group head epidemically, as seen in Figure 3.1(b). Once the other group head confirms the ratio, it measures the delay of the invitation message. If the delay is greater than a certain period, the merge is refused; otherwise, the group head sends out a kill message to remove the other group and adds that group's member list to its own. This process is detailed in Algorithm 13.



(a) Joint members of the Dot Group and Stripe Group send a suggestion to the Stripe Group head

(b) Stripe Group head confirms the group merge and sends a confirmation message to the Dot Group head

(c) The Dot Group head confirms again and adds all Stripe members to the Dot Group

Figure 3.1. Three stages of merging groups (figures provided by [16])

Algorithm 13 Merging Groups

Notation
$Group_Y$, $Group_Z$ - Any social groups in MANET
$Node_A$ - Member of both $Group_Y$ and $Group_Z$
$Node_B$ - Group head of $Group_Y$, $GroupHead_Y$
$Node_C$ - Group Head of $Group_Z$, $GroupHead_Z$
τ - Threshold for Merging a Group
MTT - Max Transit Time

Trigger - Periodically in $Node_A$
for all $Group_Y, Group_Z$ of which $Node_A$ is a member **do**
  if $\frac{|Group_Y \bigcap Group_Z|}{|Group_Y \bigcup Group_Z|} > \tau$ **then**
    Send SUGGEST Message to arbitrary Group ($Group_Y$ or $Group_Z$)
  end if
end for

Trigger - $Node_A$ contacts $Node_B$
In $Node_A$...
for all $Group_Y$ which contain both $Node_A$ and $Node_B$ **do**
  if $Node_A$ has Control Messages for $Group_Y$ **then**
    $Node_A$ sends Control Messages to $Node_B$
  end if
end for

Nodes pass Control Messages epidemically to all Group Members

Trigger - $Node_B$ receives SUGGEST message
if $Node_B$ is $GroupHead_Y$ **then**
  if $\frac{|Group_Y \bigcap Group_Z|}{|Group_Y \bigcup Group_Z|} > \tau$ and transit time $< MTT$ **then**
    Send CONFIRMATION Message to all members of $Group_Z$
  end if
end if

The Group Head confirms the suggestion, and sends a Confirmation to the other group

Trigger - $Node_C$ receives CONFIRMATION message
if $Node_C$ is $GroupHead_Z$ **then**
  if $\frac{|Group_Y \bigcap Group_Z|}{|Group_Y \bigcup Group_Z|} > \tau$ and transit time $< MTT$ **then**
    $Group_Z = Group_Z \bigcup Group_Y$
    $Group_Y = \varnothing$
    Send KILL message to $Group_Y$
  end if
end if

If both Group Heads approve, the arbitrary group is added to the other, and then removed.

3.1.4. Dynamic Grouping. Due to changing social patterns, nodes can become alienated from a group. Such alienation is discovered by a node periodically reviewing its group list and determining whether or not the contact frequency ratio meets the group's required value of ψ. If not, the node must be removed from that group by sending a resign message to the group head, as shown in Algorithm 14. To determine whether a node should resign, it periodically calculates its $Group\lambda_{A,Y}$, which is the average $\lambda_{A,B}$ between itself and all members of the group, and compares that value to ψ. If the average is less than ψ, the node will send a resignation message to the group head. In order to reduce group fragmentation, the resignation is not recognized until the group head receives the message and then propagates a group update through all members. Until the update has propagated through the group, the group data is considered fragmented.

Algorithm 14 Dynamic Groups

Notation
$Node_A$ - Member of $Group_Y$
$Node_B$ - Group Head of $Group_Y$, $GroupHead_Y$
$NodeList_Y$ - All nodes in $Group_Y$
$Group\lambda_{A,Y}$ - Average Contact Strength between $Node_A$ and $NodeList_Y$
ψ - Threshold for Forming a Group

Trigger - $Node_A$ checks it's groups
In $Node_A$
$Group\lambda_Y = Average(\lambda_{A,Y})$
if $Group\lambda_Y < \psi$ **then**
  Send RESIGN to $Node_B$
end if

Trigger - $Node_B$ receives RESIGN from $Node_A$
$NodeList_Y = NodeList_Y - Node_A$
Update all nodes in $Group_Y$ of removal and $Node_A$

3.1.5. Group Versions. An ongoing issue when using dynamic groups is transmitting group membership changes to all members of that group. Data fragmentation occurs

whenever two nodes, members of the same social group, have different data regarding that group's membership. To prevent this issue, all changes are controlled by the group's head. This node, arbitrarily chosen, holds the group's master member list.

Social patterns changing or being revealed will cause changes to a group's membership, as defined in Sections 3.1.3 and 3.1.4. The group head will increment the version value of the group as changes are made, including merges and resignations from the group. Whenever two members of the same group meet, they compare the version number of their local copy of the group. The higher version number is the one whose information regarding the group came from the group head more recently. This information is copied over to the less recent node, along with the version number. This process is detailed in Algorithm 15.

Algorithm 15 Group Updates

Notation
$Version_{A,Y}$ - Version Number of $Group_Y$ kept by $Node_A$

Trigger - $Node_A$ contacts $Node_B$
In $Node_A$
for all $Group_Y$ which contains both $Node_A$ and $Node_B$ **do**
  Send $Version_{A,Y}$ to $Node_B$
  Receive $Version_{B,Y}$ from $Node_B$
  if $Version_{A,Y} > Version_{B,Y}$ **then**
    Send $Group_Y$ to $Node_B$
  else if $Version_{A,Y} < Version_{B,Y}$ **then**
    Receive $Group_Y$ from $Node_A$
  else if $Version_{A,Y} = Version_{B,Y}$ **then**
    Do nothing
  end if
end for

3.2. ROUTING

Having organized the nodes into social groups, the algorithm can now use this information to route data to the destination, which is either the basestation or another node.

The probability σ is assigned to each node, depicting an estimate of its ability to transmit a message. This metric is changed to reflect changes in the message delivery or contact patterns, as described below.

3.2.1. Basestation Routing. Basestation routing is based on nodes gathering data and then transmitting that data to the nearest basestation. No other messages are transmitted in this architecture - nodes receive no feedback, nor do they send messages to each other. Nodes initially are assigned a default probability σ, while base stations are assigned a probability of 100%. When two nodes encounter one another, they compare probabilities to determine routing. The member with the lower probability will forward all its messages to the other before updating its σ to reflect the ability of the node to deliver either directly or indirectly to a base station. The result is that nodes with immediate access to a base station achieve a higher probability, and the probability cascades outwards through node contacts. The cascade rate is determined by a control variable α, which is similar to ϕ in that it controls how quickly the probability changes based on new data. For example, when $Node_A$ transmits a message to $Node_B$...

$$\sigma_A = (1-\phi)\sigma_A + \phi\sigma_B \tag{2}$$

To reflect a given node's inability to transmit messages, the algorithm also decays σ. When a message is transmitted to a node, it logs the reception time. If a node holds a message for a certain duration without transmitting it, its probability is reduced, again in ratio to ϕ.

$$\sigma_A = (1-\phi)\sigma_A \tag{3}$$

This can result in a message lowering its probability below that of neighboring nodes. For example, if $Node_A$ receives a message, it logs the time and compares its probability (55%) to that of its neighbors (40%). The neighbor nodes are less likely to deliver the message, so $Node_A$ holds the message, slowly lowering the probability as time passes. If $Node_A$ never encounters another node, eventually its probability will drop below that of the neighboring nodes, at which point the message will be transmitted. The result of this algorithm is that the probability will be reduced if a node is incapable of delivering a message in a timely manner.

3.2.2. Node-to-Node Routing. Although the algorithm for routing messages to nodes is larger than that used in base-station routing, requiring more space and more processing power, the two are not substantially different. The routing probability metric is a vector of numbers, one for each node. There are two possible methods by which to determine a node's ability to transmit a message to a destination node. The first, called Performance-Based Probability, is based on the assumption that nodes that have successfully delivered a message are more likely to do so again. When nodes transmit messages to the destination or to nodes with a higher probability, they increase the probability metric. The Contact-Based Probability method ignores the message history in favor of the node's contact patterns.

3.2.3. Performance-Based Probability. This algorithm (Algorithm 16) is similar to base-station routing in that a node's probability metric σ is updated to reflect its ability or inability to transmit a message. When a node successfully transmits a message to another node with a higher probability, the first node's probability of delivery is increased to reflect this.

In addition to message transmission, a node's individual probability can be updated to reflect its inability to deliver a message. Messages eventually time out in a network, which reduces a node's probability, thereby reflecting its inability to transfer a message to the destination node.

This algorithm is best used in a high-traffic environment because less message traffic can result in an inaccurate σ. It functions well when nodes are heterogenous because, by tracking the message performance, the algorithm does not assume that all nodes react identically to messages.

Algorithm 16 Performance Based Probability

Notation
$Node_A$, $Node_B$, $Node_C$ - Nodes in MANET
$\sigma_{A,B}$ - Probability of $Node_A$ to deliver a message to $Node_B$
$message_i$ - Message in MANET
$timeRcvd_i$ - Time $message_i$ was received at current node
TTL - Parameter determining duration of messages (time to live)
ϕ - Control Parameter determining Probability decay ratio

Trigger - $Node_A$ transmits $message_i$ to $Node_B$
if $Node_B$ is destination of $message_i$ **then**
  $\sigma_{A,C} = (1-\phi)\sigma_{A,C} + \phi$
else
  $\sigma_{A,C} = (1-\phi)\sigma_{A,C} + \phi\sigma_{B,C}$
  $timeRcvd_i = time_{current}$
end if

Trigger - Periodical maintenance in $Node_A$
for all $message_i$ in $Node_A$ **do**
  if $time_{current} - timeRcvd_i > TTL$ **then**
    $\sigma_{A,C} = (1-\phi)\sigma_{A,C}$
  end if
end for

3.2.4. Contact-Based Probability. This method is more useful in an environment with lower message traffic or uniform message failure probabilities. Using this scheme, the probability is based on contact with other nodes, rather than message transmission (Algorithm 17). Upon contact, the probability increases (the exact increase is a function of ϕ and the time the nodes spend in contact with each other). This is shown in equation 4.

$$\sigma_{A,B} = \sigma_{A,B} + (1 - \sigma)\phi \tag{4}$$

In addition, nodes increase their probability when they encounter other nodes capable of transmitting messages to a destination. This method, called either transitive probability or indirect contact, occurs when two nodes encounter each other and then. After adjusting their probability to deliver to each other, they adjust the probabilities of all other nodes. The control variable β is used to determine the impact of transitive probability, as seen in equation 5.

$$\sigma_{A,B} = \sigma_{A,B} + \beta(1 - \sigma_{A,B})\sigma_{A,C}\sigma_{C,B} \tag{5}$$

Finally, to ensure that the probability decays for two nodes not in contact for extended durations, a decay is implemented. Periodically, the node adjusts its probabilities based on the decay variable γ. A higher γ indicates that the environment should remain relatively stable for extended durations.

$$\sigma_{A,B} = \sigma_{A,B}\gamma \tag{6}$$

Algorithm 17 Contact Based Probability

Notation

$Node_A$, $Node_B$, $Node_C$ - Nodes in MANET

$\sigma_{A,B}$ - Probability of $Node_A$ to deliver a message to $Node_B$

ϕ - Control Parameter determining direct probability impact

β - Control Parameter determining transitive probability impact

γ - Control Parameter determining Probability decay ratio

Trigger - $Node_A$ comes into contact with $Node_B$

$\sigma_{A,B} = \sigma_{A,B} + (1 - \sigma)\phi$

for all $Node_C$ in environment **do**

$\sigma_{A,C} = \sigma_{A,C} + \beta(1 - \sigma_{A,C})\sigma_{A,B}\sigma_{B,C}$

end for

Trigger - Periodical maintenance in $Node_A$

for all $Node_C$ in environment **do**

$\sigma_{A,C} = \sigma_{A,C}\gamma$

end for

3.2.5. Group Probability. The group probability, depicted as ρ_Y, is a given node's estimate of a group's ability to deliver a message. This is the estimated average probability of delivery, based on the node members encountered frequently by the node. This method begins when $Node_A$ contacts $Node_B$ and both are in $Group_Y$. As the σ for all other nodes in $Group_Y$ are unknown, they are assumed to be the current ρ_Y. This method then calculates the average probability of all members of the group based on previous ρ_Y, σ_A, and σ_B, as detailed in Algorithm 18.

Each node will encounter different nodes in a group, so it is likely that different group members will have different estimates of that group's probability. Because group membership is kept current with dynamic grouping, a group's ρ will still be a reasonable estimate of the group's actual probability of delivering, with added emphasis on the members of the group that a given node is likely to encounter.

Algorithm 18 Calculating Group Probability

Definition
ρ_Y - Group Probability for $Group_Y$

Trigger - $Node_A$ contacts $Node_B$
$Node_A$ sends σ_A to $Node_B$
$Node_A$ receives σ_B from $Node_B$
for all $Group_Y$ which contain $Node_B$ and $Node_A$ **do**
$\rho_Y = \frac{\sigma_A+\sigma_B+\rho_Y\times(|Group_Y|-2)}{|Group_Y|}$
end for

3.2.6. Using Probabilities to Route. When two nodes contact each other, they independently determine their γ value. This is considered the cahnce of either the node itself or any of the groups it participates in successfully deliverying the message. The simplest method to calculate it is the inverse of the probability that all groups fail. The node with the higher value is assumed to either have more consistent contact with the base station or to be a member of a group that has consistent contact. In either event, the

node with the lower value transfers all messages to the node with the higher probability, and it then (if it is using Performance-Based Probability) updates its individual probability based on the successful delivery of a message. The basestation version of this algorithm is detailed in Algorithm 19. To expand on it for node-to-node routing, it is repeated for all destination nodes in either buffer.

Algorithm 19 Routing Algorithm

Notation
ρ_{Node_A} - List of all Group Probabilities in $Node_A$
γ_A - Composite Probability of $Node_A$

Trigger - $Node_A$ contacts $Node_B$
In $Node_A$
$\gamma_A = 1 - (1 - \sigma_A) \times \prod(1 - \rho_{Node_A})$
Transmit γ_A to $Node_B$
Receive γ_B from $Node_B$
if $\gamma_B > \gamma_A$ **then**
  Transmit messages to $Node_B$
  $\sigma_A = (1 - \alpha)\sigma_a + \alpha\gamma_B$
else
  Receive messages from $Node_B$
end if

3.3. HYBRID DSG-N^2 / EPIDEMIC ROUTING

As an alternative to a composite probability based on the groups in which a node participates, the routing algorithm can route through all members of a group containing the destination node. When the message initially is sent, it is routed 'near' the destination node by a probabilistic method. Upon finding a member of the same social group as the destination, the message is routed epidemically to all members of that group, quickly arriving at the destination. This algorithm is designed to achieve the high delivery ratio and low delivery time of epidemic by ensuring that all nodes that come into contact with the

destination have a copy for delivery, while reducing the number of redundant copies by only transmitting a single copy around nodes not in contact.

This algorithm (Alg 20) functions best when the identified social groups have a high correlation with the contact patterns. If the groups are too large, then the number of redundant messages increases the transmission costs, overloads buffers, and causes transmission collisions. If the groups are too small, then the algorithm functions the same as PRoPHET, ignoring any available social data.

Algorithm 20 Hybrid Algorithm

Notation
$BuddyList_A$ - List of nodes that are in same group as $Node_A$
$Group_X$ - Group containing $Node_A$
$\sigma_{A,B}$ - $Node_A$ probability to deliver to $Node_B$
$\hat{\sigma_A}$ - Vector of $Node_A$ probability to all other nodes
$dest_i$ - Destination node of msg_i

Trigger - $Node_A$ contacts $Node_B$
In $Node_A$
$BuddyList_A = \emptyset$
for all $Group_X \in Node_A$ **do**
  $BuddyList_A$ = $BuddyList_A \cup$ Members of $Group_X$
end for
Send $BuddyList_A$, $\hat{\sigma_A}$ to $Node_B$
Receive $BuddyList_B$, σ_B
for all msg_i in $Node_A$ **do**
  if $dest_i \in BuddyList_B$ **then**
    Transmit msg_i to $Node_B$
  else if $\sigma_{A,dest_i} < \sigma_{B,dest_i}$ **then**
    Transmit msg_i to $Node_B$
  end if
end for

4. ANALYSIS - DSG

4.1. EXPERIMENTAL SETUP AND EVALUATION

To evaluate this algorithm and the impact of control variables, a simulation was designed and implemented using MATLAB. Once the routing efficiency was determined, a power consumption comparison was performed in NesC using a TOSSIM simulator modified with PowerTossim Z. For comparison, both simulations also ran Epidemic Routing and base Probabilistic Routing schema. Epidemic Routing is considered the ideal in terms of the message transmission time and delivery ratio, and is therefore used for comparison, while Probabilistic Routing represents an improvement over Epidemic Routing in terms of the resources used. Other comparable algorithms were reviewed, but they either use previous knowledge of the environment (such as Bubble Rap) or are targeted to solve different communication issues (Simbet performing node-to-node communication, and SocialCast performing publish/subscribe broadcasts). For these reasons, the Probabilistic and Epidemic schemas were chosen as comparable algorithms. The control variables tested, along with ideal values identified, can be found in Table 4.1.

