MELHORANDO O ENCAMINHAMENTO DE MENSAGENS EM REDES D2D COM MÚLTIPLOS SALTOS

MICHAEL DOUGRAS DA SILVA

MELHORANDO O ENCAMINHAMENTO DE MENSAGENS EM REDES D2D COM MÚLTIPLOS SALTOS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Antônio Alfredo Ferreira Loureiro

Belo Horizonte - Minas Gerais

Junho de 2018

MICHAEL DOUGRAS DA SILVA

IMPROVING D2D MULTI-HOP MESSAGE FORWARDING

Dissertation presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Antônio Alfredo Ferreira Loureiro

Belo Horizonte - Minas Gerais

June 2018

da Silva, Michael Dougras

Improving D2D Multi-Hop Message Forwarding / Michael Dougras da Silva. — Belo Horizonte - Minas Gerais, 2018
xxii, 36 f. : il. ; 29cm

Dissertação (mestrado) — Federal University of Minas Gerais
Orientador: Antônio Alfredo Ferreira Loureiro

1. d2d. 2. multi-hop. 3. network. 4. forwarding. I. Título.

[Folha de Aprovação]

Quando a secretaria do Curso fornecer esta folha,
ela deve ser digitalizada e armazenada no disco em formato gráfico.

Se você estiver usando o `pdflatex`,
armazene o arquivo preferencialmente em formato PNG
(o formato JPEG é pior neste caso).

Se você estiver usando o `latex` (não o `pdflatex`),
terá que converter o arquivo gráfico para o formato EPS.

Em seguida, acrescente a opção `approval={`*nome do arquivo*`}`
ao comando `\ppgccufmg`.

Se a imagem da folha de aprovação precisar ser ajustada, use:
`approval=[`*ajuste*`][`*escala*`]{`*nome do arquivo*`}`
onde *ajuste* é uma distância para deslocar a imagem para baixo
e *escala* é um fator de escala para a imagem. Por exemplo:
`approval=[-2cm][0.9]{`*nome do arquivo*`}`
desloca a imagem 2cm para cima e a escala em 90%.

Dedication text goes here.

Acknowledgments

Here goes aknowledgement text. texto muito importante

“Not everything that counts can be counted, and not everything that can be counted counts.”

(William Bruce Cameron - "Informal Sociology: A Casual Introduction to Sociological Thinking")

Abstract

Abstract text goes here.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Motivation

Mobile devices have become smaller and more powerful over the years. Smartphone-based computing and communication have become ubiquitous. Connected services and applications like social networks, instant messaging, content distribution systems, and games, for example, have imposed considerable traffic growth on the mobile web. One approach to solve this problem is to improve the network infrastructure. In indoor environments, this strategy makes sense, because usually, they are smaller in the number of devices. However, in outdoor environments this approach can be unfeasible because they have a high number of devices that can vary due to people mobility.

Since improving the infrastructure can be insufficient or even unfeasible, researchers have been discussing alternative solutions. The current mobile communication model is driven by base stations, in the sense that devices must contact base stations to communicate with other devices or services. This centralized architecture has imposed a bottleneck on network evolution, so recent works have explored a new communication model in which mobile devices can bypass the base station to communicate directly with near devices. This model is called Device to Device communication or D2D [Yang et al., 2013], and it has many applications like traffic offload from base stations [Aijaz et al., 2013; Andreev et al., 2014; Bastug et al., 2014; Nunes et al., 2016c; Pyattaev et al., 2013; Yang et al., 2013], proximity-based services [Lin et al., 2014] and extended network coverage under emergency scenarios [Babun et al., 2015], for example.

1.2 The Problem

The D2D communication shows itself as a sensible solution to evolve the mobile network system and improve the devices experience. However, the direct communication nature of this model makes the regular Internet communication protocols unapplicable. One of the primary reasons for this is the absence of an end-to-end path between two nodes since messages are transmitted when there are contacts between two or more nodes and these contacts are driven by people mobility. This fact by itself makes the IP protocol routing mechanism unapplicable, which makes the routing problem one of the significant topics of study [Misra et al., 2016]. Another reason is the intermittent connections caused by nodes mobility, which makes impossible for a node to forward a packet immediately after receiving it, as is expected in the regular IP protocol.

Solutions for these problems are proposed in the context of Disruption Tolerant Networks (DTNs) [Fall, 2003]. Regular networks rely on the *store-and-forward* paradigm, in which nodes after receiving a message immediately forward it to another connected node that can help to deliver the message to the destiny. DTN networks define the *store-carry-and-forward* paradigm, in which a node after receiving a message, it stores it in a persistent buffer, and carries it until there is a proper contact to forward the message. This model makes it possible to deliver messages in D2D networks. However, in this scenario, there is no predefined end-to-end path between two nodes, which makes the regular IP routing mechanism unapplicable [Misra et al., 2016]. There are several proposals of routing algorithms in this context. The majority of them are based on flooding variations, utility functions derived from probability theorems or social context exploration.

Routing protocols based on utility functions have shown the best balance between message delivery and the number of transmissions. However, the major solutions measure utility functions at the individual level, which leads to traffic concentration on nodes with higher utility values, penalizing them [Chilipirea et al., 2013]. This is a remarkable problem because in networks that use the *store-carry-and-forward* paradigm nodes need to allocate a buffer to store messages until there is a chance to forward it. If the traffic is too high, some nodes can have problems of buffer overflow [Silva et al., 2015].

1.3 Contributions

This work has two main contributions. First, we propose a new buffer management strategy called *Space Time Drop* or ST-Drop that aims to solve the problem of buffer management under high traffic demands. The proposed solution shows a notable performance when combined with social aware and probability based routing algorithms, outperforming classic approaches. The second contribution is the definition of a distributed implementation of a new social aware routing algorithm called Groups-NET proposed in [Nunes et al., 2016c]. This algorithm does not rely on individual utility functions, which alleviates the problem of traffic concentration on some nodes. We offer a distributed algorithm for detecting and manage groups. The initial experiments show that the algorithm outperforms the BubbleRap algorithm on network overhead metric with a compatible delivery ratio.

1.4 Structure

The work is organized as follows. In chapter 2 we discuss the main approaches and algorithms to forward messages in D2D networks. This chapter presents essential concepts to understand the proposed solutions. In chapter 3 we discuss the classic approaches of buffer management and introduce the proposed algorithm ST-Drop. In chapter 4 we introduce the distributed implementation of the Groups-NET algorithm and present some initial experiments results. And in chapter 5 we conclude the work with a summary of the main findings and discuss future works to expand the research.

Chapter 2

Message Forwarding in D2D Networks

In D2D networks messages are forwarded using intermediate nodes as relays through the *store-carry-and-forward* paradigm. Nodes store messages in a persistent buffer until there is a proper contact to forward them. The big problem here is how to decide when to forward a message upon a contact. If too many copies of a message are generated, the network performance degrades because nodes energy and memory consumption get high. With fewer message copies, fewer paths are explored, so in general, the delivery probability degrades as well. This chapter presents the main forwarding protocols proposed to deal with this problem.

Forwarding protocols built on top of D2D networks can be classified in single-hop and multi-hop protocols. Single-hop protocols use only the direct contacts of a node to provide services. Multi-hop protocols use intermediate nodes to expand the service coverage. This work focus on multi-hop protocols mainly, but in this chapter, we take a quick overview of single-hop protocols too.

2.1 Single-Hop Protocols

Single hop protocols address the routing problem by exploring only the direct neighbors of a node. Protocols in this category usually explore people group or cluster formation to provide local services for data offloading. One example of a solution that uses this approach is a protocol called WiGroup [Wang et al., 2015]. This protocol defines user groups in which there is a group owner that is responsible for connecting to the base station and actuate as an intermediate access point. The other members of the group use the group owner as a relay to access network services. A similar solution is proposed in [Zheng et al., 2014] which uses social relationship analysis in the group formation algorithm.

The major benefits of these solutions are the reduced number of connections to the base station, and the possibility of defining cache strategies at the group level. Single hop protocols are a sensible solution for data offloading. They are especially good in environments with a high number of people in the same area, like in stadiums and big concerts [Wang et al., 2015]. However, in environments with high mobility, they do not perform well, because in this scenario groups are unstable. To handle these scenarios solutions based on multi-hop protocols are recommended.

2.2 Multi-Hop Protocols

Multi-hop protocols use intermediate nodes to forward messages in the network. These protocols can be used for *unicasting* communication, in which a message is forwarded from a source to a single destination node. Also, they can be used for *multicasting* communication, in which a message is forwarded from a source to multiple destination nodes. Forwarding algorithms usually explore the unicasting communication to validate the solution, because if unicasting works well, the protocol can be extended to work in multicasting mode [Misra et al., 2016].

Multi-hop protocols can be classified as single copy and multi copy. Single copy protocols, as the name suggests, keep just one copy of a message in the network as in traditional networks. First Contact and Direct Delivery are the major protocols in this category [Misra et al., 2016]. In the First Contact protocol messages are forwarded every time a node get in contact with other nodes. The message is forwarded until it reaches the destination. After forwarding a message the node drops the local copy. In Direct Delivery a node carries a message until it gets in contact directly with the destination node. These protocols are too simple and show low performance in dynamic scenarios.

