APPLICATION-AWARE DELAY TOLERANT NETWORK PROTOCOL

CHOO FAI CHEONG

NATIONAL UNIVERSITY OF SINGAPORE

2012

APPLICATION-AWARE DELAY TOLERANT NETWORK PROTOCOL

CHOO FAI CHEONG
(B. Computing (Hons), NUS)

A DOCTORAL THESIS SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF SINGAPORE

2012

Acknowledgements

I am very thankful to my parents who have been very supportive throughout my education journey. They have always encourage me to do the things I like, and pursue it as far as possible.

I also like to thank to my advisor Professor Mun Choon Chan who has been providing me with invaluable guidance and support throughout the research in this dissertation. He has pushed me to question and think more critically, and sharpened my research skills. He is a friendly advisor who make my research work more interesting.

Throughout my stay in the Communication and Internet Research Lab (CIRL) in School of Computing (NUS), I have made many wonderful friends in the lab. Their friendship, encouragement, and insightful discussion have really make a difference in my stay in CIRL. Specially, I would like to thank the following friends in CIRL: Hwee Xian Tan, Padmanabha Venkatagiri. S, Xiangfa Guo and Shao Tao.

Finally, I would also like to take this opportunity to thank the following people from National University of Singapore (NUS), who have given me much advice and encouragement during my studies: Professor A. L. Ananda and Professor Ee-Chien Chang.

Contents

Summary

Unlike traditional networks, DTNs are characterized by intermittent connectivity. Nodes may experience frequent disconnections and long communication delay in DTNs. While there are existing works that focus on improving the performance of DTN, they do not look into how it may benefit or improve the applications running on the DTN. With the unique characteristic of DTN, we believe that it is beneficial to the applications if the DTN protocols are designed with application in mind.

In this dissertation, we design protocols to better manage resources in the DTN. We show how routing can be improved when the routing protocol is application-aware. In addition, we consider the resource management for a class of applications in which nodes perform cooperative tasks apart from merely relaying messages. Finally, we look at the security implications that affect the resource usage in the DTN.

In our first work, we look into application-aware routing in DTN. We show that in the face of inherent intermittent connectivity, knowledge of application semantics can be exploited in routing to improve application performance in the network. We then propose a mechanism to capture application semantics based on dependency relationships. The mechanism is general and can be used to model a large class of applications. We show how to incorporate dependency relationship into existing DTN routing algorithms to enhance application performance. Specifically, our approach allows a relay node to prioritize the sending/buffering

of messages with the goal of optimizing the completion of application tasks.

In our second work, we consider resource management in a class of applications in which nodes in the DTN participate in completing application-related tasks in the system. We assume that tasks may appear dynamically in the system without a priori knowledge. As a result, it is not possible to pre-plan the task allocation to nodes in the system. We look at a possible real life taxi scenario that falls into the described class of applications and proposed the Taxi Advisory Dispatch System (TADS). TADS is a distributed taxi advisory system in which taxis collaboratively monitor and advise some free taxis to move to regions with higher ratio of clients. We perform evaluation of TADS based on traces obtained from a large Singapore taxi company that operates more than 15,000 taxis. Our results show that TADS can reduce the number of clients with wait times longer than 60 minutes by over 30%.

Finally, we look at the routing security issues that affect the resource usage in the DTN. In particular, we revisit the Haggle and DieselNet DTNs that Burgess et al. [1] have previously reported that both DTNs (with no authentication mechanisms) are robust against even a large number of attackers. We show how techniques that are employed by many routing protocols to improve resource usage can be exploited by attackers. Specifically, we demonstrate how to exploit routing metadata to improve the effectiveness of attacks and we identify scenarios where DTNs are most vulnerable to such attacks. In addition, we show how attackers can increase the effectiveness of their attacks in our application-aware routing protocols via manipulation of dependency relationships. Finally, we give a discussion on the level of authentication that is required to secure the attacks that we presented.

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Recent advancement in technology created a trend in which many low power, small portable devices are carried by humans as well as embedded in the environment as smart devices. A typical example is cellphones, which are carried by billions of users in the world for communication purposes. Unlike the early generation of cellphones, newer cellphones (such as smartphones) are often equipped with some short range wireless communication capability such as Bluetooth and WiFi.

While these devices may communicate over the cellular network, the use of such short range communication over the ISM band does have its advantages. First, communication over short range wireless network (eg. Bluetooth, ZigBee) can reduce energy consumption by up to 90% over cellular wireless networks (e.g. 3G networks) [2]. This may be a major consideration for power constrained mobile devices. Second, if the devices are in close communication range, these short range communication links often provides higher bandwidth than cellular wireless networks. Third, the use of short range communication links over the ISM band is much less costly to deploy as there is neither infrastructure nor subscription cost. Finally, cellular network service may not be available, especially in

rural areas or underground tunnels etc.

With a large number of short range wireless equipped portable devices being carried by humans or vehicles, these devices can potentially form a network for communication and information sharing purposes. Use of short range radios, coupled with mobility and energy saving mechanisms that turn off the network interface opportunistically lead to intermittent connectivity and the formation of a Delay/Disruption Tolerant Network (DTN). A DTN is characterized by the lack of a contemporaneous path between the source and destination at any given time. DTN connectivities are typically intermittent, and it may experience frequent, long duration partitioning.

Due to intermittent connectivity and difficulty in establishing an 'instantaneous' end-to-end route, traditional internet routing protocols and ad hoc routing protocols such DSR [3] and AODV [4] do not work well in DTN. In DTN, routing is typically opportunistic. Nodes do not pre-establish a routing path before data transfer. Rather, data are transferred in a store-and-forward manner from node to node whenever the link between two nodes is up. In the DTN literature, this is known as a *contact*. The amount of data that can be transferred during a contact opportunity is known as the *contact capacity*. The frequency of contacts, the contact capacity, and the buffer space (storage) available at each node are important factors that affect the routing performance of the DTN.

Applications that run on a DTN are typically non-real-time and are tolerable to various degrees of communication delays. Example of some possible familiar applications include email, file sharing/transfer applications, twitter-like or micro-blogging applications, social networking applications such as "who's near me", rss feeds, and etc.

In addition, there are also more specialized applications in which the DTN is formed mainly for the purpose of supporting the particular application. Such applications may involve nodes given some tasks to perform in the system. For example, consider a Search and Rescue operation. In this case, the agents (nodes)

come together (forming a DTN) solely for the purpose of rescuing the victims.

Another possible example is distributed taxi booking system [5] [6]. In a distributed taxi booking system, each taxi (node) is equipped with a short range communication device such as WiFi. A client uses a WiFi device to communicate with a taxi in wireless communication range. If the taxi is busy, it communicates with other taxis in the DTN to locate a free taxi in vicinity. The free taxi will then move to the client location to pick up the client.

Regardless of the kind of applications and their tolerance to delay, applications can generally still benefit from a better performance of the DTN. It is hence important to manage the resources in the DTN effectively to improve the application performance.

1.2 Application-Aware Protocols and Resource Management in DTNs

In this thesis, we focus on enhancing application performance through the better management of resources in the DTN.

For DTN routing, the important resources include the contact capacity and buffer size of the nodes in the network. We believe that if routing algorithms take into consideration of the application semantics, then the resources can be better utilized to improve the application performance.

For example, consider a file sharing application. All data blocks of a single file must be received by the destination for the file transfer to be deemed successful. However, due to intermittent connectivity in DTN, different data blocks of the file might be split, drop, and transferred by different relay nodes to the destination. When relay nodes are aware that all data blocks of the file must be successfully transferred before it is considered successful, the missing components of ongoing file transfers can be assigned with higher priorities to increase the number of successful file transfer. For example, it is much better

that 50% of file downloads completes successfully (ie. with 100% of the blocks received in each file transfer), as compared to the case whereby 100% of file downloads receive 50% of their individual data blocks - resulting in no successful file downloads at all. Existing routing protocols are optimized to deliver as many data blocks as possible, but do not take into consideration of the fact that a file transfer application requires all data blocks of the file to be received before it is considered successful.

The above consider the role of the nodes in the DTN as simply sending/relaying messages. However, nodes can perform cooperative tasks apart from merely relaying messages. Consider a class of applications in which nodes in the DTN participate in completing application-related tasks in the system. We assume that the system consists of an unknown number of tasks and the tasks may appear dynamically in the system without a priori knowledge. As a result, it is not possible to pre-plan the task allocation to nodes in the system. In addition, we assume that each node can only process one task at any single point of time and tasks can be allocated/distributed to other nodes in the form of request messages. Nodes in the DTN collaboratively try to allocate the tasks in the system with an objective in mind (such as minimizing average task completion delay, minimizing the maximum task completion delay etc). However, due to time constraints, nodes may only gather partial knowledge of the system before a decision has to be made.

As an example, consider a distributed taxi booking system. A taxi (node) would pick up a client and sends the client to his/her destination. Picking up a client and sending the client to his/her destination is considered as a task in the system. Taxis may choose to collaboratively pickup clients such that the maximum waiting times of clients can be minimized. Key challenges of such a system design will be: (i) how to quickly inform other taxis on the locations of known clients, and (ii) how the taxis can collaboratively determine the client that each taxi is supposed to pick up to avoid dispatching more than one taxi to

the same client.

1.3 Research Goals and Contributions

The main objective of our research work is to enhance application performance through better resource management in the DTN. In addition, we also look at the security implications that affect the resource usage in DTN.

The contributions of this dissertation are as follows:

1. **Application-Aware routing protocols for DTN**.

   In this work, we show that in the face of inherent intermittent connectivity, knowledge of application semantics can be exploited in routing to improve application performance in the network. To enable DTN routing algorithms to take into consideration application semantics, we proposed a mechanism to capture application semantics based on dependency relationships. The mechanism is general and can be used to model a large class of applications.

   We show that for many applications, data do inherently have some form of dependency relationships. For example, web pages and related multimedia objects reference each other using hyperlinks. Media file formats such as MPEG have the I-P-B frame structure for compression/decompression. Even normal data files have dependencies in which all data blocks of a file are required for a file transfer to be considered successful. Such data dependency can be algorithmically be extracted and use in routing algorithms.

   With dependency relationships extracted, we show how dependency relationships can be incorporated into existing DTN routing algorithms to enhance application performance. Specifically, it allows a relay node to

prioritize the sending/buffering of data blocks with the goal of optimizing the completion of application *tasks* (eg. a file transfer).

We perform evaluation using simulation on two real life traces and the simulation results show that when application semantics are exploited, we can substantially improve application performance from 60% to 583% over the baseline DTN routing algorithms.

2. **Application Resource Management in DTN**

In this work, we consider resource management in a class of applications in which nodes in the DTN participate in completing application-related tasks in the system. We assume that the system consists of an unknown number of tasks and the tasks may appear dynamically in the system without a priori knowledge. As a result, it is not possible to pre-plan the task allocation to nodes in the system. In addition, we assume that each node can only process a task at any single point of time and tasks can be allocated/distributed to other nodes in the form of request messages. Nodes in the DTN collaboratively try to allocate the tasks in the system with an objective in mind (such as minimizing average task completion delay, minimizing the maximum task completion delay etc). Due to time constraints and communication delay, nodes may only gather partial knowledge of the system before a decision has to be made.

There are many possible applications that falls into the above considered model (eg. search and rescue, distributed taxi booking system and etc). In this work, we work on a realistic example of such an application that we called the Taxi Advisory Dispatch System (TADS). TADS is a distributed taxi advisory system in which advisory tokens are sent to advise free taxis to move to regions that have a higher number of clients. Taxis would then have a higher chance of picking up clients. A task in TADS involves a taxi

picking up the client and sending the client to its intended destination. The objective of TADS is to reduce the long waiting times of clients. Taxis (nodes) in TADS are assumed to be equipped with a location-aware device (eg. GPS) and a short range wireless communication device (eg. WiFi).

TADS runs on DTN and is particularly challenging to design because taxis have limited communication opportunity. Due to limited communication and delay, each taxi would only have partial knowledge (eg. number of free taxis and clients in a region) about the system. In addition, information received from other taxis may be outdated quickly as taxis may move and change their status quickly. Clients may also be picked up by other taxis and communication delay may hinder such information from being made known to other taxis.

Our solution involves heuristics in which taxis collaboratively monitor to determine the number of free taxis and clients in each region over a period of time. At the end of each monitoring period, an elected leader taxi for each region determines if it is necessary to request for more taxis to move into its region. For regions that require more taxis, advisory tokens are generated (by their respective region leader) and forwarded to taxis in nearby regions. When a taxi receives the token, it moves to the requested region if it locally determines that moving to the requested region will improve the objective.

We perform evaluation of TADS based on traces obtained from a large Singapore taxi company that operates more than 15,000 taxis. Our results show that TADS can reduce the number of clients with wait times longer than 60 minutes by over 30%.

3. **Robustness of DTN against attacks**.

Routing attacks misuse resources in the DTN and may cause severe per-

formance degradation. In this work, we evaluate the robustness of DTN against attacks.

Due to difficulty of key management in DTN, some authors have looked into the possibility of forgoing authentication [1] [7]. Burgess et al. work [1] has suggested that some DTNs coupled with replication-based routing protocols are intrinsically fault tolerant, and robust against a large number of attackers even in the absence of authentication [1]. This poses the question on the necessity of authentication or the level of authentication required especially since authentication imposes overhead. Without authentication, it may encourage more nodes to join the network (providing more resources such as buffer storage) due to simplicity of joining.

First, we revisit the Haggle and DieselNet DTNs that Burgess et al. [1] have previously reported that both the DTNs (with no authentication mechanisms) are robust against even a large number of attackers.

We investigate two routing techniques that have been popularly used to improve resources in DTN routing. One technique is the contact history that is used in many DTN routing protocols [8] [9] [10]. These DTN routing protcols often flood their contact history into the network so that relay nodes can better allocate their resources to packets that are deemed to have a higher chance of delivery. Another technique is the network-wide packet delivery acknowledgement that is used to prevent copies of a delivered message from further replication [9] [10].

Using techniques that exploit the contact histories, we devised an attack called *non-deliverable flooding* attack. Non-deliverable flooding attack floods non-deliverable packets into the network. It also falsified contact history in an attempt to make the non-deliverable packets to be given higher priority for replication and buffering in the network. This causes severe resource contention in the network. In addition, we introduce an-

other attack called *identity impersonation* that exploits the lack of identity authentication and packet delivery acknowledgements. Identity impersonation attack causes relay and source nodes to drop packets that have not been delivered. We show the effectiveness of our attacks and we identify scenarios where DTNs are most vulnerable to such attacks.

Finally, we study how attackers can attack our application-aware routing protocols. Our application-aware routing protocols flood dependency graphs as routing metadata into the network so that relay nodes can determine the packets which are more crucial to complete application tasks. This however, may also give attackers the chance to exploit the dependency graphs in their attacks. We show how attackers can increase the effectiveness of attack through the manipulation of dependency relationships.

1.4 Organization

The rest of this dissertation is organized as follows: Chapter 2 discusses background and related work. It also discusses various security and possible attacks on a DTN. In Chapter 3, we present our application-aware routing protocols. Chapter 4 describes our proposed Taxi Advisory Dispatch System (TADS) and chapter 5 describes our work on the robustness of DTN against routing attacks. We conclude our work in Chapter 6 with directions for future research.

Chapter 2

Background

DTNs are a class of emerging networks that are characterized by intermittent connectivity. Such networks are often assumed to experience frequent, long-duration partitioning and may never have an end-to-end contemporaneous path.

In this chapter, we first give a short discussion on some possible applications in DTN. We then describe some terminologies and popular performance metrics that are frequently used in the context of DTN. Next, we provide a survey on some general strategies to improve performance in DTN. In addition, we discuss in more detail some popular routing algorithms. Finally, we look at routing security issues related to DTN.

2.1 DTN Applications

Applications that run on a DTN are typically non real-time and are tolerable to various degree of communication delays. Example of some possible familiar applications include email, file sharing/transfer applications, twitter-like or micro-blogging applications, social networking applications such as "who's near me", rss feeds, and etc.

In addition, there are also more specialized applications in which the DTN is formed mainly for the purpose of supporting an application. Such applications

typically involve nodes given some tasks to perform in the system. For example, in Search and Rescue operations, humans and even robots [11] [12] may come together with the purpose of searching and rescuing victims.

In participatory sensing applications, the applications involve using everyday mobile devices, such as cellular phones, to form participatory sensor networks to gather information [13] [14] [15] [16]. Each sensing devices are tasked to collect information and the sensed data may be relayed through the DTN network. For example, in GreenGPS [15], participatory sensing data collected by individuals from their vehicles are consolidated into a system that predicts the fuel consumption of an arbitrary car on an arbitrary street. In ParkNet [16], vehicles are tasked to collect parking space occupancy information while driving by. The collected information in ParkNet can actually be similarly routed to nearby vehicles using a DTN.

In a distributed taxi booking system [5] [6], clients use short range communication devices such as WiFi to communicate with taxis in communication range. Due to mobility of the taxis, communication between nodes (clients and taxis) is intermittent, forming a DTN. Client booking requests are forwarded to nearby free taxis through the DTN. Taxi nodes in the booking system are tasked to pickup and send their clients to their destination.

2.2 DTN Terminologies

Message. A message (or bundle) is a protocol data unit for data transfer in DTN. In this dissertation, we shall use the term message, bundle, packet and data block interchangeably.

Contact. When a link between two nodes is up, they exchange data with one another. This connection opportunity is referred to as a *contact* in the DTN literature [17].

Contact Duration. Contact duration is the time duration that the contact

lasted. The time duration between two contacts of a pair of nodes is known as the inter-contact duration.

Contact Capacity. Contact capacity depicts how much data can be exchanged when two nodes are in contact. A contact duration that is longer will tend to have a higher contact capacity (assuming other factors like interference being equal). Knowing the contact schedule and contact capacity allows DTN routing protocols to better schedule messages for transmission on certain paths.

Buffer Space (Storage). Due to unavailability of an end-to-end connectivity, DTN routing protocols adopt a store-and-forward approach. Messages are stored at relay nodes and forwarded/replicated to the next hop when they are in communication range. Messages may be buffered for a long period of time since disconnection period may be long. In the event of buffer contention, buffer management policy such as FIFO or other more complex strategies are used to drop messages.

2.2.1 Routing performance metrics in DTNs

The popular routing performance metrics that are used in the DTN literature are *delivery ratio*, *latency*, and *transmission overhead*.

Delivery ratio is defined as the fraction of generated messages that are delivered to the final destination within a given period of time.

Another frequently used metric is latency. It is the time a message is generated at the source to the time the message is delivered to the destination. While DTNs can generally tolerate delay, many applications can still benefit from shorter delivery latency.

Finally, transmission overhead is the amount of contact capacity consumed by a protocol to deliver a message. Replication-based routing strategies typically transmit more copies of the same message than forward-based routing strategies. Transmission overhead can also be different because routing strategies make different decisions about the next hop or routing path.

2.3 General Performance Improvement Strategies

In the literature, a number of performance improvement strategies have been used to improve the performance of DTN. They can be broadly classified into the following categories.

2.3.1 Adding more nodes

The most straightforward approach is to introduce more nodes into the sparse network. If additional nodes are introduced at different parts of the network, they provide more contact opportunity, capacity and buffer space for the network. In [18] [19], additional nodes in the form of *throwboxes* are added to the DTN to enhance performance. Throwboxes are stationary wireless devices with storage that acts as a relay, creating contact opportunity where none existed before. Another way to have more nodes in the network is to encourage more nodes to join the network by making it easier for them to join. Burgess et al. have suggested to forgo authentication (hence no administrative difficulty and overhead) to encourage more volunteer nodes into publicly deployed DTNs [1]. Using a number of possible attacks, the authors show that some DTNs coupled with replication-based routing protocols are intrinsically fault tolerant and robust against a large number of malicious nodes. In such DTNs, it may be possible to forgo authentication to encourage more nodes into the network.

2.3.2 Replication

DTN routing protocols can be broadly classified into forward-based or replication-based. In the forward-based approach, only a single copy of message is maintained in the network [20] [21] [22] [23] [24] while the replication-based approach replicates a message at many different nodes [25] [8] [26] [9] [10]. Replication is useful when nodes are unreliable, or when contact schedules are unpredictable. In such cases, replicating messages increases the chance of delivery and also re-

duces latency. The main idea of replication is to have many copies of the same message in the network to increase the probability that one of them will find its way to the destination. This also leads to a shorter delivery latency for the message.

The downside of replication is however the higher cost incurred in terms of transmission overhead and buffer space usage. Some routing protocols such as Spray and Wait [26], Spray and Focus [27] etc sets a predefined limit on the number of replications allowed per message to reduce cost. Others use various knowledge such as delivery predictability, acknowledgements, to limit or prevent further replication of messages [8] [28] [9] [10]. Some have explored the use of erasure and network coding schemes [29] [30] in an attempt to keep the benefit of replication while minimizing resource usage.