4.2. SIMULATION DATA SOURCE

To implement this algorithm accurately, the node contacts must follow social patterns. Potential sources of this information include real-life tracking data or the results from a social prediction algorithm, such as the Caveman Model. For this reason, the simulation utilized data obtained from an experiment conducted by the University of Cambridge at the 2005 Grand Hyatt Miami IEEE Infocom conference. Participants were asked to carry iMotes with them during the conference for 3 to 4 days to capture social interaction data [17]. Although the experiment did not include base stations capable of

Table 4.1. Variable Reference Chart

Control Variables		Range Tested	Ideal Value
α	Contact Decay Ratio	0.1 - 0.7	0.4
ϕ	Contact Impact on Probability	0.1 - 0.7	0.6
β	Transitive Impact on Probability (CBP)	0 - .5	.015
γ	Probability Decay (CBP)	.95 - .999	.998
ψ	Group Formation Threshold	0.1 - 0.7	0.3
τ	Group Merge Threshold	0.1 - 0.33	0.3
Ideal values tested by experimentation			
Variables			
λ	Contact Strength between nodes		
σ	Node's Probability of Delivery		
β	Group's Probability of Delivery		
γ	Composite Probability		

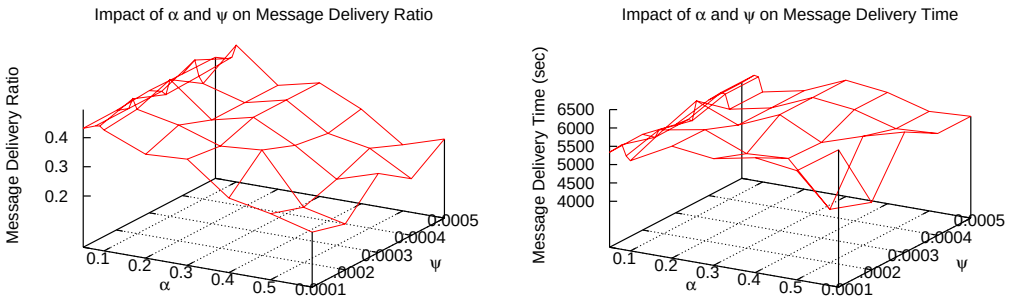
collecting information, they did include static, immobile nodes. For simulation purposes, these units were treated as data sinks.

4.3. IMPACT OF α AND ψ

Based on the utilized dataset, the α setting is directly influenced by the dynamic nature of the contact patterns. In both long-range simulations and simulations in which nodes consistently follow similar contact patterns, a lower α value can result in more effective simulations. In contrast, a higher α results in the algorithm placing more weight on recent data. If the probability is more dynamic, it can more rapidly respond to changes in the layout.

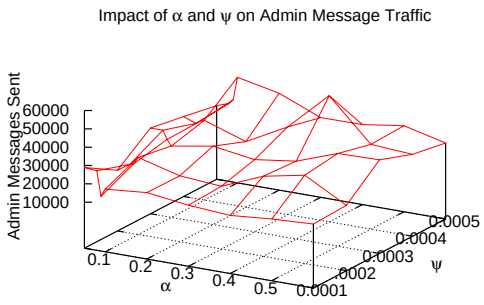
The ψ setting influences the ease with which the nodes form social groups. In an environment in which nodes only encounter other nodes in the same social group, a lower value can result in the network being rapidly organized, whereas a higher ψ results in groups forming slowly, and only with regular contacts.

The results (as shown in Figure 4.1) indicate a peak Message Delivery Ratio as α decreases and ψ increases. The lower α seems to be due to the large amount of 'garbage' data - a node that successfully delivers to the base station is quite likely just passing through and will not likely serve as a good long-range contact. A lower α allows the algorithm to ignore these incidental contacts and concentrate on nodes that deliver successfully with regularity. Similarly, the ψ peak value is based on ignoring the large number of casual contacts. Because this dataset was gathered at a conference, there are several regular contacts between nodes that are not members of the same group. To ignore this noise, a higher ψ setting is optimal.



(a) Message Delivery Ratio

(b) Message Delivery Time



(c) Admin Message Traffic

Figure 4.1. Impact of α and ψ

4.4. IMPACT OF τ AND ϕ

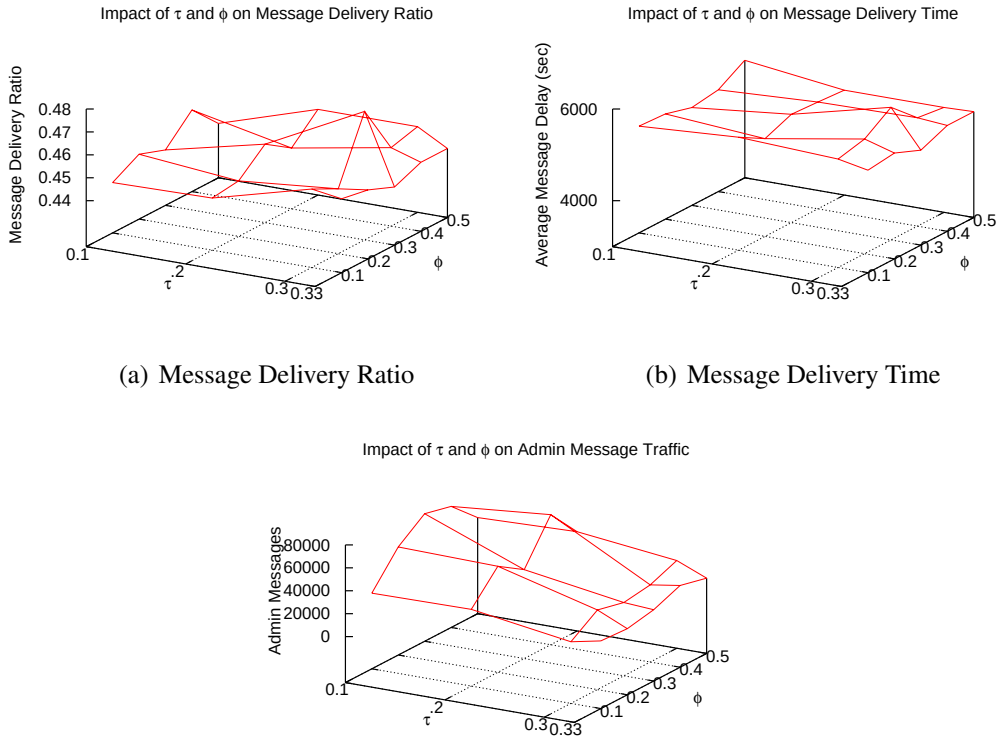
The control variable τ represents the level of similarity necessary for groups to merge. A higher τ value results in fewer groups merging and maintains a small group size, but it does not influence the number of groups being created. A higher τ value will result in several smaller groups. Due to the ratio needed to ensure that any group merges occur, the maximum value for τ is $\frac{1}{3}$. If it is higher, the initial 2-member groups will not have enough common members to merge.

The contact deterioration ϕ determines how quickly the contact strength changes over time. A higher ϕ value results in the contact strength changing more rapidly. Higher values would be used in more dynamic environments, so the nodes' contact strengths are influenced more by recent events than historical data. This value is closely linked to the ψ value given earlier; a more rapidly-changing contact strength results in a lower required value of ψ to successfully form groups.

The results, as shown in Figure 4.2, indicate that the algorithm functions best when the τ value is nearly, but not quite, at its maximum. A τ of 0.3 allows groups to merge regularly, reflecting larger groups while still ignoring the casual group affiliations. Similarly, the ϕ results show peak functionality at 0.3, although this may be due to the ψ value already inserted. These metrics show the largest increase in the delivery ratio, but with a corresponding increase in the average message delivery time (Figure 4.2(b)). These observations are due to this time reflecting only delivered messages; by increasing the number of delivered messages, the system includes several messages that were ignored.

4.5. COMPARISON OF ROUTING ALGORITHMS

A review of the results (Figure 4.3) shows that the DSG Routing Scheme performs better than the Probabilistic Scheme in all metrics and is comparable to Epidemic Routing in terms of both the delivery ratio and delivery time while having considerably less message traffic. Some of this is due to the applicability of the dataset – the social environment



(a) Message Delivery Ratio

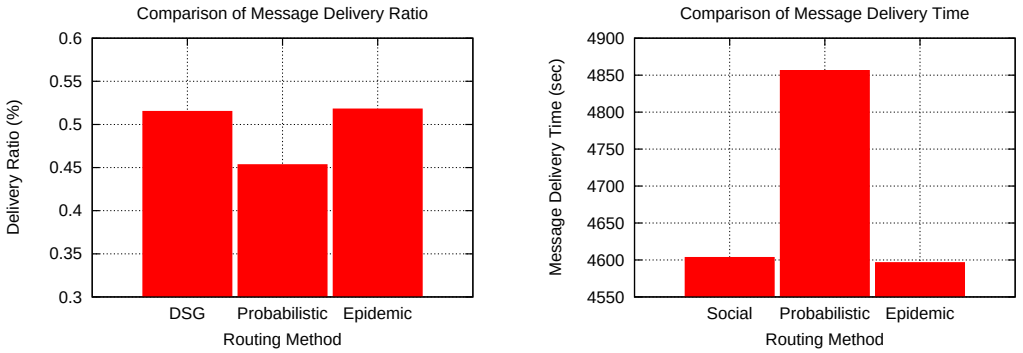
(b) Message Delivery Time

(c) Admin Message Traffic

Figure 4.2. Impact of τ and ϕ

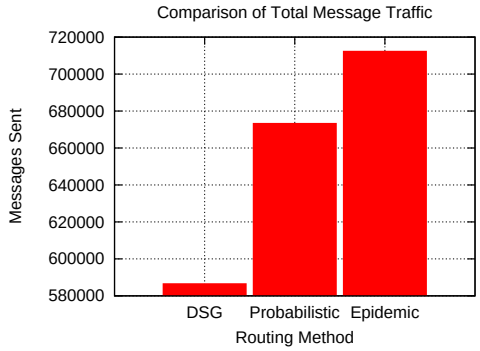
provided at the conference was ideal for forming short-term dynamic groups. In contrast, the Probabilistic Scheme had difficulty adjusting to the social groups that formed and took too long to cascade the probabilities that would result from successful delivery. The results indicate that DSG is comparable to the ideal routing method, and with considerably less cost.

A cost comparison of the algorithms was performed in the TOSSIM simulation using the PowerTOSSIMZ [18] model, as shown in Figure 4.4. The baseline shown in the graph indicates the simulation's power consumption when no routing occurs and serves as a comparison point. We use this to find that the Dynamic Social Grouping's cost is 16% of Epidemic's cost and 18% of the base Probabilistic algorithm's cost. This includes



(a) Message Delivery Ratio

(b) Message Delivery Time



(c) Total Message Traffic

Figure 4.3. Comparison of DSG, Probabilistic, and Epidemic for Base-Station Routing

the overhead from group management, such as group merge invitations, suggestions, and membership updates, but not standard maintenance, such as neighbour discovery or message generation.

When considering a routing algorithm for large-scale deployment, there is always concern regarding how well the algorithm scales to a higher number of nodes. For this reason, scalability experiments were performed to investigate how well the DSG algorithm performs with different numbers of nodes. Because the experiment was performed using a real-world dataset, it is impossible to add additional nodes to the experiment, but relatively simple to randomly remove nodes from consideration. The results show that the ratios from

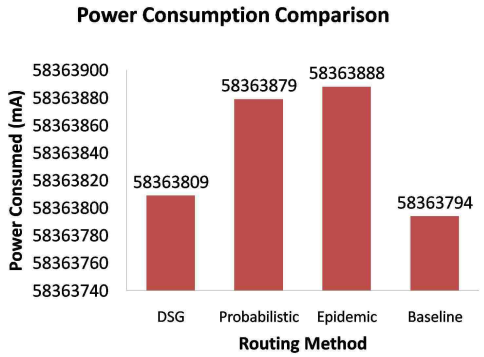
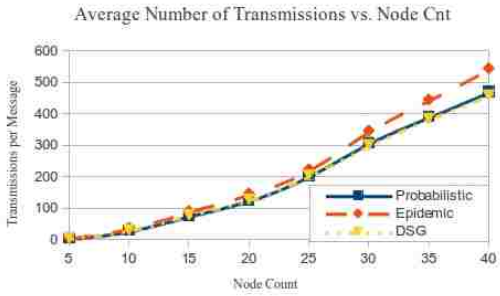
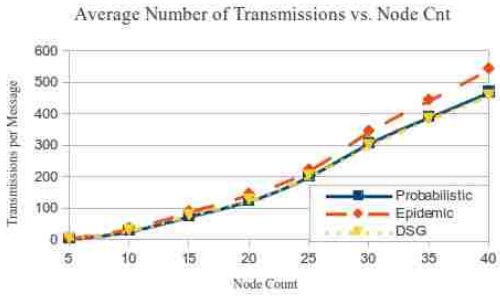


Figure 4.4. Comparison of DSG, Probabilistic, and Epidemic Power Consumption

the three algorithms tend to increase as the number of nodes increases, improving connectivity, but the DSG algorithm does not show a marked improvement until the full 40 nodes are in use (Figure 4.5(a)). This is because the same control variables are used throughout the entire experiment set, from 5 to 40 nodes. As indicated earlier, the algorithm is sensitive to the control variables used during execution (detailed in Table 4.1); using these variables despite the different environments causes performance to fluctuate wildly. Nonetheless, the cost of this system remains consistently lower than the alternatives (Figure 4.5(b)) because of the nature of the administration traffic. As the number of nodes decreases, reducing node connectivity, the administration overhead also decreases. Fewer groups are being formed, merged, or updated. As the connectivity increases, the routing performance improves by enough to offset the overhead.



(a) Comparison of Delivery Probability versus Node Count



(b) Comparison of Total Message Traffic versus Node Count

Figure 4.5. Scalability Experiments

5. ANALYSIS - DSG-N^2

5.1. EXPERIMENTAL SETUP AND EVALUATION

DSG-N^2 was first implemented in MATLAB to evaluate the control parameters and as a proof of concept. After optimal values were determined, the algorithm was implemented using TOSSIM, and a virtual hardware simulation was performed. Data were output from the simulation depicting the time at which messages were generated and received, the packet size of messages sent at each node to determine power consumption, the average delivery time and the delivery ratio. The SimBet and Epidemic Routing Algorithms also were implemented in the hardware simulation, and the results were compared to DSG-N^2. Other routing algorithms were reviewed, but they either use previous knowledge of the environment (Bubble Rap [12]) or solve different communication issues (Probabilistic [5], which performs node-to-base-station communications). For these reasons, the SimBet and Epidemic schemes were chosen as comparable algorithms.

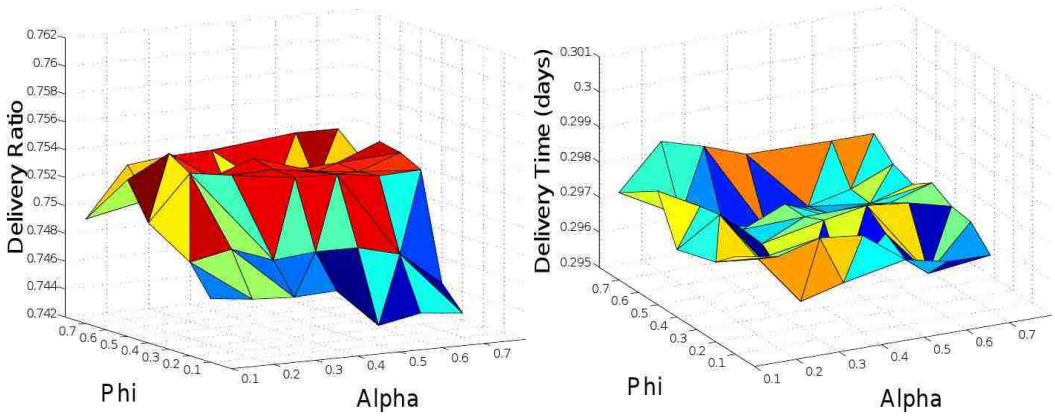
5.2. SIMULATION DATA SOURCE

In order to test our algorithm in a simulated environment, a record of social interactions was needed. The data used for these simulations originated from the MIT Reality Dataset [19]. The study tracked 100 nodes (people) with cell phones and Bluetooth devices over the course of 9 months. A set of Symbian-based phones was programmed to measure contact data and messages, and participants were asked to carry and use them. Multiple social patterns developed between participants, which resulted in an ideal dataset for testing social algorithms.

5.3. IMPACT OF α, ϕ, AND ψ

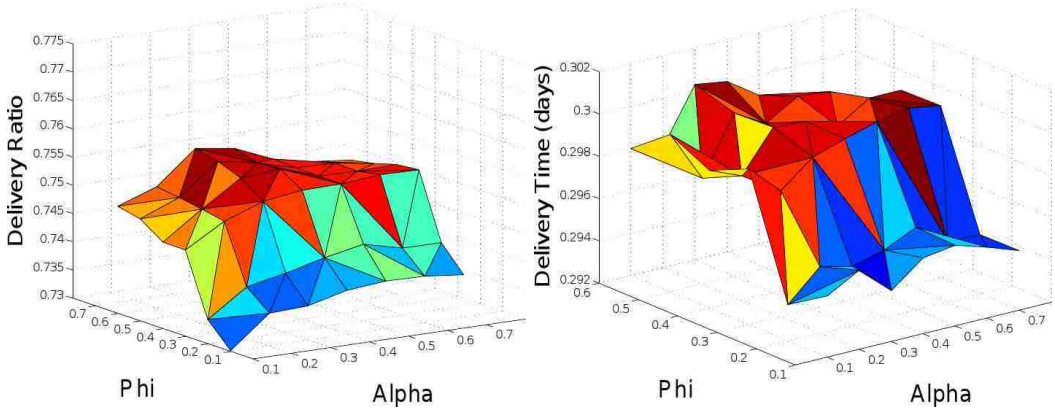
Based on the simulations, when ψ is low (Figure 5.1), the value of α will have no effect on the delivery ratios because any node-to-node contact will result in a new group. These groups do not accurately reflect social groups formed by nodes, thus lowering the delivery ratio. As ψ increases (Figure 5.2), α will begin to behave differently and make the groups more relevant. Based on current results, the best ϕ values seem to be midrange because such values appear to hold probabilities between nodes at a relevant level. Conversely, it appears as if upper and lower bound values of ϕ hold onto or ignore the probability history, respectively.