Multi-copy protocols create multiple copies of each message to explore multiple paths. This approach is sensible because since there isn't a predefined path, exploring multiple paths can improve the delivery ratio. However creating a high number of message copies in the network will increase nodes' power and memory consumption, what in practice can degrade the network performance. So a good forwarding algorithm should achieve the right balance of delivery ratio and message copies overhead. Message delivery is not guaranteed is this scenario, so the delivery ratio refers to the percentage of created messages delivered to the destination. The message overhead refers to the proportional number of copies per created message. It is also important to notice that multi-hop protocols are designed to work with content that does not have strict delivery time constraints. Traditional networks have an acceptable delivery time of few milliseconds, in contrast in D2D multi-hop we consider delivery times of some days, a week or higher depending on the used approach.

The Epidemic protocol is the simplest multi-copy algorithm [Vahdat et al., 2000]. It uses the flooding strategy, in which at each encounter nodes exchange all messages they have. Considering a scenario with unlimited nodes' buffer capacity, the Epidemic achieves the upper bound delivery ratio result, at the cost of the highest network overhead. However, due to the enormous number of message copies in the network, when the buffer sizes are limited, this strategy may not achieve a high delivery ratio. There are some approaches derived from the epidemic that uses a controlled flooding strategy to reduce the number of message copies in the network and improve the performance under limited buffer capacity. The best example is the Spray and Wait protocol [Spyropoulos et al., 2005], in which the source node forwards a predefined number of message copies during its contacts. Nodes that received the message them uses the direct delivery approach to forward the message to the destination. There is also a variation called Binary Spray and Wait [Spyropoulos et al., 2005] in which the source node starts with L messages. At each encounter the source forwards $\frac{L}{2}$ copies until there is only one copy left, then it uses the direct delivery strategy. Other nodes take the same approach of the source, but they start with the number of copies received. Both approaches show good delivery ratio with low overhead, however, the authors discuss that this approach has low performance in scenarios with low mobility because nodes with only a copy will carry the message until they encounter with the destination. So they propose a modified version called Spray and Focus [Spyropoulos et al., 2007], in which nodes with only one copy left can forward the message like a single copy protocol, dropping the local copy after forwarding it. The forwarding decision is based on a utility function that uses a timer that counts the last time a node had contact with the destination. The authors have shown that Spray and Focus has better performance than Spray and Wait in scenarios with lower mobility.

More advanced protocols reduce the number of message copies adding a decision mechanism to define if a node should send a message copy or not when an encounter occurs. The Prophet protocol is a good example of such protocols [Lindgren et al., 2003]. Prophet uses nodes' contacts history to measure the probability of a given node to deliver the message. When two nodes get in contact, they will exchange only the messages for which the other node has a higher delivery probability. The basic idea in Prophet is to assign higher probabilities of meeting again to pairs of nodes that have met more recently. Through experiments, the authors show that Prophet outperforms previous solutions.

In recent years, researchers have been exploring social aware forwarding strategies. The idea is that since contacts are driven by human mobility if we can understand social relationship patterns we can improve protocols. A good example of such algorithms is Bubble Rap [Hui et al., 2011]. It is a social-aware algorithm that exploits the concept of community and network node's popularity. In this algorithm, each node is assigned to at least one social community. Social communities are defined as sets of more densely interconnected nodes (groups of nodes that have more contacts among themselves). The popularity of the nodes is measured by the number of distinct contacts a given node in the network has along the time. The basic idea of Bubble Rap is to forward the message to nodes with higher global popularity until it reaches a member of the destination communities. Then the message is forwarded based on the local popularity (node popularity considering only members of the destination community) until it reaches the destination node. Through experiments, the authors show that Bubble Rap outperforms previous solutions including the Prophet protocol. Other recent social aware protocol is the Groups-NET [Nunes et al., 2016b]. This solution explores the regularity of people encounters in groups to forward the messages. This protocol is explored in depth in chapter 4.

Chapter 3

ST-Drop: A Novel Buffer Management Strategy

In D2D multi-hop forwarding protocols nodes need to cooperate and act as relays to messages of other nodes. As a result of this cooperative behavior, each node needs to share its resources with the network allocating memory to the message in a buffer. This buffer is used to store messages being forwarded temporarily, and its capacity is limited. Consequently, the network traffic load, combined with the way a given forwarding algorithm operates, can overflow the devices' buffers. In single-copy protocols this problem is alleviated because the number of messages in the network is relatively lower. However, in multi-hop protocols this is a real problem because multiple copies of a message are spread in the network, so the traffic can grow very quickly. Besides allowing multiple copies of a message, a central entity with global network knowledge usually does not exist. As a consequence, nodes do not know when a message is delivered to the destination. Even when the message is delivered in multi copy forwarding algorithms, copies of this message remain in the network until they exceed their TTLs (time to live) or are dropped by other mechanism [Bindra and Sangal, 2012]. Therefore, a good buffer management mechanism is a critical requirement in this scenario.

This chapter presents a buffer management algorithm for D2D multi-hop and multi copy forwarding algorithms, named Space-Time Drop (ST-Drop). This algorithm explores the local information to measure the time and space coverage of the message in the network. The basic idea is that a message with higher time and space coverage is more likely to have been delivered, so it can be dropped first. Through experiments with a real mobility trace containing 115 nodes and a synthetic mobility trace containing 500 nodes, we show that ST-Drop enables routing protocols to achieve a good delivery ratio with low network overhead. With the purpose of evaluating the performance of ST-

Drop when applied to different types of opportunistic routing algorithms, we consider an epidemic based, a probabilistic, and a social-aware algorithm. Using three well-known forwarding algorithms, we show that ST-Drop helps the forwarding protocols to achieve a good delivery ratio with low overhead.

3.1 Buffer Management in D2D Networks

In D2D networks, users must devote a fraction of their devices' storage to be used as a buffer that temporarily stores the messages carried by them. The basic idea to manage such buffer, to avoid its overflow while maintaining opportunistic forwarding efficiency, is to use congestion control mechanisms. Silva et al. [2015] have classified congestion control mechanisms and provided an extensive review of message drop policies used in opportunistic networks. In this section, we go over some of the state-of-art message drop policies highlighting their specific features.

Lindgren and Phanse [2006] proposed the Evict Most Forwarded (MOFO) drop policy, which uses the idea that a message forwarded a large number of times has a higher probability to be delivered, even if it is dropped locally. They also proposed the Evict Shortest Life Time First (SHLI) drop policy, which is based on the idea that it makes more sense to spend resources on messages that have higher TTL, because, intuitively, these messages have a higher probability to be delivered. The Drop Largest policy [Rashid and Ayub, 2010] selects big size messages to drop first, because, by doing this, more space is freed from the buffer. Another basic policy is FIFO, in which, as the name suggests, messages that arrived first are dropped first.

Rashid et al. [2013] proposed the Message Drop Control Source Relay (MDC-SR) buffer management algorithm, in which nodes stop receiving incoming messages when the buffer occupancy achieves a defined upper bound, except when the node is the destination of the message. With this approach, the algorithm decreases the number of message drops in the network. Through experiments, they showed that MDC-SR optimized the performance of Epidemic, Prophet and First Contact routing algorithms regarding delivery ratio and network overhead.

Krifa et al. [2008] proposed the Global Knowledge Based Drop (GBD-Drop) buffer management algorithm, in which they first assume a global knowledge of the network to decide which messages to drop, achieving optimal performance. They proposed a distributed version of the algorithm that uses statistic learning to approximate the global knowledge of the network. Through experiments, they showed that the centralized and distributed versions of the algorithm outperform the basic drop policies. The

GDB-Drop can also be configured to achieve a better average delivery delay or better average delivery ratio.

Rashid et al. [2011] proposed the E-Drop buffer management algorithm, in which messages with size greater or equal to the incoming message are dropped first. In the case of an incoming message with a size greater than the messages in the buffer, it works like the FIFO policy. This policy minimizes the drop of messages because it will usually remove only one message from the buffer to receive the incoming message. Through simulations, using mobility models, they show that E-Drop outperforms the MOFO algorithm.

Li et al. [2009] proposed the N-Drop buffer management algorithm, in which messages transmitted more than a predefined constant are dropped first. Through simulations, they showed that this algorithm outperforms FIFO when used with Epidemic routing algorithm. Ayub and Rashid [2010] also proposed the T-Drop buffer management algorithm, in which messages with size in a predefined range T are dropped first. Through simulations, the authors show that this algorithm outperforms FIFO when applied to the routing algorithms Epidemic and Prophet.

Naves et al. [2012] proposed two new buffer management policies. The first one is named Least Recently Forwarded (LRF), and it drops messages that were forwarded more recently. The second one is named Less Probable Sprayed (LPS), and it drops messages that have less probability to be delivered first. Through experiments with Epidemic and Prophet routing algorithms using real-data traces, the authors showed that both algorithms outperform FIFO and MOFO in delivery ratio and overhead.