2.3.3 Knowledge

Embedding additional information and propagating them into the network has been shown to be beneficial to the performance of the DTN. Such information is often propagated as *routing metadata*. Although metadata takes up additional bandwidth, it however allows nodes in the network to be better informed and make better decision for forwarding and management of buffer space. The following paragraphs describe three common types of information that are propagated in the network to improve delivery performance.

The following describes 3 common types of information that are propagated into the network to improve performance.

Contact History

Depending on the network scenario, different levels of information may be available to the nodes. An example is the contact information. On one extreme end, all contact schedules are known beforehand; hence a routing protocol can compute the lowest delivery latency path to the destination. If nodes are also

reliable, such scenario negates the need to replicate messages as the most feasible path can be pre-computed and messages can be sent along the path reliably. On the other end is the scenario whereby there is no information about contacts in advance. Many routing protocols have been developed to address this scenario. The main strategy taken is to predict the future contacts based on past contact history [20] [31] [32] [8] [9] [10] [33]. In MaxProp [9], previous contact history is converted into meeting probabilities. These meeting probabilities are then used to estimate the delivery cost of a message to certain destination. In Rapid [10], each node keeps track of the average inter-contact duration and contact capacity of another peer based on previous observations.

The knowledge of these information allows a node to determine how likely it is able to deliver a message to its destination. Priority is usually given to messages that a node has higher chances of delivering. In order to provide estimation of delivering messages in multiple hops, such routing information is usually propagated to all the nodes in the network.

Acknowledgements

Besides information about contacts, information about delivery status of messages can also help to improve performance. In [9], the use of network-wide acknowledgements to inform all other nodes about the delivery of a message has been shown to improve the performance of replication-based routing protocols. Acknowledgements consisting of the cryptographic hash of the content, source, and destination of delivered messages are sent to the entire network (ie. flooding acknowledgements). Each acknowledgement incurs a small transmission overhead and is intended to be kept in each node for a long period of time. The advantage of network-wide acknowledgements is that it allows nodes holding copies of delivered messages to drop them. It also prevents further replication of delivered messages.

Recipient List

In [10], information about the nodes (source and relay nodes) having a copy of a message is propagated to other nodes. This allows nodes in the network to know who is holding a copy of the message and estimate the likelihood and latency that a message is going to be delivered. The estimated values are used for computing utility values which allow messages to be sorted for replication decision during a contact or for dropping when buffer is full. [10] shows that using recipient list can help to improve performance.

2.3.4 Buffer Management

Buffer management is important when using replication-based DTN routing protocols. Due to message replication, buffer tends to be filled quickly and dropping policies have direct impacts on the delivery performance of the routing protocols. Early DTN routing protocols manage finite buffer by using FIFO queues [25] or Drop Least Encountered (DLE) approach [34] [32] [8] [35]. Later works use delivery probability [9] or utility value [10] to decide on the messages to drop. In general, routing algorithms in DTN favours messages that can be delivered quickly.

2.3.5 Mobility

In [36] [37], the authors have shown that mobility can increase the capacity of mobile ad hoc networks or DTNs. To further enhance performance, nodes may actively move in response to communication needs [32] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47].

In [38], when a network is partitioned and a node is unable to send a message to a destination, the node will compute a shortest time strategy (including nodes mobility as part of the strategy) to deliver the message assuming that other nodes location are known and is willing to move to relay message.

In [39], message ferries are controlled to move in a Levy walk pattern to maximize the opportunity of meeting the destinations. Levy walks are known to show optimal searching efficiency for sparse and randomly distributed targets.

In [41], two message ferry schemes are proposed - Node Initiated Message Ferry (NIMF) and Ferry Initiated Message Ferry (FIMF). In NIMF, ferries move around the deployed area according to known routes. With knowledge of ferry routes, nodes periodically move close to a ferry and communicate with the ferry. In FIMF, ferries move proactively to meet nodes. When a node wants to send/receive messages to/from other nodes, it generates a service request to a chosen ferry using a long range radio. Upon reception of a service request, the ferry will move to meet up with the node and exchange packets using short range radio.

In [32], additional participants (autonomous agents) are introduced to augment the performance of DTN routing. Nodes in the network flood their status (bandwidth, latency etc.) so that the autonomous agents know about the global state of the network. Autonomous agents can then adapt their movements in response to variations in network demand to optimize for one or more metrics (eg. average delivery time).

2.4 Popular DTN routing protocols

In the forward-based routing protocols [20] [21] [22] [23] [24], message is passed from one node to the other until it reaches the destination. Forward-based routing protocols are more suitable in situations where nodes are reliable and contact schedules are a priori or highly predictable. For DTNs without known contact schedules, routing protocols typically use the replication-based approach.

In this dissertation, we look at the case whereby contact schedules are not known a priori. We discuss some popular replication-based routing protocols as follows.

2.4.1 Epidemic

Epidemic routing protocol [48] replicates messages to all contactable nodes in the network. The protocol uses significant resources (contact capacity and storage) due to replication. However, as long as the load does not stress the available resources, Epidemic gives the best performance in terms of delivery ratio and latency.

2.4.2 Spray And Wait

Spray and Wait [26] is a replication-based routing protocol that limits the number of copies of each message in the network. It consists of two phases: Spray phase and the Wait phase.

In the Spray phase, for each message originating from a source node, L message copies are initially forwarded by the source and relay nodes. If the destination is not found in the Spray phase, then at the Wait phase, nodes carrying a message copy will only perform a direct transmission (ie. forward the message only to its destination).

Multiple variants of spray phase are possible. In particular, the Binary Spray and Wait scheme has been proven to have the minimum expected delay among all variants of Spray and Wait routing protocols when nodes movement is IID. In the Binary Spray and Wait scheme, the source node initially starts with L copies of the message. When node i that has $n > 1$ message copies (source or relay) encounter another node j (with no copies), it forwards to node j $\lfloor n/2 \rfloor$ and keeps $\lceil n/2 \rceil$ for itself. When a node is left with only one copy, it switches to direct transmission.

2.4.3 Prophet

Prophet uses a probabilistic metric called *delivery predictability* for routing.

Initially, when node i meets node j, the delivery predictability p_{ij} is updated

to be:

$$p'_{ij} = p_{ij} + (1 - p_{ij}) \times p_0$$

where p_0 is an initialization constant.

When node i does not meet node j for some time, Prophet decreases the delivery predictability using

$$p'_{ij} = p_{ij} \times \gamma^t$$

where $\gamma \in [0, 1)$ is an aging constant, and t is the number of time units that have elapsed since the last time the metric was aged.

In addition, when node i receives delivery predictabilities from node j, node i may compute the transitive delivery predictability to node k using

$$p'_{ik} = p_{ik} + (1 - p_{ik}) \times p_{ij} \times p_{jk} \times \beta$$

where β is a scaling constant that decides how large the impact of transitivity should have on delivery predictability.

In Prophet, a message is replicated to the other node if the delivery predictability of the destination of the message is higher at the other node.

2.4.4 MaxProp

MaxProp is a popular replication-based routing protocol that has been shown to achieve better throughput than several other strategies such as Epidemic, Spray and Wait and Prophet [9].

MaxProp uses additional knowledge of contact history and message delivery status (acknowledgements) to better utilize resources. When two nodes meet, MaxProp replicates messages in the following order:

1. Messages destined to the contacted node.

2. Routing metadata (estimations of the probability of meeting every other

node)

3. Acknowledgements of delivered data

4. Messages in ascending order of hop count for hop count below a certain threshold. This threshold is adaptive and is determined by using the average contact capacity measured from previous encounters.

5. Messages in descending order of delivery likelihood

When a node needs to make space for buffer, MaxProp removes messages from its buffer in the following order:

1. Acknowledged messages

2. Messages in ascending order of delivery likelihood for messages with hop count above a certain adaptive threshold

3. Messages in descending order of message hop count

In contrast to Prophet which replicates a message only if the other node has higher delivery predictability to deliver the message to the destination, MaxProp always replicates all messages (except for those that the other node already has) as long as the contact capacity allows. To reduce resource usage, MaxProp sends network-wide acknowledgements for messages that have been delivered so that source and relay nodes can stop further replication and remove delivered messages from their buffer.

2.4.5 Rapid

Rapid [10] is a utility driven routing protocol that assigns a utility value for each message to determine the priority for replication and buffer management. When node i encounter j, the following steps are performed:

1. *Initialization*: Receive metadata from Y

2. *Direct Delivery*: Send messages destined to Y in decreasing order of utility

3. *Replication*: For each message that is not in node Y, compute the marginal utility of replicating the message to Y. Replicate the messages in descending order of marginal utility normalized by message size.

Similar to MaxProp, Rapid always replicates all messages as long as the contact capacity allows (except for those that the other node already has). It also uses network-wide acknowledgements for messages that have been delivered. In addition, Rapid keeps track of the nodes holding a copy of each message so that it can better estimate the expected delivery delay of messages. The expected delivery delay is used as part of the utility computation of Rapid.

Rapid has been shown to perform extremely well, outperforming both Spray and Wait and MaxProp [10]. It uses more information in its utility computation compared to MaxProp.

2.5 DTN Routing Security

2.5.1 Key setup in DTNs

Till date, approaches for securing routing in DTN largely depends on using public key cryptography to limit participants to a set of authorized nodes and using class of service for the allocation of buffering and link capacity [17] [49] [50] [51].

Public keys for verification can be pre-distributed before deployment, but this approach is more difficult when incremental deployment of network nodes is desirable. Alternatively, a public key infrastructure (PKI) may be used. The use of traditional PKI however, is not suitable for disconnected environments such as DTN, since access to online servers for fetching public keys and checking Certificate Revocation Lists (CRL) cannot be assumed. As such, Identity Based Cryptography (IBC) schemes have been proposed for use in DTN environments [50] [51]. With IBC scheme, the recipient public key is simply a function of a

public identification string of the recipient, hence the recipient identity implicitly validate the recipient public key. As for CRL, Seth et al. propose to use a short timeout period for signing keys (eg. a day) [50]. Signer is expected to get new time-stamped private key from the PKG before the previous key timeout.

The use of cryptography to authenticate every single participants and messages in DTN can dramatically reduce the attack surface of the network. However, such schemes have high processing overhead and administrative or key management difficulty in certain environments, and hence may not always be possible to implement [1].

2.5.2 Routing Attacks

While routing security has been studied extensively in traditional ad hoc networks [52] [53] [54] [55] [56] [57], the work cannot be easily extended to DTNs due to different routing style and network characteristics. For example, route in ad hoc networks are typically established before any data transfer. Routing disruption attacks such as black hole [52] [53], flood rushing [57] and wormhole [54] attack the route establishment process. It either causes route establishment to fail, or establishing a route that data will not be delivered to its destination. In DTNs, routing are opportunistic and they typically do not pre-establish a route before sending data.

In this section, we survey possible routing attacks on DTNs.

Blackhole/DropAll Attack

In this attack, attacker simply drops all the messages that it receives [52] [53]. In forward-based DTN routing protocols, attackers may advertise themselves as the most suitable relay to the destination so that messages are routed to the attackers and then dropped. Since forward-based DTN routing protocols do not replicate messages, the message will be lost and not be delivered to the destination. Using replication-based routing protocols is a natural defence against drop all attacks.

Route Metadata Falsification

Even though routing in DTN may be opportunistic, studies have shown that the use of routing metadata such as expected meeting time or probability can improve routing performance. Each node's routing metadata is propagated to other nodes in the network so that every node will have a more global view of the network. Unfortunately, an attacker may inject falsified routing metadata into the network and cause performance degradation.

Routing metadata falsification attack has been studied extensively in ad-hoc networks literature [52] [53] [54] [55] [56]. An example of routing attack in ad-hoc networks includes creating a routing loop, so that packets traverse nodes in a cycle without reaching their destination, thus consuming energy and available bandwidth. Another example is the blackhole attack whereby nodes advertise fake routing metadata to attract packets and then drop the packets.

For forward-based DTN routing protocols, attacks can advertise themselves as the most suitable relays to the destinations so that packets can be forwarded to them which they can launch blackhole attacks.

For replication-based DTN routing protocols, due to the availability of many paths to destination, routing metadata falsification causes performance degradation in a different manner compared to ad hoc network or forward-based DTN routing protocol.

Routing metadata falsification attack in replication-based routing protocols works by causing messages to be replicated in the wrong order or even not replicated at all. For example in Prophet [8], if the other node has a lower delivery predictability than the destination, then current node will not replicate the message to the other node. In addition, when there is buffer contention, it causes messages to be dropped in the wrong order. Hence when contact capacity is low or when buffer contention is high, routing metadata falsification can cause severe performance degradation.

Defence against routing metadata falsification in DTN includes getting every node in the network to sign its routing metadata, and checking if metadata claim is consistent. For example, if node A claims frequent meeting and message exchanges with node B, but node B claims otherwise, further investigation might be needed.

Acknowledgement Counterfeiting

For replication-based routing protocols in DTN, the use of network-wide acknowledgements to inform all other nodes about the delivery of a message has been shown to be very beneficial to the routing performance. Acknowledgements in [9] [10] is simply the 128-bits cryptographic hash of the content, source and destination of each message. Unfortunately, this does not prevent attackers who have seen the message from flooding fake acknowledgments into the network. Fake acknowledgements causes relay nodes holding a copy of the acknowledged message to be dropped and can cause severe degradation of performance.

A solution to this will be for the destination of messages to sign every acknowledgement that it created. Relay nodes can hence verify the authenticity of acknowledgements. However, the signature size itself incurs a large overhead. Alternatively, Burgess et al have proposed to ignore and delete an acknowledgement if a node has not previously seen the message that the acknowledgement is acknowledging [1]. This leverage on the fact that messages should normally propagate from nodes closest to a message's source to nodes closest to a message's destination. Consequently, message acknowledgements should propagate in the reverse direction. Burgess et al. show that this defence reduces the effectiveness of acknowledgement counterfeiting in the MaxProp protocol [1].

Recipient List Attack

In [10], Aruna et al. show that replicating messages together with the list of nodes holding a copy of the message can help to improve performance as nodes

can better estimate on how likely the message is to be delivered soon. Unfortunately, similar to acknowledgement counterfeiting, attackers who know about the message can easily fake the list of nodes holding a copy of the message. Note that even if the list is to be signed by each individual node itself, compromised insider nodes sharing their keys can still sign and make it look as though there are already many nodes holding a copy of the message. This may cause the message to be regarded as being sufficiently replicated and hence dropped in the event of a buffer contention.

Resource Consumption Attack

Resources such as the contact capacity, buffer space, and battery of mobile nodes are limited. Due to replication of messages and metadata, performing a resource consumption attack is easier in replication-based routing protocols than in traditional ad hoc network routing protocols.

Contact capacity is an important resource in sparse DTNs as links may be up only for a short time and down for a long period of time. A simple way to drain contact capacity is to flood new messages into the network. Replication-based routing protocols will replicate the messages, draining up contact capacity and buffer space of nodes in the network. To counter this attack, admission control should be performed to limit the amount of messages that each node can inject into the network.

Chapter 3

Application-Aware Routing

In DTN, nodes typically do not pre-establish a routing path before data transfer. Data are transferred in a store-and-forward manner from node to node whenever the link between the two nodes is up. During a contact, data can either be *forwarded* or *replicated* to the other node. In the forward-based approach, only a single copy of the message is maintained in the network [20] [21] [22] [23]. Forward-based DTN routing algorithms usually assume a DTN environment whereby nodes are reliable, and contact schedules are a priori or highly predictable. When nodes are unreliable or contact schedules are not a priori, replication-based routing algorithms are usually preferred. Replicating messages in such cases increases the chance of delivery and also reduces latency [25] [8] [26] [9] [10].

To further improve performance, existing routing algorithms exploit various information such as contact history, acknowledgements, limiting number of copies in the network, and even list of nodes having a copy of a message [8] [26] [9] [10]. In [58] [59], exploitation of mobility patterns based on social network context is applied. Nodes that are in the same communities or more popular are identified so as to improve delivery probability.

While the above mentioned routing algorithms exploit various additional information to improve routing performance, none of these routing algorithms are

designed to exploit application semantics to enhance routing. Existing literature on routing in DTNs ([10] [60] [9] [20] [21] [61] [62] [8] [26]) assumes that data packets are independent and uncorrelated with one another. In this work, we assert that in the face of inherent intermittent connectivity, knowledge of application semantics can be exploited to improve resource allocation and application performance in the network. We support our claim with the following scenarios.

Scenario 1 In Twitter-like or micro-blogging applications where update messages are distributed over time, it is necessary that a subscriber receives these updates in an ordered sequence. Messages that arrive out of order cannot be displayed or used by the subscriber, and have to be either buffered or discarded until earlier updates have all arrived. With knowledge of application semantics, intermediate relaying nodes can assign higher priorities to earlier updates so that they are scheduled to be sent or routed before later updates.

Scenario 2 In file sharing applications, *all* the data blocks (or components) of a single file must be received by the destination for the file transfer to be deemed successful. When relay nodes are aware of such dependencies between blocks of data, the missing data blocks of an ongoing file transfer can be assigned with higher priorities to increase the number of successful file transfers, which can be especially significant in resource-constrained DTNs. For example, it is much better that 50% of file downloads complete successfully (i.e. with 100% of the blocks received in each file transfer), as compared to the case whereby 100% of file downloads receive 50% of their individual data blocks - resulting in no successful file downloads at all.

Scenario 3 In applications whereby data blocks in a stream of data have varying importance, application semantic awareness enables the network to manage its resources in a more efficient manner, resulting in better application performance. For example, video frames that are encoded in MPEG format can be classified as I, P or B frames - in order of descending importance. In the event of the loss of an I frame, all the corresponding P and B frames in the same

data stream cannot be used. In contrast, the loss of a P frame affects only the corresponding B frames in the same stream.

Based on the scenarios that we have illustrated, it is obvious that existing DTN routing algorithms are inadequate and do not optimize application-level performance as they: (i) ignore application semantics; and (ii) are evaluated based on conventional networking metrics such as packet delivery ratio and latency - which have little correlation with application-level performance metrics.

To enable DTN routing algorithms to take into consideration of application semantics, we propose a mechanism to capture application semantics based on dependency relationships. By translating dependency relationships into hints that can be used by resource allocation schemes, the mechanism allows better resource allocation to improve the application performance. In addition, the mechanism is general and can be used to model a large class of applications.

Information about dependency relationship can be tagged along with little overhead to control messages disseminated by existing DTN routing algorithms. The DTN routing algorithms can potentially make use of the information to enhance performance. In this work, we show how a simple DTN routing algorithm - Epidemic, and a complex Rapid-like [10] DTN routing algorithm can utilize the dependency relationships for routing purposes.

3.1 Dependency Graph

3.1.1 Obtaining dependency information from Application

As mentioned earlier, there are many cases where awareness of application semantics can be exploited to provide benefits to the network nodes for resource allocation purpose. However, in order to exploit these dependencies, we need a way to capture them in an organized manner so that they can be exploited algorithmically. While this imposes requirements on the applications we support, we show that such information is already available from many existing popular

applications as follows:

Web pages

Web pages and related multimedia objects reference each other using hyperlinks. This information can be harnessed to build the dependency graph. Application such as HTTrack (http://www.httrack.com) uses hyperlink information to extract the dependencies among different web pages.

Social network applications

Social network applications such as Facebook Graph API and Twitter API allow access to data generated by the users. The data is available in a structured format. For instance, facebook wall data for a user is returned in JSON (JavaScript Object Notation) format. Wall objects contain posts by the user, comments of friends of a user and other information which can be represented as dependency graphs.

Software update management tools

Software updates are often automated. Most software update management utilities are accompanied by tools to extract the dependencies of the updates. For instance, *apt-cache* (http://linux.die.net/man/8/apt-cache), a package handling utility, exports the dependencies of a package using DOT, a plain text graph description language. This information can be used to build a dependency graph. A tool that performs a similar function for Microsoft Windows based platform is *dependency walker* (http://www.dependencywalker.com).

Media files

Media file formats such as MPEG have the I-P-B frame structure which facilitates the building of dependency graph. The I-P-B frame structure is well established and used extensively in media applications.

Normal data files

Data files are the simplest case of dependencies where all data blocks are required for the file to be useful.

3.1.2 Application Model

We assume that different application types can run on a DTN and many instances of an application are possible. We define the terminologies used below.

- **Block:** A block of data is the unit of data transmitted or received over a wireless link. We assume that there is a maximum block size similar to the idea of a maximum transmission unit (MTU). Note that each data block is sent in the form of a message in the DTN.

- **Update:** An update is a set of data blocks which forms an unit of data input to/from the user application. An update may be fragmented in the delivery process. Such fragmentation will lead to blocks from the same update becoming reordered or being delivered over different paths due to intermittent connectivity.