Regarding delivery time, a low ψ will yield evenly-distributed results. This is due to groups forming easily and nodes transmitting messages liberally rather than finding efficient delivery routes. When ψ is increased and ϕ is low, meaning that the probability history is ignored, the delivery time will be lower at any α. The delivery ratio suffers due to the contact history being ignored, but the average transmission time is better.



(a) Message Delivery Ratio (b) Message Delivery Time

Figure 5.1. Impact of α and ϕ with $\psi = .1$

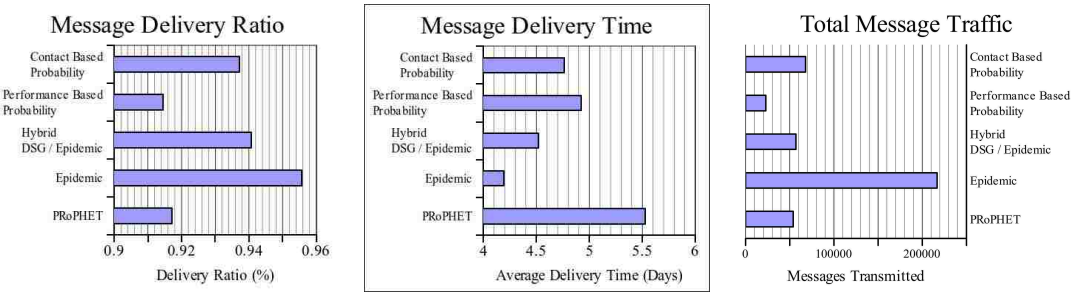


(a) Message Delivery Ratio (b) Message Delivery Time

Figure 5.2. Impact of α and ϕ with $\psi = .2$

5.4. COMPARISON SIMULATIONS

With ideal control parameters tested, a simulation to compare the routing algorithms was implemented. Epidemic and PRoPHET were implemented as a baseline comparison and compared to DSG-N^2 using both the Performance-Based Probability and the Contact-Based Probability. In addition, the Hybrid Algorithm was tested. The results, as presented in Figure 5.3, show that the Hybrid Method generally has the best tradeoff. While Epidemic's delivery ratio of 95.5% is slightly higher than Hybrid's 94.1%, it achieves this ratio at the expense of 4 times as many message transmissions. In contrast, the PBP version of DSG-N^2 is by far the least burdensome to use, requiring only 22,797 transmissions over a simulated 9 months, as opposed to PRoPHET's 53,874. This performance benefit has a tradeoff of a lower delivery ratio. These results are based on a relatively lower message count - 6,400 messages constitute a poor basis of probability. In conclusion, the Hybrid transmits the most messages, while the Performance-Based Probability is the most energy efficient.



(a) Message Delivery Ratio (b) Message Delivery Time (c) Total Message Traffic

Figure 5.3. Comparison of DSG-N^2, Hybrid, Epidemic, and PRoPHET for Node-to-Node Routing

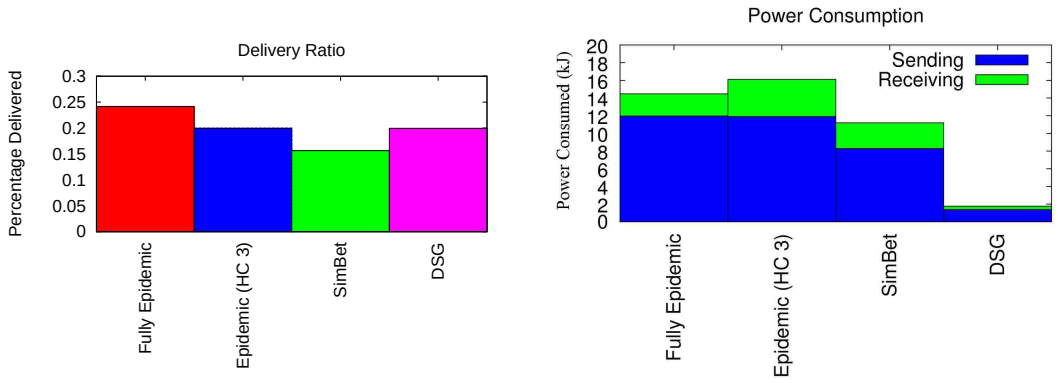
5.5. TINYOS SIMULATIONS

To validate the above simulations, the algorithms were simulated in a TOSSIM environment. This simulation tool allows individual nodes to track additional factors, such as buffer overflow, message collision, and power consumption. Although it is infeasible to implement for more complicated algorithms, it was used to validate the above results in a realistic environment.

When simulating the Epidemic routing protocol, two cases were tested based on the propagation distance. The baseline version limited the hop count to three, while the full execution had an unlimited hop count for which messages only ceased when they expired. SimBet was executed with default control parameters, and DSG-N^2 with control parameters of $\alpha = 0.4$, $\phi = 0.6$, and $\psi = 0.3$, based on optimal values found in MATLAB simulations. As seen in Figure 5.4, DSG-N^2 performed much better than both SimBet and Epidemic Routing. Epidemic underperformed in the TOSSIM environment due primarily to the realistic implementation of limiting factors such as buffer sizes, message collision, and limited message transmission time. Additionally, the Dynamic Social Grouping algorithm consumed far less power (Figure 5.4(b)). Similarly, SimBet did not perform as well in regards to both the delivery ratio and power consumption due primarily to the disconnected nature of the environment. SimBet would perform optimally in a static or strongly connected environment, which does not apply to the real-world dataset used.

5.6. COMPARISON TO ORACLE

The ideal routing algorithm will make its decision based on total knowledge of future events. Its ability to identify the shortest, most reliable route makes the Oracle routing algorithm a useful comparison tool for other algorithms [20]. Although impossible to implement in real life, it is possible to use Oracle in simulation to compare its ideal routes to another algorithm's estimates. During simulation, measurements were taken periodically to identify which nodes had the simplest, easiest path to a given destination. Similar measure-



(a) Message Delivery Ratio (b) Power

Figure 5.4. Comparison of DSG-N^2 Routing, SimBet, and Epidemic Routing

ments were taken during DSG-N^2 simulations to identify what the algorithm identified as the ideal path. For comparison, PRoPHET also was submitted to identify which nodes were more capable of delivering a message. The lists then were compared to identify how many 'swaps' would be necessary to make the probability metrics identical to the ideal case; a higher swap count indicates that the simulations did not reflect an accurate measurement of 'real' probability. The results, shown in Figure 5.5, show that the DSG-N^2 matches the ideal Oracle routing scheme, having a value equal to Oracle's ideal route in 86.23% of the cases, as opposed to PRoPHET's 64.98%.

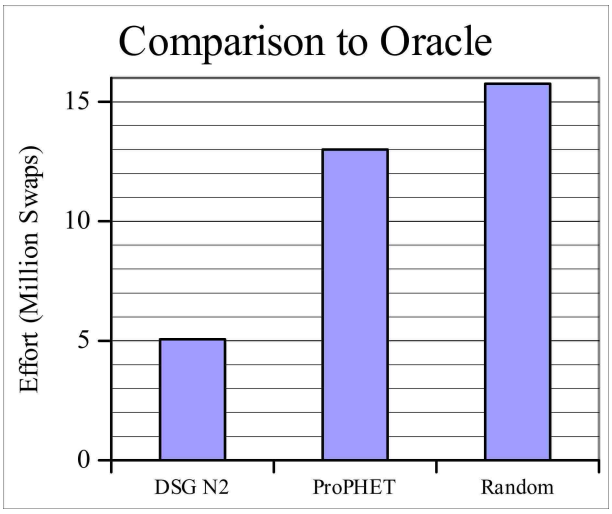


Figure 5.5. Comparison of DSG-N^2 and PRoPHET to Oracle

6. CONCLUSION

The Dynamic Social Grouping (DSG) algorithm has been shown to provide a significant increase in efficiency over Probabilistic routing and Epidemic routing, while Dynamic Social Grouping with Node-to-Node Transmissions (DSG-N^2) performs better than SimBet and Epidemic routing in an applicable social environment. While maintaining a lower overhead, the grouping methods described achieve as high a message delivery ratio and as low a delivery time as can be expected. This will lead to better data aggregation and longer battery life, both of which are primary goals in modern wireless sensor networks.

Further research remains to be conducted to consider the implications of alternative merge methods. The current method using a simple ratio depicting commonality may not be optimal, and experiments in alternative methods may provide further improvement. The current method may also be improved by introducing a load-balancing method to equalize the drain on commonly-used nodes, further improving the network's lifespan. Load-balancing techniques also can change the clusterhead to distribute the energy drain among the group. A final improvement to consider is the expansion of this algorithm to include point-to-point communication, allowing it to impact a wider range of applications.

7. ACKNOWLEDGEMENTS

We would like to thank James M. Bridges, Andrew Wilson, George Rush and Abbey Trotta for their hard work on implementing the simulations.

8. BIBLIOGRAPHY

[1] W. Chen, R. K. Guha, T. J. Kwon, J. Lee, and Y.-Y. Hsu, "A survey and challenges in routing and data dissemination in vehicular *ad hoc* networks," *Wireless Communications and Mobile Computing*, vol. 11, no. 7, pp. 787–795, 2011.

[2] C.-M. Cheng, P.-H. Hsiao, H. T. Kung, and D. Vlah, "Maximizing throughput of UAV-relaying networks with the load-carry-and-deliver paradigm," in *Proceedings of the 2007 IEEE Wireless Communications and Networking Conference (WCNC 2007)*, Kowloon, China, Mar. 2007, pp. 4417–4424. [Online]. Available: http://dx.doi.org/10.1109/WCNC.2007.805

[3] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. 5, pp. 96–107, Oct. 2002. [Online]. Available: http://doi.acm.org/10.1145/635508.605408

[4] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 145–158, Aug. 2004. [Online]. Available: http://doi.acm.org/10.1145/1030194.1015484

[5] Y. Wang and H. Wu, "Delay/fault-tolerant mobile sensor network (dft-msn): A new paradigm for pervasive information gathering," *IEEE Transactions on Mobile Computing*, vol. 6, no. 9, pp. 1021–1034, 2007.

[6] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of Predictability in Human Mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, Feb. 2010. [Online]. Available: http://dx.doi.org/10.1126/science.1177170

[7] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Duke University, Tech. Rep., 2000.

[8] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," New York, NY, USA, pp. 32–40, 2007.

[9] E. P. Jones and P. A. Ward, "Routing strategies for delay-tolerant networks," *Submitted to ACM Computer Communication Review (CCR)*, 2006.

[10] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in *INFOCOM*, 2006.

[11] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," in *SIGMOBILE Mobile Computing and Communication Review*,

2004, p. 2003.

[12] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: social-based forwarding in delay tolerant networks," pp. 241–250, 2008.

[13] P. Costa, C. Mascolo, M. Musolesi, and G. Picco, "Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks," *Selected Areas in Communications, IEEE Journal on*, vol. 26, no. 5, pp. 748–760, June 2008.

[14] J. A. Bitsch Link, N. Viol, A. Goliath, and K. Wehrle, "Simbetage: utilizing temporal changes in social networks for pocket switched networks," in *U-NET '09: Proceedings of the 1st ACM workshop on User-provided networking: challenges and opportunities*. New York, NY, USA: ACM, 2009, pp. 13–18.

[15] H. Dang and H. Wu, "Clustering and cluster-based routing protocol for delay-tolerant mobile networks," *IEEE Transactions on Wireless Communications*, vol. 9, no. 6, pp. 1874–1881, 2010.

[16] R. Cabaniss, S. Madria, G. Rush, A. Trotta, and S. S. Vulli, "Dynamic social grouping based routing in a mobile ad-hoc network," in *HiPC*, 2010, pp. 1–8.

[17] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD data set cambridge/haggle (v. 2009-05-29)," Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle, May 2009.

[18] E. Perla, A. O. Catháin, R. S. Carbajo, M. Huggard, and C. Mc Goldrick, "Powertossim z: realistic energy modelling for wireless sensor network environments," in *PM2HW2N '08: Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*. New York, NY, USA: ACM, 2008, pp. 35–42.

[19] N. Eagle, A. Pentland, and D. Lazer, "Inferring social network structure using mobile phone data," Downloaded from http://reality.media.mit.edu/download.php, pp. 15 274 – 15 278, 2009.

[20] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Single-copy routing in intermittently connected mobile networks," in *In IEEE SECON*, 2004, pp. 235–244.

IV. Three Point Encryption (3PE) - Secure Communications in Delay Tolerant Networks

Roy Cabaniss, Vimal Kumar, and Sanjay Madria

Department of Computer Science

Missouri University of Science & Technology, Rolla, Missouri 65401

Mobile Ad Hoc Networks (MANET) are a subset of Delay Tolerant Networks (DTNs) composed of several mobile devices. These dynamic environments make conventional security algorithms unreliable; nodes located far apart from each other may not have access (available) to each other's public keys or have doubt on validity of the public-key, making secure message exchange difficult. Furthermore, *ad hoc* networks are likely to be highly compromised and therefore may be untrusted. Other security methods, such as identity-based encryption and Kerberos, rely on requesting key data from a trusted third party, which can be unavailable or compromised in a DTN-like environment. The purpose of this paper is to introduce two security overlay networks capable of delivering messages securely, preventing both eavesdropping and alteration of messages. The first algorithm, Chaining, uses multiple midpoints to re-encrypt the message for the destination node. The second, Fragmenting, separates the message key into pieces that are routed and secured independently from each other. Both techniques improve security in hostile environments; under test conditions, Chaining reduces the number of messages intercepted by 90%, and Fragmenting by 83%. This improvement has a performance trade-off, however, reducing the delivery ratio by 63% in both algorithms.

1. INTRODUCTION

Secure communication is a base requirement for many wireless computing applications. Thus, any effective Delay Tolerant Networking (DTN) implementation must have a security system capable of routing messages without allowing an adversary to access or modify those messages. With the advent of public-key cryptography, private communication without any direct interaction is now feasible by providing the public key over unsecured networks. To communicate securely, the origin node must verify the key's association with the destination. In conventional communications, this problem is solved by using a trusted key repository to store and verify the keys used. However, this technique is infeasible in dynamic DTN in which the trusted party may be unreachable. This limitation forces nodes to handle their own key distribution and verification [1].

Methods are available that verify a node's identity, ranging from the very low tech (physical contact) to the more sophisticated (measuring the performance traits of the devices) [2]. However, these methods require point-to-point contact and therefore are difficult over any distance. When a node wants to send a message to another in direct contact or has already stored the destination's public key, the problem of secure transmission is trivial. However, in a DTN, validation of a public key [3] may not be feasible. Also, in DTN networks, connections can be short-lived and message delivery may get delayed. Thus, public keys may be inaccessible or untrusted. In such an environment, a node can encrypt a message with a key of a trusted node or multiple semi-trusted nodes and these nodes will be responsible for secure delivery of messages when connections are established. The intent of Three Point Encryption (3PE) is to send a message securely to a destination without having access to the destination's public key for reasons explained above. This task is performed by routing the message through the keys of other nodes and asking the midpoint to re-encrypt the message for the destination node.

This algorithm could possibly be applied to the task of routing messages securely through a series of allies (any of whom may or may not be compromised). If a member of the Red army wishes to communicate securely with another Red, he can do so using his public key, which he should already have. If he wants to communicate with a member of the Blue army, however, he may not have the key available or trust the public key. Rather than wait for direct contact with the destination to obtain the key, he can route the message through a series of members to ensure its security. Additional applications include a user in a battlefield trying to communicate when surrounded by untrusted nodes, and secure interactions between independently deployed wireless sensor fields.

The objective of this paper is to analyze two 3PE schemes: Chaining (Section 3.1) and Fragmenting (Section 3.2) in a DTN environment for secure delivery of messages. The chaining algorithm secures messages by sending them through a series of midpoints in the delay tolerant network; each midpoint must decrypt and re-encrypt the message before forwarding it to the final destination. Because the message is always encrypted with multiple layers, no midpoint will have access to the original message. The fragmenting algorithm functions similarly, sending the message through multiple midpoints in parallel. The message is broken into several fragments. Any node attempting to read the original message must have access to many (although not necessarily all) of the fragments. Simulation results (shown in Section 6) indicate that these schemes increase security significantly. Chaining reduces the number of compromised messages by as much as 90%, while fragmenting does so by 83%. However, there is a trade-off of reduced performance and reliability in terms of message delivery ratio.

2. BACKGROUND

A wide variety of advanced encryption methods are available to enable secure communications. Key distribution techniques give nodes access to public keys in the network. Our algorithms approach the problem of secure communications by expanding on these base methods.

2.1. COMMUTATIVE ENCRYPTION

Encryption algorithms, by definition, convert a coherent message into an unreadable ciphertext. Normally, when two encryption keys can be used on the same message, they must be encrypted and decrypted in a last-in-first-off order. Otherwise, the result is incomprehensible. Commutative encryption is a class of encryption algorithms that can be applied and removed in an arbitrary order, as shown in Eqn 1.

$$dec_A(enc_B(enc_A(msg))) = enc_B(msg) \tag{1}$$

This class of encryption techniques allows multiple encryptions to be applied to a message, thereby increasing the security of the system. Sample commutative encryption techniques include the Shamir algorithm [4], the Massey-Omura algorithm [5], and the El-Gamal re-encryption scheme[6].

2.2. THRESHOLD ENCRYPTION

Delay Tolerant Networks (DTNs) are designed to work in unreliable environments in which messages are corrupted or dropped. While this unreliability can be reduced by both message replication and resubmission, these solutions complicate security. Alternative methods can be used to allow a message to be decrypted if portions of the original key are lost. Using Shamir's (k,n) scheme [7], a message can be encrypted with n keys, requiring k

shares to decrypt (where k is always less than or equal to n). This technique allows complex security systems to be implemented in unreliable networks.