The aforementioned buffer management algorithms are based on only one local metric, for example, buffer time, forward count, TTL, and message size. These metrics represent a good choice because they are independent of the routing algorithm. However, their isolated meaning can be insufficient to decide whether to drop a message or not. We have combined basic metrics to create a new buffer management algorithm named Space-Time Drop (ST-Drop). In the following section, we describe the ST-Drop algorithm.

3.2 The ST-Drop Algorithm

The ST-Drop formulation starts with the idea that a message with a greater space and time coverage in the network has a greater probability to have been already delivered, so it can be dropped first. Based on this idea, we have defined a space coefficient S_c that measures the space coverage of the message in the network, and the time coefficient T_c

that measures the time coverage of the message in the network. Therefore, to compute the space-time coverage ST_c, we use the simple function:

$$ST_c = S_c \cdot T_c. \tag{3.1}$$

With this first formulation, we do not define how to compute S_c and T_c. Buffer management algorithms for D2D networks are ideally implemented in a distributed way, so we have to be restricted to local information. Besides this, we want to keep ST-Drop independent of the routing algorithm. Consequently, we decided to combine the metrics used by the basic algorithms to compute our defined metrics S_c and T_c.

The first issue is how to measure space coverage. In cellular networks, the contacts are driven by human mobility. Intuitively we can say that a message carried by only one person will have fewer opportunities to be forwarded than a message carried by two or more people because it will be restricted to the mobility of only one person. Thus, we can say that the number of devices carrying a message is a sensible measure of the space coverage of the message, because the space coverage of a message is a direct result of the combination of the devices' mobility. Therefore, we use the same metric used by the MOFO algorithm to measure the space coverage, i.e., the number of times a message was forwarded by a node. The forward counter is a local measure of the number of devices carrying a message.

The second issue is how to measure the time coverage. FIFO exploits the time the messages have been carried by a node to decide which message to drop first. Using only this metric we could drop a recently created message and let a message with almost ending TTL in the buffer. SHLI uses the TTL of messages to decide which message should be dropped first. Using only this metric we could let a message with a huge TTL indefinitely in the buffer of a node that has a low delivery probability for it. Also, in a real scenario, messages may have different TTL values. Thus, messages with relatively low initial TTL would be penalized. Therefore, ST-Drop combines the idea of FIFO and SHLI, normalizing the carrying time CT, that is the time a message is stored in the buffer of a node, with the TTL of the message, defining T_c as:

$$T_c = \frac{CT}{TTL}. \tag{3.2}$$

With this formulation, we can now compute the space-time coverage using only local information. The ST-Drop policy is formalized in Algorithm 1. ST-Drop is called only when there is no available space in the buffer to receive an incoming message. The algorithm sorts the messages in reverse order by their ST_c values and stores them in a list. The messages with ST_c equals 0 are removed from the list, which implies that they will never be dropped. The algorithm considers that dropping a message with space-time equals 0 is not fair, because the message was not forwarded so far. Then, the first message of the list is dropped until enough space is freed for the new message or the list becomes empty. If enough space is freed, then the algorithm returns a confirmation to receive the incoming message. Otherwise, the device must reject the incoming message.

Algorithm 1 ST-Drop algorithm

Input: incomingMessage
Input: buffer
Output: receive
if incomingMessage.size > buffer.capacity **then**
    return false
end if
Sort the messages in reverse order by ST_c :
msglist = sort(buffer)
Remove messages with $ST_c = 0$
msglist.filter()
Drop messages until enough space is freed
incsize ← incomingMessage.size
available ← buffer.availablespace
while available < incsize and not msgList.empty **do**
    msg ← msglist.removefirst()
    buffer.drop(msg)
    available ← available + msg.size
end while
if available < incsize **then**
    return false
else
    return true
end if

3.3 Simulation Methodology

In this section, we describe the simulations we have performed to evaluate ST-Drop. Firstly, we describe the metrics for D2D opportunistic communication that we have evaluated, and then we present the opportunistic routing algorithms in which we have applied ST-Drop. Next, we describe the two mobility traces used to emulate the network nodes' mobility and their proximity contacts, and finally we discuss the parameters and configurations we used in our experiments.

3.3.1 Metrics

With the goal of comparatively evaluating ST-Drop with other message drop policies, we used the following traditional opportunistic network metrics:

- **Delivery ratio:** evaluates the percentage of successfully delivered messages along the time;
- **Network Overhead:** measures the number of retransmissions per created message in the network, i.e., the number of D2D transmissions that each algorithm performs along the time normalized by the number of created messages.

Message drop policies for D2D networks should achieve cost-effective delivery considering these metrics, i.e., the highest possible delivery ratio with the lowest possible network overhead. Successfully delivered messages are those which the base station will not need to deliver itself, thus using less bandwidth. A high number of message re-transmissions (high overhead) may negatively impact the users' experience by, for example, increasing devices' energy expenditure.

3.3.2 Opportunistic Forwarding Algorithms

We used algorithms based on three different approaches to study the behavior of our solution when integrated with different routing algorithms. The first routing algorithm is the Epidemic [Vahdat et al., 2000], also known as Flooding. This algorithm is considered one of the most basic ones as it is not based on utility functions or similar metrics. In this algorithm, at each encounter between two nodes, they exchange all the messages they have in their respective buffers. The second routing algorithm that we considered is the Prophet [Lindgren et al., 2003]. This algorithm uses a delivery probability as a utility function to decide whether to forward a message or not. And the third is the Bubble Rap [Hui et al., 2011]. It is a social-aware algorithm based on the concept of community and network node's popularity. These algorithms are presented with more details in chapter 2.

We are using routing algorithms based on epidemic routing, probabilistic functions, and social-aware strategies. In this way, we can evaluate the impact of different types of routing algorithms when using ST-Drop.

3.3.3 Mobility Traces

We have used two mobility traces to emulate nodes' mobility. The first one is the NCCU trace [Tsai and Chan, 2015], a real-world dataset that monitors the mobility of 115 students inside a campus for 15 days. The second one is a synthetic trace generated by the Small World in Motion (SWIM) mobility model [Kosta et al., 2010]. This trace is based on a state-of-the-art mobility model that captures the existence of social communities influencing human mobility, and it simulates the mobility of 500 people for 11 days. We have chosen the two traces to test ST-Drop under different network scales.

3.3.4 Execution

We used The ONE simulator [Keränen et al., 2009] to emulate the execution of the opportunistic routing algorithms using different message dropping policies. It is a discrete event simulator focused on opportunistic networks. By default, the simulator does not support custom buffer management algorithms and the Bubble Rap routing algorithm, therefore we implemented these features[1].

To test the ST-Drop, we defined the buffer capacity and a network traffic model. As discussed in [Grasic and Lindgren, 2012], there is no pattern when it comes to defining a traffic model to test solutions in this scenario. Thus, we chose sensible values in our simulation. The buffer capacity was defined as 1GB. Messages have seven days of TTL and are generated at random times within the trace duration. Message sizes are chosen from the interval [50MB,100MB] with uniform probability. The source and destination pair for each created message is also selected with uniform probability among the nodes in the network.

We have defined two network traffic load levels based on the number of nodes of each scenario. The first load level generates messages equals to 50% of the number of nodes at each day, and the second generates 100%. Table 31 summarizes the number of messages generated in each traffic load level:

Table 31: Network Traffic Load Scenarios

Scenario		**Messages/Day**	**Total**
NCCU	50	58	522
	100	115	1035
SWIM	50	250	1250
	100	500	2500

We have emulated each scenario with each traffic level 10 times using different seeds to generate message's sources and destinations, and messages size. The average

[1] The source code is available at https://github.com/micdoug/the-one/tree/58e207ac44d5012fac5d103da781d30266164216

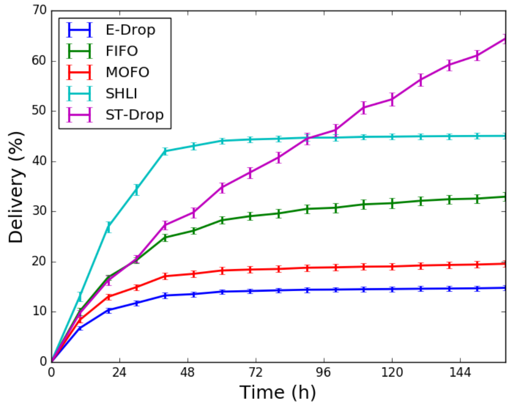
delivery ratio and the average network overhead of each scenario were computed and presented with 95% confidence intervals.