- **Channel:** A channel is defined as a sequence of *updates*. A user application is a consumer of this channel and each channel is associated with an application type. For instance, a channel that produces twitter feed will be consumed by twitter clients. The updates belonging to a channel could originate from any of the participating nodes in the DTN and be consumed by any of the participating nodes.

The relationships among block, update and channel are shown in Figure 3.1.

Users express their interest to a channel, for example, a twitter feed, a particular file sharing session or a podcast feed by subscribing to that channel. Updates can be periodic or aperiodic depending on the application. The size of

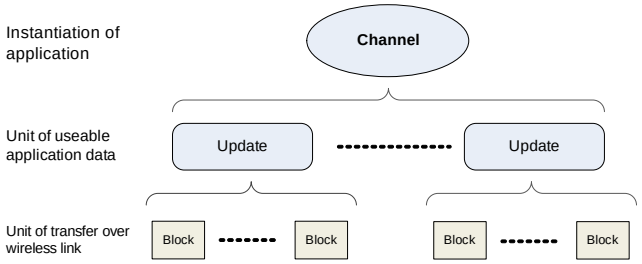


Figure 3.1: Relationships among block, update and channel.

each update may also vary.

Due to intermittent connectivity, blocks of an update can follow different paths to reach the subscriber nodes where they are re-assembled. Further, blocks may arrive at the subscriber nodes out-of-order.

3.1.3 Dependency Model

The definition and structure of the dependency graph used in this work is presented below.

Definition: Dependency Graph A dependency graph $G_d = (V_d, E_d)$ where V_d is the set of vertices and E_d the set of edges, is a directed graph where if there is a directed path from node $A \in V_d$ to node $B \in V_d$, then node A **depends on** node B.

Definition: Dependency set of node A Given a dependency graph G_d, the dependency set of node A, $G_d(A)$, is defined as the set of nodes reachable from node A in G_d. Therefore, a node $j \in G_d(A)$ if there is a directed path from node A to j.

One way to think of the dependency set of node A is that all nodes in this set must be received by the subscriber before node A can be used by the application. Dependencies can be defined among blocks as well as updates. Hence, the nodes in G_d can be either blocks or updates. Finding the dependency set of a node involves a simple depth first search and is hence very efficient.

Definition: Task For each update u, there is an associated task. When

all updates in $G_d(u)$ are received by the subscriber, update u can be utilized by the application and the task associated with u is completed.

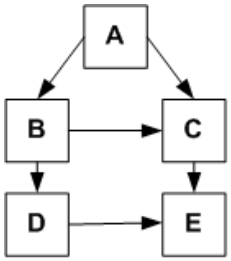


Figure 3.2: Example of a dependency graph

Figure 3.2 shows a simple dependency graph. Following from the definition for task, for each of the updates (A, B, C, D, E) in Figure 3.2, the tasks are given below:

$$G_d(A) = \{A, B, C, D, E\}$$

$$G_d(B) = \{B, C, D, E\}$$

$$G_d(C) = \{C, E\}$$

$$G_d(D) = \{D, E\}$$

$$G_d(E) = \{E\}$$

From the application's perspective, task completion and latency are more useful than measures looking at only data block delivery statistics. We use the following three applications to further illustrate the dependency graph concept.

1. streaming data/messaging, where delivery must be in order;
2. file download where delivery is completed only when all blocks are received by the subscriber;
3. video file compressed in MPEG format with I, P and B frames.

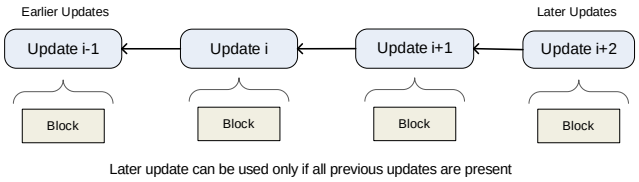


Figure 3.3: Dependency graph of messaging application.

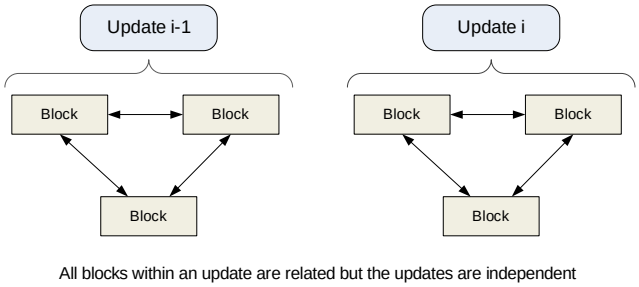


Figure 3.4: Dependency graph of file sharing application.

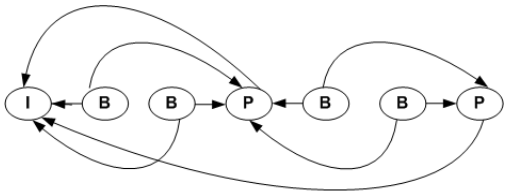


Figure 3.5: Dependency graph of IPB group of pictures

These three applications represent a range of different dependency characteristics. In Figure 3.3, each message update is assumed to be relative small and can fit into a single data block. In this case, we only need to consider update (as there is only one block per update). Since the dependency is strictly based on message creation time, the dependency graph structure is a single linked list, with each update pointing to the earlier update. Hence, if we consider an update i created at time t_i, all updates with creation times less than t_i must be received by the subscriber before update i can be delivered/processed. Therefore, for update i, $G_d(i) = \{..., i-1, i\}$.

Figure 3.4 shows the case for a file download/sharing application. In this case, a single file is one update. If the file size is larger than a data block, the file (a single update) will have multiple data blocks. In the figure shown, each file has 3 data blocks.

First, consider the dependency on the block level, within an update. Since all blocks must be received by the subscriber for the file download to be successful, the dependency graph for all blocks within an update is a complete graph where all blocks depends on each other. For a block b, $G_d(b)$ is the set of all blocks of the same update. Also, all blocks of the same update have the same dependency set.

At the update level, we assume that the updates are independent. Therefore, subscribers can receive and utilize these update files independently. For update i, $G_d(i) = i$.

In Figure 3.5, we consider the case where the dependencies are more involved than the previous two examples. The example is based on the types of pictures or frames (the I-, P- and B- frames) used in MPEG video compression algorithm that exploits temporal redundancy. An update can be either a I-, P- or B-frame. An I-frame is intra-coded, which in effect is a fully specified picture, like a conventional static image file. A P-frame may reference previous I- and P-frames while a B-frame may reference both previous and next I- and P- frames

for decoding.

Each frame is in turn made up of a single or multiple data blocks. The I-frames and P-frames are usually bigger than the B-frames. Depending on the video bit rate, one or more B-frames may be transmitted as a single data block but the I-frames are likely to be fragmented into multiple data blocks. Within a frame, as in the previous case, if there are multiple blocks, the dependency graph for these blocks forms a complete graph.

Take the example of a B-frame. The task associated with this B-frame is only completed when all frames/updates that this B-frame depends on are received by the subscriber.

Once the dependency relationships among data blocks can be represented in a structured way, it is possible to design algorithms that take into account these dependencies. In section 3.2, we describe how a given DTN routing algorithm can be enhanced to take advantage of this information.

3.2 Depedendency-Aware Routing

As highlighted earlier, all data blocks of a task must be received by the destination before these blocks can be consumed by the application. Due to intermittent connectivity and limited buffer storage available at relay nodes, it is likely that blocks belonging to the same task will be split into many smaller parts and relayed through different nodes before arriving at the destination. If any one of these parts does not reach the destination, blocks from the same task that have been delivered cannot be used for that task.

Our proposed dependency graph approach allows relay nodes to be aware of the dependency relationships of the data blocks in a task. As a result, relay nodes can make use of these dependency relationships to schedule important and related blocks with higher priority.

Dependency-aware DTN routing algorithms need not be designed from scratched.

Existing DTN routing algorithms can be enhanced with dependency awareness to improve on its routing decision. We show in Section 3.2.1 how we can modify a simple Epidemic routing algorithm into a dependency-aware routing algorithm. Next, we show in Section 3.2.3 how we modify a more complex, utility based RAPID-like routing algorithm into a dependency-aware routing algorithm.

3.2.1 Dependency-Aware Epidemic

In this section, we show how one can enhance the simple Epidemic algorithm to be dependency-aware. We called this enhanced Epidemic algorithm *D-Epidemic*.

Conceptually, D-Epidemic improves on Epidemic by performing scheduling in a FIFO manner based on tasks rather than on data blocks. Recall that each task has one or more data blocks that are associated with it. For relay nodes to identify the blocks that are associated with a task, the dependency relationships are sent in the form of metadata to the relay nodes.

When a task is scheduled to be sent to a relay node, all data blocks in the node's buffer associated with the task will be sent together. However, note that if a data block has already been sent due to its association with other scheduled tasks, it will not be sent again.

In addition, D-Epidemic maintains two logical FIFO queues, a *high priority* and a *low priority* queue. During a contact, a node always sends tasks from the high priority queue first, follow by the low priority queue. In the event of buffer contention, nodes always drop tasks (and their associated data blocks) from the low priority queue first, unless the data blocks are also associated with other tasks.

Mapping of a task to either the high or low priority queue depends on how many of the task's data blocks are present in the node's buffer. If all of the data blocks associated with a task is present in the node's buffer, we call such task a *full task*. Otherwise, a task is a *partial task* since only part of the task is stored in the node's buffer.

Full tasks are always stored in the high priority queue. On the other hand, a partial task is given a one time probability of being stored in the high priority queue. This probability, α, is a system parameter. When $\alpha = 0$, full tasks are given strictly higher priority than partial task. In this case, full tasks may end up being in buffer for extended period of time, starving newer partial tasks. In our evaluation, we set $\alpha = 0.5$, giving higher priority to full task, and yet do not excessively starve partial tasks.

Note that while D-Epidemic only exploits dependency in the simplest form, it is sufficient to demonstrate how dependency awareness can be used to enhance a simple algorithm.

3.2.2 SAR

In this section, we describe a DTN routing algorithm we named **S**ubscriber **A**ware **R**outing or *SAR*. SAR is a utility based algorithm that has been specifically modified from RAPID [10] algorithm to suit a publish/subscribe environment. We believe that a publish/subscribe environment is more suitable for supporting the DTN applications that we used in the evaluation. SAR is considered to be complex as it incorporates a number of well known techniques that exploit contact history and keeps track of data availability on nodes in its utility computation to enhance delivery ratio and latency. We will modify SAR to be dependency-aware in section 3.2.3.

Overview of SAR

When two nodes using SAR are in contact, they first exchange metadata and then compute a utility value for every data block in its buffer. Each node then computes the utility gain value for each data block assuming that the data block is replicated to the peer node. Data blocks are then replicated to the other peer in descending order of utility gain value. In addition, each node computes a utility loss value for each data block in buffer assuming that the data block is

dropped. In the event of buffer contention, SAR drops data blocks in ascending order of utility loss value.

We describe the operations of SAR in more detail below.

Metadata exchange

When two nodes using SAR are in contact, the following metadata is exchanged:

(a) Contact history (inter-contact time)

(b) List of channels subscribed by each known nodes

(c) Known data blocks and their list of nodes known to have a copy of it

In (a), a table of node contact history that contains the durations between each successive contact for each node (*inter-contact duration*) is sent. Using history of inter-contact duration, nodes can estimate the delivery latencies of data blocks to their subscribers.

To allow relay nodes to know the subscribers of each data block, subscription details of each known node is sent in (b). This allows relay nodes to make informed decision on how to route the data blocks. The subscription details contain the matching tags for matching published contents to the subscribers.

In (c), known data blocks and their list of nodes known to have a copy of it is sent. This allows a node to compute and estimate the delivery latency of a data block to its subscribers, and how many subscribers already received a copy of the data block.

Estimating data block delivery latency

The contact (or meeting) times for nodes made up a large portion of delivery latency. Nodes estimate the expected meeting time from the source to the destination node as follows:

Node A estimates $E(M_{AC})$, the expected time for node A to meet node C using the contact history (inter-contact duration) that is exchanged as part of

the metadata. $E(M_{AC})$ is approximated using the expected time taken for A to meet C in at most h hops. For example, if A meets C via an intermediary B, the expected inter-contact time is computed as the expected time for A to meet B and then B to meet C. We restrict h=3 in our implementation. RAPID and several other DTN routing protocols [10] [9] [8] [32] have use similar techniques to estimate meeting probability among peers. If two nodes never meet even via three intermediate nodes, the expected inter-contact time is set to infinity.

Suppose data block i has been replicated at nodes $R = \{r_1, ..., r_k\}$ and the subscriber is $C \notin R$. The expected time to deliver data block i to C is estimated to be:

$$D_{R,C}(i) = [\sum_{r \in R} \frac{1}{E(M_{rC})}]^{-1} \tag{3.1}$$

Note that equation (3.1) assumes the meeting times to be exponentially distributed. The distribution of meeting times in the traces (eg. DieselNet) are very difficult to model. Nodes may change their routes many times in the traces and the inter-contact distribution is noisy. Approximating meeting times as exponentially distributed makes delay estimates easy to compute and has been reported to perform well in practice, see for example, [10], [26].

The computation described above works only for a single (known) destination. In general, there can multiple destinations and this is shown in the utility computation to be described in the next section.

Utility Computation

We define $Q_{R,S}(i)$, the sum of the estimated latency that data block i will take to be delivered to all subscribers as

$$Q_{R,S}(i) \quad = \quad \sum_{s \in S} D_{R,s}(i) \tag{3.2}$$

where R is the set of nodes having a copy of data block i, and S is the set of subscribers which need data block i and has not received it. Note that $Q_{R,S}(i)$ is 0 when i has been delivered to all the subscribers.

$Q_{R,S}(i)$ allows SAR to have a quantifiable delivery metric that can be used to measure how much delivery latency is reduced if block i is replicated to another node. As data blocks may be of different sizes, replicating each of them will use different amount of resources. Hence the utility gain is normalized by the size of the data block.

For example, if data block i is replicated to node r_k, then the utility gain normalized by the size of the data block is:

$$\frac{Q_{R,S}(i) - Q_{(R+r_k),S}(i)}{size_i}, \quad where \quad r_k \notin R \tag{3.3}$$

3.2.3 Dependency and Subscriber-Aware Routing (DSAR)

In this section, we show how to modify SAR into a dependency-aware routing algorithm. We named the dependency-aware version of SAR to be **D**ependency and **S**ubscriber **A**ware **R**outing (DSAR).

We make two changes to SAR. Firstly, we need to send dependency relationship information to relay nodes. Dependency relationship information can be sent in the form of metadata together with their associated data blocks. Secondly, we need to modify the utility computation function to take into consideration dependency relationship information.

Task Utility Computation

In SAR, data block delivery latency is the main component used for utility computation. However, for DSAR, we need to take into consideration both data block and task delivery for utility computation.

We defined *task contribution* for a task as the ratio of undelivered data blocks

of the task in the node's buffer over the total number of (estimated) undelivered blocks of the task. As an illustration, consider a task with a total of 100 data blocks and 20 data blocks of this task are known to have been delivered. If a node has 60 undelivered data blocks out of the 80 remaining data blocks in its buffer, the node can contribute $\frac{60}{80}$ or 75% to the task. Note that task contribution has to be computed for each subscriber separately since each subscriber may have received different subset of a task.

The value of task contribution captures the utility of a set of data blocks for task completion. It also requires fairly minimum information to compute. The number of undelivered blocks in buffer is locally available and the number of undelivered data can be estimated based on the acknowledgement messages.

We modify the SAR utility function by including the task contribution as a parameter factor in the utility computation. Specifically, we define DSAR utility function $V_{R,S,T}(i)$ as follows:

$$V_{R,S,T}(i) = \sum_{t \in T} \sum_{s \in S_t} [D_{R,s}(i) \times C(t,s)] \tag{3.4}$$

where T is the set of tasks which depends on data block i, S_t is the set of subscribers of task t that has not received data block i, and $C(T,s)$ is the task contribution for task t and subscriber s.

We define the utility gain function of DSAR as the increase in delivery metric with respect to $V_{R,S,T}(i)$. Specifically, if node A has a data block i to replicate to node B, node A computes the normalized utility gain of replicating data block i to node B as follows:

$$G_{AB}(i) = \frac{V_{(A \in R, B \notin R),S,T}(i) - V_{(A,B \in R),S,T}(i)}{size_i} \tag{3.5}$$

Similarly, the utility loss function of DSAR is defined as the decrease in delivery metric with respect to $V_{R,S,T}(i)$, normalized by the size of data block

being dropped. During buffer contention, node A will drop the data block that has the lowest utility loss value. It computes utility loss value for data block i as follows:

$$L_A(i) = \frac{V_{(A \notin R),S,T}(i) - V_{(A \in R),S,T}(i)}{size_i} \tag{3.6}$$

When two nodes meet, DSAR nodes replicate data blocks in descending order of utiltiy gain value. In the event of buffer contention, DSAR nodes drop data blocks in ascending order of utility loss.

There are two overheads when we convert SAR to DSAR, namely sending the dependency lists and utility computation. Dependency lists need to be stored in the buffer as well as exchanged when two nodes communicate. However, the dependency lists can be stored very efficiently using a bitmap. As each single data block is in order of few KBs, the overhead is relatively small.

Utility computation of DSAR as compared to SAR involves an additional multiplication for task contribution. Task contribution requires node to keep track of the data blocks belonging to the task that has yet been delivered. As existing routing schemes like SAR already incorporates block delivery acknowledgements, computing task completion is straightforward.

3.2.4 Mixed Traffic

In the discussion so far, we assume that dependencies for all data blocks are known. In practice, this may not be possible. A mixture of data blocks with known and unknown dependencies can be handled in the following way.

For D-Epidemic, two logical queues are maintained, one for traffic with known dependencies and one for those without. For the logical queue with known dependenices, the same high and low priority queues are maintained as described in Section 3.2.1. For the logical queue with unknown dependencies, data blocks are maintained in a FIFO manner. Data blocks are sent in a round-robin manner between the two logical queues of known and unknown dependencies. For the

known dependencies traffic, all data blocks belonging to at least one task are sent before switching to the queue with unknown dependencies traffic. Similar amount of unknown dependencies traffic are then sent before switching back to the known dependencies traffic to maintain service fairness. This algorithm is clearly not optimized. While one could design a "smarter" algorithm that performs better by taking into account more information, we believe that a simple approach is better for demonstrating the effect of taking into account task information in this case.

For DSAR, the approach is to generalize the computation of task utility to traffic with unknown dependencies. The two parameters required are task contribution and dependency list. For task contribution, we estimate this value (for all unknown traffic) as the average task contribution value computed over all the known dependencies traffic in the node's buffer. For dependency, we simply assume that each data block with unknown dependency belongs to only one task.

The performance with mixed traffic is evaluated in Section 3.3.

3.3 Simulation Evaluation

In the evaluation, we compare DSAR and D-Epidemic algorithm with SAR and Epidemic algorithm respectively. Comparing SAR to DSAR and Epidemic to D-Epidemic allow us to see the performance gain by utilizing dependency information of data blocks in tasks.

The metrics used in the evaluation are Task Completion Ratio (TCR), Block Delivery Ratio (BDR) and Task Latency. TCR is computed as

$$TCR = \frac{\sum_{s \in S} |T_s|}{\sum_{s \in S} |T_{ps}|} \tag{3.7}$$

where S is the set of subscribers, T_s is the set of tasks received by subscriber s, and T_{ps} is the set of tasks being subscribed to by subscriber s. TCR captures

the ability of the algorithm to complete the generated tasks.

BDR is similar to the *delivery ratio* metric used in unicast routing algorithms. However, in our evaluation, BDR is extended to a Publish/Subscribe environment. It is defined as:

$$BDR = \frac{\sum_{s \in S} |B_s|}{\sum_{s \in S} |B_{ps}|} \tag{3.8}$$

where S is the set of subscribers, B_s is the set of data blocks received by subscriber s, and B_{ps} is the set of data blocks that are part of the tasks being subscribed to by subscriber s.

Finally, task latency is defined as the time difference between subscriber successfully received all blocks of the task and the time that the subscriber has subscribed to the task.

Two application types are used in the simulation, namely: file and linux package updates.

Linux package updates are modeled as downloads of the application TraceRoute along with their package dependencies. We obtain the package dependencies of TraceRoute using apt-cache [63]. The dependencies characteristics of this application can be described as having a total of 12 updates. 3 of the updates depend on 7 other updates and 1 of the updates depends on all the other 11 updates. In addition, each of these updates consist of a set of blocks whose dependency graph is a complete graph.

Traffic for the file application is generated as a set of blocks whose dependency graph is a complete graph. In the case of a complete graph, the reception of all data blocks in the graph is counted as one task completion.