2.3. ONION ROUTING

Onion routing, developed by Syverson et al. [8], allows secure, anonymous communications in a static network. This method is based on establishing secure communications through a series of proxies, each of which only knows their incoming and outgoing proxy. By limiting data, traffic is forwarded without the destination being aware of the source of the messages, and without any of the nodes along the route being aware of the message's contents. This technique has been implemented in the Tor anonymity network, a series of computers that act as proxies for onion routing. The algorithm cannot be implemented in a MANET due to the unavailable or untrusted list of secure nodes; however, it serves as a demonstration of multiple midpoint encryption, which was expanded upon to design the chaining encryption technique.

2.4. ENPASSANT

Node mobility means that proxies may be unavailable or, at the very least, expensive to reach when implementing Tor in a DTN. The designers of the EnPassant [9] scheme expanded on the onion routing scheme with groups of proxies. Both the delivery time and ratio improve because any group members are allowed to act as a proxy, both decrypting and forwarding a data stream. Anonymity is preserved by forcing a message to follow indirect routes. This scheme is functional, but only under certain assumptions regarding the attacker. First, this scheme is very vulnerable to global eavesdropping adversaries; if all messages can be followed through the midpoints, randomizing the route has no benefit as the messages still can be tracked. This is a general weakness among DTN security schemes, however. A larger issue is that using group keys makes the scheme very vulnerable to Byzantine attacks [10]. A single compromised node jeopardizes the security of the entire

group. If a member of each group is compromised, all traffic can be tracked. Despite these vulnerabilities, this scheme prevents traffic analyses fairly well.

2.5. SOCIAL CONTACTS FOR MESSAGE CONFIDENTIALITY

El Defrawy et al. [11] proposed a method by which to securely communicate messages in a DTN. With this algorithm, a node lacking the destination's public key can encrypt the message with the public keys of several nodes near the destination, in terms of either physical proximity or contact frequency. While secure, this algorithm has the issue of having to maintain contact information for several nodes in the network. Another algorithm proposed for comparison, the Poor Man's Approach, fractures the key into several parts. Each fragment follows a different route to the destination. The random key then is split into several other sub-keys such that all fragments must be accessed to retrieve the original message. Using a bitwise XOR is the easiest method ($K = K_1 \oplus K_2 \oplus K_3 ...$). The encrypted message and all of the fragments are sent directly to the destination, but with a time lag so that each fragment will follow a different route. The purpose is to ensure that only the destination node receives all key fragments. If an adversary can monitor all transmissions (either by global eavesdropping or by being located on the sole path between the nodes), it can retrieve the key as easily as the destination can. This method, expanded on by the 3PE algorithm, is simple to implement but is only secure as long as the routing algorithm forwards each fragment independently. Even with these limitations, these approaches to keyless secure communication serve as a foundation for future algorithms.

3. PROPOSED ALGORITHM

The algorithms presented in this section are designed to function in large-scale DTNs. They assume that the message will be processed by untrusted nodes (hence the need for encryption at every step). Each node keeps a subset of the public keys available, referred to as the keychain, maintained by any number of key distribution techniques [13] [14] [15]. The following subsections describe the algorithms designed to use partial keychains to improve message confidentiality.

3.1. CHAINING ENCRYPTION

Chaining encryption forces the algorithm to route through k randomly selected nodes, known as links, without allowing any links to access the plaintext. The original message is first encrypted with the public key of each link and then routed to the nearest link. At each link, that node's encryption layer is removed. If the link has access to the final destination's public key, then the message is encrypted with it. Otherwise, the link encrypts the message for a node in its keychain, adding that node to the link chain. Only when each layer has been replaced with the final destination's key can the message be forwarded to the endpoint. Once there, it is decrypted k times, each time removing a layer of encryption from the message. Pseudocode for this method can be found in Algorithm 21.

This method is very secure, avoiding pitfalls inherent to the normal key-exchange sequence and fragmentation method. The only way it can be broken is if all of the links are compromised by an adversary. Another advantage is that it can be either scaled up for high-risk networks or down for more casual security by changing the number of links required. The trade-off is that this algorithm requires a message to travel to numerous midpoints, increasing the message delivery time considerably. Another trade-off is that a link compromised by the adversary can stop message delivery. The compromised link

either can refuse to forward a message or can reencrypt it with its own key. The latter can allow the adversary to read the message if all of the link's encryption layers are replaced with the adversary's. If the message uses a compromised node as a link, the message cannot be read by the adversary unless all links are compromised. However, it cannot be read by the destination either, resulting in a dropped message.

For example, Alice, a member of the Red army, sends a secure message to Bob, a member of the Blue army. If Alice possesses Bob's public key, she simply encrypts the message and forwards it. Failing that, she may attempt normal key distribution techniques, asking other members of the Red army (who she assumes are trusted) if they have the destination key. If they do not, she will select two midpoints, Chuck and David (Alice must possess the public key for both), and encrypt the message for both - $enc_{Chuck}(enc_{David}(Msg))$. Because she may not trust them, she layers the encryption to ensure that the message cannot be read by either party. The message is routed to Chuck first simply by virtue of his proximity. Chuck then removes his layer of encryption, leaving $enc_{David}(Msg)$. Chuck cannot read the message, so he obeys the protocol, re-encrypts it with Bob's key, and forwards it to David - $enc_{Bob}(enc_{David}(Msg))$. David cannot read the message either, so he follows the algorithm, removing his layer of encryption and forwarding it to Bob - $enc_{Bob}(enc_{Bob}(Msg))$. Once Bob receives the message, he has no problem removing both layers of encryption and retrieving the original message.

This example of execution changes if one of the midpoints is compromised. If Chuck had been compromised, he would not have encrypted the message for Bob. Even if Chuck cannot read the message, he can refuse to forward it, causing the message delivery to fail. If he wants to read the message, he can re-encrypt it with the adversary key and forward it to David - $enc_{Adv}(enc_{David}(Msg))$. David, in this case, does not possess Bob's public key. Thus, he randomly selects midpoint Eric, forwarding the message on - $enc_{Eric}(enc_{Adv}(Msg))$. If Eric is compromised and working with Chuck, he can remove his layer of encryption and Chuck's adversary key to retrieve the original message.

Only through their collaboration can the message be compromised. If Eric is not compromised, he removes his layer of encryption, encrypts the message for Bob, and forwards it - $enc_{Bob}(enc_{Adv}(Msg))$. In this case (when a portion of the chain was compromised), the final intended destination cannot read the message, but neither can the adversary.

The chaining method, therefore, has three possible outcomes. If all midpoints are uncompromised, the message is delivered successfully and securely. When all members are both compromised and collaborating, the message is compromised. If some are compromised and some not, or if they are not collaborating, the message delivery fails - a midpoint either refuses to forward the message or encrypts it with the wrong key. A detailed look at these relative probabilities is presented in Section 4.

3.2. FRAGMENTING ENCRYPTION

The trade-off for chaining encryption's increased security is its significantly increased delivery time. The fragmenting encryption method, rather than sending messages sequentially through links, will fragment the message into different pieces. Using threshold encryption, the message is encrypted into several subkeys. This allows the final destination to decrypt the message even if fragments are compromised by the adversary or dropped. Each fragment is encrypted and forwarded through a single link. Because each fragment is routed through a single midpoint, this technique takes less delivery time than chaining. Pseudocode for this method can be found in Algorithm 22.

One drawback of the fragmented method is that threshold encryption makes it less secure than chaining. Also, the adversary can read the original message if enough of the fragments are sent through compromised nodes. Additionally, because a copy of the message must be sent with each fragment, the system's energy costs are considerably higher. Section 4 offers detailed information regarding these trade-offs.

A performance weakness in the scheme is that the midpoint is selected randomly from the available keychain rather than from among those related to either the origin or desti-

Algorithm 21 Chained Encryption

Notation
k - The number of links through which a message must be routed
$Node_{Origin}$ - Origin of msg
msg_{dest} - Final destination of msg
$plaintext$ - Original message to be sent to msg_{dest}
$Node_A$ - Arbitrary node in MANET
$Keychain_A$ - Set of Public Keys to which $Node_A$ has access
$Enc_A(text)$ - Encrypted form of $text$ using public key of $Node_A$
$Dec_A(text)$ - Decrypted form of $text$ using private key $Node_A$

Trigger - $Node_{Origin}$ wants to send msg to $Node_{dest}$
$msg_{text} \leftarrow plaintext$
for $i = 1 \rightarrow k$ **do**
  $Node_i \leftarrow$ Random Node from $Keychain_{Origin}$
  $msg_{mid} \leftarrow msg_{mid} \cup Node_i$
  $msg_{text} \leftarrow Enc_{Node_i}(msg_{text})$
  /* Message is now encrypted commutatively */
end for
Route to nearest node in msg_{mid}

Trigger - $Node_A$ receives msg
if $Node_A = msg_{dest} \&\& msg_{mid} = \emptyset$ **then**
  for $i = 1 \rightarrow k$ **do**
    $msg_{text} \leftarrow Dec_{dest}(msg_{text})$
  end for
  Message has been delivered
else
  if $Node_A \in msg_{mid}$ **then**
    $msg_{mid} \leftarrow msg_{mid} - Node_A$
    $msg_{text} \leftarrow Dec_A(msg_{text})$
    $msg_{text} \leftarrow Dec_A(msg_{text})$
    /* Since msg was encrypted commutatively,
      order of node delivery / decryption is irrelevant */
    if $msg_{dest} \in Keychain_A$ **then**
      $msg_{text} \leftarrow Enc_{msg_{dest}}(msg_{text})$
    else
      $Node_i \leftarrow$ Random Node from $Keychain_A$
      $msg_{mid} \leftarrow msg_{mid} \cup Node_i$
      $msg_{text} \leftarrow Enc_{Node_i}(msg_{text})$
    end if
  end if
  Route to nearest node in msg_{mid}
end if

Algorithm 22 Fragment Encryption

Notation

$Node_{Origin}$ - Origin of msg
$plaintext$ - Original message to be sent to msg_{dest}
$Keychain_A$ - Set of Public Keys to which $Node_A$ has access
Nodes use $Threshold(k,n)$ algorithm
  $EncTH_{encKeyset}(text)$ - Encrypts $text$ using Threshold encryption.
  $DecTH_{decKeyset}(text)$ - Decrypts $text$ using Threshold encryption.
  $decKeyset$ must contain at least k keys of original $encKeyset$
  Key_i - Fragment i of a total of n fragments.
$msgFrag$ - Message Fragment. Contains...
  $msgFrag_{text}$ - encrypted original message
  $msgFrag_{key}$ - One key for encrypted message
  $msgFrag_{dest}$ - Final destination of message
  $msgFrag_{mid}$ - Midpoint of this fragment
$RcvdKeys$ - Keys received by $Node_{dest}$, initially $\emptyset$

Trigger - $Node_{Origin}$ wants to send message $plaintext$ to msg_{dest}
Generate random keys $\{Key_1, Key_2, ..., Key_n\}$
for $i = 1 \rightarrow n$ **do**
  $Node_j \leftarrow$ Random Node from $Keychain_{Origin}$
  Generate new message fragment $msgFrag$
  /* In message, include encrypted form of original $plaintext$ and Key_i */
  $msgFrag_{text} \leftarrow EncTH_{Key_{1..n}}(plainText)$
  $msgFrag_{key} \leftarrow Enc_j(Key_i)$
  $msgFrag_{dest} \leftarrow msg_{dest}$
  $msgFrag_{mid} \leftarrow j$
  Send $msgFrag$ to $Node_j$
end for

Trigger - $Node_A$ receives $msgFrag$, $A = msgFrag_{mid}$
if $msgFrag_{dest} \in KeyChain_A$ **then**
  $msgFrag_{key} \leftarrow Enc_{dest}(Dec_A(msg_{key}))$
  /* At this point, $Enc_{dest}(Dec_A(msg_{key})) = Enc_{dest}(Key_i)$ */
  $msgFrag_{mid} \leftarrow \emptyset$
  Send $msgFrag$ to $Node_{dest}$
else
  $msgFrag_{mid} \leftarrow$ Random Node from $Keychain_A$
  $msgFrag_{key} \leftarrow Enc_{mid}(Dec_A(msg_{key}))$
  /* At this point, $Enc_{mid}(Dec_A(msg_{key})) = Enc_{mid}(Key_i)$ */
  Send $msgFrag$ to $msgFrag_{mid}$
end if

Trigger - $Node_{dest}$ receives $msgFrag$, $msgFrag_{mid} = \emptyset$
$RcvdKeys \leftarrow RcvdKeys \cup Dec_{dest}(msgFrag_{key})$
if $|RcvdKeys| = k$ **then**
  /* $Node_{dest}$ has enough keys to decrypt the message */
  $plaintext \leftarrow DecTH_{RcvdKeys}(msgFrag_{text})$
  Message has been delivered
end if

nation of the message. While this selection technique reduces performance by potentially sending messages from one end of the network to the other, it is necessary for security. Any metric that would allow nodes to identify themselves as high-value midpoints would allow adversaries to falsely identify themselves, resulting in a large number of messages routing through compromised nodes. For this reason, midpoints are chosen randomly, rather than using any awareness of the environment.

Consider that Alice again wants to send a message securely to Bob. Lacking the public key, Alice encrypts the message using threshold encryption. Three keys are generated, two of which must be possessed to read the message - $enc_{k_1,k_2,k_3}(Msg)$. The encrypted message is sent to each of the three untrusted midpoints, Chuck, David, and Eric, along with a copy of a single key encrypted with that army's public key - $enc_{Chuck}(k_1), enc_{k_1,k_2,k_3}(Msg)$ is sent to Chuck, and so forth. Each midpoint, upon receiving the message, should decrypt the key, then encrypt it with Bob's public key, and finally forward the message to Bob - $enc_{Bob}(k_1), enc_{k_1,k_2,k_3}(Msg)$. If Chuck has been compromised, he can access a single key that is insufficient for reading the message. This demonstrates the trade-off between security and reliability; by forcing the message to require a larger number of keys in order to be read (such as needing three out of four created keys, for example) the algorithm is more secure. A larger number of midpoints must be compromised by the adversary before it is able to read the original message. This increases security at the cost of preventing the destination from reading the message until it receives more of the keys, thus limiting both its successful delivery ratio and its time to delivery.

4. TIME AND ENERGY ANALYSIS

The cost of both implementing and maintaining a security infrastructure is a critical consideration. Thus, this section contains an analysis of both the expected time required to deliver a message and the cost of said delivery for both chained and fragmented encryption. For comparison, an overview of the null security scheme and the key-request scheme (also referred to as the reflection scheme) also is provided.

Although the total source-to-destination cost of a message is based on the routing algorithm rather than the security system, the costs will still increase when a message must be re-encrypted and forwarded multiple times [1]. The exception to this is null security, which will only encrypt a message if it already has the key immediately available. Otherwise, the message will be sent in plaintext. The energy cost to transmit this message is simply the cost required to forward a given message based on the routing protocol - $E(J)$. Similarly, the time required to deliver a message, T, is based solely on the routing method used - $E(T)$. Both the distribution and expected values of J and T are undefined because they can change based on the protocol used. A list of the symbols used for comparison can be found in Table 4.1.

A major factor influencing the efficiency of a security schema is the probability that any given node will have the public key of any other node. Such techniques as caching and distribution can increase this probability but generally have their own security risks [16]. For the purpose of these calculations, the probability that a node will contain another node's key is assumed to be independent of neighboring nodes. Intelligent caching schemes, for instance, are implemented such that if a node does not have a key, nearby nodes are more likely to have them. Naive caching schemes tend to fill the local keyspace with the first keys available, which means that nearby nodes likely will not have the key. As the probability

[1] The costs of encrypting the message are negligible compared to the transmission costs. During experiments with Mica2 nodes, for example, encrypting a 1kB message required 12.96μJ. Transmitting the message required 1.5mJ.

of codependence is a function of the distribution and mobility schemes, for calculation purposes they are assumed to be independent.

Table 4.1. 3PE Variable Reference Chart

P_{key}	Probability that a node chosen at random has the Public Key for another node chosen at random
$E(T)$	Expected time for the routing algorithm to deliver a message from src to dest
$E_{req}(T)$	Expected time for Key Request scheme to securely deliver msg
$E_{chain}(T)$	Expected time for Chaining scheme to securely deliver msg
$E_{frag}(T)$	Expected time for Fragmenting scheme to securely deliver msg
$E(J)$	Expected energy cost for the routing algorithm to deliver a message from src to dest
$E_{req}(J)$	Expected energy cost for Key Request scheme to deliver msg
$E_{chain}(J)$	Expected energy cost for Chaining scheme to deliver msg
$E_{frag}(J)$	Expected energy cost for Fragmenting scheme to deliver msg

4.1. KEY REQUEST ANALYSIS

The key-request scheme begins by determining whether or not the node has the key for the destination in question. If it does, the algorithm simply sends the encrypted message. Otherwise, it sends a key request to the destination, along with the public key. Then, the destination node sends an encrypted, symmetric key back to the source, where the original message is encrypted and sent. The expected energy cost and required transmission time therefore are based on the probability that the source already has the destination's key, represented as P_{key}.

$$E_{req}(J) = P_{key} * E(J) + (1 - P_{key}) * 3 * E(J)$$

$$E_{req}(T) = P_{key} * E(T) + (1 - P_{key}) * 3 * E(T)$$

For comparison purposes, consider a large-scale environment in which nodes are capable of carrying 30% of the total number of public keys. In such a network, 30% of the messages will be delivered in a single origin-to-destination transmission. The other 70% will require three such transmissions. Messages thus have an expected delay and cost of 2.4 times that of a single transmission - $E_{req}(J) = .3 * E(J) + .7 * 3 * E(J) = 2.4E(J)$.