3.4 Results

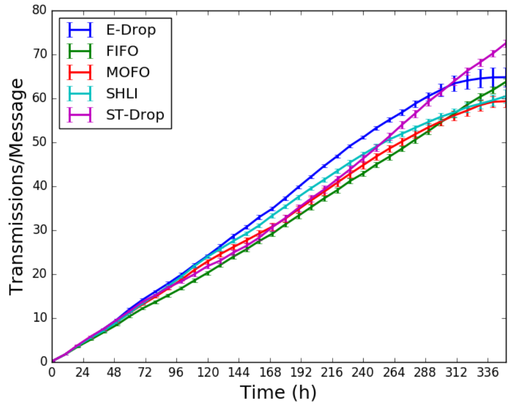
The simulation results for NCCU100 and NCCU50 are presented in Figures 31 and 32, respectively. The results of SWIM100 e SWIM50 are expressed in Figures 33 and 34, respectively. The delivery ratio values were obtained considering the time spent to delivery each message since its creation time. The result is presented in a way that it can be analysed with different TTL values, for example, considering only points under the time value 48 in the x axis, we will get the delivery ratio with TTL of 48 hours.

Considering the Epidemic routing, the delivery ratio in the NCCU100 and NCCU50 scenarios is expressed in Figures 31a and 32a, respectively. We can see that the delivery ratio of ST-Drop policy is higher than E-Drop and MOFO for almost all the simulation time. It is also higher than FIFO for time values higher than 48 hours and is higher than SHLI for time values higher than 120 hours. We can also see that the different traffic loads have not changed the behavior of the policies. The delivery ratio changed proportionally for the tested policies. The overhead results of NCCU100 and NCCU50 are expressed in Figures 31b and 32b, respectively. In these two figures, we can see that the overhead results in the two traffic levels have similar behavior. All drop policies have overhead values very similar for almost all the simulation time. But we can see that in the last four days, ST-Drop continues to grow while others get small values. This is happening because in the last four days the delivery probability of ST-Drop continues to grow too, while other policies do not, consequently the overhead increase is expected in Epidemic when more messages are delivered. The delivery ratios in the SWIM100 and SWIM50 scenarios are expressed in Figures 33a and 34a, respectively. In these figures, we can see that all drop policies have similar results for delivery ratio considering the confidence interval. And, as in NCCU scenario, with both traffic levels, we can see a similar behavior. The overhead in the SWIM100 and SWIM50 scenarios is expressed in Figures 33b and 34b, respectively. Again, we can see similar behavior under the two traffic levels.

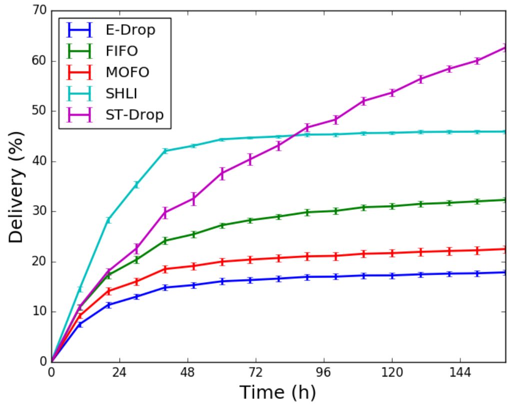
Considering the Prophet algorithm, the delivery ratio in the NCCU100 and NCCU50 scenarios is expressed in Figures 31c and 32c, respectively. We can see that the delivery ratio of ST-Drop is higher than FIFO, MOFO and E-Drop for almost all time values, and is higher than SHLI for time values higher than 120 hours. Again, we can see that under different traffic levels the results show similar behavior. The



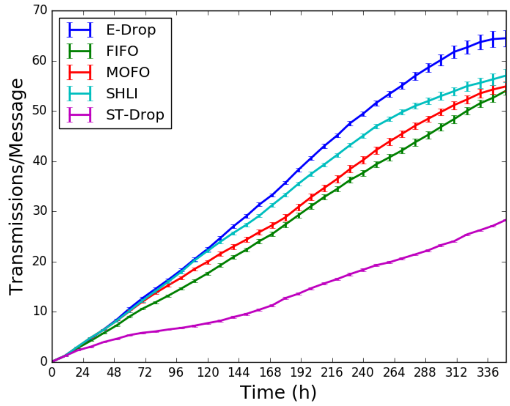
(a) NCCU100 - Delivery Epidemic



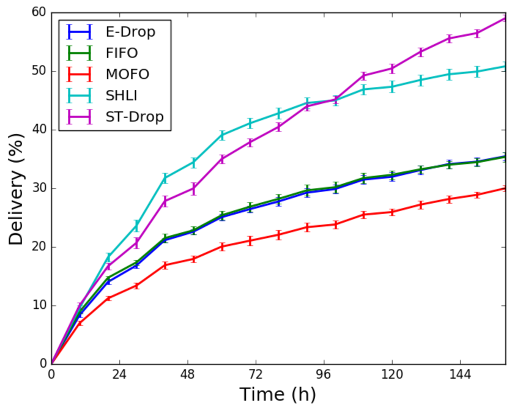
(b) NCCU100 - Overhead Epidemic



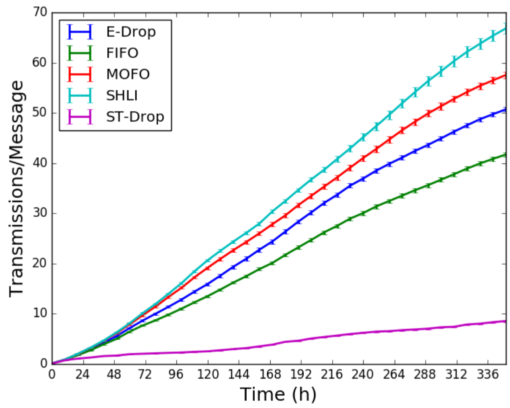
(c) NCCU100 - Delivery Prophet



(d) NCCU100 - Overhead Prophet

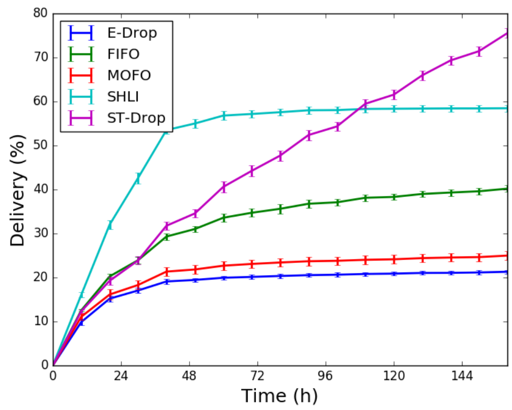


(e) NCCU100 - Delivery Bubble

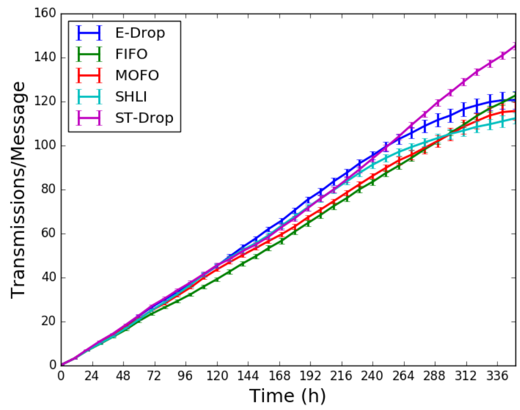


(f) NCCU100 - Overhead Bubble

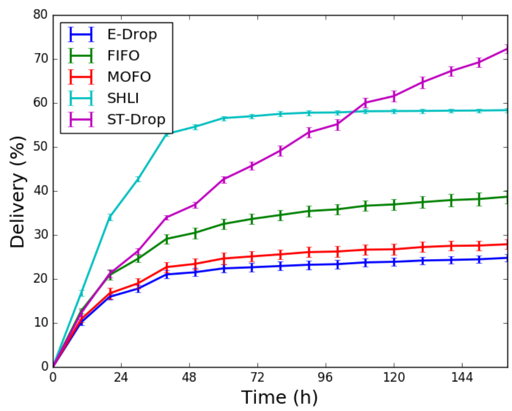
Figure 31: Comparison of message drop policies in the NCCU trace with 100% traffic