Our evaluation are based on the mobility traces of taxi cabs in San Francisco [64], and the Haggle Infocom 05 [65] trace.

The San Francisco taxi (SanCab) trace consist of the locations of approximately 500 taxis collected over 30 days in the Sans Francisco Bay Area. Using

the reported GPS locations, we assume a wireless range of 100m and create connections for each pair of nodes if they are within a distance of 100m. We (randomly) picked the first and the tenth day of the trace for simulation. The trace for each day is further split into 2 segments, giving a total of 4 traces each having 12 hours of trace data. The simulation results reported are the average of the 4 traces.

The Haggle trace consists of a 3 day long trace that is based on a human mobility experiment in Infocomm 2005. 41 volunteers joined the experiment and was each given an iMote device that can communicate with one another using Bluetooth. The trace includes connection events among these iMotes devices (Class 1 devices) and also to other Bluetooth-capable devices (Class 2 devices). We removed connection events from the Haggle trace that lasted less than one second or involved the singular appearance of a node since meaningful data transfer is likely to require setup time and nodes incapable of routing data may be ignored. After the transformation, the traces left with only events involving the 41 iMotes devices. We further split the Haggle trace into 3 segments, and the results reported are the average of the 3 segments.

Table 3.1 gives the default configuration for the simulation scenarios discussed later. Note that each data block is equivalent to one bundle and for each channel, a node is randomly chosen to be the publisher. Unless otherwise stated, these parameters will be used throughout the simulation.

3.3.1 Effect of varying buffer

We first study the effect of varying buffer size. Figure 3.6 shows the Task Completion Ratio (TCR) and Block Delivery Ratio (BDR) results. In terms of TCR, DSAR and D-Epidemic performs substantially better than their respective counterparts SAR and Epidemic, especially when buffer size is very limited. At 12MB, DSAR is able to complete 74% and 17% more tasks than SAR in the SanCab and Haggle trace respectively, while D-Epidemic is able to complete 583% and

Table 3.1: Simulation Parameters

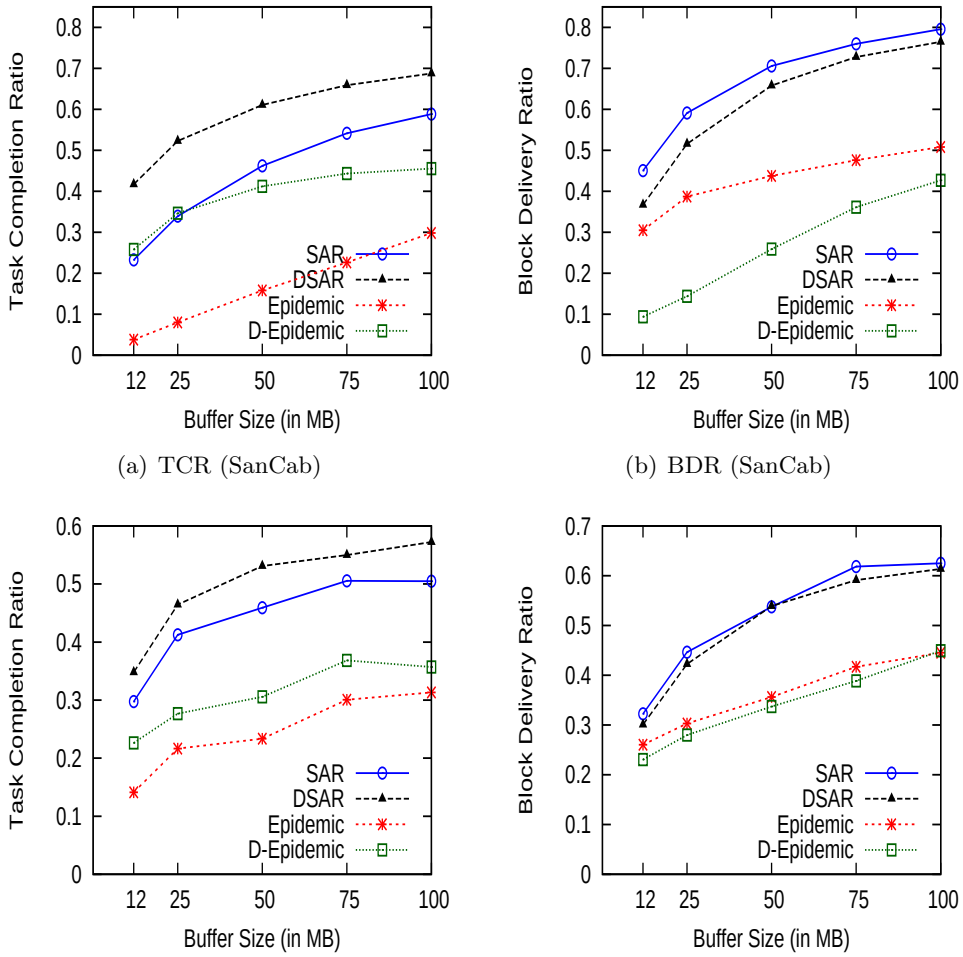
Parameter	**SanCab**	**Haggle**
Number of publishers	24	8
Number of subscribers	24	8
Channels subscribed per subscribers	8	8
Number of blocks (File application)	40-140	40-140
Avg injection rate in blocks (File application)	180/hr	180/hr
Number of blocks (Linux package)	10-180	10-180
Avg injection rate in blocks (Linux package)	150/hr	150/hr
Size of the data block	50KB	50KB
Transmission Rate	2Mbps	1Mbps
Buffer Size	50MB	50MB
Bundle Expiry Time	6 hours	6 hours

60% more tasks than Epidemic in the SanCab and Haggle trace respectively.

In terms of BDR, SAR outperforms DSAR for both SanCab and Haggle traces. This is expected since SAR is optimized for BDR while DSAR is optimized for TCR. In addition, DSAR has overhead of sending the dependency list. Note however, TCR is the more relevant metric for application performance. Hence, higher BDR is not as useful as higher TCR.

While DSAR is able to achieve higher TCR performance, it is clear that the improvement is much larger in the SanCab trace than in the Haggle trace. This can be explained by looking at how likely a task will be split through multi-hop delivery. If the task is not split and is delivered to a subscriber in a single contact, dependency awareness would not matter much since the entire task is already being delivered as a single block.

From the simulation results, considering only tasks with at least one block being delivered, we observe the following when SAR is used on the Haggle trace: 56.7% of the tasks are delivered in full using 1 contact, 16.3% are delivered in full using multiple contacts and 27% are partially delivered. As a result, the possible performance gain achievable by DSAR is limited since it can only improve on



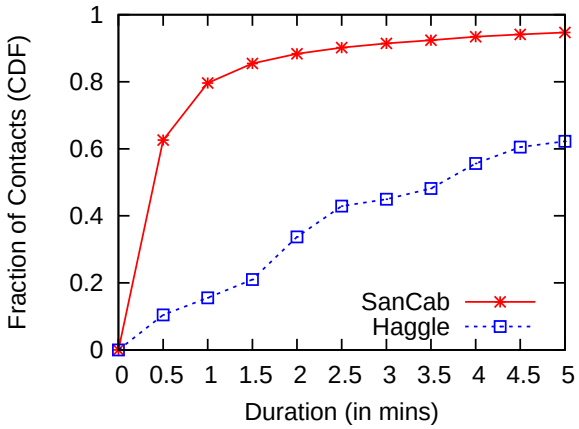
(a) TCR (SanCab) (b) BDR (SanCab)

(c) TCR (Haggle) (d) BDR (Haggle)

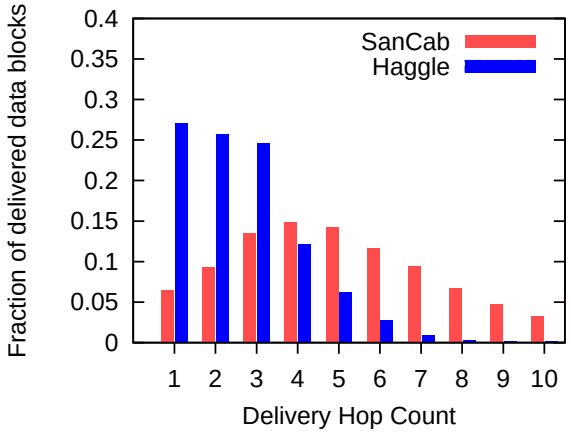
Figure 3.6: Effect of Buffer Size

the remaining 27% that are partially delivered. On the other hand, when SAR is used on the SanCab trace, only 10.9% of the tasks are delivered in full using 1 contact, 40.8% are delivered in full using multiple contacts and 48.3% are partially delivered. DSAR is thus able to improve on the remaining 48.3% that are partially delivered. Therefore, it is clear that the characteristics of the traces play a very important role in determining how effective task awareness can be.

Figure 3.7(a) shows the CDF graphs of contact duration of the SanCab and Haggle traces. Clearly, the SanCab trace consists of many more short duration contacts than the Haggle trace, causing much more connection breaks during



(a) CDF of Contact Duration



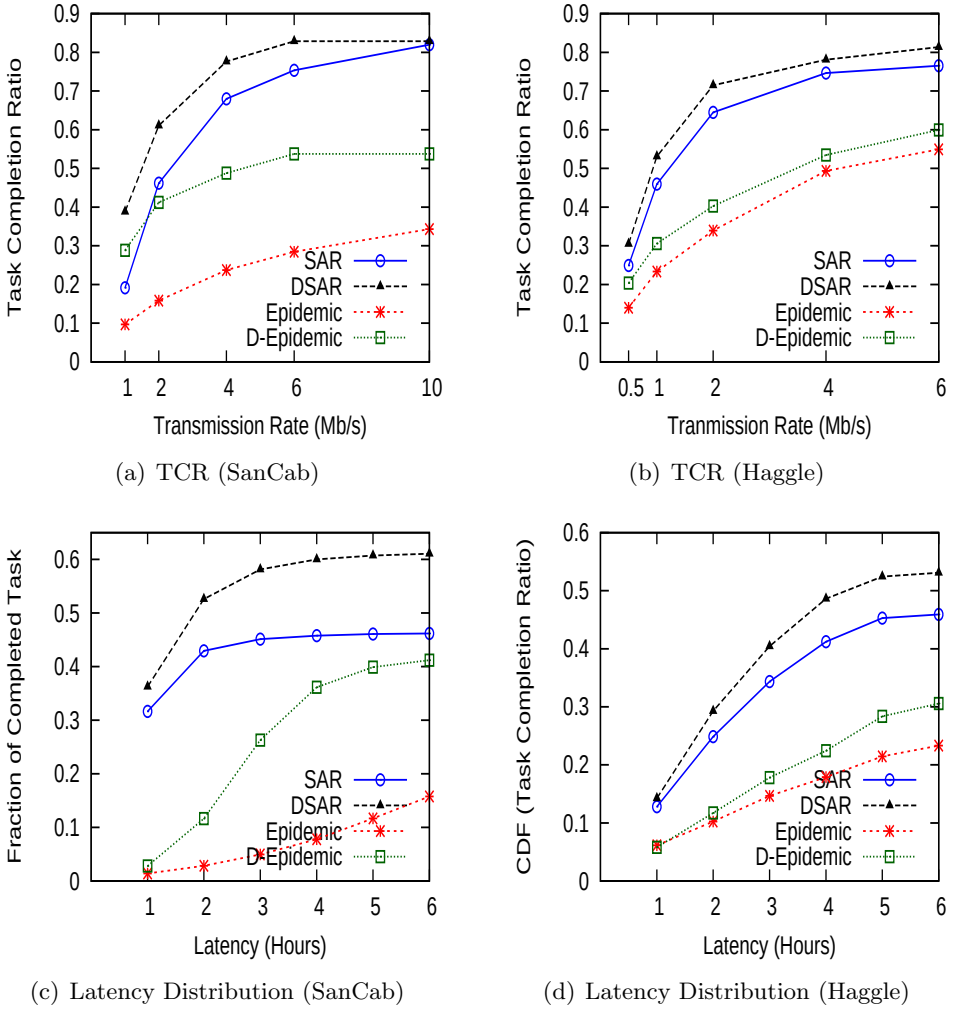
(b) Distribution of Hop Count

Figure 3.7: Trace statistics

data transfer. For example, over 60% of the contacts in the SanCab trace are shorter than 30 seconds, while only about 10% of the contacts in the Haggle trace are shorter than 30 seconds. Similarly, Figure 3.7(b) shows the hop count distribution of SAR when the data blocks are delivered. 95.7% of the packets in the Haggle traces are delivered in 5 hops or less. On the other hand, in the SanCab trace, 41.8% of the packets takes 6 or more hops to reach the subscriber.

The results from both figures show that tasks are much more likely to split and be delivered over multiple paths in the SanCab trace. Without any dependency relationship information, nodes in SAR (or Epidemic) may end up wasting resources delivering many partially completed tasks to the subscribers.

In summary, while our dependency-aware routing scheme is able to improve performance in both traces, it perform much better when contacts are intermittently short, and whereby delivery would occur over more hops and different paths. Our approach thus works best in DTN environments with high mobility, for example a urban vehicular DTN, as in the SanCab scenario.



(a) TCR (SanCab)

(b) TCR (Haggle)

(c) Latency Distribution (SanCab)

(d) Latency Distribution (Haggle)

Figure 3.8: Effect of Transmission Rate

3.3.2 Effect of Varying Tranmission Rate

In this section, we study the effect of varying transmission rate. We vary transmission rate from 0.5Mbps to 10Mbps. Figure 3.8(a) and 3.8(b) shows the TCR

results of the simulation.

The results here are quite similar to varying buffer sizes. In all cases, the dependency-aware routing algorithms DSAR and D-Epidemic outperform their respective dependency ignorant counterparts, and the performance gain is much larger in the SanCab trace than in the Haggle trace. For TCR, the improvement of DSAR over SAR reaches up to 100% in the SanCab trace. For D-Epidemic, the improvement over Epidemic reaches up to 197% in the SanCab trace.

3.3.3 Task Latency Distribution

We analyze the task latency distribution in this section. The default simulation parameter is used (see table 3.1). Figures 3.8(c) and 3.8(d) show the task latency distribution.

The dependency-aware DSAR and D-Epidemic is able to consistently deliver more tasks in a shorter time. In terms of task completion latency, in the SanCab trace, 58% of the tasks in DSAR completes within 3 hours. For SAR, the ratio is 45% in 3 hours. The ratios for D-Epidemic and Epidemic are 26% and 5% respectively for task completion within 3 hours.

3.3.4 Task Completion for Different Task Sizes

Results in the previous sections only consider aggregated TCR across all task sizes. While dependency-aware routing algorithms significantly outperform non-dependency-aware routing algorithms, it is also important to look at the task completion ratio for different task sizes. If the gain in overall TCR is at the expense of lowering TCR for large tasks (which takes up more resources to complete) to benefit small tasks, the performance gain will not be as attractive.

Tables 3.2 and 3.3 show the breakdown in task delivery over different task sizes for both traces. The results shown are measured using the default simulation parameter shown in Table 3.1. There are improvements in task delivery for DSAR for both SanCab and Haggle trace. The result for D-Epidemic is less

impressive. For the larger task sizes (100 blocks or more), there is a decrease in task delivery though a substantial number of the large tasks are still able to complete and large tasks are not overly penalized. This trade-off can be attributed to the fact that D-Epidemic has strong preference over complete task in buffer, and hence indirectly favours smaller tasks as they are less likely to be split.

Table 3.2: Total Delivered Task Distribution (SanCab)

Task size (blocks)	**Epidemic**	**D-Epidemic (α=0.5)**	**SAR**	**DSAR**
10	1877	5667	4923	6597
15	1731	4154	3491	4518
40	291	855	630	841
60	110	256	496	599
80	133	167	935	1291
100	66	63	535	622
120	99	64	1012	1423
140	15	14	383	447
180	13	7	255	345
All	96535	199100	505815	664240

Table 3.3: Total Delivered Task Distribution (Haggle)

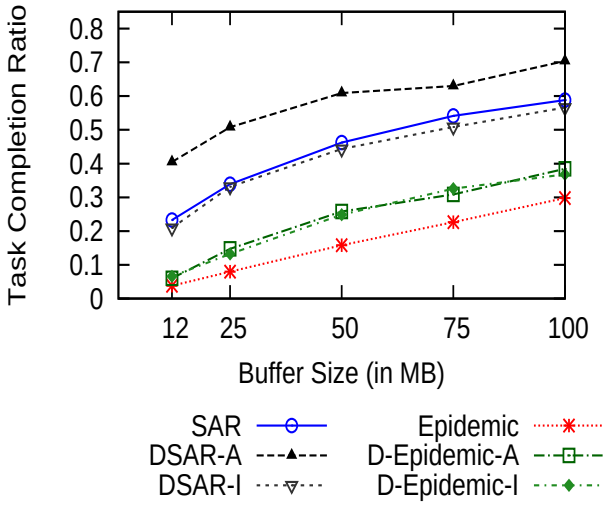
Task size (blocks)	**Epidemic**	**D-Epidemic (α=0.5)**	**SAR**	**DSAR**
10	129	211	255	302
15	106	129	218	293
40	18	27	78	80
60	51	52	84	88
80	59	84	115	137
100	38	50	47	54
120	54	54	106	107
140	40	49	57	53
180	9	5	19	21
All	28880	34205	52000	56295

3.3.5 Mixed Traffic Type

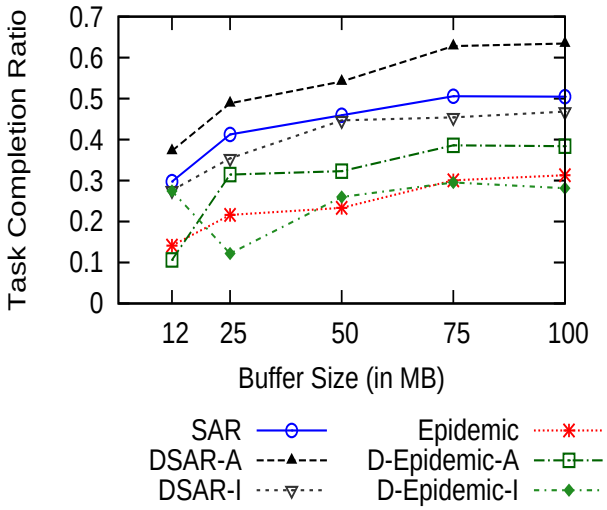
In this section, we evaluate the performance when there are traffic with both known and unknown dependencies. In the simulation, the default setting (see Table 3.1) is used, except that 50% of the traffic does not provide dependency data. The results are shown in Figure 3.9(a) and 3.9(b). DSAR-A and DSAR-I show the TCR for known and unknown dependency traffic respectively for DSAR. For DSAR, there is improvement for known dependencies traffic when compared to their dependency ignorant counterparts. For unknown dependencies traffic, DSAR provides rather similar performance compared to SAR.

3.4 Conclusion

We have shown that in the context of DTN where connectivities are intermittent, it is very beneficial, if not crucial, for application semantics to be known to the network nodes in order to improve performance. We proposed to use dependency graph to capture the application semantics based on dependency relationships and showed that dependency relationships can be added to existing DTN routing algorithms. We illustrated our approach by showing how two routing algorithms (Epidemic and SAR) can be enhanced to be dependency-aware. In addition, the enhanced routing algorithms can be further modified to handle mixed traffic type - mixture of data blocks with known and unknown dependencies.



(a) TCR (SanCab) - Mixed traffic



(b) TCR (Haggle) - Mixed traffic

Figure 3.9: TCR for mixed traffic

Chapter 4

Application Resource Management in DTN

The previous chapter consider the role of nodes in the DTN as simply sending/relaying messages. However, nodes can perform cooperative tasks apart from merely relaying messages. In this chapter, we consider resource management in a class of applications in which nodes in the DTN participate in completing application-related tasks in the system. Nodes typically come together (forming a DTN) to support the application by completing application-related tasks. We consider a simple model as follows.

We assume that the system consists of a number of nodes in the system with similar task processing capability. Each node is able to process at most one task at any point of time. When a node is processing a task, its status is *busy*, otherwise its status is *free*.

The system consists of an unknown number of tasks that may appear in the system at different times without a priori knowledge. Each task may be handled by any free nodes in the system. We assume that all the nodes in the system have similar processing capability, taking similar amount of time t_i to complete task i. However, there may be a cost $C_i^x(t)$ associated with node x for accepting

task i at time t. For example, if at time t, task i and node x are at different locations, then node x would need to travel to task i location, incurring a cost $C_i^x(t)$.

Each node only has partial knowledge of the existence of all the tasks in the system. If a node discover a new task and is busy, it may try to inform other nodes in the DTN about the task. Nodes in the DTN collaboratively tries to allocate the tasks in the system with an objective in mind (such as minimizing average task completion delay, minimizing the maximum waiting time to start processing a task etc). However, due to time constraints and communication delay, nodes are likely to only able to gather partial knowledge of the system before a decision has to be made.