4.2. CHAINING ANALYSIS

Similar to the key-request scheme, the chaining method begins by determining whether or not the source has the destination's key. If it does not, it selects k midpoints, as described in Algorithm 21. This analysis is based on k being two nodes, although a system with better security will have a higher k. The expected hop count is based on how many nodes the message must visit before it is received by k nodes that have the destination key. The probability that the hop count is equal to the probability of the source having the key for the destination is $P(HC = 1) = P_{key}$. For the hop count to be 3, the first node will not have the key. Both midpoints will, however, and therefore they will not redirect the message at all. The probability of this is $P(HC = 3) = (1 - P_{key}) * P_{key} * P_{key}$. For the hop count to exceed three, either of the midpoints must be forced to redirect the message. The number of redirects is equal to the hop count minus 2, including the source's redirect to the two midpoints. A summary of these events can be found on Table 4.2.

The expected number of node-to-node messages can be derived when the probability of the various hop counts is known. The expected delivery cost and time can be calculated from the hop count.

Table 4.2. Chaining Algorithm Events

Event	Expected Delivery Time	Probability
Src Node has Key_{dest}	$E(T)$	P_{key}
Src does not have Key_{dest}, both Chosen Links have Key_{dest}	$3E(T)$	$\overline{P_{key}}P_{key}^2$
Src does not have Key_{dest} One link must redirect once	$4E(T)$	$\overline{P_{key}} * \overline{P_{key}} * P_{key} * P_{key} + \overline{P_{key}} * P_{key}\overline{P_{key}} * P_{key} = 2\overline{P_{key}}^2 * P_{key}^2$
Links must redirect j times	$(j+3)E(T)$	$\overline{P_{dest}}^{j+1}P_{dest}^2$

$$
\begin{aligned}
E_{chain}(T) &= E(T) \times \left(P_{key} + \sum_{i=3}^{\infty} i * (i-2) * (1-P_{key})^{i-2} * P_{key}^2 \right) \\
&= E(T) \times \left(P_{key} + \frac{2 - 5 * P_{key} + 3P_{key}^2 + P_{key}^3 - P_{key}^4}{(1-P_{key})^2 * P_{key}} \right) \\
E_{chain}(J) &= E(J) * \left(P_{key} + \frac{2 - 5 * P_{key} + 3P_{key}^2 + P_{key}^3 - P_{key}^4}{(1-P_{key})^2 * P_{key}} \right)
\end{aligned}
$$

Because the algorithm is based on a single message being forwarded through numerous midpoints, both the expected delivery time and energy cost are based directly on the hop count. For purposes of comparison, when an individual node can carry 30% of the public keys in the network, the average hop count is $5\frac{2}{3}$.

4.3. FRAGMENTED ANALYSIS

Fragmented encryption is the first algorithm discussed in which the delivery time and the energy consumed are not directly proportional. As in the chaining algorithm, the message routes through a set number of midpoints and continues routing until all fragments are received. For this reason, the energy cost is nearly identical to chaining; only the

number of fragments is different. The following equations assume that three fragments are sent to the destination, two of which are needed to decrypt the message. In order to send only the message through a single hop, the source node must have the destination key, so $P(HC = 1) = P_{key}$; otherwise, there will be at least six transmissions - the source will send the message to each of the three midpoints, and each of those three will send it to the destination if all three have the key - $P(HC = 6) = (1 - P_{key}) * P_{key}^3$. If a single midpoint must redirect, the hop count is 7, and the probability of all three midpoints redirecting is $P(HC = 7) = 3(1 - P_{key})^2 P_{key}^3$.

$$\begin{aligned} P(HC = 1) &= P_{key} \\ P(HC = 6) &= (1 - P_{key}) * P_{key}^3 \\ P(HC = 7) &= 3(1 - P_{key})^2 * P_{key}^3 \\ P(HC = i) &= \frac{(i-4)(i-5)}{2}(1 - P_{key})^{i-5} * P_{key}^3 \end{aligned}$$

This allows us to track the total number of transmissions, which in turn is used to calculate the total energy consumed per message.

$$\begin{aligned} E_{frag}(J) &= E(J) \times \\ &\quad \left(P_{key} + \sum_{i=6}^{\infty} i * \frac{(i-4)(i-5)}{2}(1 - P_{key})^{i-5} * P_{key}^3 \right) \\ &= E(J) \left(\frac{3}{P_{key}} - 2 * P_{key} \right) \end{aligned}$$

Using the previous 30% example, this method will require each message to be transmitted an expected 9.1 times before all fragments are delivered.

Because the fragmented encryption scheme sends each message independently of the others, the total delivery time is actually much shorter. Considering the example in which three fragments are sent, two of which are needed to decrypt the message, the delay will be the time the second fragment takes to reach the destination. Each fragment has a minimum of two hops - one to reach the midpoint, and another to be forwarded to the destination. If the fragment must be redirected to find the destination key, another hop is added. This means that the probability of two fragments reaching the destination in two hops is the probability of all three midpoints immediately having the key, or two of the midpoints having the key and the third midpoint being greater. $P(HC = 2) = P_{key} * P_{key} * P_{key} + P_{key} * P_{key} * (1 - P_{key})$. This can be expanded to show the fragment's hop-count probability.

$$\begin{aligned} P(HC = i) &= P_{key} \times (1 - P_{key})^{i-2} \\ P(HC > i) &= (1 - P_{key})^{i-1} \\ P(HC < i) &= 1 - P(HC = i) - P(HC > i) \\ &= 1 - P_{key} * (1 - P_{key})^{i-2} - (1 - P_{key})^{i-1} \end{aligned}$$

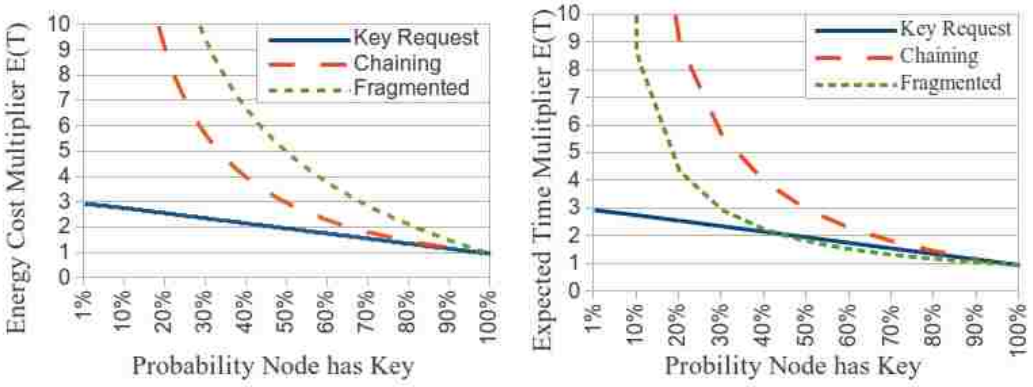
Calculating the hop count of the message is feasible when the individual fragment's hop count is known. A message will be delivered in i hops if one fragment is delivered in exactly i hops, one fragment is delivered in i or less hops, and the third is delivered in i or more hops (independent of order). There are four discrete possibilities: 1) All three fragments can be delivered in exactly i hops, 2) Any one of the fragments can be delivered in less than i hops (because it does not matter which fragment is delivered, three combinations exist), 3) Any one can be delivered in more than i hops, and 4) One can be delivered in less than i hops, while another is delivered in greater than i hops (likewise, this distribution can occur in six different ways). These possible delivery hop counts, shown in

Eqn. 2, can be used to derive the expected delivery time of the fragmented method, shown in Eqn. 3.

$$\begin{aligned} P(HC_{msg} = i) &= P(HC = i)^3 + \\ &\quad 3 * P(HC < i) * P(HC = i)^2 + \\ &\quad 3 * P(HC = i)^2 * P(HC > i) + \\ &\quad 6 * P(HC < i) * P(HC = i) * P(HC > i) \end{aligned} \tag{2}$$

$$E_{frag}(T) = E(T)\frac{P_{key}^4 - 2P_{key}^3 + 5P_{key}^2 + P_{key} - 5}{P_{key}^4 - 5P_{key}^3 + 9P_{key}^2 - 6P_{key}} \tag{3}$$

To follow our original example, the fragmenting security scheme, with each node carrying 30% of the total number of keys in the system, will deliver a message in roughly 3.84 hops. This analysis confirms our earlier assertion that this scheme will deliver a message in much less time than the chaining method, but with more energy consumption. Comparisons showing how the energy costs and delivery times vary with the P_{key} can be found in Figure 4.3.



(a) Increased Energy Cost

(b) Increase of Time to Deliver

Figure 4.1. Expected Performance Comparison

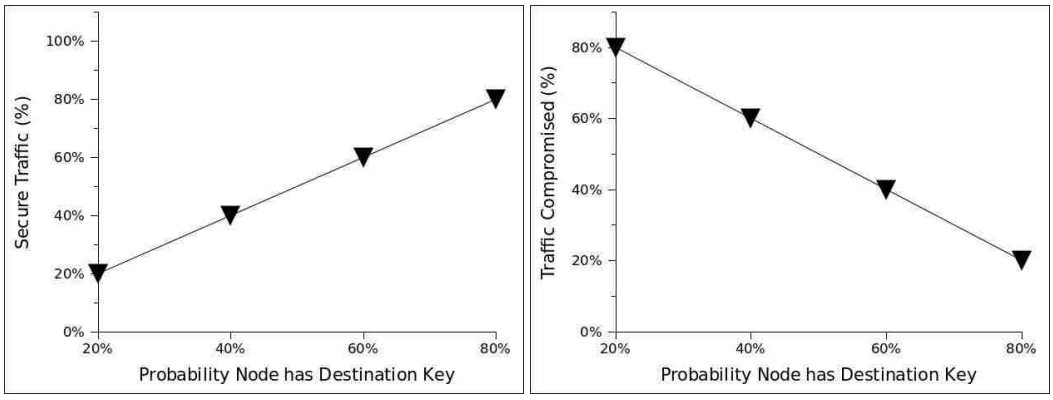
5. SECURITY ANALYSIS

The purpose of 3PE techniques is to provide security in unreliable networks. In environments with either unreliable, easily-compromised communications or nodes that have been compromised by an adversary, both chaining and fragmentation provide some measure of security, but at the cost of reduced performance and increased energy consumption. The conditions under which 3PE fails must be determined to identify whether or not these techniques are beneficial despite their drawbacks.

Certain assumptions were made in evaluating the system. For example, the encryption method itself was considered secure. The network uses a node identification method that functions while the nodes are in direct contact. A certain percentage of the nodes, however, were assumed to have been compromised by an adversary. Another assumption was that messages were compromised if they were sent without encryption, even if they did not pass through a compromised node. However, in the simulations in Section 6, a link-layer encryption scheme was implemented, securing messages unless they were routed through compromised nodes.

5.1. NULL ENCRYPTION

Despite the title, the base security infrastructure will encrypt all of the messages it can. If a node does not have the public key, it will send the message unencrypted. Due to an adversary's ability to eavesdrop, a message can be read any time the source key does not have the destination key. Thus, a linear relationship exists between the number of keys a single node possesses and the number of compromised messages (Fig. 5.1).



(a) Message Delivery Rate (b) Compromised Message Rate

Figure 5.1. Null / Key Request Security Analysis

5.2. KEY-REQUEST SCHEME

A key-request scheme generally is broken by a man-in-the-middle attack, which is more difficult to implement in a MANET. Because nodes are mobile, messages tend not to follow the same message route continuously. The required position between the source and the destination is therefore more difficult to maintain. In practice, this means that a compromised node must lie somewhere on the path of the original key request (so the adversary can alter the public key), on the path of the reply containing the symmetric key (to read the encrypted key), and on the path of the encrypted message. Because the compromised nodes work together, message security can be violated if one node on each path is compromised. However, if it has a dedicated communication channel, an eavesdropping adversary can reply to the source node with false key information immediately upon the message being sent. This attack renders the key-request scheme no more secure than employing no encryption scheme at all, although the attack is more difficult to implement.

5.3. CHAINING ANALYSIS

The major advantage of the chaining method is that all messages are encrypted in one form or another. Eavesdropping attacks are thus rendered useless, so an adversary must rely on compromised nodes to intercept any traffic. To determine the security of this scheme, a Markov State process was used to simulate the current message status (Figure 5.3). A message is sent securely from its origin to the destination only if the origin node already possessed the key. Otherwise, the node uses the chaining algorithm, sending the message to the first link on the chain. The first link will either be compromised by the adversary, have the destination's public key, or redirect the message to another link.

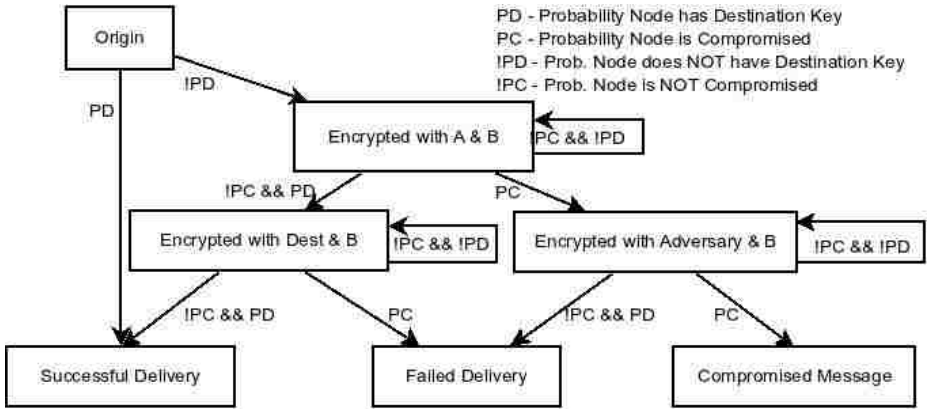
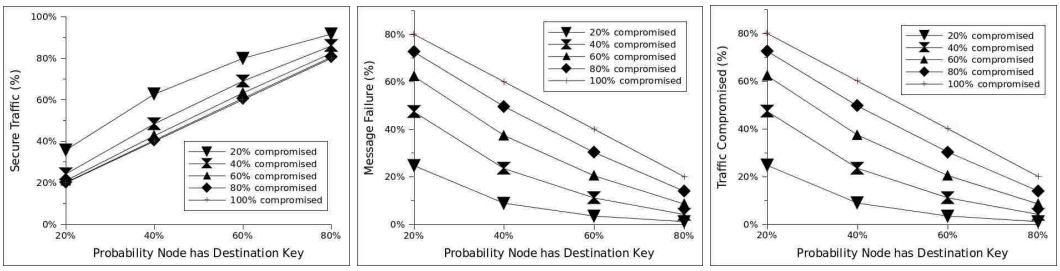


Figure 5.2. 2-Link Chain Process

Three possibilities exist according to how many of the two links are compromised. If neither is compromised, the message is sent successfully to the destination. If only one is compromised, neither the destination nor the adversary can read the result. If both are compromised, the adversary can read the message. The probabilities of these possible scenarios are demonstrated in Figure 5.3.



(a) Message Delivery Rate (b) Message Failure Rate (c) Compromised Message Rate

Figure 5.3. Chaining Security Analysis

5.4. FRAGMENTED ANALYSIS

As with chaining, the adversary's inability to eavesdrop on any traffic means that the focus must be on compromised nodes. When using a threshold encryption scheme, an adversary can sometimes read a message with only a portion of the traffic read. This algorithm has no middle ground when using a 2 of 3 threshold encryption scheme. Eventually, either the adversary will read the message by intercepting two of the three fragments, or the message will be delivered successfully (Fig. 5.4).

The results of this analysis (Fig. 5.4) illustrate that both chaining and fragmenting considerably reduce the percentage of messages compromised by the adversary. In an environment in which more than half of the nodes are compromised, neither algorithm provides enough security to operate safely. Messages can be compromised even when only small portions of environments are compromised. Regardless, both of these algorithms reduce the expected percentage of messages compromised, doubling the number of messages securely transmitted in the base case in which a given node has 30% of the keys available and 20% of the nodes have been compromised, though at the cost of multiplying the total energy consumed by 10 and tripling the transmission time (refer to Figure 4.3). This increase in security is nec-

essary in compromised environments, such as either ubiquitous or social ad hoc networks.

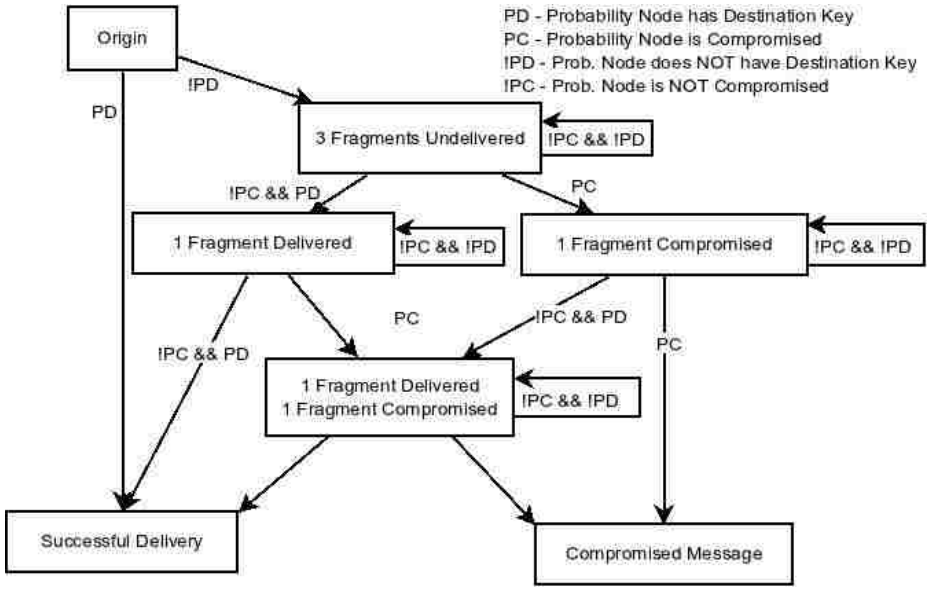
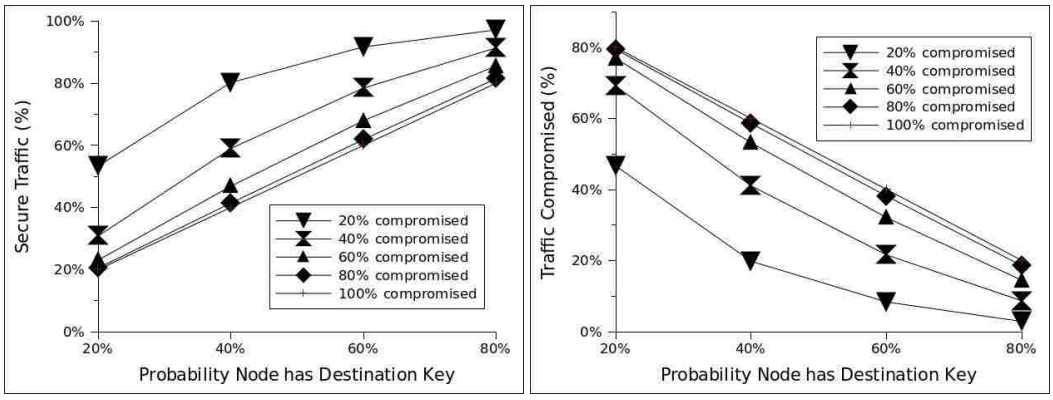


Figure 5.4. 2 of 3 Fragment Process



(a) Message Delivery Rate (b) Compromised Message Rate

Figure 5.5. Fragmented Security Analysis

5.5. OTHER ATTACKS

A Byzantine attack [10] is when nodes are compromised and then work in collusion to compromise security. While considered an advanced attack, other types of attacks also are available to the adversary.