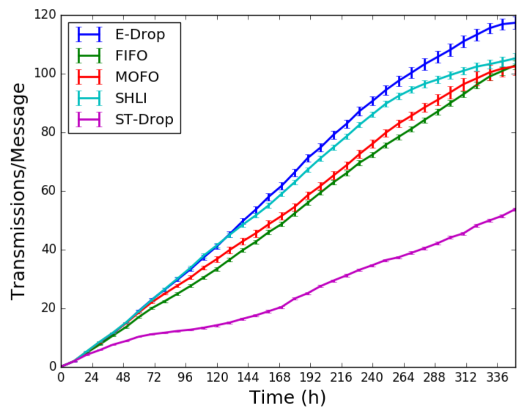
(a) NCCU50 - Delivery Epidemic



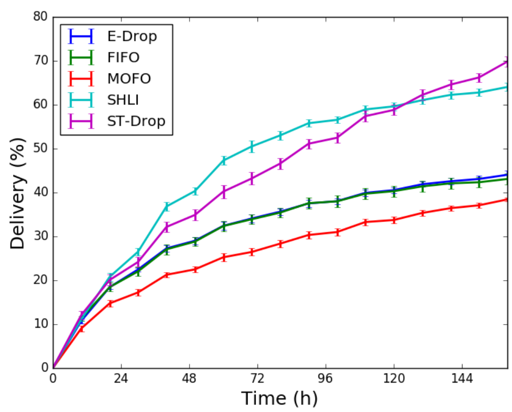
(b) NCCU50 - Overhead Epidemic



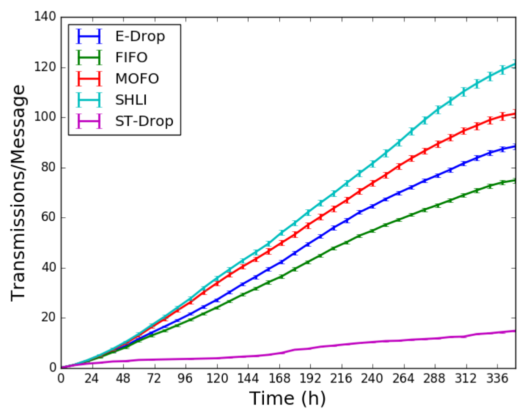
(c) NCCU50 - Delivery Prophet



(d) NCCU50 - Overhead Prophet

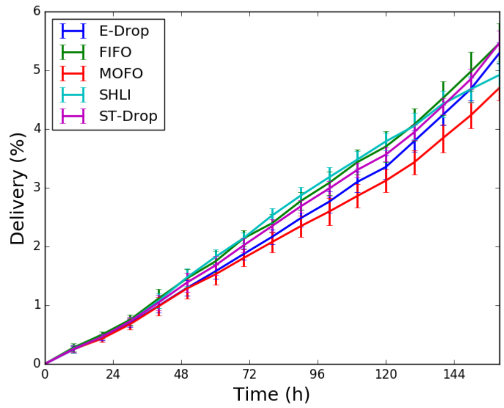


(e) NCCU50 - Delivery Bubble

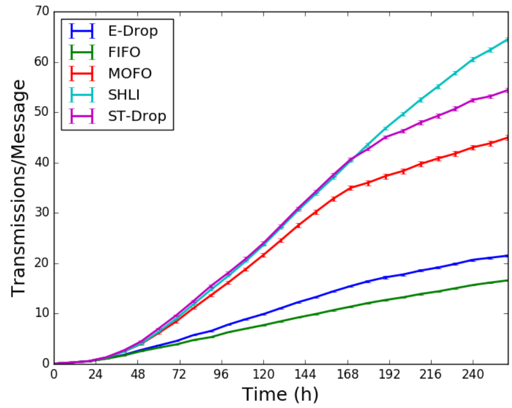


(f) NCCU50 - Overhead Bubble

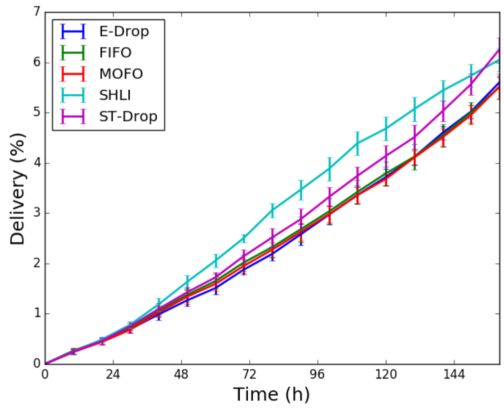
Figure 32: Comparison of message drop policies in the NCCU trace with 50% traffic



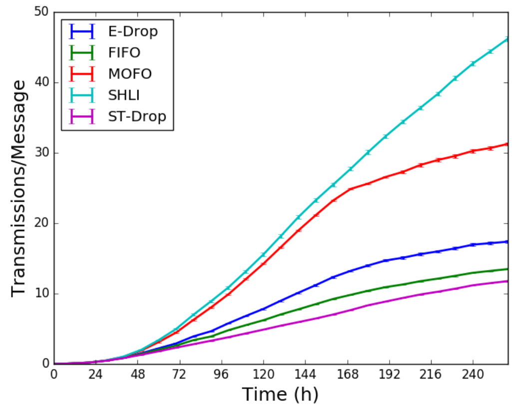
(a) SWIM100 - Delivery Epidemic



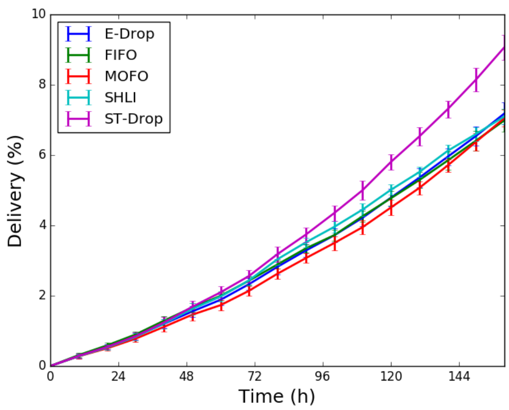
(b) SWIM100 - Overhead Epidemic



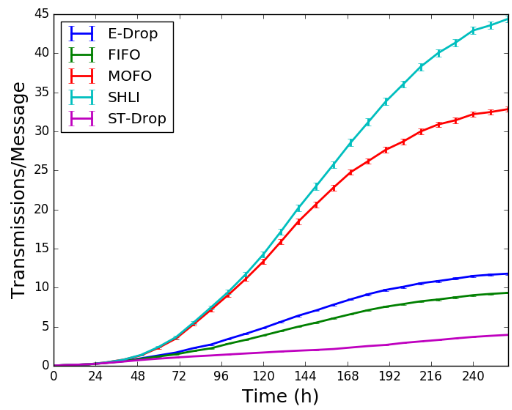
(c) SWIM100 - Delivery Prophet



(d) SWIM100 - Overhead Prophet

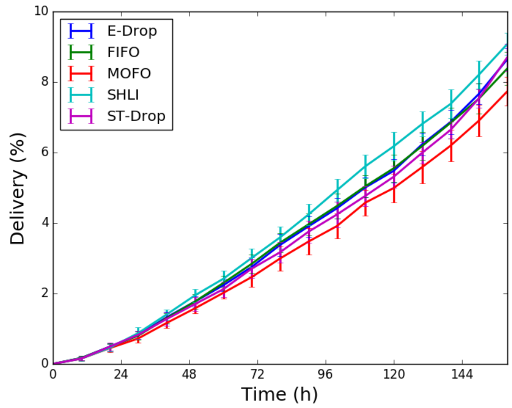


(e) SWIM100 - Delivery Bubble

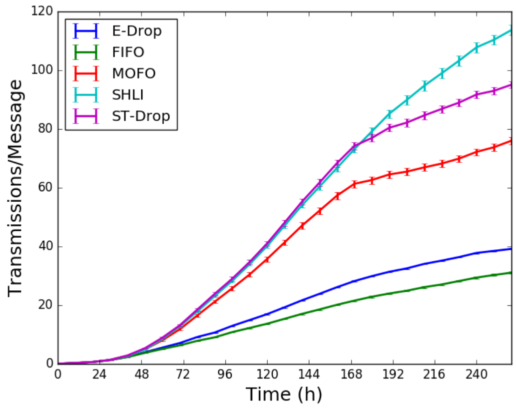


(f) SWIM100 - Overhead Bubble

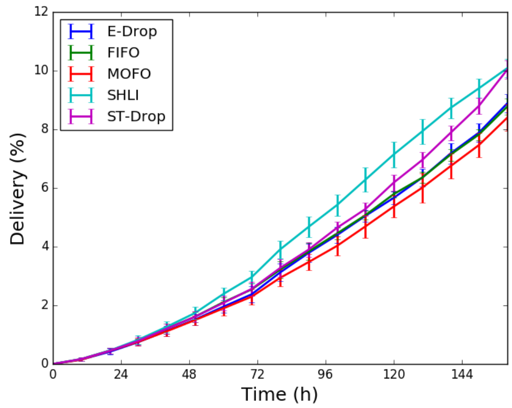
Figure 33: Comparison of message drop policies in the SWIM trace with 100% traffic