Example of applications that falls into the above described model includes Search and Rescue operations in which nodes actively search for victims in the disaster location. Each task involves picking up the victim and sending the victim to a safe location (such as hospital). If a victim is found by a busy node, the busy node may sends out request to other nodes nearby to pickup the discovered victim. Another possible example of applications that falls into the above described model includes participatory sensing applications. In participatory sensing applications, nodes may be tasked to perform sensing operations at certain locations.

In this work, we look at a possible real life taxi scenario that falls into the above described model. We consider the case of a DTN that is formed by taxis and waiting clients. Both taxis and (some) clients are equipped with short range wireless devices such as WiFi to allow them to communicate in the DTN. In addition, taxis are equipped with location-aware devices such as a GPS device. A task in this application would be a taxi picking up a client and sending the client to its destination. Client demands at each region vary over time and is not known to the taxis beforehand. The objective is to minimize the number of clients that have waited for a long period of time (eg. waited for an hour).

We propose Taxi Advisory Dispatch System (TADS), a distributed system in which taxis collaboratively determine the regions with high client demands and avoid having excessive free taxis moving into the same region. TADS runs on DTN and is particularly challenging to design because taxis have limited communication opportunity. Due to limited communication and delay, each taxi only has partial knowledge (eg. number of free taxis and clients in a region) about the system. In addition, information received from other taxis may be outdated quickly as taxis may move and change their status quickly. Clients may also be picked up by other taxis and communication delay may hinder such information from being made known to other taxis.

Exising work on distributed taxi booking system [5] [6] cannot be applied to TADS because they do not consider a DTN environment whereby disconnection is frequent. Ideally, we should assign taxis to regions so that the overall time taken by the taxis to travel to the requested region is minimal. This is related to the Linear Assignment Problem (LAP). Even though decentralized solutions for LAP have been proposed [66] [67], they are not suitable for our work due to very fast changing and intermittent connectivity environment.

Our approach to the problem involves nodes collaboratively estimating the supply/demand of each region. A node that supposedly has the most information on the supply/demand of a region sends out advisories to guide or suggest some free taxis to move into the region. Advisories are generated carefully to avoid having excessive free taxis moving into the same region.

We perform evaluation of TADS based on traces obtained from a large Singapore taxi company that operates more than 15,000 taxis. Our results show that TADS can reduce the number of clients with wait times longer than 60 minutes by over 30%.

In the following sections, we first give an introduction of the taxi transportation situation in Singapore. We then describe our system model and then give a detail description of TADS.

4.1 Introduction

Taxis form an important means of public transport for many countries. It provides a comfortable and convenient alternative to public transport such as buses and rail. In Singapore, in particular, taxis are widely available and relatively low-priced. As of November 2011, the number of taxis in Singapore is about 27,018 [68]. Compared to major cities like Hong Kong, London and New York, Singapore has one of the highest number of taxis per million population [69].

However, even though Singapore has a high number of taxis, waiting time for a taxi can still be long, depending on the location, time of day and weather condition. For example, statistics [70] show that the average taxi waiting time in the main shopping district can vary from 5 minutes to 38 minutes during the same hourly interval among different locations that are only 1 to 2 kilometers apart.

This large difference in waiting times among nearby locations suggests that there exist significant imbalance in the supply and demand of taxis. If a number of the free taxis were to be given advice to move to regions with higher demand, long taxi waiting times can be reduced. In addition, it may also benefit the free taxis since they were given advice to move to regions with higher chances of getting clients.

Our proposed system, Taxi Advisory Dispatch System (TADS), addresses the above problem by sending advisory tokens to request some nearby taxis to move to regions with a higher demand.

Note that such advisories may (slightly) increase the waiting times for clients in regions with lower demand for taxis, while substantially reducing the waiting time of regions with high demand.

The main objective of TADS is to minimize long waiting times rather than the average waiting time over all regions. In addition, unlike a taxi booking system, TADS deals with "flag down" clients and does not handle taxi bookings.

TADS complements and is compatible with the existing taxi booking systems.

TADS assumes that taxis are equipped with short/mid range communication devices such as WiFi for communication. Communication can be among taxis or between a taxi and a client's smartphone. Use of WiFi keeps the running cost of the system low compared to the use of cellular network.

In TADS, each taxi estimates the number of free taxis and clients in its vicinity. Such information is used by a leader to determine the number of advisory tokens to generate. A key challenge of TADS is to be able to perform state estimation accurately in a distributed manner in a DTN environment. An important aspect of TADS is to prevent excessive generation of advisory tokens as this may lead to too many taxis moving into the same region which is undesirable.

4.2 System Model

Clients are classified into either *simple* clients or *smart* clients. A simple client is someone who simply waits along the road and flags down a free taxi in sight. A smart client is someone who uses a WiFi device (such as a smartphone) to communicate with taxis to signal his/her intention to hire a taxi. Each smart client has a unique client ID.

Taxis are equipped with GPS and WiFi. Each taxi is also given a unique ID and time is synchronized through GPS. When two taxis are in wireless communication range, they exchange data. Other taxis who can overhear the communication can update their records as well. Taxis essentially come together forming a peer-to-peer Delay Tolerant Network (DTN).

Taxi state is either *FREE*, *POB* (Passenger-On-Board) or *BUSY*. A taxi state is in POB if it has a passenger on board. If the taxi is not available to pickup clients due to other reasons (eg. taxi was booked, driver going for lunch etc), then the taxi state will be BUSY. Note that both the POB and BUSY state cannot pickup a client. However, it can be travelling on the road and can relay

data or advisory tokens.

An **advisory token** is a message that request free taxi to move into a particular geographical region. Advisory tokens for a region are generated periodically by a chosen *leader taxi*. For each region, during each monitoring period, taxis floods taxi and clients records they have in possession to other taxis in vicinity. The taxi that has the most number of records is chosen to be the leader for the region. The leader taxi then generates a number of advisory tokens for the region based on the information that it has.

A token includes an expiration time and can be exchanged wirelessly among taxis. When a taxi receives a token, it may choose to accept or reject the advice. If a taxi accepts the advice, the taxi changes its current path and move towards the indicated region. The token is thus *consumed*.

It is important to note that an advisory token only advises a taxi to move into a region. It does not assign a client to a taxi and hence no client information is provided in the advisory. In addition, even after a taxi accepts an advice, it is not necessary for the taxi to move into the targeted region. For example, along its way to the region, the taxi may pick up a client. In which case, the advisory is regenerated with the same expiration time as the initial advisory and forwarded to other taxis. On the other hand, if the taxi rejects the advice, the taxi continues to be the relay node of the advisory token until the token expires.

4.3 TADS in detail

TADS consists of 3 components: (1) state estimation (number of free taxis and clients in each region), (2) generation of advisory tokens based on the estimated states, and (3) distribution of advisory tokens. We assume that the entire region of interest (e.g. the entire Singapore island) is divided into regions.

Table 4.1 lists the commonly used notations in the following sections. Throughout this chapter, unless otherwise stated, operations are *implicitly described with*

reference to the current monitoring period. For instance, $\mathbb{T}_r$ refers to the set of taxis that have driven past or entered region r in the current monitoring period.

Table 4.1: Notations in TADS

$\mathbb{T}_r$	Set of taxis that have driven past or entered region r.
cnt_r^{POB}	Total number of POB taxi records for region r.
cnt_r^{FREE}	Total number of FREE taxi records for region r.
tot_r^{FREE}	Total number of free taxis that have driven past or entered region r.
tot_r^{simple}	Total number of simple clients in region r.
tot_r^{smart}	Total number of smart clients in region r.
ldr^j	Node j's replica copy of a leader record.
$ldr.l$	The leader identity as indicated in the leader record.
$ldr.r$	The region as indicated in the leader record.
$ldr.score$	A score value as indicated in the leader record.
ID^i	Unique identity of node i.
M_s	Start time of a monitoring period.
M_g	Start time of a monitoring guard.
M_e	End time of a monitoring period (also the end time of monitoring guard).
$dist$	A system parameter that determines how far a record is allowed to replicate.

4.3.1 State Estimation

State estimation is performed for each region periodically through monitoring by taxis in vicinity. Figure 4.1 illustrates multiple monitoring cycles in TADS.

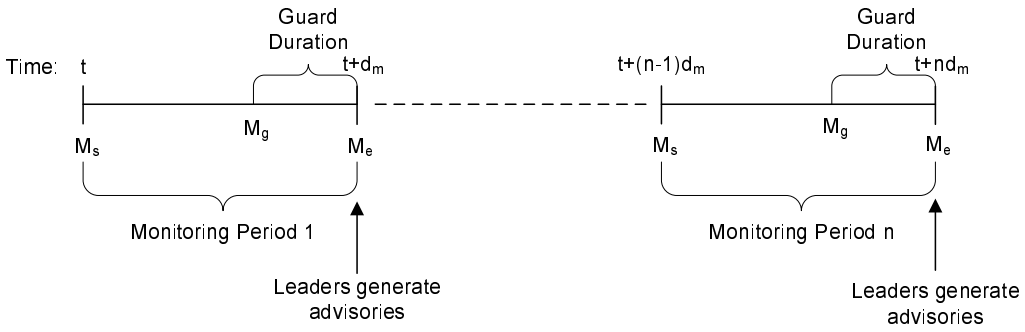


Figure 4.1: Monitoring cycles.

Let M_s and M_e be the start and end of a monitoring period respectively. In addition, let M_g be the start of the guard duration. During [M_s,M_g), taxi $i \in \mathbb{T}_r$ participates in monitoring region r. Specifically, taxis in $\mathbb{T}_r$ collaboratively try to estimate the number of free taxis and clients in the region.

To estimate the number of free taxis moving into a region, whenever a free taxi enters a region or when its status changes to free in the region, the free taxi generates a unique *FREE taxi record*. This taxi record is then flooded to all other taxis within a pre-defined distance or in pre-defined neighbouring regions. Flooding allows data to spread faster and is a popular technique used in DTNs [9] [10]. The taxi current location can be made known to the routing layer, so that the routing layer can prevent further replication of a record that is outside the area that it is supposed to be replicated.

Estimating the number of clients in a region is challenging since simple clients are impossible to detect accurately without additional sensory inputs. TADS obtains a conservative estimate on the number of simple clients in the region through observing changes of taxi status from FREE to POB. When a free taxi picks up a simple client, it generates a unique POB taxi record. The POB taxi records are then flooded to other taxis in the region and its neighbouring regions.

For estimating the number of smart clients in a region, each smart client generates a unique client record per monitoring period and forwards it to any taxi that is within communication range. Note that smart clients do not need to have their time synchronized through GPS. Their time can be synchronized when they encounter a taxi. Taxis are time synchronized through GPS. Similar to the POB taxi records, client records are flooded to other taxis in the region and its neighbouring regions.

Note that no taxi or client record is generated during the guard duration period [M_g,M_e). The purpose of the guard duration is to allow some time for generated records to "spread" to the leader taxi of the respective region before the end of the monitoring period (see next section 4.3.2).

In addition, note that for a smart client to be counted, it has to be able to communicate to at least one taxi in the monitoring period. If there is no taxi within communication range of the smart client, TADS will not know the existence of the smart client.

To summarize, *FREE* taxi records are used for estimating the number of available taxis. The total number of (simple and smart) clients is estimated by adding the number of "flag down" pickups and number of smart clients records seen in a monitoring period. TADS makes two assumptions. First, the size of the data to be exchanged is small relative to the WiFi throughput available even though contact times are not large. Second, while contacts among taxis are intermittent due to their mobility, the density of taxis is sufficiently large so that messages can reach neighbouring regions within minutes.

4.3.2 Generating Advisory Tokens

Advisory tokens are generated at the end of each monitoring period by elected taxis (leaders). Throughout the monitoring period, taxis collaboratively elect a leader for each region r. However, due to time constraint and long communication delay, TADS may not give a unique leader at the end of the monitoring period. Hence, it includes heuristics to minimize the number of duplicate leaders for a region.

TADS leader election scheme comprises of two components: (1) initial selection and (2) leader transfer/elimination.

For initial selection, suppose a taxi generates a taxi record (for monitoring) or receives a client request directly from a client in the region. If the taxi does not have any existing record for the region in the current monitoring period, it volunteers itself to be a leader of the region. Hence, whenever there are any events detected in a region, there will be at least one leader representative for the region. If there are already existing records for the region in the current monitoring period (which it receives from some other taxis), then it implies that

there are already at least one leader in the region. To minimize the possibility of duplicate leaders in the region, it shall not volunteer to be a leader.

Whenever a taxi encounters another taxi, they exchange records. Each taxi will send their taxi, client and leader records to the other taxi. If leader records for a region are different, the two taxis need to harmonize their leader records for the region. The recorded leader with a larger leader record score (ie. larger number of taxi and client records) is chosen and the taxi identifier is used as the tie breaker.

Next, for each region that a taxi is currently a leader, leader election is performed between the two connected taxis. Each taxi sends their current location to the other taxi. TADS picks the taxi that is currently closer in distance to the *vantage point* of the region that they are electing to be the leader. A vantage point is a pre-chosen location in the region that is assumed to have a higher chance of meeting other taxi nodes. A typical vantage point can be a road intersection whereby most taxis would pass by.

EncounterNode() in figure 4.2 shows the pseudocode when two taxis encounter and ElectLeader() in figure 4.3 shows the pseudocode for leader election based on vantage point.

Line 2 and 3 of EncounterNode() (figure 4.2) simply replicates *eligible* taxi, client and leader records to the other node (as required for monitoring). A record is considered *eligible* if the current node distance is within a pre-defined distance *dist* from the record respective region's vantage point. Both *dist* and regions' vantage point are pre-defined system parameters that is known to all the taxis in the system.

Having a large *dist* value allows taxis in vicinity to be more informed about the situation in region r. However, it results in more resource usage since records are now replicated to more taxis. Figure 4.4 shows an example in which the square in the centre represents the region of interest. The dark grey circle represents a taxi and its associated arrow represents its current trajectory. The

```
1: EncounterNode_i(Node j) {
2:     Send/Receive eligible taxi/client records.
3:     Send/Receive eligible leader records.
4:     For each leader records ldr^j received,
5:         if ((ldr.score^j > ldr.score^i) or (ldr.score^j = ldr.score^i and
   ldr.l^j > ldr.l^i))
6:             ldr^i ← ldr^j
7:
8:     new_Score ← compute_node_new_Score()
9:     For each leader records ldr in node i,
10:        if (ldr.l = i)
11:            ldr.x ← new_Score
12:            new_leader ← ElectLeader(ldr.r, i, j)
13:            if new_leader=j
14:                Send new_leader of ldr to j
15:                if send successful
16:                    ldr.l ← new_leader
17: }
```

Figure 4.2: EncounterNode()

```
1: ElectLeader(r, i, j) {
2:     dist^i ← getDistance(i, VantagePoint(r))
3:     dist^j ← getDistance(j, VantagePoint(r))
4:     if dist^i > dist^j
5:         new_leader ← j
6:     else if dist^i < dist^j
7:         new_leader ← i
8:     else
9:         if ID^i > ID^j
10:            new_leader ← i
11:        else
12:            new_leader ← j
13:    return new_leader
14: }
```

Figure 4.3: ElectLeader()

dotted circle represents the taxi wireless communication range. Suppose taxi A has some observation records for the region but it has now moved out of the region. It encounters taxi C who is travelling in the reverse direction, moving into the region soon. Assuming that their position is still within the distance *dist*, taxi A replicates its records to taxi C. Taxi C eventually enters the region, encounter taxi B, and replicates the records to taxi B. Noticed that in this case if replication is not performed beyond the region of interest, then records in taxi A would have been deleted and will not be replicated to taxi C.

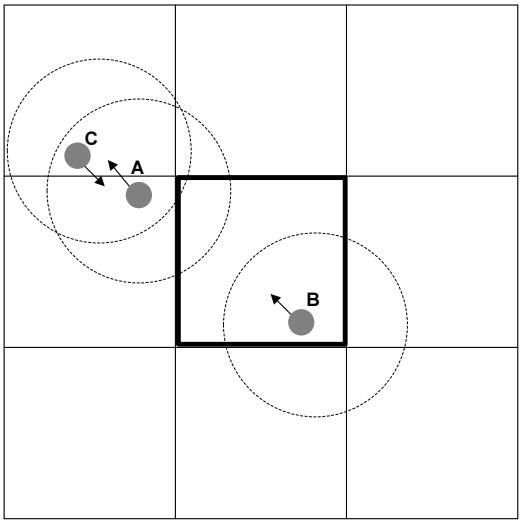


Figure 4.4: Example of replication beyond region of interest.

For line 4 to 6, taxi updates existing leader records if the newly received records show that the claimed leader has a higher ldr.score. ldr.score is defined to be the number of received client and taxi records. For example, if the leader record in taxi A for region r shows that taxi C is the leader, and the newly received leader record for region r shows that taxi D is the leader with a higher IQ score, then taxi A will update the leader records to indicate that taxi D is the leader for region r.

A taxi that has more records supposedly has a better picture of the state of the region compared to a taxi that has fewer records. Preferably, a leader taxi should have a better ldr.score so that it can make better decision in generating

tokens. Note that ldr.score is a monotonically increasing value.

For line 8-11, taxis update their ldr.score since they may have received new records due to line 2. In addition (line 12-16), for each region that the taxi is currently a leader, taxi calls the function ElectLeader(). ElectLeader() picks the taxi that is currently closer in distance to the vantage point of the region that they are electing to be the leader. The rationale for line 12-16 is to enable a leader to "transfer" leadership to a taxi that is closer to the vantage point. Since vantage point is a location chosen to have higher chances of meeting other taxis, it gives the leader higher chances of meeting other leaders (for leader elimination), and receiving new records (for better ldr.score).

Figure 4.5 shows the scenario of four regions in which taxi A is the elected leader for region W, Z and taxi B and C are leaders for region X and Y respectively. Taxi A has earlier driven through region Z and is at region W at the end of the monitoring period. It participated in the leader election for both regions and became the leader due to having a higher ldr.score for both regions. Even though taxi D has the same path as taxi A in this monitoring period, it is not chosen as the leader due to a lower ldr.score.

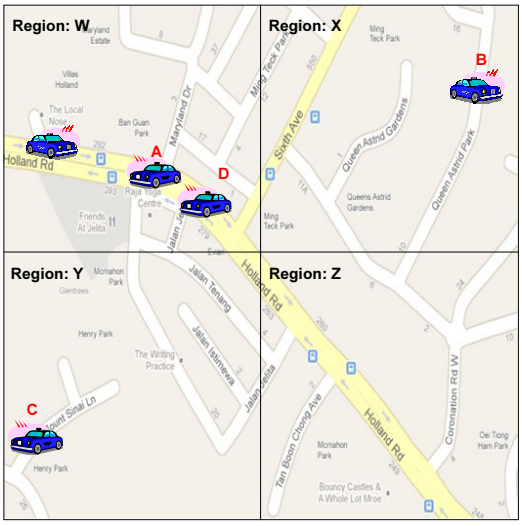


Figure 4.5: Example of leaders in regions.

4.3.3 Determining Number of Advisory Tokens

At the end of a monitoring period, each leader will individually determine the amount of advisory tokens to generate. As explained in section 4.3.1, the number of client records is often fewer than the actual number of clients. If we simply use the client record count without making any adjustment, we will miss out many regions that require more free taxis.

Instead of using the number of taxi and client records directly, we take into account the duration in which a taxi spends in either *FREE* or *POB* state in the region, as well as the amount of time a client has been waiting for a taxi (applicable only to smart client) to further improve the estimates. Intuitively, if a taxi spends more time in *FREE* than *POB* in a region, supply of free taxis may be higher in the region. Similarly, regions where clients have been waiting for longer periods of time may have a higher demand for taxi.

TADS utilizes a score-based system to better reflect the supply and demand for taxis. The score-based system also attempts to compensate for the fact that some advisory tokens may be unable to find a free taxi.