Two attacks to consider in tandem are Black Hole [17] [18] and Wormhole attacks [19][20]. The Black Hole attack is based on nodes identifying themselves falsely as being of high utility in order to direct all traffic through that node. Similarly, Wormhole attacks identify themselves as having high utility. In this case, however, the utility is at least partially correct because messages are routed with both high speed and reliability through a dedicated channel. When applied to a routing scheme, either attack can impact the number of messages delivered successfully. The fact that all message traffic is encrypted means that even directing all traffic through a particular node will not allow the adversary to read it. Incidentally, these attacks are the reason that the midpoint nodes are selected randomly. While delivery speed and reliability may be increased by assigning a utility value to a node (thus indicating its function as a midpoint), an adversary can use this function to route message traffic through compromised nodes. Thus, the current 3PE model is based on midpoints being selected randomly.

Man-in-the-Middle (MitM) attacks are based on a midpoint intercepting a key exchange message and then altering that key to one that the adversary controls. When Alice sends her public key to Bob, midpoint Eve can replace Alice's key with her own. When Bob uses the key, Eve can easily read all of the messages that Bob sends to Alice. Most techniques for preventing the MitM attack are based on a trusted third party verifying the key, which does not work in a MANET, although other techniques are designed to function in such an environment [13] [14] [15]. Both the chaining and fragmenting algorithms are designed to avoid this problem by only accepting keys in direct contact, thus preventing an intermediate node from replacing the keys used.

A final attack to consider is the Sybil attack [21], which is based on an adversary injecting simulated nodes into the network. Because the algorithm selects keys from existing nodes, a large number of false nodes increases the probability that a compromised node will be selected as a midpoint. A review of solutions to this attack is provided by Levine et al. [22].

In conclusion, both 3PE algorithms render the majority of network attacks useless for the purpose of reading encrypted messages being sent across the network. While the Sybil attack remains a threat, solutions are available that can identify simulated nodes with high degrees of accuracy.

6. PERFORMANCE EVALUATION

While all security systems have trade-offs regarding performance, the amount of performance delay should be compared to the increase in performance before implementation. For this reason, both the chaining and fragmenting algorithms were simulated using the Omnet++ Network Simulator. Performance and security metrics were gathered and compared to the null encryption scheme and the key-request scheme. When evaluating mobile networks, the mobility patterns followed by the devices should follow realistic patterns. The Small World in Motion mobility model [12] is a synthetic trace generator used to match real world datasets. It assumes nodes visit locations, with the probability of visiting a location determined by distance and popularity of the area. The performance evaluation was simulated using control parameters set to match the Cambridge '05 experiment.

6.1. EXPERIMENTAL SETUP

The algorithms were simulated in a 1000m x 1000m area, in which 100 mobile nodes were generated. Two seperate simulations were performed. The first simulation set, designed to test the concept and optimize control variables, was implemented using simple nodes and the random waypoint mobility pattern. Each iteration varied the algorithm, the key space, and the buffer space available on each node. A message generation schedule was likewise generated in advance following a Gaussian distribution, with messages being sent every 30 seconds. For message routing, a PRoPHET routing algorithm was used [23] to route messages across the network. This routing algorithm was selected because it functions well in disconnected networks, especially where the nodes have a limited range. However, the random patterns followed by nodes reduced the efficiency of PRoPHET, which typically relies on long-term historical patterns to determine optimal paths. A summary of this setup can be found in Table 6.1.

The second simulation set was performed using more realistic parameters. The SWiM trace generator was implemented to simulate human-carried devices in a large-scale area. By measuring the social patterns, the routing algorithm PRoPHET was capable of identifying reliable message routes. This resulted in greater message efficiency, allowing more accurate measurements of the impact of the 3PE security schemata. A summary of this setup can be found in Table 6.2.

Table 6.1. Random Waypoint Simulation Parameters

System Parameters	Settings
Length x Width	1000m x 1000m
Number of Nodes	100
TX Power (tx)	0.25 mW
Signal-to-Noise Threshold (snr)	$3.98 * 10^{-9}$
Carrier Frequency (cf)	$2.4 * 10^{9}$ Hz
Transmission range between nodes	53m
Message Generation Rate	30 sec mean
Mobility Pattern	Random Waypoint
Movement Speed	5 meters/second
PRoPHET α	.1
PRoPHET β	.05
PRoPHET γ	.95

6.2. COMPARISONS

6.2.1. Simulation Attack Model. The potential attacker's capabilities are a primary factor when evaluating a security system. MANET uses radio communications, which allow attackers to eavesdrop on all unencrypted traffic. The simulation assumes that nodes use link-layer security, meaning that all traffic is encrypted. Breaking message encryption is not considered feasible. Adversaries can, however, inject themselves into the network,

Table 6.2. SWiM Simulation Parameters

PRoPHET Settings		
α	Direct Contact Impact Setting	.007
β	Inirect Contact Impact Setting	.008
γ	Probability Decay Rate	.9992

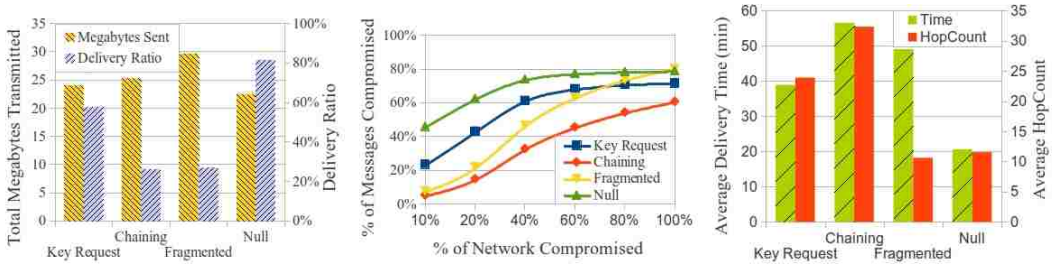
SWiM Control Settings		
Wait Time Exponent	Exponent of the power-law of the waiting time distribution	1.35
Wait Time Upper Bound	Upper bound of the waiting time distribution	12h
α	Distribution of home nodes	.75
Buckets per Side	Bucket number per network area side (Used for performance improvements)	14

disguising themselves as normal nodes. The simulation is based on the adversary compromising a certain percentage of the nodes. These nodes can read and modify any unencrypted messages going through them but are unable to break any encryption scheme used.

6.2.2. Random Waypoint Simulation Results. Based on the analysis described above, the chaining method was expected to be the most secure, but at the cost of reduced performance, both in terms of the ratio of messages successfully delivered and the delivery time.

For general comparison, all algorithms were submitted multiple times using a finite buffer from 10 to 1000 messages. The keychain also varied, holding from 10% to 100% of the available keys in the network. The chaining algorithm varied the number of required links from 2 to 5. Likewise, the fragmented algorithm varied both the number of fragments sent and the number needed for decryption from 2 to 5. Due to limited space, full results are shown for runs with buffers capable of carrying 400 messages, and 20% of the node keys are displayed. The chaining algorithm results for two links are shown, as are the fragmenting results for sending three fragments, two of which are required to decrypt the message. The results (shown in Figure 6.2.2) match

the expected results; 3PE algorithms offer much better security but at a greatly increased cost.



(a) Energy Costs and Delivery Ratio

(b) Security Analysis

(c) Message Delivery Time Analysis

Figure 6.1. Simulation Results, 400 message buffer, 20% of keys

One valuable trait of the 3PE methods is that they are scalable. Security can be increased to suit compromised networks, though at a higher cost. The simulation (Figure 6.2.2) shows that increasing the number of midpoints increases security at a faster rate than the performance degrades. Scaling the system up has drawbacks, including higher transmission costs and longer delivery times. In theory, however, the security can be improved indefinitely.

Similar experiments were run for the fragmented method with various numbers of fragments. The results are complicated by the extra variable. The overview in Figure 6.2.2 shows that performance varies in the same manner as in chaining, but much faster. The performance starts off slightly worse than in chaining and then drops quickly. The security, however, improves just as quickly. For comparison, when chaining into 5 links, the delivery ratio is 16%, with 1% of the messages being compromised. A comparable delivery ratio can be found when fragmenting to 3 messages, in which the compromised ratio is 2.6%. In the same environments, by fragmenting into 5 messages (all of which are needed), the delivery ratio is 14%, but 0.1% of the messages are compromised. This indicates that, while the

fragmented algorithm is more flexible, accomodating a wider range of environments with a faster delivery time, chaining is the more secure method of the two.

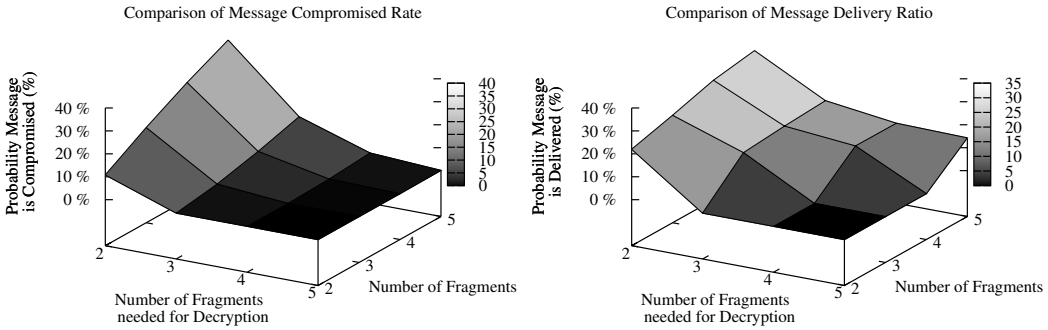


Figure 6.2. Results of Varying Fragment Count

For general comparison, all algorithms were submitted multiple times using a finite buffer from 10 to 1000 messages, and capable of holding from 10% to 100% of the available keys in the network. The Chaining algorithm varied the number of links needed from 2 to 5. Likewise, the Fragmented algorithm was submitted varying both the number of fragments sent and the number needed for decryption, ignoring those in which the number required to decrypt was larger than the number sent. Due to limited space available, full results were shown for runs with buffers capable of carrying 400 messages and 20% of the node keys were displayed, with the number of links used in the Chaining algorithm was set to 2, and the Fragmenting algorithm sending 3 fragments, requiring 2 to decrypt the message. The results, shown in Figure 6.2.2, match the expected results - much larger costs, much better security for the 3PE algorithms.

One valuable trait regarding the 3PE methods is that they are scalable, able to increase security at a higher cost. By increasing the number of midpoints, the simulation shows (Figure 6.2.2) that security increases at a faster rate than the performance degrades. Scaling

the system up has other drawbacks, such as higher transmission costs and longer delivery time, but in theory the security can be improved indefinitely.

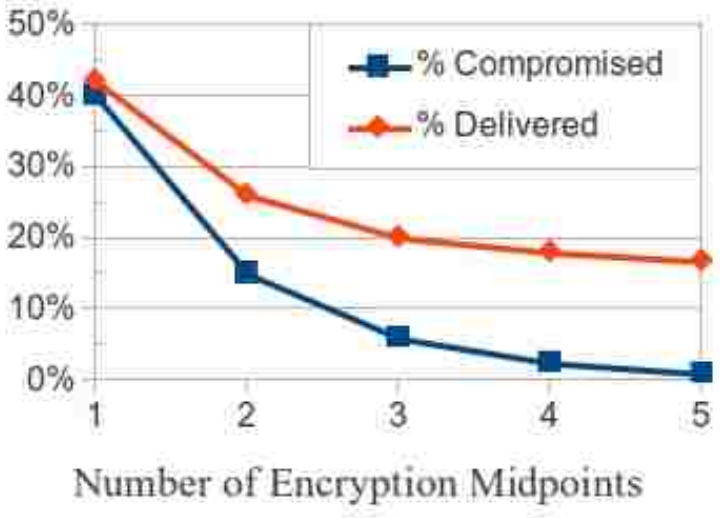
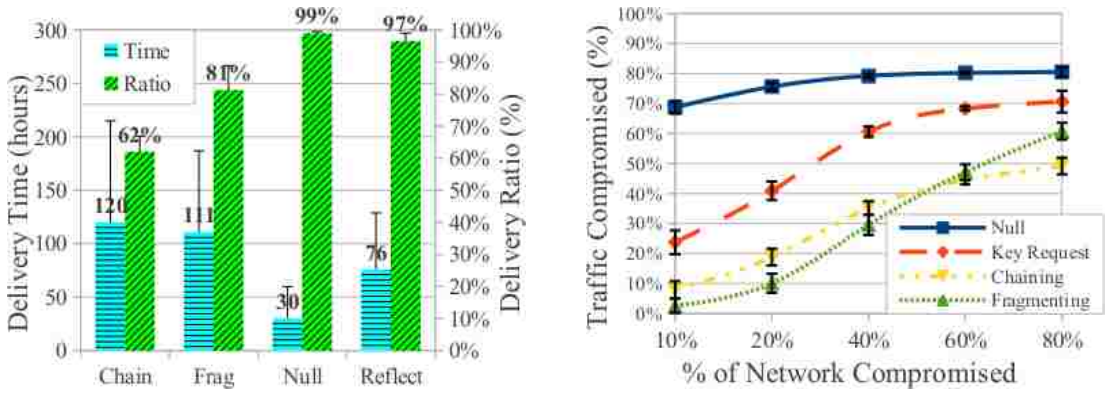


Figure 6.3. Results of Varying Link Count

6.2.3. Small World in Motion Simulation Results. The second set of simulations allowed more accurate measurements of the performance considerations in a realistic environment. The results, show in Figure 6.2.3, indicate that changing the mobility pattern allowed much more reliability and improved security in the environments. This indicates that the 3PE security overlay networks are susceptible to a disruptive environment. The Fragmenting algorithm performed very well in this environment, only exceeding the Chaining algorithm's security when more than half of the network was compromised. Based on these results, the Fragmenting algorithm is more secure and better performing except at very insecure networks.



(a) Simulated Performance

(b) Simulated Security

Figure 6.4. Results of SWiM Experiments

7. CONCLUSION

Both the experimental results and the analytical models indicate that 3PE algorithms can improve the security in an otherwise unsecured network at the cost of increased network traffic and slower performance. While exact numbers vary based on both the network environment and the degree of security needed, results suggest that 3PE algorithms serve as secure methods for routing without public keys. Additionally, both 3PE schemes can be implemented without prior knowledge or trust schemes. They can be fine tuned to the degree of performance and security required by increasing the number of links or fragments. While the chaining system boasts higher security and lower system costs, fragmenting has a faster delivery time and can be modified more easily to suit a wider range of environments.

An ongoing issue to be addressed is the integration of 3PE methods with a proper key management system. Because key management systems can use a variety of trust models to indicate the security of the individual key, a feasible approach is to merge the two, using key management when trust exceeds a certain threshold, and using a 3PE method otherwise. In theory, this would achieve the best of both algorithms, making this a promising area for future development.

8. BIBLIOGRAPHY

[1] S. A. Camtepe and B. Yener, "Key distribution mechanisms for wireless sensor networks: a survey," tech. rep., 2005.

[2] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," pp. 172–194, Springer-Verlag, 1999.

[3] A. Menezes and B. Ustaoglu, "On the importance of public-key validation in the mqv and hmqv key agreement protocols," in *Proceedings of the 7th international conference on Cryptology in India*, INDOCRYPT'06, (Berlin, Heidelberg), pp. 133–147, Springer-Verlag, 2006.

[4] D. R. Stinson, *Cryptography: Theory and Practice, Third Edition (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, Nov. 2005.

[5] R. E. Lewand, *Cryptological Mathematics (Classroom Resource Materials)*. The Mathematical Association of America, Dec. 2000.

[6] P. Golle, M. Jakobsson, A. Juels, and P. Syverson, "Universal re-encryption for mixnets," in *IN PROCEEDINGS OF THE 2004 RSA CONFERENCE, CRYPTOGRAPHERS TRACK*, pp. 163–178, Springer-Verlag, 2002.

[7] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[8] P. F. Syverson, M. G. Reed, and D. M. Goldschlag, "Private web browsing," *Journal of Computer Security*, vol. 5, no. 3, pp. 237–248, 1997.

[9] G. Vakde, R. Bibikar, Z. Le, and M. Wright, "Enpassant: anonymous routing for disruption-tolerant networks with applications in assistive environments," *Security and Communication Networks*, vol. 4, no. 11, pp. 1243–1256, 2011.