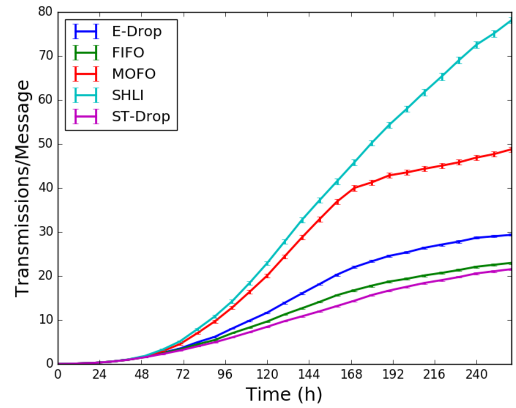
(a) SWIM50 - Delivery Epidemic



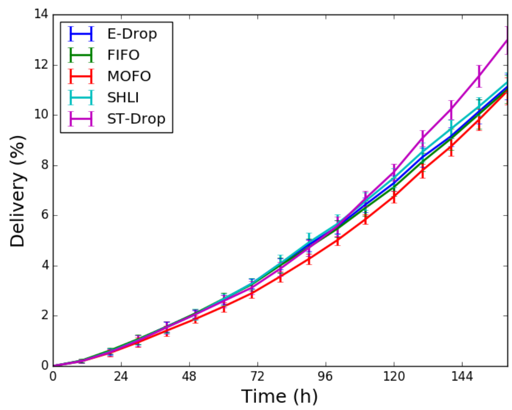
(b) SWIM50 - Overhead Epidemic



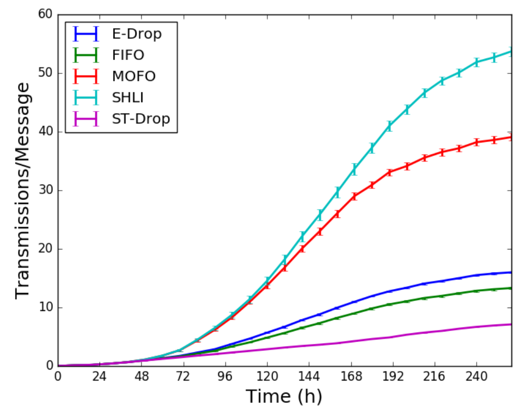
(c) SWIM50 - Delivery Prophet



(d) SWIM50 - Overhead Prophet



(e) SWIM50 - Delivery Bubble



(f) SWIM50 - Overhead Bubble

Figure 34: Comparison of message drop policies in the SWIM trace with 50% traffic

overhead in the NCCU100 and NCCU50 scenarios is expressed in Figures 31d and 32d, respectively. We can see again that the results show similar behavior for both traffic levels. In this case, ST-Drop obtained the best results during all simulation time. The delivery ratio in the SWIM100 and SWIM50 scenarios is expressed in Figures 33c and 34c, respectively. Again, we see similar behavior under both traffic levels. The delivery ratio is similar for all drop policies, except for time values between 72 and 144 hours where SHLI obtained slightly better results. The overhead in the SWIM100 and SWIM50 is expressed in Figures 33d and 34d, respectively. In this case, we obtained the same behavior of NCCU scenarios, in which ST-Drop obtained the best results during all simulation time.

Finally, we analyze the policies when used with Bubble Rap routing. The delivery ratio in the NCCU100 and NCCU50 scenarios is expressed in Figures 31e and 32e, respectively. We can see similar behavior under both traffic levels. The ST-Drop obtained a higher delivery ratio than FIFO, E-DROP and MOFO for almost all time values, and higher values than SHLI for time values higher than 120. The overhead in the NCCU100 and NCCU50 scenarios is expressed in Figures 31f and 32f, respectively. As with Prophet routing, in this scenario the ST-Drop obtained the best results during all simulation time, and we can see similar behavior under both traffic levels. The delivery ratio in the SWIM100 and SWIM50 scenarios is expressed in Figures 33e and 34e, respectively. Again, we see similar behavior under both traffic levels. The delivery ratio is similar for all drop policies, but ST-Drop obtained slightly better results for time values greater than 120 hours. The overhead in the SWIM100 and SWIM50 is expressed in Figures 33f and 34f, respectively. In this case, we obtained the same behavior of NCCU scenarios, in which ST-Drop obtained the best results during all simulation time.

In general, the ST-Drop policy obtained better delivery ratio values for all the three routing algorithms. For the overhead ratio, ST-Drop obtained the best results in all scenarios with Prophet and Bubble Rap routing algorithms. The SHLI policy obtained better results with low TTLs because in this policy, messages that spent more time in the network are dropped first, so it is expected that most delivered messages, when using this policy, have low delivery time. But when we consider all TTL range, ST-Drop obtained better results than SHLI.

It is worth mentioning that, in the Prophet algorithm, the ST-Drop overhead was much lower when compared to the other strategies. With respect to Bubble Rap, this result is even better, since ST-Drop reaches overhead values that are three times smaller than the second-best policy. This latter observation is particularly important because, as of today, social-aware algorithms such as Bubble Rap are considered the state-of-art for cost-effective opportunistic routing. This assertion can be reassured by contrasting the delivery ratio and overhead values for Epidemic, Prophet, and Bubble Rap. In both mobility traces, we can see that Bubble Rap achieves the best delivery ratio and the lowest overhead, especially when used together with ST-Drop.

3.5 Conclusion and Future Work

This work proposed a new buffer management algorithm for D2D opportunistic networks named Space-Time-Drop (ST-Drop). This algorithm uses the idea that a message with more space and time coverage can be dropped first, because it is more likely that such a message has already been delivered. We have combined the idea of basic drop policies to measure the space and time coverage of a message using local information. Therefore, our solution can be easily deployed in distributed environments. We evaluated our solution with simulations in Epidemic, Prophet, and Bubble Rap routing algorithms using three different traffic loads with two different mobility traces. We showed that our solution achieves higher delivery ratio with the three routing algorithms and much lower overhead values with Prophet and Bubble Rap. Considering all combinations of the evaluated opportunistic routing algorithms and buffer management policies, the ST-Drop combined with Bubble Rap achieved, in all experiments, the best cost-effectiveness, i.e., the highest delivery ratio with the lowest network overhead. This result shows that ST-Drop significantly contributes to improving the cost-effectiveness of opportunistic D2D Networks.

As future work, we plan to apply ST-Drop to other opportunistic D2D routing algorithms. Since ST-Drop exhibited good performance in social-aware algorithms, it is interesting to apply it in other recent social-aware routing strategies, such as GroupsNet Nunes et al. [2016b]. We also highlight the possibility of exploring more metrics to measure space and time coverage. Information obtained from the device's context, such as battery information and actual spatial information (GPS), could be used to improve ST-Drop.

Chapter 4

Groups NET Distributed Implementation

In D2D networks communication is established when a contact occurs, and the contacts are driven by human mobility. Therefore understanding human mobility and how people interact is a fundamental requirement to create suitable routing algorithms. Recent works have shown that the best routing solutions for D2D communication are those based on social context, mainly those that explore how people interact in groups or communities. In this chapter we propose a distributed implementation of Groups-NET, a forwarding algorithm based on group encounters [Nunes et al., 2016c]. We propose an algorithm for group detection and tracking in a distributed environment and used it to implement Groups-NET. Through experiments with a real mobility trace containing 115 nodes we show that the solution achieves a good delivery ratio with an expressive lower network overhead when compared with the state-of-art BubbleRap algorithm.

<Write down the chapter structure when finished>

4.1 The Groups NET Algorithm

Groups-NET is a multi hop and multi copy forwarding algorithm for D2D networks that explores people group meetings to propagate content [Nunes et al., 2016c]. A group is defined as a set of people that are placed near each other in a given time due to a common goal. For example, people working in the same company or students attending the same class. The algorithm is based on the idea that group encounters show some regularity along the time, because in general people have regular schedules and routines. Therefore we can explore group encounters to forward messages. In their previous work, Nunes et al. [2016a] proposed an algorithm for group detection and tracking, in which detected groups show encounters regularity along the time, mainly on daily and weekly basis.

This algorithm operates on contact traces and work as follows. First, the contact trace is split considering time windows of a predefined size. Contacts in each time window are used to create contact graphs, in which vertices denotes devices and edges denotes contacts in that time window. Then a sequence of graph contacts along the time is created, representing people contacts. For each of these graphs the clustering algorithm Clique Percolation [Derényi et al., 2005] is applied to find clusters based on the contacts of that time interval. A cluster in a contact graph represents a group encounter in a that time window. In order to find groups that have encounters in different time windows the authors introduced the *Group Correlation Coefficient* metric, which measures the similarity of two group captured in distinct contact graphs. This metric is defined as the number of intersection members of the two groups divided by the number of members of the union of the groups, therefore returning a value between 0 and 1. Groups detected in distinct time windows with *Group Correlation Coefficient* greater or equals 0.5 are considered as the same group with multiple encounters. So after applying the group detection algorithm we have a set of groups with its encounters along the analysed time.

Groups-NET forwards messages using the most probable group-to-group route based on the output of the group detection algorithm. Routes are computed using a graph in which groups are the vertices and there is an edge between each possible pair of groups. Edges are weighted considering the group correlation coefficient between the two groups and the probability of each group of encountering in the near future. The group correlation coefficient is expressed in equation 4.1. The probability of a group encountering is defined in equation 4.2, in which TTL represents the time before a message is expired (we are interested in the probability of a group encountering before a given message has TTL expired) and N represents the number of times a group has met considering the last T hours. Based on these two values the edge weight is computed according to equation 4.3.

$$GC_{G_1G_2} = \frac{(G_1 \cap G_2)}{(G_1 \cup G_2)} \tag{4.1}$$

$$P_G = 1 - e^{(-\frac{TTL*N}{T})} \tag{4.2}$$

$$E_{G_1G_2} = -\log(GC_{G_1G_2} * P_{G1} * P_{G2}) \tag{4.3}$$

With the groups graph defined it is possible to find the route to forward a message. Given a message with defined source and destination nodes, we compute shortest paths for each pair of groups ($G_s \rightarrow G_d$), in which G_s is a group that the source is present and G_d is a group that the destination node is present. From this list of paths the shortest one is chosen as the most probable group-to-group path. The message is then forwarded upon a contact if the other node is present in one of the groups of the chosen path. Thought experiments, the authors show that Groups-NET outperforms Bubble-Rap in terms of network overhead. However, this solution relies on the global network knowledge to detect and track groups, which is unfeasible in real scenarios. In the next section we go over a proposed solution to detect groups in a distributed environment. We then use this definition later combined with the original forwarding decision mechanism of Groups-NET to build the distributed solution.