GetNumberOfAdvisoriesToGenerate() presents the pseudocode for a leader to determine the number of advisory tokens to generate for a region.

```
1:  GetNumberOfAdvisoriesToGenerate(Region r)
2:      S_r^supply ← cnt_r^FREE
3:      S_r^smart ← cnt_r^smart + inc_r^smart
4:      S_r^simple ← cnt_r^POB + inc_r^simple
5:      S_r^demand ← S_r^smart + S_r^simple
6:      n ← (S_r^demand − S_r^supply)
7:      if n > 0
8:          return n
9:      else
10:         return 0
```

inc_r^{smart} (line 3) adjusts for the number of smart clients. It is computed as follows:

$$inc_r^{smart} = \sum_{i \in r_{smart}} [k_1(i) + k_2(i)] \tag{4.1}$$

where

- $k_1(i) = 2^{\lfloor \frac{w_i}{d_x} \rfloor}$ if $2^{\lfloor \frac{w_i}{d_x} \rfloor} <= y$, otherwise $k_1(i) = y$,
- $k_2(i) = 0$ if $2^{\lfloor \frac{w_i}{d_x} \rfloor} <= y$, otherwise $k_2(i) = \lfloor \frac{w_i}{d_x} \rfloor - \lfloor \log_2(y) \rfloor$.
- $y = \frac{1}{\%\_smart\_in\_system} - 1$

w_i is the amount of time that a smart client i has waited, and d_x is a pre-defined duration that is currently set to be the advisory token lifetime. y is the number of simple clients per smart client in the system. The taxi company can provide an estimated value of y based on the recent data that has been collected by TADS. For example, taxis can upload their smart and POB records to the taxi company and these records can be used to predict y on a monthly basis.

Intuitively, inc_r^{smart} is assigned a higher value when smart clients in the region have waited for a longer period of time. It is initially set to increase exponentially, but slows down to increase linearly once it reaches some threshold (y) so that the number of tokens generated will not be too excessive. The idea is that if a smart client has waited for a long time, chances are that simple clients in the region may also have waited for a long time. (Note that there is no mechanism in TADS that can detect the waiting time for simple clients). Hence, TADS initially increases S_r^{smart} exponentially in the hope that it will also benefit the simple clients in the same region.

Next, let

- $d_r^{POB}(i)$ be the total duration that a taxi $i \in \mathbb{T}_r^{POB}$ was in status POB in the region r after picking up a client in region r.
- $d_r^{FREE}(i)$ be the total duration that a taxi $i \in \mathbb{T}_r^{FREE}$ was in status FREE in the region r.

Define cf_r as

$$cf_r = \frac{\sum_i d_r^{POB}(i)}{\sum_i d_r^{POB}(i) + \sum_i d_r^{FREE}(i)} \tag{4.2}$$

cf_r has a range of [0,1] and is set based on how long free taxis take to find and pickup clients in the region. We assume that the longer it takes, the fewer simple clients there are and cf_r is set to a smaller value.

$$inc_r^{simple} = cnt_r^{pob} \times cf_r \tag{4.3}$$

inc_r^{simple} (line 4 in GetNumberOfAdvisoriesToGenerate()) is a correction factor that tries to compensate for the number of simple clients that are not accounted for.

4.3.4 Forwarding Advisories

When two taxis meet, they exchange advisory tokens such that each taxi will keep half of the total number of advisory tokens per region. For example, if taxi A initially has $n > 0$ copies of advisories, and it encounters another taxi B (with no copies of advisories), taxi A will forward to taxi B $\lceil \frac{n}{2} \rceil$ copies of advisories and keeps $\lfloor \frac{n}{2} \rfloor$ copies of advisories itself. This approach of spreading advisories is similar to the Binary Spray and Wait approach in [26], which is shown to have the minimum expected delay in routing for nodes with IID mobility.

4.4 Simulation Evaluation

4.4.1 Evaluation Methodology

We evaluated our proposed system using traces obtained from a large taxi operator that operates more than 15,000 taxis in Singapore. The taxi trace includes the location and status (*FREE*, *POB* and etc) information but does not contain any connectivity information. In the evaluation, we use data from a particular

week in November 2010.

We assume that each taxi has a communication range of 50m, while client to taxi communication range is 30m (smartphone WiFi radios have a lower transmission range). A free taxi will move to pick up a client if it is within 30m from the client. To simulate the effects of queueing in taxi stands, if there are multiple clients at a single location, the taxi picks up the client with the longest waiting time.

The number of clients served in an hour varies from a few thousand to about twenty thousand, depending on the time of the day. In the results presented, "Current" refers to the existing system without TADS. We compare how TADS performs compared to "Current". We divide the area of interest (Singapore island) into fixed-size regions. Each region is a $400m \times 400m$ grid in our simulation. In the experiments, unless otherwise stated, the following parameters are used:

- Percentage of smart clients: 50%
- Guard duration: 30 seconds
- Monitoring period: 3 minutes
- Flooding radius ($dist$): 2 km
- Vantage point: Center of each region

Generating Client Request

For each taxi status that changed from *FREE* to *POB* in the trace, we create a client request record using the associated location and time. To avoid the same taxi picking up the client immediately when the client appears in the simulation, we defer the client arrival time by 10 minutes. Hence during simulation, this taxi would have likely left the location and it will be up to another taxi to pickup this client.

In addition, we note that such a way of generating records loses the "burstiness" of requests since client arrivals are constant offsets from the service times (taxi pickups) rather than the actual arrival times of clients which are not available.

In order to compensate for the loss of burstiness (which leads to shorter waiting time), we increase the number of client requests by 10% in the system. These additional arrivals follow the same time and location distributions of client arrivals obtained from the taxi traces.

Generating Taxi Paths

Taxis in TADS change their paths after they accept an advisory token or pickup a client. Since the clients' pickups are not the same as those in the original traces, paths taken by the taxis are also different. In the simulation, based on the traces, we create a database of taxi paths taken by a taxi.

Specifically, each taxi path extracted from the taxi trace consists of the following:

1. Taxi movement information (GPS coordinates and time)
2. *previous_status* - previous status of taxi
3. *current_status* - current status of taxi

Whenever a taxi reaches the last point in the path that it has been assigned, it will select a new path with the path starting location being closest in distance to the current taxi location. In addition, the path being picked must be time and status *compatible* with the current taxi. A path is time compatible if the path start time is within 90 minutes from the current time. A path is status compatible if the *previous_status* of the path matches the current status of the taxi.

In addition, we prevent taxi from picking a path with status FREE to POB.

For a taxi to change status from *FREE* to *POB*, it must physically pickup a client in our simluation.

Our scheme for selecting a taxi path has the following advantages:

- ability to capture current road conditions (eg. traffic jam).
- ability to capture the drivers' preferred routes at different times of the day (such as drivers moving to areas which they believe clients can be picked up easily).

Accepting/Rejecting Advisory Tokens

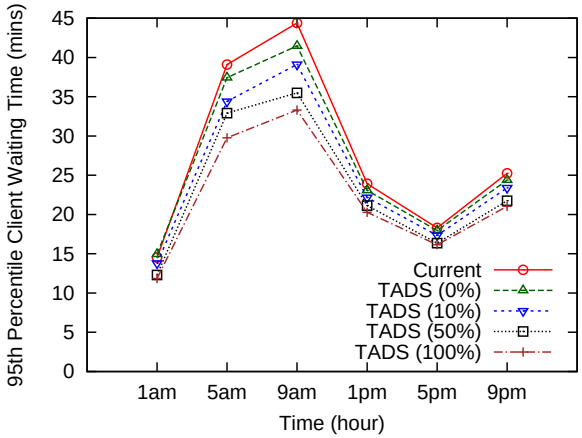
In the simulation, we use the following procedure to simulate acceptance/rejection of advisory token. When an advice is presented to a taxi, it randomly picks a time compatible path that allows the taxi to move from the current region to the requested region. If no such path exists or the path chosen takes more than the remaining lifetime of the token to reach the requested region, the taxi rejects the advice. Lifetime of each advisory token is set to 6 minutes.

In addition, the taxi will also reject an advisory if a client pickup is *eminent* (eg. taxi already saw a client nearby). If the taxi current path is able to pick up a client within 15 seconds, then we assume a pickup is eminent and the taxi will reject any advisories presented to it.

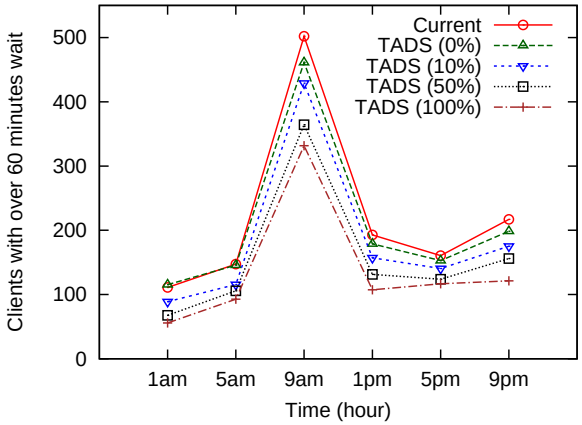
4.4.2 Varying Time of Day and Number of Smart Clients

In this section, we evaluate the performance of TADS with varying numbers of smart clients. As expected, TADS performs best when there are only smart clients (ie. 100% smart clients) in the system. In the busy hour of 9am to 10am, there is a 25% reduction in the 95th-percentile waiting time and 34% reduction in the number of clients with over 60 minutes wait.

With only simple clients in the system, in the 9am to 10am interval, there is only 7% reduction in the 95th-percentile waiting time and 8% reduction in



(a) Client Waiting Time



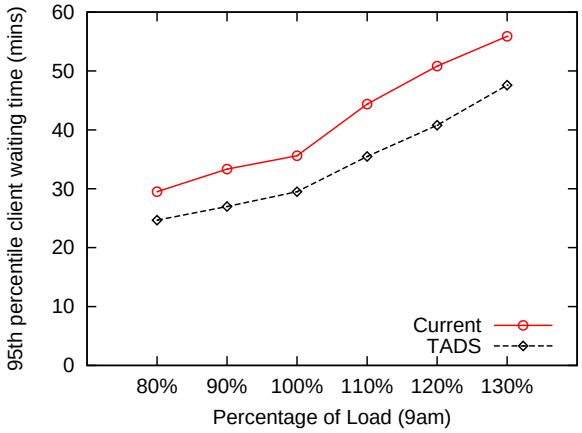
(b) Clients with over 60 minutes wait

Figure 4.6: Varying Time of Day and Smart Clients

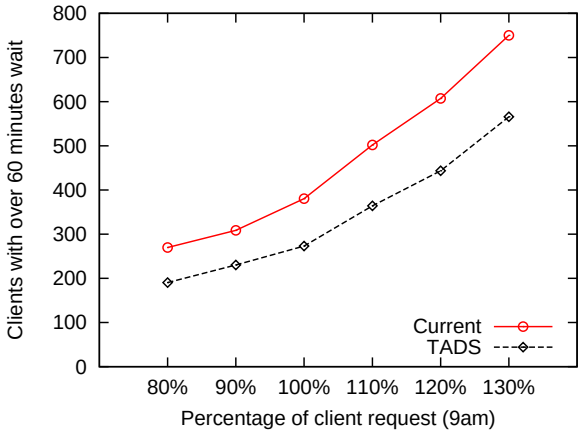
the number of clients with over 60 minutes wait. However, even with a modest 10% of smart clients in the system, the improvement is still noticeable, a 12% reduction in the 95th-percentile waiting time and 15% reduction in the number of clients with over 60 minutes wait.

In terms of average waiting time from 9am to 10am (not shown in the graph), the reduction in client waiting time is only 9% and 12% for 50% and 100% of smart clients in the system respectively. As we have highlighted previously, we do not expect the average times to reduce as much since TADS basically moves free taxi from "free" to "busy" regions and such load balancing occurs on the order of minutes.

4.4.3 Varying Load



(a) Client Waiting Time



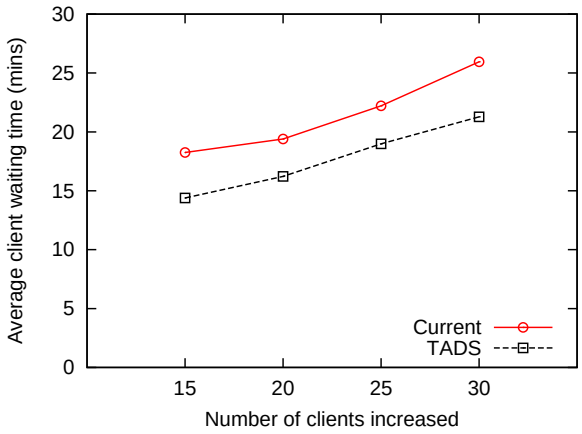
(b) Clients with over 60 minutes wait

Figure 4.7: Varying Load at 9am

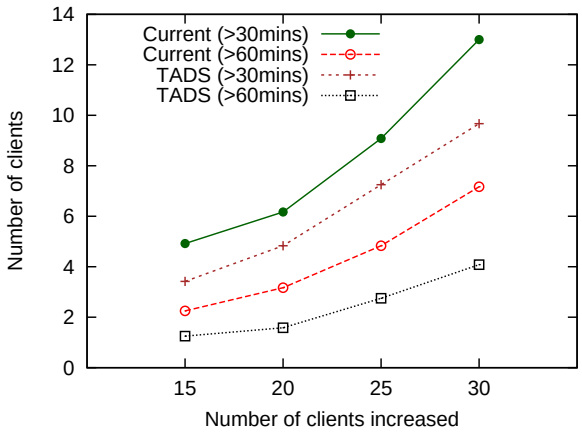
In figure 4.7, we vary the number of clients request at 9am. The load is varied as a percentage of the original number of client request at 9am. In all cases evaluated, TADS is able to reduce the client waiting time and the number of clients that has waited for over 60 minutes. At 130% load, the number of clients with over 60 minutes wait is 566 in TADS and 750 when TADS is not used.

4.4.4 Bursty Client Arrivals

In this section, we look at the effect of a sudden increase in client requests in a region. This corresponds to the situation such as a major show/game ending, resulting in a sudden influx of clients. We randomly select 1 region and increase the number of clients in the region over a 15 minutes period, starting at 9pm. Figure 4.8 shows the results for the selected regions.



(a) Average client waiting time



(b) Over 30 and 60 minutes wait

Figure 4.8: Increasing number of client request in a region

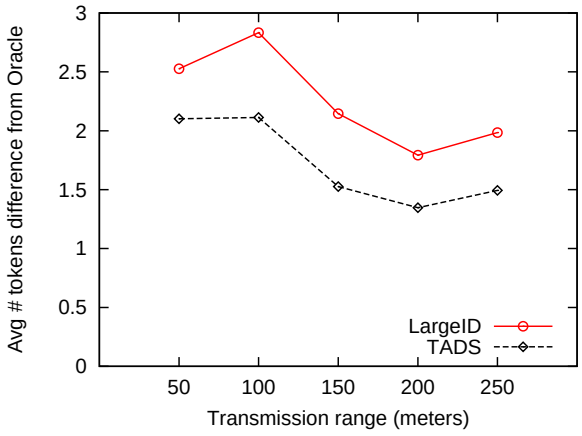
As expected, as the number of clients increase, the client waiting time also increases. However, TADS is able to reduce the average waiting time, as well as the number of clients who waited for more than 30 and 60 minutes significantly.

When we increase the number of clients by 30, the number of clients with over 60 minutes wait for TADS is 4 and 7 when TADS is not used.

4.4.5 Accuracy in Generating Advisories

In this section, we look at the accuracy in determining the number of advisories to generate. Due to nodes mobility and intermittent connectivity, a leader node often has incomplete information about a region. In addition, duplicate leaders for a region may also be possible. TADS mitigate the problem by performing flooding and leader elimination/transfer in an attempt for leaders to gain more knowledge and eliminate duplicate leaders for a region. We compare TADS against a simpler scheme (LargeID) in this section. In the LargeID scheme, nodes floods the known largest node ID in $\mathbb{T}_r$. The node with the largest known ID will be the leader for the region r.

Figure 4.9 shows the results comparing the accuracy in determining the number of advisories to generate. It shows the difference in the number of advisory tokens generated when compared to an oracle. To avoid comparing regions where there are no advisories generated, we only consider regions that generate at least one advisory tokens in this experiment.



(a) Average tokens difference

Figure 4.9: Accuracy in generating advisory tokens

As expected, TADS performs better than LargeID scheme for all the evaluated cases. Unlike LargeID scheme, TADS perform leader election to "transfer" leadership to a node closer to the vantage point of a region. This allows a leader to be closer to the region, and hence have higher opportunities to meet other "duplicate" leaders for the region. It also allows a leader in TADS to have higher chances of receiving new taxi records.

With a transmission range of 200m, the difference in the average number of tokens generated for TADS is 1.3 compared to Oracle. Note that the average number of tokens generated by the Oracle is 6.7.

4.4.6 Different strategies for Generating Advisories

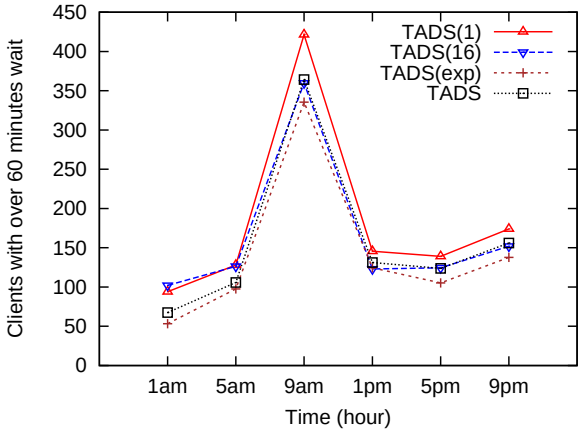
In this section, we evaluate 3 more different strategies for determining the number of advisory tokens to generate and compare them to TADS.

For TADS(1) and TADS(16), inc_r^{smart} in equation 4.1 is always set to 0. For every advisory token to be generated, the leader always generates 1 and 16 copies for TADS(1) and TADS(16) respectively. For TADS(exp), inc_r^{smart} in equation 4.1 is allowed to increase exponentially (ie. y in equation 4.1 set to ∞). Note that more copies of advisories lead to a higher chance of an advisory reaching a free taxi in a shorter time. The downside is that it risks requesting excessive taxis into the same region.

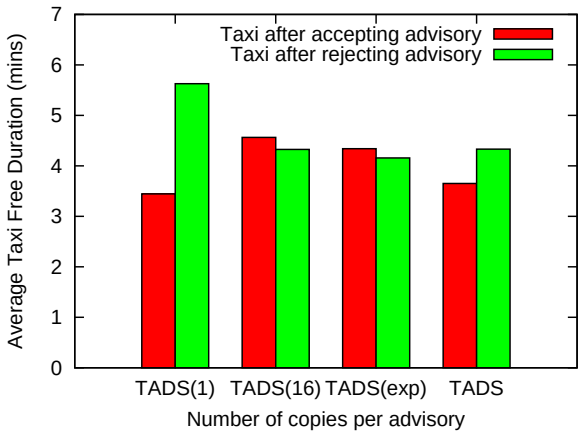
Figure 4.10 shows the result using the different schemes. TADS(1) performs the worst in terms of clients with over 60 minutes waiting time. However, in terms of average free taxi duration after a taxi accepts/rejects an advisory, it is the most attractive from the taxi driver's point of view. After accepting an advisory, the average time that the taxi takes to pickup a client is much shorter than if the taxi were to reject the advisory.

On the other hand, for TADS(16) and TADS(exp), too many tokens are generated and a taxi is better off rejecting the advisory than to accept.

The result shows that the proposed TADS scheme that dynamically adjusts



(a) Clients with over 60 minutes wait



(b) Average taxi free duration after accepting/rejecting advisory (9am)

Figure 4.10: Comparing different TADS schemes

the number of tokens based on load and waiting time is able to both reduce client waiting time as well as shorten taxi cruising time before picking up a client.

4.4.7 Related work

A number of research works have been proposed to improve taxi booking service. In [71], several temporal and spatial related advance bookings are chained and assigned to a single taxi. This reduces the cruising time between free to busy state when a taxi accepts a taxi booking.

In [72], the authors proposed to group taxi bookings within some time window

and then dispatch them to a group of taxis. This allows for better proximity assignment of taxis to clients.

In [5] [6], a three way handshake protocol is used to book taxi in a distributed taxi booking system. Their work however consider scenarios whereby multi-hop connectivity is present. Our work consider also the more general scenarios whereby disconnection may occur frequently due to mobility and the sparsity of taxis in some regions. Such multihop connectivity cannot be assumed.

In addition, our work is different from the above taxi booking systems in that we target the taxi service for "flag down" clients rather than clients served through taxi booking. No taxi bookings are involved in our system.

Ideally, we should assign taxis to regions so that the overall time taken by the taxis to travel to the requested regions is minimal. This is related to the Linear Assignment Problem (LAP). Even though decentralized solutions for LAP have been proposed [66] [67], they are not suitable for our work due to very fast changing and intermittent connectivity environment.

Part of TADS algorithm involves electing a taxi leader for each region. In the literature, there are a number of leader election algorithms that have been proposed for mobile ad hoc networks. We can broadly classify them here based on whether the algorithms use geographical information.

Leader election algorithms that do not use geographical information includes [73] [74] [75] [76] [77] [78].