[10] B. Wu, J. Chen, J. Wu, and M. Cardei, "A survey of attacks and countermeasures in mobile ad hoc networks," in *Wireless Network Security* (Y. Xiao, X. S. Shen, and D.-Z. Du, eds.), Signals and Communication Technology, pp. 103–135, Springer US, 2007.

[11] K. El Defrawy, J. Solis, and G. Tsudik, "Leveraging Social Contacts for Message Confidentiality in Delay Tolerant Networks," in *2009 33rd Annual IEEE International Computer Software and Applications Conference*, pp. 271–279, IEEE, July 2009.

[12] *Proceedings of the Seventh Annual IEEE Communications Society Conference on*

Sensor, Mesh and Ad Hoc Communications and Networks, SECON 2010, June 21-25, 2010, Boston, Massachusetts, USA, IEEE, 2010.

[13] L. Zhou and Z. J. Haas, "Securing ad hoc networks," *IEEE NETWORK MAGAZINE*, vol. 13, pp. 24–30, 1999.

[14] S. Capkun, L. Buttyn, and J.-P. Hubaux, "Self-organized public-key management for mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 2, pp. 52–64, 2002.

[15] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing robust and ubiquitous security support for manet," in *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, November 2001.

[16] Y. Kong, J. Deng, and S. R. Tate, "A distributed public key caching scheme in large wireless networks," in *Proc. of IEEE Global Telecommunications Conference - Communication and Information System Security (GLOBECOM '10)*, (Miami, FL, USA), December 6-10 2010.

[17] M. Al-Shurman, S.-M. Yoo, and S. Park, "Black hole attack in mobile ad hoc networks," in *ACM Southeast Regional Conference*, pp. 96–97, 2004.

[18] J. Yin and S. K. Madria, "A hierarchical secure routing protocol against black hole attacks in sensor networks," in *SUTC (1)*, pp. 376–383, 2006.

[19] M. Jain and H. Kandwal, "A survey on complex wormhole attack in wireless ad hoc networks," in *Proceedings of the 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies*, ACT '09, (Washington, DC, USA), pp. 555–558, IEEE Computer Society, 2009.

[20] S. K. Madria and J. Yin, "Serwa: A secure routing protocol against wormhole attacks in sensor networks," *Ad Hoc Networks*, vol. 7, no. 6, pp. 1051–1063, 2009.

[21] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: Analysis & defenses," 2004.

[22] B. N. Levine, C. Shields, and N. B. Margolin, "A Survey of Solutions to the Sybil Attack," Tech report 2006-052, University of Massachusetts Amherst, Amherst, MA, October 2006.

[23] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, pp. 19–20, July 2003.

V. Content Distribution in Delay-Tolerant Networks using Social Context

Roy Cabaniss and Sanjay Madria

Department of Computer Science,

Missouri University of Science and Technology, Rolla, Missouri 65401

Wireless communication technologies have given rise to the development of Delay Tolerant Networks, a collection of devices which route data opportunistically. These networks can be used to distribute data, such as news articles, advertisements, or media, to interested clients. To deliver content efficiently, the Social Content Distribution (SCD) schema collects social context data and request patterns. Nodes are then designated as mobile repositories, or throwboxes, at which large content stores are kept. Clients request specific articles from nearby repositories. By predicting which nodes request data, based on request and social patterns, the SCD schema services content requests quickly and efficiently. Applying the SCD protocol to a simulated environment of social entities resulted in improvement of the delivery ratio and reducing the average delivery time to a third of the static repository.

1. INTRODUCTION

Wireless ad-hoc networks are a developing field in which devices exchange information and resources to perform tasks beyond their individual capabilities. These tasks can include large-scale computing, message delivery, navigation, or sharing sensory data. Many such networks are examples of Delay-Tolerant Networks (DTNs). Devices in these architectures store messages in their buffer to be forwarded to the destination as opportunities for routing occur. One task applicable to DTNs is that of efficient content distribution.

Content distribution is defined as the task of providing requested data to the clients. Requests are constructed by the client and transmitted to a node capable of servicing them, usually a server or repository. This is different from caching in the distribution of the content requested. Caching is used ideally to provide the same data item to a large number of clients, which means having individual nodes keep a copy of the data item allows them to serve requests just as well as the server[1]. If the content repository contains a large number of related data items that are requested individually, caching a particular data item will only allow the node to serve that specific data item. A mobile repository, sometimes called a data store or a throwbox, contains a large collection of related items which allow the device to serve the majority of requests.

An example execution is a mobile advertisement server. Since each consumer is interested in a different set of articles, caching is of limited benefit. Although a node can cache several items, it is unlikely that nodes it encounters will be interested in that specific advertisement. However, if a node can act as a throwbox containing a collection of advertisements it can respond to requests, reducing average access time. Clients in the shopping environment send a request to the nearest throwbox, including any patterns they wish to share, and the throwbox can process it without waiting for the request to be processed by the main server. Other use cases include media files, navigation data, or a news site.

In the area of delay tolerant networks, social context can be used to improve the efficiency of certain tasks such as routing or caching. Clients tend to be influenced by social patterns regarding their content requests [2]. In general, if the client's contacts request a specific data item, the probability of that client requesting it are increased. To take advantage of this tendency a node must first identify the consistent contacts of a node. In a dynamic environment, in which nodes are added and removed, the social structure must accommodate such changes.

The purpose of the Social Content Distribution schema is to locate the optimal position for mobile repositories. This is accomplished by evaluating the existing social structure of the network, identifying which nodes and groups issue frequent content requests, and locating the devices most capable of fulfilling these requests. Initially, a node identifies its frequent contacts, forming small groups. These groups are merged, joined and left by nodes to accurately reflect the network's social structure. Simultaneously, nodes track the request frequency of themselves and their groups. This data is reviewed to determine a metric for the benefit of positioning a throwbox at that node. The end result is to position repositories in close proximity to requests, increasing content availability and reducing the average round-about time.

1.1. RELATED WORK

1.1.1. Bubble Rap. The Bubble-Rap grouping method [3] allocates nodes into groups based on direct contacts. They distribute using a method called $k - cliques$, in which all fully connected groups of k members are considered a distinct group. They are then merged with all other $k - cliques$ which share $k - 1$ members. This provides an accurate representation of the groups formed by a set of nodes. However, it relies on global knowledge of the nodes' contacts, and it must have them before the algorithm can group them. As such, it cannot be used in an on-

going DTN, although it provides a valuable tool for analysis and comparison.

1.1.2. R-P2P. Another system designed to allocate data through a delay tolerant network is the R-P2P system [4]. It is designed to ensure content, such as advertising or media content, is consistently available to a client. This is performed by designating certain nodes in the environment as throwboxes, responsible for serving data queries and maintaining updated content. Throwboxes maintain a Distributed Hash Table among themselves, enabling them to locate data without storing is directly. When a client requests a data item, it contacts the nearest throwbox. The distributed hash table is consulted to determine which throwbox contains the data item in particular, then dedicated communication channels obtain the item. The use of throwboxes in the DTN is a valuable contribution to the field of content distribution, which the SCD algorithm utilizes along with a social aspect.

1.1.3. OnMove. An earlier attempt to position throwboxes in a DTN uses social context, among other metrics. The OnMove protocol [5] is designed to determine which nodes are ideal for distributing content based on a series of parameters. By calculating the repository rank of a node based on social similarity, meeting frequency, connection quality, content similarity, and 'betweenness' to other nodes, the protocol can establish ranking metric. While the overview is promising, the complete protocol was never developed. The SCD algorithm expands on this concept, applying the social context of nodes to identify ideal content repository nodes.

2. PROPOSED SCHEMA

2.1. GROUPING

All nodes in a mobile ad hoc network are aware of their immediate surroundings, including neighboring nodes, via wireless communications. A social group, in this context, is a collection of nodes which have regular contact with each other. Nodes of a social group maintain group data, including group membership and the metrics of all group members. By tracking the contact patterns of a node it is possible to extrapolate the social patterns a node follows. Node's maintain their group list in three ways. When two nodes have regular contact with each other, they form a new social group. Two social groups with similar members can merge. Finally, nodes can resign from groups which they no longer participate in.

2.1.1. Forming Groups. To determine any groups formed by nodes, the first step is to establish a metric by which the distance between two nodes can be measured. The SCD schema calculates the percent of time spent in direct contact, using an exponential moving average formula to adjust the current estimate. Initially the direct contact strength, $\lambda_{A,B}$ is set to 0 between all nodes. Whenever nodes enter contact, nodes measure how long they have spent in and out of contact. This data is used to calculate $\lambda_{A,B}$ as the estimated percent of time the nodes spend in contact. As an exponential moving average, it is based on historical data, with a higher emphasis placed on more recent data. The control variable α determines how quickly the contact strength changes, as shown in eq. 1. A higher α will result in more emphasis being placed in recent changes, allowing nodes to swiftly adapt but also be fooled or confused by brief changes.

$$\lambda_{A,B} = (1-\alpha)\lambda_{A,B} + \alpha \frac{time_{contact}}{time_{contact} + time_{nocontact}} \tag{1}$$

Although $\lambda_{A,B}$ measures the direct contacts of a node fairly well, it tends not to accommodate indirect contacts. Indirect contacts, also called transitive contacts, indicate a node's ability to communicate to another node via intermediate nodes. Although $Node_A$ may only occasionally encounter $Node_B$, if it constantly encounters other nodes which have high connection to $Node_B$ there is still a high level of association. The control variable β is used to determine the impact of transitive contacts to a node. With this a cumulative estimate of the contact strength, $CS_{A,B}$, is calculated and compared to a group formation threshold, ψ. When the contact strength exceeds a control threshold ψ, this indicates the nodes are considered close enough to form a new group. The updates to λ and forming new groups are detailed in Algorithm 23.

Algorithm 23 Forming New Groups

Notation
$Node_A$, $Node_B$ - Active nodes in the network
$\lambda_{A,B}$ - Direct contact strength between $Node_A$ and $Node_B$
$CS_{A,B}$ - Cumulative contact strength between $Node_A$ and $Node_B$
α - Control variable, rate of direct contact change
β - Control variable, impact of transitive contacts
ψ - Threshold for forming a new group

Trigger - $Node_A$ contacts $Node_B$
$\lambda_{A,B} = (1-\alpha)\lambda_{A,B} + \alpha \frac{time_{contact}}{time_{contact}+time_{nocontact}}$
$CS_{A,B} = \lambda_{A,B} + (1-\lambda_{A,B}) \times \beta \times \prod_{Node_B} \lambda_{A,C} \times \lambda_{C,B}$
if $CS_{A,B} > \psi$ **then**
  Form new $Group_Y$, consisting of $Node_A$ and $Node_B$
end if

Since groups will change as time passes, through group merges or resignations, distributing the group data through all the participating nodes is difficult. Attempts to allow any node in the group to make changes resulted in data fragmentation, which is when two nodes of the same group had different group data. To avoid this, one of the nodes is arbitrarily selected as the group head. The group head also maintains data on member

nodes, specifically their relative connection strengths. This data is used to validate group merges.

2.1.2. Merging Groups. By tracking the contact strength between nodes, a series of two-node social groups can be formed. The next step is to integrate these links into larger groups by merging similar groups. Nodes which are joint members of two groups will periodically review the group data for merges. Although further checks will occur at the group head, the joint member's only concern is group similarity, which is defined as the ratio of joint group to the union of the two groups - $\frac{|Group_Y \cap Group_Z|}{|Group_Y \cup Group_Z|}$. The control variable τ determines what percentage of nodes must be in both groups. If the joint node determines the two groups are similar enough, then a $SUGGEST$ message is sent to the group head of the smaller of the two groups. This message indicates that the node believes there is a potential group merge available.

When the group head receives the suggestion, it confirms that the two groups are similar. It then performs a check to ensure that it has enough contact strength with the new group; if the group head's average *CS* to the merged group is below ψ, it will discard the suggestion. Otherwise, an $INVITE$ message is generated and sent to the other group head. This message indicates that one of the groups has already approved the merger, and the other group must still approve the merge. The other group head confirms the similarity and contact strength as well. At this point, both group heads have confirmed that the two groups should be merged. A *KILL* message is sent to all members of the smaller group, and all members are added to the larger group. The process is detailed in Algorithm 24.

Performing merges in this manner updates groups in a limited environment. It has the drawback in that it does not ensure all nodes of the group are strongly connected to all members of the new group. To address this issue, nodes can resign from groups they no longer have a strong attachment to.

Algorithm 24 Merging Groups

Notation
$Group_Y$, $Group_Z$ - Social groups in DTN
$Node_A$ - Member of both $Group_Y$ and $Group_Z$
$Node_B$ - Group head of $Group_Y$
$Node_C$ - Group Head of $Group_Z$
τ - Threshold for Merging a Group
ψ - Grouping Threshold

Trigger - Periodically in $Node_A$
for all $Group_Y, Group_Z$ of which $Node_A$ is a member **do**
  if $\frac{|Group_Y \bigcap Group_Z|}{|Group_Y \bigcup Group_Z|} > \tau$ **then**
    Send $SUGGEST$ Message to Group Head ($Node_B$)
  end if
end for

Trigger - $Node_B$ receives $SUGGEST$ Message
if $\frac{Group_Y \bigcap Group_Z|}{|Group_Y \bigcup Group_Z|} > \tau$ **then**
  Calculate average $CS_{B,Group_Z \cup Group_Y}$
  if $Avg(CS_{B,Group_Z \cup Group_Y}) > \psi$ **then**
    Send $INVITE$ Message to Group Head $Node_C$
  end if
end if

The Group Head confirms the suggestion, and sends a Confirmation to the other group

In $Node_C$...
if $\frac{|Group_Y \bigcap Group_Z|}{|Group_Y \bigcup Group_Z|} > \tau$ **then**
  Calculate average $CS_{B,Group_Z \cup Group_Y}$
  if $Avg(CS_{B,Group_Z \cup Group_Y}) > \psi$ **then**
    $Group_Z = Group_Z \bigcup Group_Y$
    Send $KILL_Y$ Message to all members of $Group_Y$
  end if
end if

If both Group Heads approve, one group is added to the other and then removed

2.1.3. Resignation. Periodically nodes will review their group list to ensure they are still participating, as shown in Algorithm 25. The average contact strength to all group members is calculated, and if it is beneath ψ the node resigns from the group. A RESIGN message is sent to the group head which requests that this node be removed from the group data. To avoid fragmentation, the sending node will not remove group data until the message has been confirmed. This stage limits the groups size, and ensures members of a group are still regularly encountering other members.

Algorithm 25 Resigning from a Group

Notation
$Node_A$ - Active node in the network
$Group_Y$ - Group containing $Node_A$
$Node_B$ - Head of $Group_Y$
ψ - Group Threshold

Trigger - Periodically in $Node_A$
$sumCS = 0$
for all $Node_B \in Group_Y$ **do**
  $sumCS = sumCS + CS_{A,B}$
end for
$AvgCS_{A,Y} = \frac{sumCS}{|Group_Y|}$
if $CS_{A,Y} < \psi$ **then**
  Send $RESIGN_{A,Y}$ to $Node_B$
end if

2.2. CONTENT REPOSITORY POSITIONING

2.2.1. Request Frequency. To determine the optimal position for a repository, it is necessary to measure which nodes are requesting data items frequently. The request score for $Node_A$, μ_A, is an exponential moving average which estimates the time between requests. Whenever a node makes a user request the update process estimates the average time between requests based on the previous estimate and the control variable ϕ, which

determines how much emphasis is placed on historical data. This process allows nodes to maintain an up-to-date estimate, adjusting to reflect the node's request patterns.

$$\mu_A = (1-\phi)\mu_A + \phi(time_{current} - time_{prev}) \tag{2}$$

2.2.2. Group Request Score. The SCD schema augments repository selection by considering the social aspects of the nodes. The ranking of a node depends on its own ability to deliver a content item to requesting nodes and the ability of its contacts. Having identified the social groups in Section 2.1, a node can calculate its Request with Group Score (*RGS*). This is considered as the time for $Node_A$ or a neighboring node to request a data item.

Traffic requests tend to follow a Markov-Poisson process [6]. As such, the time between a node's estimated requests follow exponential distribution. This allows the schema to estimate the minimum of any set of node requests as the sum of the λ of these distributions (not related to the contact strength between nodes in 2.1). The λ of the distributions is the inverse of the average time between contacts, $\frac{1}{\mu_A}$. Thus, for group G_Y an estimate of the soonest request of this group is shown in Eqn 3.

$$\mu_{Group_Y} = \frac{1}{\sum_{Node_B|B \in G} \frac{1}{\mu_B}} \tag{3}$$

The *Spread* control variable is an estimate of how likely a client's request patterns are influenced by their social groups. This tends to vary based on the type of content - while close friends may watch similar videos, program update requests are not based on a contact's programs. This control variable allows different amounts of emphasis to

be placed on the social aspects of the content. These variables are used to calculate the request score with group data, RSG_A.

$$RSG_A = \frac{1}{\frac{1}{\mu_A} + Spread \times \sum_{Group_Y | A \in Group_Y} \frac{1}{\mu_{Group_Y}}} \tag{4}$$

2.2.3. Repository Position Ranking. At this stage of the process, nodes are aware of the frequency with which they contact other nodes ($\lambda_{A,B}$) and how often these nodes will request content, either on their own behalf or that of neighbors (RSG_A). The ranking algorithm establishes a metric, Rank Position Score (RPS_A) as the sum of other node's chance of requesting the data times the chance of encountering $Node_A$.

$$RPS_A = \sum_{Node_B} \lambda_{A,B} \times RSG_B \tag{5}$$

Whenever a node which contains a repository, $Node_A$, encounters $Node_B$, both calculate their respective RPS. If RPS_B is greater than RPS_A, this indicates that $Node_B$ is closer to more frequently requesting nodes than $Node_A$. A message updating the data owner is sent, and the entire repository is shifted. As this is a high-bandwidth operation, it is only performed if RPS_B exceeds RPS_A by a certain threshold.