4.2 Mobile Group Detection

The mobile group detection algorithm proposed here is derived from the ideas presented in [Nunes et al., 2016a]. In the original proposal the authors use a approach to detect groups based on global network knowledge. They basically build a contact graph considering a predefined time interval, and for each graph is used the Clique Percolation community detection algorithm. Each community detected by the algorithm is considered a group meeting in that time window. Then a metric called *Group Correlation Coefficient* is used to detect instances of the same group across multiple time windows. With this approach the authors show that groups usually have regular encounters along the time (specially in daily and weekly basis). But they also discuss that the detection approach is unfeasible for a distributed environment, because it assumes a global network knowledge.

However, authors also discuss that implementing a distributed solution should be a simple task, because in the local scope nodes can use neighborhood discover strategies and process regular neighbors encounters to decide their group encounters. In this work we expanded this idea to build a distributed solution for group detection. It is composed of four processes that are executed concurrently by each node using basically neighborhood inspection.

The first process is the **device's local group detection**. This process is responsible of inspecting a node's local contacts and decide when a group meeting happens. The algorithm is very simple. Each node keeps two lists of devices. The first is the *friends list* (*FL*) that keeps track of current nodes considered as friends, i.e, members of a recent group encounter. The second list is the *strangers list* (*SL*), that keeps track of recent nodes' contacts that are not yet considered as friends (random contacts, for example). Each entry in these lists have a contact counter that keeps track of consecutive contacts, and also have a inactive counter that keeps track of consecutive periods of time without contact with that node. Each node inspects its neighborhood at each predefined *time interval* and based on current neighbors it updates its friends and strangers list. This update is based on two predefined parameters: the *friend threshold* and *inactive threshold*. Based on these definitions this process is executed as follows:

- At each time interval each node collects its current neighbors.

- Each node use its current neighbors to update the counters of friends list members. Current neighbors that are present in the friends list have their contact counters incremented by one and its inactive counter set to zero. Current friends that are not a current neighbor have their contact counters set to zero and their inactive counter incremented by one.

- A similar update is made to strangers list. Current neighbors that are present in the strangers list have their contact counter incremented by one and their inactive counter set to zero. If a stranger contact counter reaches the defined *friend threshold*, the stranger is promoted to a friend and moved to the friends list. Current strangers that are not a current neighbor have its inactive counter incremented by one and its contact counter set to zero. If a stranger inactive counter reaches a *inactive threshold* the node is removed from strangers list.

- Current neighbors that are not in friends and strangers list are added in the strangers list with their contact counter set to one and inactive counter set to zero.

- As a final step, the number of inactive nodes in the friends list that are checked. A node is considered inactive when the inactive counter reaches the *inactive threshold*. If more than 50 percent of the nodes in the friends list are inactive, the friend list is archived and considered as a group meeting that happened in the past. When this happens, both friends and strangers list are clear and the algorithm is restarted.

In the end this algorithm creates a list of local detected group meetings. This list is used by the **local group combination** process to detect instances of the same group with multiple encounters along the time. In this process each mobile device analyzes groups discovered by the previous process to define groups that have multiple encounters along the time. The algorithm is based on the *Group Correlation Coefficient* metric that measures the proportion of nodes shared between two sets. Each node keeps a list of *combined groups*, that keeps track of current known groups with multiple encounters along the time. For each local group detected in the previous process, it checks if there is a combined group with *Group Correlation Coefficient* greater or equals 0.5. If there is, the local group is merged into the combined group adding its members to it and registering a new encounter. If there isn't, a new combined group is initialized with the local group data.

The combined groups generated are then used by the **neighborhood inspection** process to compose the final groups considered in the forwarding algorithm. This process is responsible of at each considered time interval collecting a node's neighbors combined groups. The collected groups are then merged with the local combined groups using the same approach described previously, in which groups that have correlation coefficient greater of equals 0.5 are merged. The purpose of this process is to expand the nodes global knowledge. The list of combined groups are then used for the **groups graph creation** process to create the graph used by the forwarding decision mechanism of Groups-NET. The next section describe the experiments used to validate the proposed solution.

4.3 Simulation Methodology

In order to validate the proposed solution we executed some message forwarding simulations comparing the performance of distributed Groups-NET against the performance of the distributed version of Bubble Rap [Hui et al., 2011] based on the distributed community detection algorithm presented by Hui et al. [2007]. Both forwarding algorithms were implemented[1] as an extension to the ONE (Opportunistic Network) simulator. The distributed group detection algorithm presented in the previous section was implemented as an external script[2] that generates the group graph structure and give as input to the ONE simulator. The reason of this is that adding parallelism in the ONE simulator would require modification on the core of the simulator, and without parallelism the simulation performance highly degrades due to the massive groups graph manipulation requirement.

We have used for simulation the NCCU mobility trace [Tsai and Chan, 2015], a real-world dataset that monitors the mobility of 115 students inside a campus for 15

[1]The code is available at https://github.com/micdoug/the-one/tree/groups-net.
[2]The code is available at https://github.com/micdoug/mgb.

days. We considered the entire trace duration to run the group detection algorithm for Groups-NET and community detection algorithm for BubbleRap. And we used the second week to simulate message forwarding. We decided to use the 15 days for the group detection algorithm to capture the weekly regularity of group encounters, that is presented in the results of next section. Another reason is that the to compute the groups graph edges weight we need to consider a message TTL and the number of encounters in the last T hours, so we defined T as the first seven days and the message TTL as the next 7 days.

To compare the two forwarding algorithms we evaluated the delivery ratio and network overhead. The delivery ratio evaluates the percentage of delivered messages along the time. The network overhead evaluates the number of retransmissions per created message in the network, i.e., the number of D2D transmissions that each algorithm performs along the time normalized by the number of created messages. Forwarding algorithms seek for the highest delivery ratio with the lowest possible network overhead. Successfully delivered messages are those which the base station will not need to deliver itself, thus using less bandwidth. A high number of message re-transmissions (high overhead) may negatively impact the users' experience by, for example, increasing devices' energy expenditure.

The parameters used for simulation are summarized in table 41. In the next section we present the results of group detection and message forwarding using the proposed solution.

Table 41: Simulation Parameters

Parameter	**Value**
friend threshold	10
inactive threshold	10
scan time interval	60 seconds
message TTL	7 days
T (past encounters)	7 days
number of messages	5000

4.4 Results

We executed the mobile group detection on NCCU trace using using the parameters presented in table 41. Other combination of parameters were used, however this one has showed the same behavior of the centralized version in terms of group reencounters distribution. The figure 41 shows the distribution of group reencounters. The group

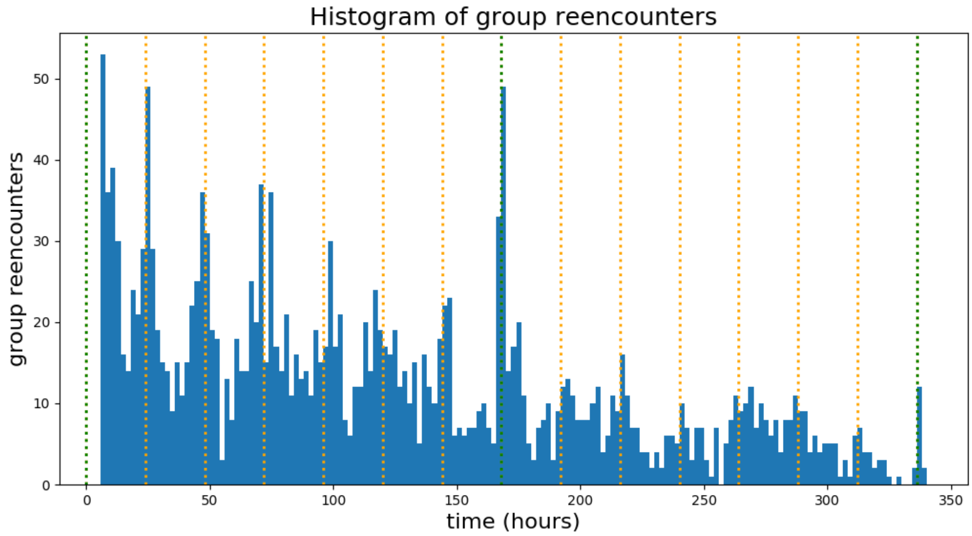
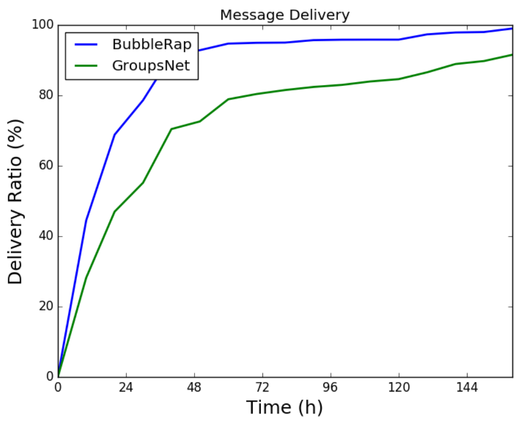