In [74], a leader-oriented directed acyclic graph (DAG) within each connected component is maintained using link reversal techniques. This scheme is later extended by [75] to handle multiple topology changes. When the network stabilizes, the algorithm always give a unique leader for each connected component.

In [73] [77] [78], the authors present "extrema-finding" leader-election algorithms for mobile networks with the goal of electing as leader the node with the highest priority according to some criterion (eg. battery life). Both proposed algorithms are highly adaptive with ad hoc networks in the sense that it can tol-

erate link failures, network partitions, nodes recovery, and merging of connected network components associated with ad hoc networks.

The algorithms described above attempt to find a leader within a connected component, and do not consider geographical information. Leader election algorithms that include geographical information includes [79] [80] [81].

Hatzis et al. [80] presented a leader election algorithm for mobile ad hoc networks. When two or more mobile nodes meet, they exchange identities and the winner is the one with the higher identity. Similar to our work, this simple scheme does not guarantee that a unique leader can be found. A probabilistic analysis is then given assuming nodes perform random walks.

In [79], the entire geographical space is divided into regions and leader election is performed for each region assuming a single hop network. The leaders are elected for the purpose of forming a backbone for message propagation.

Chung et al. [81] presented a leader election algorithm for a region of interest. Nodes are assumed to be mobile and may fail, enter or exit the region of interest at any time. The work requires the existence of certain communication paths in the network to give a bound on the time for propagation of information within the region. With the assumption that such paths exist to bound communication delay between nodes, they show the correctness of their algorithm in electing a unique leader for the region. It is however, not clear how the mobile nodes can collaborate to form such a path if it doesn't already exist in the region.

Unlike most work presented above, our leader election involves electing a leader for a region and nodes may be required to communicate using multihop in the region. The closest work related to our leader election requirement is by Chung et al. [81]. However, there are a number of differences:

1. We only require node $i \in N_r$ to know whether it is a leader for region r. If node i is not a leader for region r, it does not need to know who is the leader for region r.

2. Node is elected to be a leader based on the evaluation of information that it has rather than some criteria such as node id. The information that a node has may change over time (eg. when it sensed or receive new information), this greatly differs from node id which is static.

3. Our leader election algorithm have to give an output at a fixed deadline (at the end of each monitoring period). It is not possible to guarantee that a algorithm will output a unique leader at the deadline. In [81], their work is focus on giving a unique leader, with no regards to time frame or any deadline.

4.4.8 Security Issues in TADS

In this section, we briefly discuss some security issues related to TADS. TADS relies on advisory tokens to guide free taxis to regions with more demand. Since every taxi can claim to be a leader taxi, this easily allows dishonest taxis to generate advisory tokens to mislead other taxis. A dishonest taxi may want to mislead other taxis to the wrong regions to increase its chances of picking up a client.

A possible mitigation to this problem is to have a unique ID for each advisory token generated by a taxi. The taxi is required to digitally sign the advisory token. In addition, any records generated or received must also be digitally signed. In the event of suspected dishonesty, the taxis can later come together for verification. The taxi can be checked against under/over generation of advisory tokens given the records that it has received. While this doesn't solve the problem online, the verification will serve as deterrence for dishonesty or malicious behavior. Other possible solutions include reputation schemes [82] [83] to mitigate the problem.

Note that it is out of the scope of this thesis to give a detailed solution for the security issues of TADS.

4.5 Conclusion

We have extended the role of nodes in the DTN to include completing application-related task in the system. Using the taxi scenario as a real life specific example, we provide a solution by proposing the Taxi Advisory Dispatch System (TADS). Unlike a taxi booking system that improves the service for call booking clients, TADS improves the service for "flag down" clients. Through simulation results, we showed that TADS is particularly useful for reducing long waiting times of clients. In addition, for regions with sudden influx of client request, TADS also perform very well in reducing the client waiting times.

Chapter 5

Robustness of Routing in DTN

The sucessful and efficient operation of a DTN involves cooperation among nodes in the network. Nodes in the network contribute but may also consume some resources (such as buffer space) in the network. Unfortunately, like any other network, malicious nodes may join and attack the network.

Early work on securing DTN largely depends on using public key cryptography to limit participants to a set of authorized nodes and using class of service for buffer space and link capacity allocation [17] [49] [50] [51]. In addition, packets injected into the network are authenticated at every intermediate hops. Such approaches incur considerable overhead and have to deal with the difficulty of key management in DTNs where communication delay may be long.

Due to difficulty of key management in DTN, some authors have looked into the possibility of forgoing authentication [1] [7]. Burgess et al. [1] study the robustness of DTN routing in the absence of authentication. They evaluated both forward-based and replication-based routing protocols using trace-based evaluation. A set of actions that are fundamental to any attacks in their model were investigated: dropping packets, route falsification, flooding packets and counterfeiting delivery acknowledgements. Their trace-based study suggests that some DTNs coupled with replication-based routing protocols are intrinsically fault tolerant, and robust even against a large number of attackers. This poses

the question on the necessity of authentication or the level of authentication required especially since authentication imposes overhead. Without authentication, it may encourage more nodes to join the network due to simplicity of joining. Having more nodes in the DTN provide more contact capacity and buffer resources for the network.

While Burgess et al. work suggest that some DTNs coupled with replication-based routing protocols are robust even against a large number of attackers, it remains unclear on why the protocols are robust or the kind of scenarios whereby the DTNs will be robust against the attackers. Nevertheless, Burgess's work has gained a number of citations in the research community. At the time of writing this thesis, their paper [1] has been cited over 60 times according to Google Scholar. We believe that having a better understanding on the robustness of DTN routing is important. In addition, we are also interested to know the security implications when application-aware routing protocols are employed in the DTN.

We revisit the Haggle and DieselNet DTNs that Burgess et al. have previously reported to be both robust against even a large number of attackers in the network. First, we note that while Burgess et al. have evaluated several attacks on the two DTNs, other variants of attacks are still possible. In this work, we devised a flooding attack called the *Non-Deliverable Packet Flooding* attack. Our non-deliverable packet flooding attack exploits metadata such as contact history to improve its flooding effectiveness. In addition, we present another attack called *Identity Impersonation* attack that causes relay and source nodes to drop packets that have not been delivered.

We observe that minimum hop count for packet delivery has a strong influence on the robustness of the DTN routing protocols. Generally, attacks become increasingly effective when the minimum hop count required increases. An observation we make in this work is that the small number of hops needed to deliver a packet in some DTNs (Haggle and DieselNet) is one of the reasons why they

are robust against the evaluated routing attacks.

In addition, we note that the increased effectiveness of non-deliverable packet flooding attack comes from the exploitation of routing metadata. Routing metadata such as the contact histories are flooded into the network so that relay nodes can make better decision in replicating/buffering packets. Given that our application-aware routing protocols also flood routing metadata (eg. dependency graphs) into the network, attackers can similarly exploit them to increase effectiveness in their attacks.

In the following sections, we first describe our system model followed by a description and evaluation on attacks in DTN routing. Next, we extend the attacks to exploit application-aware routing protocols. Specifically, we extend MaxProp routing protocol to become application-aware and show how attackers can exploit dependency graphs in their attack.

Our work suggests that for routing protocols which utilize routing metadata to improve routing performance, it is important to validate the contents of their control messages as attackers may be able to exploit such metadata to improve the effectiveness of their attacks. In addition, due to the replicative nature of replication-based routing protocols, it will be useful to employ rate limiting on the number of tasks/packets that each node can inject into the network.

5.1 System Model

In this section, we describe the security assumptions, mobility models used and properties of the routing protocol evaluated.

5.1.1 Security Assumptions

We assume that nodes do not perform authentication of relay nodes in the network. Similarly, no authentication is performed on the authenticity of messages. As a result, attackers can spoof their MAC layer addresses to appear to be

any node, including destinations of packets. Routing metadata and packets can also be spoofed and relay nodes have no means to verify their authenticity. We do, however, assume that end-to-end communicating parties may have keying materials that provide them with confidentiality, integrity and authenticity.

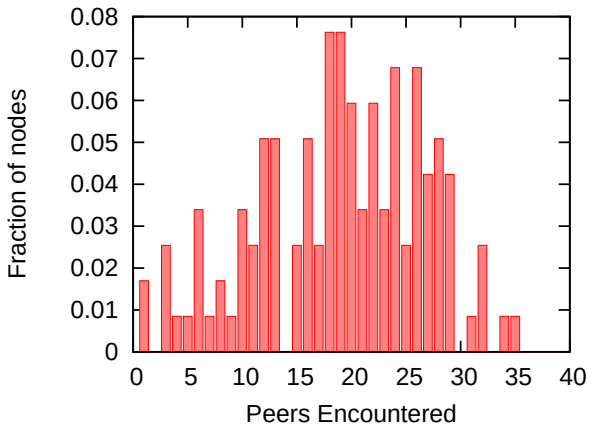
Finally, we also assume that attackers do not have global knowledge of DTN topology and future transfer opportunities. Stronger attackers with global knowledge and choice of location will be able to inflict much more damage than our attacker model here. However, we show that even with a weaker attacker model, attackers can still degrade the performance of the network considerably such that we need to be wary about DTN without authentication.

5.1.2 Mobility Models

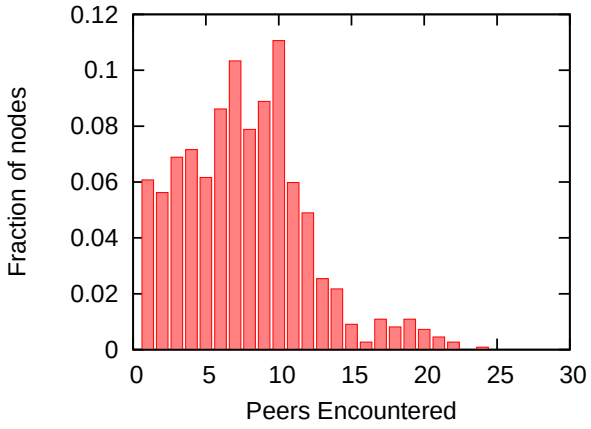
Our evaluation is based on real network traces, namely the DieselNet [9] and Haggle project [84] traces which are similarly used in [1]. The Haggle trace consists of a 3 days long trace that is based on a human mobility experiment in Infocomm 2005. A total of 41 volunteers joined the experiment and each were given an iMote device that can communicate with one another using Bluetooth. The iMotes are also capable of connecting to other Bluetooth-capable devices in the environment. Similar to the experiment in [1], we removed connection events from the Haggle data that lasted less than one second or involved the singular appearance of a node since meaningful data transfer is likely to require setup time and nodes incapable of routing data may be ignored. After the transformation, the traces are left with events involving 41 Class 1 devices (the iMotes devices) and none of the Class 2 devices (other Bluetooth-capable devices). In order to limit a single simulation interval to be 24 hours or less, we split the Haggle trace into 3 segments, each lasting about 1 day.

The DieselNet trace comprises of roughly 30 buses (with specific number varying according to the bus schedule). The median number of DieselNet buses in each trace is 19. Buses are outfitted with wireless transmitters and receivers.

Communications between the buses are done via the 802.11b protocol. DieselNet trace consists of 60 days of traces (captured during January to May 2005).



(a) Haggle



(b) DieselNet

Figure 5.1: Unique Peers Connected Daily

Figure 5.1 shows the number of peers encountered per mobile node for the two traces. Nodes in the Haggle traces have a broader distribution of the number of peers encountered. The median number of peers contacted by each node in the Haggle trace is 19, about 45% of the network. The median number of peers contacted by each node in DieselNet is 7, about 39% of the median number of buses in each trace.

5.1.3 Routing Protocol

The routing protocol used in our evaluation is MaxProp [9], a replication-based DTN routing protocol. MaxProp has been shown to provide robustness against various attacks [1]. It offers better throughput than several other strategies such as Epidemic [25], Prophet [8], Spray and Wait [26] and even Dijkstra algorithm with an oracle of future transfer opportunities [20].

While we use MaxProp in our study, our study is applicable to other replication-based routing protocols that use flooded routing metadata to guide replication and buffer management.

In terms of packet scheduling/replication, MaxProp replicates packets in the following order:

1. Packets destined to the contacted node
2. Routing metadata (estimations of the probability of meeting every other node)
3. Acknowledgements of delivered data.
4. Packets in ascending order of hop count for hop count below a certain threshold. This threshold is adaptive and is determined by using the average contact capacity measured from previous encounters.
5. Packets in descending order of delivery likelihood.

In terms of buffer management, MaxProp removes packets from its buffer in the following order:

1. Acknowledged packets.
2. Packets in ascending order of delivery likelihood for packets with hop count above a certain adaptive threshold.
3. Packets in descending order of packet hop count.

MaxProp uses network-wide acknowledgements to remove delivered packets from relay or source nodes, clearing up buffer and also prevent nodes from receiving packets that have already been delivered. An acknowledgement is created whenever a packet first reaches its destination. The acknowledgement is then flooded to all other nodes in the network.

As mentioned by Burgess et al. [1], to defend against acknowledgement counterfeiting, a node should ignore an acknowledgement if it has not seen the packet being acknowledged beforehand. In all our experiments, we implement this defense against acknowledgement counterfeiting.

In MaxProp and many similar DTN routing protocols [8–10, 32], routing metadata is kept and exchanged when two peers meet. Each node maintain a copy of its own table that describes the node contacts that it has observed in the past. Each contact history has an associated timestamp, indicating the time in which the direct contact occurs. These contact information or routing metadata will be replicated to other nodes during contact so that other nodes are aware of each others' contact history. When two nodes in the contact have different versions of the routing metadata entry, the copy with the earlier timestamp will be replaced. Figure 5.2 shows an example of MaxProp routing metadata that is stored at node A.

Based on the routing metadata exchanged, the data maintained in each contact table is used to estimate delivery likelihood. In the case of MaxProp, this likelihood is computed in the form of path cost. The higher the delivery likelihood, the lower the path cost. The cost of the path $i, i+1, ..., d$ is the sum of the probabilities that each connection on the path does not occur: $c(i, i+1, ..., d) = \sum_{x=i}^{d-1} \left[1 - \left(f_{x+1}^{x}\right)\right]$. In figure 5.2, the most likely path for delivering a packet from A to D is through node B, since the path cost ABD has the minimum value. The cost is computed as ABD=0.3 ((1 - 0.9) + (1 - 0.8)).

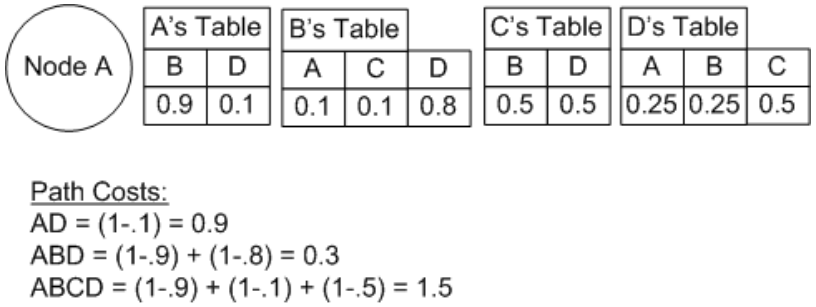


Figure 5.2: The organization of routing metadata in a node

5.2 Attack Model

In [1], four general attacks *Drop All*, *Random flooding*, *Invert routing metadata*, and *Acknowledgement counterfeiting* were experimentally shown to be ineffective. We briefly describe some of the factors that may limit the effectiveness of these attacks.

Drop All attack is not effective as there are still many possible paths to the packet's destination that does not involve the attackers.

In Random flooding attack, attackers randomly choose a known destination and flood packets to the chosen destination. The priority to replicate or drop the attackers' packets is on the same level as the non-attackers' packets. Hence the effectiveness of Random Flooding attack is limited by how fast the attackers can inject packets into the network to cause resource contention.

For Invert routing metadata attack, attackers invert the metadata of the probability of meeting other nodes. The goal is to cause the list of packets to be transmitted or dropped in the reverse order. Its effectiveness is limited by how resource constrained the network is. For example, if two peers meet and they have enough contact capacity to transmit all their buffer contents to the other node, then even if the list of packets are transferred in the reverse order, there is no performance degradation at all. Perhaps a more severe limitation of invert routing metadata attack is that inverting the routing metadata even number of

times will give the correct version of the routing metadata. For example, if an attacker sees the same routing metadata the second time, inverting it the second time gives correct version of the routing metadata.

In Acknowledgement counterfeiting, attackers send out falsified acknowledgements of known and yet to be delivered packets in the network. To counterfeit acknowledgement of a packet in the network, attackers must first know the existence of the packet in the network.

While the above attacks may be ineffective, many variations of these attacks are still possible. Furthermore, these attacks can be combined to reinforce one another.

5.2.1 Proposed Attack

Our proposed attack combines and uses a variation of the attacks in the previous section in an attempt to overcome the described limitations. It consists of two components. The first component, called *non-deliverable packet flooding* floods data to non-existent nodes to cause resource contention. It also includes *routing metadata falsification* that spoof routing metadata so that the flooded packets gets higher priority in replication and lower priority in being dropped. The second component, *identity impersonation*, impersonates different identities to act as destinations for packets. In addition, upon knowing the existence of a packet, attackers flood network-wide acknowledgements of the packet in an attempt to purge the packet out of the network.

The primary purpose of the first component is to cause network congestion, and to make relay nodes having a higher tendency to replicate attackers' packets and drop non-attackers' packets from their buffer. The objective of the second component is to purge packets from both the source and the relay nodes. We explain in detail the two components in the following sections.

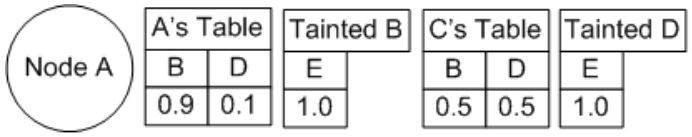
Non-Deliverable Packet Flooding

In Non-Deliverable Packet Flooding attack, attacker floods new packets to some non-existent destinations so that the flooded packets will not be delivered and stay in the network until expiry. However, with MaxProp or other routing protocols that rely on contact histories to estimate delivery time or probability, non-deliverable packets are actually given the lowest priority. Hence flooded packets are replicated last and dropped first. This is undesirable from the attacker's point of view. To counter that, attackers can perform routing metadata falsification by spoofing every node's routing metadata and claim that the node can reach the non-existent destination with high probability. More specifically in our experiments, the attacker removes all entries in a routing metadata table and creates an entry with meeting probability 1 to the non-existent destination.

Figure 5.3 shows node A's routing metadata with and without routing metadata attack. Node E is a non-existent destination and attackers flood packets to node E. With routing metadata attack, node A will give replication priority for packets in the order B, E, D, C. Further, if there is contention for buffer, packets destined to C will be dropped first. Without routing metadata attack, node A will give replication priority for packets in the order B, D, C, E and if there is contention for buffer, packets destined to E will be dropped first. This illustrates that routing metadata attack can successfully raise the priority of the attackers' flooded packets.

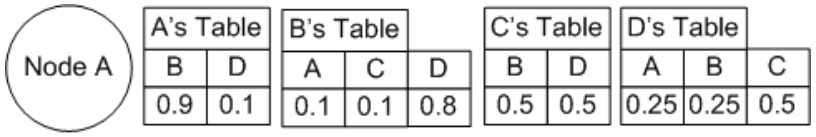
Effectiveness of metadata falsification

Let N_A be the number of attackers in the system. Consider an attack where N_A attackers keep injecting false routing information of a victim node, say node D. Let's call a node who is neither an attacker nor the victim a *carrier* node. Whenever an attacker meets a carrier node, it will send a *tainted* table (see figure 5.3 for example) of node D, which contains false information and time-stamped with the latest time. On the other hand, the victim also injects the correct table



Minimum costs to destinations:
B => AB = (1-.9) = 0.1
C => Infinity
D => AD = (1-.1) = 0.9
E => ABE = (1-.9) + (1-1) = 0.1

(a) Under Attack



Minimum costs to destinations:
B => AB = (1-.9) = 0.1
C => ABC = (1-.9) + (1-.1) = 1.0
D => ABD = (1-.9) + (1-.8) = 0.3
E => Infinity

(b) Without Attack

Figure 5.3: Comparison of node's routing metadata with/without attack.

to the carrier nodes. Recall that whenever two nodes meet, they will exchange and update their routing metadata to the one with the later timestamp. Note that only the attackers and victim will set the timestamp, carrier nodes simply help replicate the routing metadata without modifying the timestamp.