For reference, a summary of the control variables and metrics used has been provided in Table 2.1. The ideal control variables were identified through experimentation, submitting repeatedly under baseline conditions before implementing experimental algorithms.

Table 2.1. Variable Reference Chart

Control Variables		Range Tested	Ideal Value
α	Contact Strength Change Rate	0 - 1.0	0.4
β	Transitive Contact Impact	0 - 1.0	0.7
ψ	Group Formation Threshold	0 - 1.0	.2
ϕ	Request Strength Change Rate	0 - 1.0	0.85
$Spread$	Impact of group on node	0 - 0.1	.05
Ideal values tested by experimentation			
Variables			
$\lambda_{A,B}$	Direct Contact Strength between nodes		
$CS_{A,B}$	Contact Strength between nodes, including transitive contacts		
μ_A	Estimated request frequency of $Node_A$		
μ_{GroupY}	Estimated request frequency of $Group_Y$		
RSG_A	Estimated request frequency of $Node_A$ and participating groups		

3. ANALYSIS

To determine the performance of the SCD schema the schema was implemented in *The One* DTN simulation environment[7]. This tool, originally written in Java, is designed to realistically implement traits such as buffer overflow, transmission speeds, and disruption (Table 3.1). The analysis was generated using 120 mobile nodes using Bluetooth to communicate. Nodes used the Dynamic Social Grouping routing algorithm, which functions well in social, largely disconnected environments [8].

Table 3.1. The One Control Variables

Simulation Variables	
Buffer Size	5Mb
Range	40m
Bandwidth	250kBps
Message TTL	12h
Duration	30 days

3.1. MOBILITY

To simulate a schema which relies on the social aspects of mobile nodes, the mobility patterns must incorporate social dynamics. For this reason a series of patterns were generated using the Small World in Motion (SWiM) algorithm, which is based on nodes repeatedly revisiting either locations that have meaning to them as individuals (house, lab), or locations where they see a large number of other nodes (restaurant, park)[9]. The time these nodes remain at a given location follows a power law distribution with a set upper limit. Another variable, α, determines how the nodes' home points are distributed. An α of 0 represents uniform distribution, while 1 indicates all nodes have the same home. These variables can be tuned to match sets of real world data. A control parameter set was

generated to match the Cambridge dataset (Table 3.2), which is a real-world collection of Bluetooth contacts [10]. The SWiM tool created 120 nodes and deployed them in a 1km by 1km area without features. Using the control parameters to match the Cambridge dataset, a period of 1 month was randomly generated 20 times. The result was several sets of mobility patterns, each modelling human behaviour.

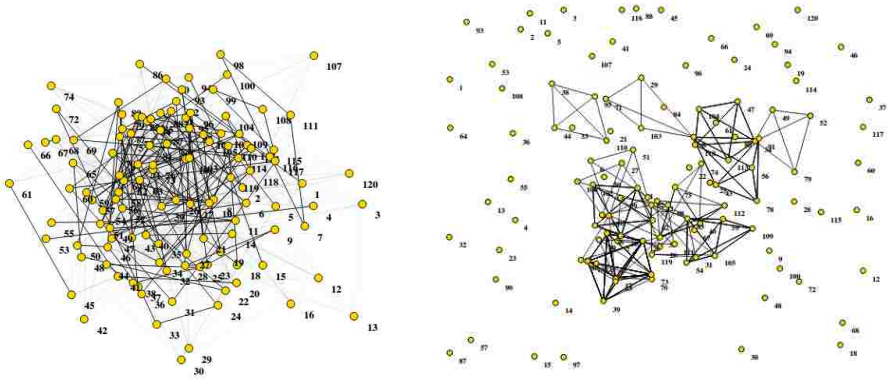
Table 3.2. SWiM Cambridge Control Variables

Nodes		120
Wait Time Exponent	Exponent of the power-law of the waiting time distribution	1.35
Wait Time Upper Bound	Upper bound of the waiting time distribution	12h
α	Distribution of home nodes	.75
Buckets per Side	Bucket number per network area side (Used for performance improvements)	14

3.2. TRAFFIC PATTERN

Message traffic was generated using a Poisson process in which each node has a different interest level. The interest level was generated to average at a user requesting a data item per hour, with a variance of 12 hours. Afterwards, the tendency of a social group to influence interest was considered. The mobility data was reviewed to determine which nodes spent over 2% of their time in another node's company (a sample linkage display is shown in Figure 3.1(a), figure produced by SocNetV[11]). A MATLAB script was implemented to review the links, generating a series of groups using the k-clique grouping algorithm for analysis (Figure 3.1(b)). After the cliques were prepared, the participating nodes' interest levels were adjusted towards the group average. The result was a series of

contact request patterns in which nodes would request data based on individual interest and group tendencies.



(a) Connectivity Display (b) $k-clique$ Display

Figure 3.1. Sample Network Layout

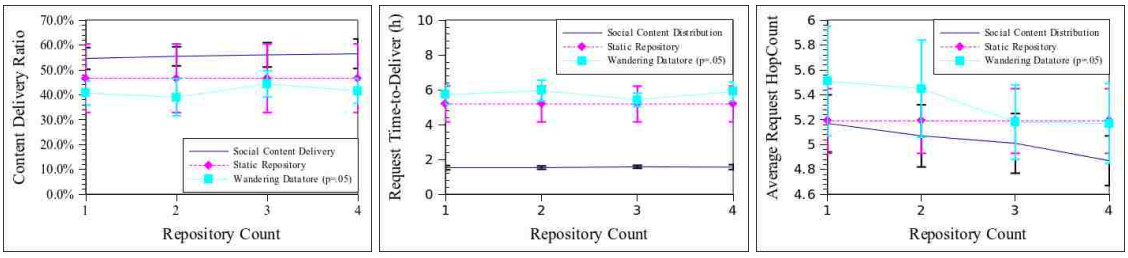
3.3. COMPARISON

The results of the SCD schema were compared against two baselines. The first is the Static Repository, which has no mobile throwboxes. Instead, one node acts as the server. This baseline has the advantage of no update cost, since the server node is also the one generating all updates. However, all requests are sent to the server, and the server's location in the DTN is randomly selected, resulting in poor performance. In contrast, the Wandering Datastore baseline has multiple repositories that transfer themselves randomly. On each contact, a throwbox has a set probability, p, to transfer to the neighboring node. A range of probabilities were simulated, resulting in the better performance with $p = .05$. This method has a tendency to spread datastores throughout the network, allowing them to service the whole environment. By positioning randomly, however, it does not take advantage of any context data, and thus performs poorly.

3.4. RESULTS

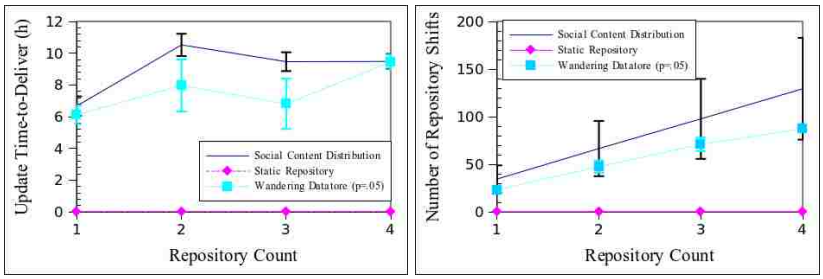
The simulation results indicated that applying social context to the task of throwbox positioning improves the performance considerable. The delivery ratio shows a 21% improvement, and the delivery time is reduced to 30%, as shown in Figure 3.4. This performance increase has a trade-off, however, of higher costs to update the repositories (Figure 3.3). A static repository (which remains on a single device) can be updated at no cost, since all the requests come to the device. In contrast, allowing nodes to place themselves optimally requires the various repositories be updated by sending file updates through the network. As the simulation uses a sparse DTN of relatively low mobility (see section 3.1), this is a significant delay, taking between 6 to 10 hours. This delay is acceptable to some applications (such as advertisement or navigation data), but other applications require updates be implemented much more quickly (such as a news site).

In addition to the update delay, the throwbox must also deploy itself to the optimal position. On contact, a repository will transfer, or shift, when the *RPS* of the contact exceeds the current node's ranking. Considering that the content repository can range from large to very large, the costs of such transmissions are considerable. A single throwbox must reposition itself on average once every day to maintain the optimal position, as shown in Figure 3.3(b). Even a large repository (500Mb) can position itself optimally by using 4J of energy per day (based on the 100nJ per bit cost [12]).



(a) Content Delivery Ratio (b) Content Delivery Time (c) Content HopCount

Figure 3.2. Request Performance



(a) Update Delivery Time (b) Number of Repository Shifts

Figure 3.3. Repository Maintenance

Allocating additional throwboxes apparently had very limited benefit to the algorithm. Although both delivery ratio and time-to-delivery improved with added repositories, the performance increase was marginal. This is possibly due to the tendency of throwboxes to find the optimal position independent of the activity of other throwboxes. Although there is a mechanism to prevent them from inhabiting the same device, all throwboxes will identify the most heavily trafficked network area and position themselves in the same region. Future experiments should be considered to implement a repulsion element to the throwboxes, ensuring they maintain a reasonable distance from each other.

3.5. TEST CASE

To determine the benefit of the SCD algorithm, consider an application which distributes music samples. The mobile repository contains a sample of every music the store has for sale, and clients will send requests for specific songs. If the store is using itself as the sole source, it acts as a static repository. Based on the simulations, clients who request a song will have to wait an average of 5 hours to receive the song. Assuming the song is 3Mb, it will also require 131mJ to deliver from the entire network. Since a sample smartphone battery contains 3284J of energy [13], this number is fairly low, although high traffic from other nodes can still drain the battery life.

In contrast, the store may request customers use an app implementing the SCD algorithm. If so, a clients song request time will drop from 5 hours to 1 hour and 30 min. This is still much slower than downloading from the 3G network, which can perform a 3MB download in roughly 60 seconds. However, the energy consumption (not to mention bandwidth allocation) is much cheaper using SCD. The 3G environment requires 20J to download a 100kB file, so the 3MB song will require an estimated 2000J to obtain[13]. In contrast, the SCD schema will require only 130mJ, even if only a single repository is used. Increasing the number of repositories can reduce this to 126mJ.

There are also maintenance costs to consider. Each repository will shift position roughly once per day. Assuming the repository contains 200 songs, this means the cost per day is 4.2J. Furthermore, when the server wants to update the song selection it must send an update to each throwbox. Since the average hopcount for updates is 9.13, this means that sending one song to one throwbox will consume an additional 29mJ, and will take roughly 6 hours and 40 minutes to implement.

The above analysis demonstrates SCD schema provides much faster access to data than a static and randomly selected repository although it requires some energy to maintain the throwboxes. It is much cheaper than using the 3G network to download the song, both in terms of bandwidth and energy consumption.

4. CONCLUSIONS

By taking advantage of the social context of a network, several of the network capabilities can be augmented. With the full range of social dynamics yet to be explored, applications can be developed to take advantage of the additional information, as well as developing algorithms to identify social patterns. This paper presents one such social algorithm, identifying groups dynamically by measuring the contact intervals. It then uses this data to accurately identify which nodes are optimal positions for mobile repositories. Future considerations can include further optimizing the social group detection algorithm. Identifying links between nodes based on common request patterns, for instance, may be used to establish which nodes are contacts and which are friends. This in turn can be used to augment grouping by distinguishing between common contacts and common interests.

5. BIBLIOGRAPHY

[1] Y. Ma, M. R. Kibria, and A. Jamalipour, "Cache-based content delivery in opportunistic mobile ad hoc networks.," in *GLOBECOM*, pp. 768–772, IEEE, 2008.

[2] D. J. Crandall, D. Cosley, D. P. Huttenlocher, J. M. Kleinberg, and S. Suri, "Feedback effects between similarity and social influence in online communities," in *KDD*, pp. 160–168, 2008.

[3] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay-tolerant networks," *IEEE Trans. Mob. Comput.*, vol. 10, no. 11, pp. 1576–1589, 2011.

[4] F. De Pellegrini, I. Carreras, D. Miorandi, I. Chlamtac, and C. Moiso, "R-p2p: a data centric dtn middleware with interconnected throwboxes," in *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems*, Autonomics '08, (ICST, Brussels, Belgium, Belgium), pp. 2:1–2:10, 2008.

[5] R. C. Rumin, E. Jaho, C. Guerrero, and I. Stavrakakis, "Onmove: a protocol for content distribution in wireless delay tolerant networks based on social information," in *Proceedings of the 2008 ACM Conference on Emerging Network Experiment and Technology, CoNEXT 2008, Madrid, Spain, December 9-12, 2008* (A. Azcorra, G. de Veciana, K. W. Ross, and L. Tassiulas, eds.), p. 40, ACM, 2008.

[6] A. Erramilli, O. Narayan, and W. Willinger, "Experimental queueing analysis with long-range dependent packet traffic," *IEEE/ACM Trans. Netw.*, vol. 4, pp. 209–223, Apr. 1996.

[7] A. Keränen, J. Ott, and T. Kärkkäinen, "The one simulator for dtn protocol evaluation," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, Simutools '09, (ICST, Brussels, Belgium, Belgium), pp. 55:1–55:10, 2009.

[8] R. Cabaniss, S. Madria, G. Rush, A. Trotta, and S. S. Vulli, "Dynamic social grouping based routing in a mobile ad-hoc network," in *HiPC*, pp. 1–8, 2010.

[9] S. Kosta, A. Mei, and J. Stefa, "Small world in motion (SWIM): Modeling communities in ad-hoc mobile networking," in *Proceedings of The 7th IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2010)*, (Boston, MA, U.S.A.), June 2010.

[10] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD data set cambridge/haggle (v. 2009-05-29)," May 2009.

[11] D. V. Kalamaras, “SocNetV,” October 2010.

[12] J. M. Kahn, R. H. Katz, Y. H. Katz, and K. S. J. Pister, “Emerging challenges: Mobile networking for ”smart dust”,” *Journal of Communications and Networks*, vol. 2, pp. 188–196, 2000.

[13] N. Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh, and J. P. Singh, “Who killed my battery?: Analyzing mobile browser energy consumption,” in *Proceedings of the 21st international conference on World Wide Web*, WWW ’12, (New York, NY, USA), pp. 41–50, ACM, 2012.

SECTION

4. CONCLUSION

This document presents a series of algorithms for efficient routing in a Delay Tolerant Network based on the social dynamics of the mobile nodes. It also presents algorithms enabling secure communications when the source does not have access to a destination's public key. Finally, it applies social context to the problem of optimal mobile repository selection, allowing content to be placed in the network such that it can be distributed quickly and efficiently.

The Dynamic Social Grouping algorithm is designed for efficient node-to-basestation routing. The group identification method based upon contact patterns was introduced. The group data was used to augment the probabilistic scheme. By combining a node's group data with individual probability, the algorithm could accurately estimate a node's chance of successful message delivery based on previous message performance. A simulation using real-world data from an IEEE conference showed a significant improvement over the Epidemic and Probabilistic routing algorithms.

The Dynamic Social Grouping - Node to Node algorithm is an expansion of the DSG routing algorithm. Using a similar method to identify the social groups of the environment, the routing method was expanded to deliver to any node in the network. Further experiments were performed using different methods to route through known groups. The routing could be based on the node's performance with previous messages (Performance Based Probability), or it could be based on contact frequency, both direct and indirect (Contact Based Probability). Further, the Hybrid algorithm could ignore probability within a group, routing epidemically through the destination's social groups. These algorithms were tested using the MIT Reality Project dataset, which collected contact and message

data about students over the course of 9 months, resulting in long-term views of the social patterns.

There are two Three Point Encryption algorithms designed to send a message privately in a DTN when the source node does not have the destination's public key. The Chaining method selects other nodes for which it does possess the key, then encrypts the message for both using a commutative encryption algorithm. Each midpoint, upon receiving the message, will remove their layer of encryption and then encrypt it for the final destination. This algorithm is the more secure of the two at a penalty to delivery ratio and time. To reduce the impact, the Fragmenting method was implemented. This algorithm encrypts the message using a threshold encryption technique and then sends each fragment of the key through a different midpoint. Since threshold encryption can be decrypted by a subset of the keys, the final destination can decrypt the original message even if fragments were intercepted or dropped. This algorithm has a much faster speed and better delivery ratio, but the adversary can break the security more often. Both of these algorithms were tested in a highly compromised DTN, and showed that they reduced the probability of a message being compromised considerably, with a trade-off of reduced delivery ratio and increased delivery time.

The selection of mobile repository location in a DTN can be improved by a schema which incorporates social context. While nodes can cache data items they encounter, several applications are too large to efficiently store in a single node. Further, in certain applications nodes are unlikely to request a data item multiple times, which renders caching nearly useless. Distributing repositories throughout the network can improve access time, and this can be further improved by selecting their location intelligently. By identifying groups and matching their interests, both as a group and as an individual node, a mobile repository can preemptively position data to serve client requests, consuming less bandwidth and improving access time.

VITA

Roy Cabaniss was born in Georgia of the United States of America. He earned his Bachelors of Science undergraduate degree from the University of Arkansas at Fayetteville, majoring in Computer Science. After his graduation, he become a Programmer, then Advanced Programmer for Wal-Mart ISD in the Replenishment Division. His responsibilities involved updating the Two-Tier Replenishment systems, upgrading buffer pull systems, and monitoring the background processes that order supplies for the stores.

Roy was accepted at the Missouri University of Science and Technology in 2008, where he earned his Doctorate of Philosophy in May, 2013. While there he served as a research assistant for Dr. Sanjay Madria, focusing in the areas of routing and security in Delay Tolerant Networks and Mobile Ad-Hoc Networks. He served as a teaching assistant for the Introduction to C++ classes, both the lab and the lecture, and mentored summer workshops teaching wireless sensor systems to undergraduate students.