Figure 41: Group reencounters distribution results of the distributed group detection algorithm

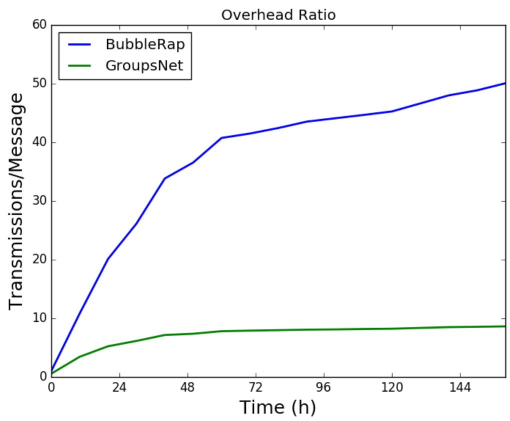
reencounters distribution shows spikes on intervals of 24 hours (dotted orange lines) and 7 days (green dotted lines). So this shows that the groups collected from the distributed algorithm captures similar characteristics of the centralized version. After validating this, we used these detected groups to run the message forwarding simulation.

Figure 42 shows the message forwarding simulation results. Groups-NET has achieved a lower message delivery ratio, about 15 percent at the end of the simulation when comparing it with Bubble Rap. However the message overhead achieved by Groups-NET is more than 4 times lower than Bubble Rap at the end of simulation, a value expressively better. It is also important to notice that the message overhead ratio of Groups-NET have established from the second day of simulation until the end, in contrast Bubble Rap number of copies continues to increase.

4.5 Conclusion



(a) Delivery ratio results



(b) Message overhead results

Figure 42: Comparison of distributed Groups-NET and BubbleRap on NCCU trace

Chapter 5

Conclusion

Put some text here

Bibliography

Aijaz, A., Aghvami, H., and Amani, M. (2013). A survey on mobile data offloading: technical and business perspectives. *IEEE Wireless Communications*, 20(2):104--112.

Andreev, S., Pyattaev, A., Johnsson, K., Galinina, O., and Koucheryavy, Y. (2014). Cellular traffic offloading onto network-assisted device-to-device connections. *IEEE Communications Magazine*, 52(4):20--31.

Ayub, Q. and Rashid, S. (2010). T-drop: An optimal buffer management policy to improve qos in dtn routing protocols. *Journal of Computing*, 2(10):46--50.

Babun, L., Yürekli, A. İ., and Güvenç, I. (2015). Multi-hop and d2d communications for extending coverage in public safety scenarios. In *Local Computer Networks Conference Workshops (LCN Workshops), 2015 IEEE 40th*, pages 912--919. IEEE.

Bastug, E., Bennis, M., and Debbah, M. (2014). Living on the edge: The role of proactive caching in 5g wireless networks. *IEEE Communications Magazine*, 52(8):82--89.

Bindra, H. S. and Sangal, A. (2012). Need of removing delivered message replica from delay tolerant network-a problem definition. *International Journal of Computer Network and Information Security*, 4(12):59.

Chilipirea, C., Petre, A.-C., and Dobre, C. (2013). Energy-aware social-based routing in opportunistic networks. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pages 791--796. IEEE.

Derényi, I., Palla, G., and Vicsek, T. (2005). Clique percolation in random networks. *Physical review letters*, 94(16):160202.

Fall, K. (2003). A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27--34. ACM.

Grasic, S. and Lindgren, A. (2012). An analysis of evaluation practices for dtn routing protocols. In *Proceedings of the seventh ACM international workshop on Challenged networks*, pages 57--64. ACM.

Hui, P., Crowcroft, J., and Yoneki, E. (2011). Bubble rap: Social-based forwarding in delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 10(11):1576--1589.

Hui, P., Yoneki, E., Chan, S. Y., and Crowcroft, J. (2007). Distributed community detection in delay tolerant networks. In *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*, page 7. ACM.

Keränen, A., Ott, J., and Kärkkäinen, T. (2009). The one simulator for dtn protocol evaluation. In *Proceedings of the 2nd international conference on simulation tools and techniques*, page 55. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Kosta, S., Mei, A., and Stefa, J. (2010). 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.

Krifa, A., Barakat, C., and Spyropoulos, T. (2008). Optimal buffer management policies for delay tolerant networks. In *2008 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 260--268. IEEE.

Li, Y., Zhao, L., Liu, Z., and Liu, Q. (2009). N-drop: congestion control strategy under epidemic routing in dtn. In *Proceedings of the 2009 international conference on wireless communications and mobile computing: connecting the world wirelessly*, pages 457--460. ACM.

Lin, X., Andrews, J., Ghosh, A., and Ratasuk, R. (2014). An overview of 3gpp device-to-device proximity services. *IEEE Communications Magazine*, 52(4):40--48.

Lindgren, A., Doria, A., and Schelén, O. (2003). Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE mobile computing and communications review*, 7(3):19--20.

Lindgren, A. and Phanse, K. S. (2006). Evaluation of queueing policies and forwarding strategies for routing in intermittently connected networks. In *2006 1st International Conference on Communication Systems Software & Middleware*, pages 1--10. IEEE.

Misra, S., Saha, B. K., and Pal, S. (2016). Opportunistic mobile networks.

Naves, J. F., Moraes, I. M., and Albuquerque, C. (2012). Lps and lrf: Efficient buffer management policies for delay and disruption tolerant networks. In *Local Computer Networks (LCN), 2012 IEEE 37th Conference on*, pages 368--375. IEEE.

Nunes, I., Vaz de Melo, P., and A.F. Loureiro, A. (2016a). Group mobility: Detection, tracking and characterization. In *IEEE ICC 2016 International Conference on Communications (ICC'16 SAC-8 SN)*, Kuala Lumpur, Malaysia.

Nunes, I. O., Celes, C., de Melo, P. O., and Loureiro, A. A. (2016b). Groups-net: Group meetings aware routing in multi-hop d2d networks. *arXiv preprint arXiv:1605.07692*.

Nunes, I. O., de Melo, P. O. S. V., and Loureiro, A. A. F. (2016c). Leveraging d2d multihop communication through social group meeting awareness. *IEEE Wireless Communications*, 23(4):12–19. ISSN 1536-1284.

Pyattaev, A., Johnsson, K., Andreev, S., and Koucheryavy, Y. (2013). Proximity-based data offloading via network assisted device-to-device communications. In *Vehicular Technology Conference (VTC Spring), 2013 IEEE 77th*, pages 1--5. IEEE.

Rashid, S. and Ayub, Q. (2010). Efficient buffer management policy dla for dtn routing protocols under congestion. *international Journal of computer and Network Security*, 2(9):118--121.

Rashid, S., Ayub, Q., Zahid, M. S. M., and Abdullah, A. H. (2011). E-drop: An effective drop buffer management policy for dtn routing protocols. *Int. J. Computer Applications*, pages 118--121.

Rashid, S., Ayub, Q., Zahid, M. S. M., and Abdullah, A. H. (2013). Message drop control buffer management policy for dtn routing protocols. *Wireless personal communications*, 72(1):653--669.

Silva, A. P., Burleigh, S., Hirata, C. M., and Obraczka, K. (2015). A survey on congestion control for delay and disruption tolerant networks. *Ad Hoc Networks*, 25:480--494.

Spyropoulos, T., Psounis, K., and Raghavendra, C. S. (2005). Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252--259. ACM.

Spyropoulos, T., Psounis, K., and Raghavendra, C. S. (2007). Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility. In *null*, pages 79--85. IEEE.

Tsai, T.-C. and Chan, H.-H. (2015). Nccu trace: social-network-aware mobility trace. *IEEE Communications Magazine*, 53(10):144--149.

Vahdat, A., Becker, D., et al. (2000). Epidemic routing for partially connected ad hoc networks.

Wang, Y., Zhang, H., Tan, C. C., and Sheng, X. D. (2015). {WiGroup}: A lightweight cellular-assisteddevice-to-device network formation framework.

Yang, M. J., Lim, S. Y., Park, H. J., and Park, N. H. (2013). Solving the data overload: Device-to-device bearer control architecture for cellular data offloading. *IEEE Vehicular Technology Magazine*, 8(1):31--39.

Zheng, Z., Wang, T., Song, L., Han, Z., and Wu, J. (2014). Social-aware multi-file dissemination in device-to-device overlay networks. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 219--220. IEEE.