Now, under the above spreading process, we want to determine the fraction of carrier nodes having the tainted table. We claim that the fraction is $N_A/(N_A + 1)$ under reasonable assumptions. Let us consider this mobility model: The time is divided into periods of unit length. During each period, two randomly chosen nodes come into contact. The random nodes chosen in each period are independent to choices made in other periods. Let $X_{t,i}$ be the random variable where $X_{t,i} = 1$ if the node i's metadata is tainted at time t, and 0 otherwise. For convenience[1], let us assume that initially (i.e. at time 0), each node has the

[1]This assumption on the initial condition is not crucial. One may consider the initial con-

probability of $N_A/(N_A+1)$ being tainted (i.e. $Prob(X_{0,i}=1) = N_A/(N_A+1)$). We can show that, for any i and $t \geq 0$,

$$E(X_{t,i}) = \frac{N_A}{N_A+1}. \tag{5.1}$$

From (5.1) and linearity of expectations, the fraction of carrier nodes having a tainted table over all carrier nodes is also N_A/N_A+1. To show (5.1), let us consider a carrier node whose routing metadata originates from a malicious node or the victim, and trace back how the routing metadata spread from the source. We say that there is a path from node p_0 at time t_0 to node p_1 at time t_1, if there is a sequence

$$j_1 = p_0,\ s_1 = t_0,\ j_2, s_2, j_3, \ldots, j_{k-1}, s_{k-1} = t_1,\ j_k = p_1,$$

where node j_i and j_{i+1} meet during time period s_i, and the subsequence $s_1, s_2, \ldots$, is strictly increasing. Let us take $(t_1 - t_0)$ as the length of the path. Note that if there is a path from node p_0 to the victim, and it is shorter than every path to a malicious node, then the metadata in p_0 will not be tainted. Similarly, if there is a path to a malicious node, and every path to the victim is longer, then the metadata will be tainted. In other words, whether the metadata is tainted or not depends on whether the nearest node is the victim or a malicious node. Due to the independencies in choosing the two nodes in each time period, the probability that the nearest node is malicious is $N_A/(N_A+1)$.

To know the effectiveness of routing metadata falsification in the Haggle and DieselNet traces, we perform simulations on them to get the fraction of nodes having a tainted routing metadata. Each simulation was run till the end of the trace and the fraction of nodes having a tainted routing metadata is noted. The result presented here is the average of the different runs of the simulation. The

dition where all metadata are untainted. In this case, the fraction approaches $N_A/(N_A+1)$, instead of the equality we obtained in (1).

description of these traces and simulations can be found in section 5.1.2. Figure 5.4 shows that the traces in our simulation exhibit similar fraction of tainted nodes compared to our stochastic model here.

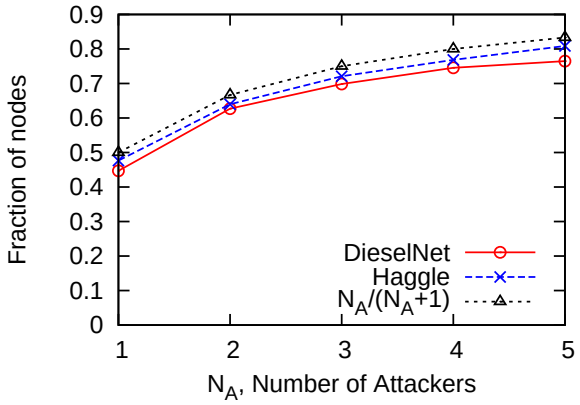


Figure 5.4: Fraction of nodes with tainted routing metadata

The result in figure 5.4 suggests that it is possible for very few attackers to launch an effective routing metadata falsification attack. This applies even for large networks with hundreds or thousands of nodes. Hence, we expect our non-deliverable packet flooding attack to benefit much from the use of routing metadata falsification.

Identity Impersonation Attack

In the identity impersonation attack, attackers impersonate different identities to act as destinations for packets so as to trick relay nodes or the packets' source node to believe that the packets have been delivered. Furthermore, upon knowing the existence of a packet, attackers flood network-wide acknowledgements of the packet into the network. Nodes that are tricked into believing that the packets have been delivered will drop the packets from their buffer.

Such attack directly exploits the lack of node authentication. In a single contact, an attacker can potentially take on the identities of many other nodes if the contact duration is sufficiently long. In the extreme case, all packets in a node's

buffer can be falsely removed. This is possible since frequent disconnections are the norms in DTN. This attack is most effective when the attacker encounters the source early in the packet forwarding process when the number of replicas of a packet in the network is low.

5.3 Evaluation

We evaluate the robustness of DTNs in the presence of attackers launching random flooding attacks (rf), non-deliverable packet flooding attacks (ndp), and identity impersonation attack (imp). For identity impersonation attack, we limit the switching of identity to at most once per second. All our evaluations were performed using our simulator that was modified from the ONE simulator [85], a simulator developed specifically for DTN simulations.

The traces used for simulation are the Haggle and DieselNet trace (see section 5.1.2 for description). In our simulation, we randomly assign nodes as honest and attacker nodes. All honest nodes generate traffic destined for other randomly chosen honest nodes. Each node have a 5MB buffer size and packets may be deleted before delivery when the buffer is full. When a packet is to be dropped due to buffer full, a node will always drop packets originating from other nodes before considering dropping its own packets. In all simulations, packets are fixed at size 10KB. Whenever load is too high, delivery rate is very low due to contention. In order to isolate the effects of the attackers, we use a moderate packet load of 10 packets/hr per honest node. Finally, in the identity impersonation attack, we assume that a malicious node can take on a new identity only once every second.

The transfer capacity of a single contact has an impact on the routing performance. In the Haggle trace, only contact duration is provided. If we assume the bluetooth device can transmit at 1Mbps, the median per-contact capacity will be approximately 25MB. In all our evaluations, unless otherwise stated, we

set the median per-contact capacity to be 25MB, including the DieselNet trace. Figure 5.5 shows the CDF graph of the per-contact capacity for the Haggle and DieselNet trace. In this setting, there is greater than 80% of the contact opportunities having enough capacity to transfer the full buffer contents of the two meeting nodes. The main resource contention here is hence the buffer.

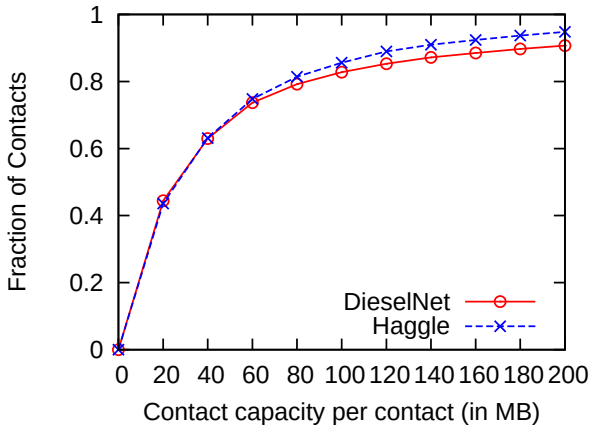
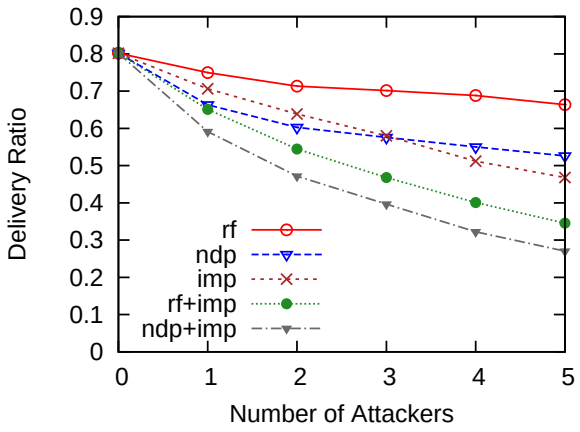


Figure 5.5: CDF of contact capacity

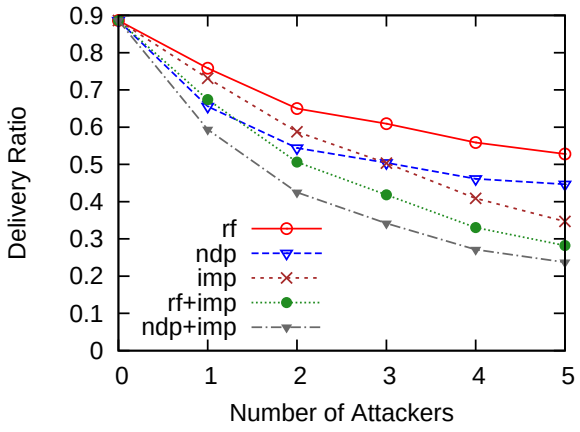
5.3.1 Impact of Varying Number of Attackers

From figure 5.6-5.8, it can be seen that non-deliverable packet flooding is effective even when there are only 1-2 attackers. In fact, the addition of more attackers does not help to bring the delivery ratio much lower. The reason is that 1-2 attackers is enough to cause the relay nodes' buffer to be filled with the attackers' packets due to the replicative nature of MaxProp protocol and high per-contact capacity. Further, since the packets are non-deliverable, they stay in the relay nodes' buffer for a long period of time, causing contention with other relay packets. For random flooding, there is less buffer contention since flooded packets may be delivered to the destination quickly, and these are removed from the relay nodes' buffer much faster. Furthermore, unlike non-deliverable packet flooding attack, random flooding attack does not manipulate the routing metadata to give the attackers' packets higher priority to stay in the buffer or be selected for

replication. Note that in the simulation, nodes always keep packets originating from itself. Hence even though relay packets are dropped, the source node still holds a copy of the packet and can still deliver the message through direct contact with the destination. Non-deliverable packet flooding fails to attack such direct contact delivery situation.



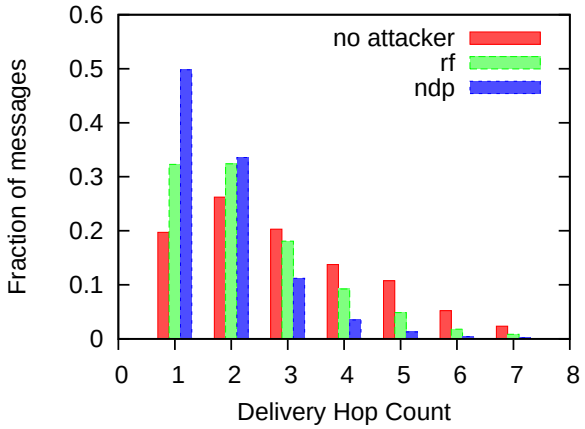
(a) Haggle



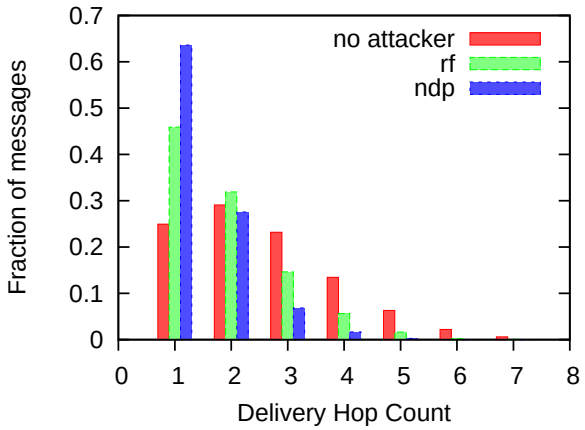
(b) DieselNet

Figure 5.6: Delivery Ratio under buffer contention

Figure 5.7 shows the hop count of messages at the time they were delivered to their destinations with 10% attackers. It can be seen that without flooding attacks, there are quite a number of packets delivered with 3-6 hop counts. On the other hand, with flooding attacks, most packets tend to be delivered with only 1 or 2 hop counts. The main reason is that flooding causes many packets



(a) Haggle

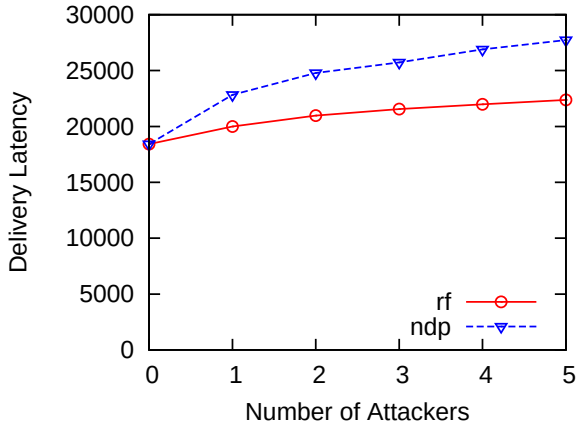


(b) DieselNet

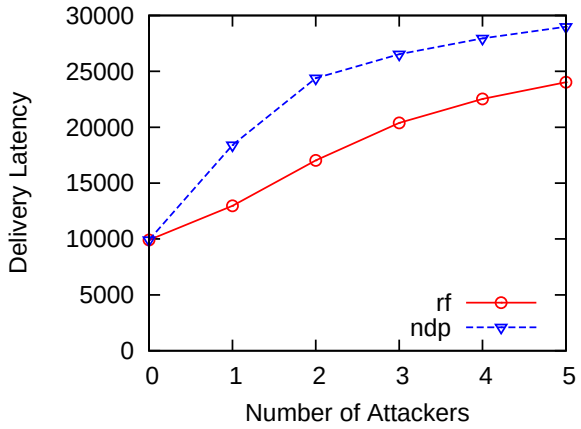
Figure 5.7: Message hop count at delivery (10% attackers)

to be dropped at relay nodes due to buffer contention. Since in our simulation, source always give higher priority in keeping its own packets, and due to mobility, the source might later meet the destination of the packets and send the packets to it directly. In other words, the capability of each node to eventually meet many other nodes provides substantial robustness against flooding attacks that causes packets to be dropped at relay nodes. Note however, the delivery latency is affected by flooding attacks, causing much higher delivery latency as shown in Figure 5.8.

Unlike non-deliverable packet flooding, impersonation attack is more effective when the number of attackers increases since launching the attacks require



(a) Haggle



(b) DieselNet

Figure 5.8: Delivery Latency (secs) under buffer contention

direct contact. The more attackers there are in the network, the more performance degradation it causes. Flooding attack and impersonation attack are complementary and can be launched together to cause more damage, as can be seen from figure 5.6.

5.3.2 Communicating Pairs Evaluation

In this section, we evaluate the routing performance of peers who are communicating across different distances (in terms of required hop counts). Our goal is to understand how non-deliverable packet flooding and identity impersonation attack affects communicating peers that communicates over different distances

in terms of hop counts. We first use a synthetic trace to better understand the effects of the attacks followed by further evaluation on the Haggle and DieselNet traces.

The synthetic trace imposes some structure so that it is possible to evaluate peers communicating with different number of minimum hop counts required. It consists of 40 nodes in a 5 by 5km area. The 40 nodes are divided into 8 different groups (each group consist of 5 nodes), and each node in a group move around an attraction point in the map with a standard deviation of 500m. The position of attraction points are randomly generated with the constraint that no two attraction points are within 1000m to each other. We generate 10 such synthetic traces for our simulation and the results reported are the averaged of the 10 traces.

In our evaluation of attacks, we place one attacker in one of the groups, call it group A. We want to evaluate the delivery ratio when an honest node in a group sends packets to another honest node in a certain group.

We divide the communicating pairs into the following category:

1. A-A: packets sent from a node in group A to another node in group A

2. A-B: packets sent from a node in group A to another node in group B. Group B's attraction point is at most 2000m from group A's attraction point.

3. A-C: packets sent from a node in group A to another node in group C. Group C's attraction point is at least 2000m away from group A's attraction point.

4. X-Y: packets sent from a node in group X to another node in group Y where there is no attacker in group X and Y. In addition, group X's attraction point is at least 2000m away from group Y's attraction point.

Figure 5.9 shows the distribution of hops taken when packets are delivered in

each category when there are no attackers. Majority of the packets in category A-A are delivered within 1-2 hops. For category A-C, C-A, X-Y and Y-X, the communication is further apart with majority of the packets delivered after going through more than 3 hops.

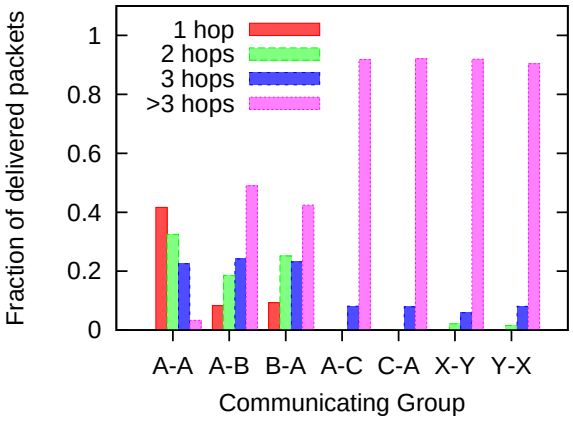


Figure 5.9: Distribution of hops taken

Figure 5.10 shows the delivery ratio of nodes communicating in different groups (recall that 1 attacker is placed in group A). When the number of hops required is low (eg. A-A), non-deliverable packet flooding do not have any effect, since eventually the source node may directly meet the destination. However, when the number of hops required is high (eg. A-C), communications between the two peers rely on relay nodes. Non-deliverable packet flooding causes relay nodes along the path to drop packets, and communications in category A-C are severely affected. The delivery ratio drops from 0.77 (without attacker) to 0.09 (one attacker). This demonstrates that for peers that require a few hops in order to communicate, non-deliverable packet flooding attack can have a serious impact on them.

Identity impersonation attack is more effective when the attacker is closer to the source of a packet, giving higher chance that the attacker eliminates the packet before it is replicated to many other nodes. This is especially clear when comparing category A-C and C-A under identity impersonation attack.

Delivery ratio for category A-C is only 0.11 compared to 0.46 for group C-A communication.

For group X-Y and Y-X communication, effectiveness of non-delivery packet flooding depends on the location of the attacker. If attacker is far from the communication path of X-Y and Y-X, then it may fail to effectively taint the relay nodes routing metadata. In such cases, relay nodes will then drop the attacker's flooded packets when there is buffer contention. Identity impersonation attack also does not work well in such cases since by the time the attacker learns about the existence of a packet and try to flood counterfeit acknowledgements into the network, the packet may have already been delivered or replicated many times and is close to being delivered.

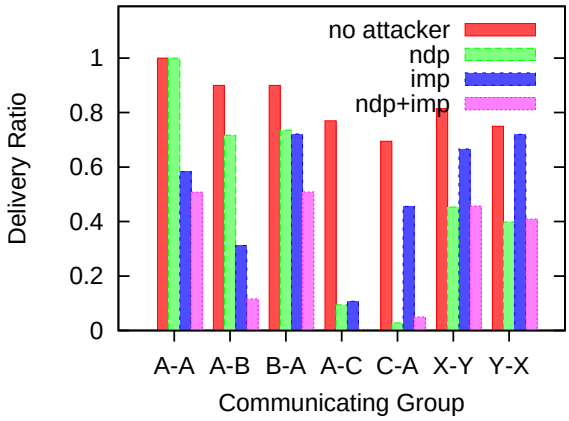
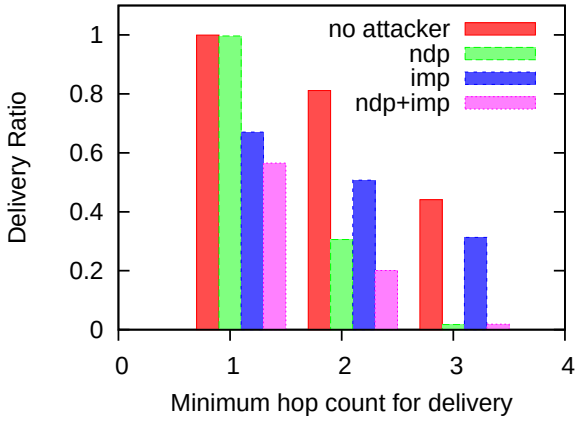


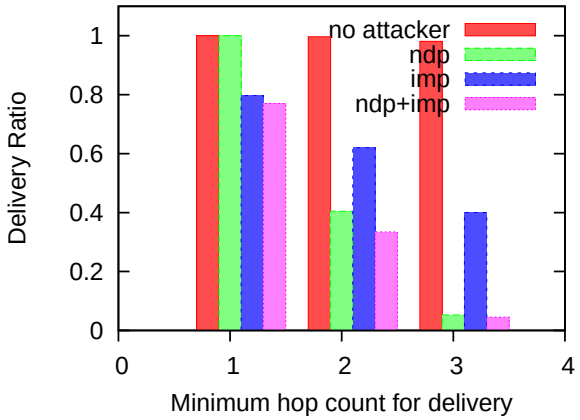
Figure 5.10: Communicating pairs evaluation (1 attacker in Group A)

We now move on to study the impact of such attacks using the Haggle and DieselNet traces. Figure 5.11 shows the delivery ratio based on the minimum hop count required for a packet to be delivered to the destination. We did not show the results for minimum hop count that is greater than 3 as the number of these packets are too little. Similar to what we observed in the synthetic trace, packets with high minimum hop count required for delivery are severely affected by non-deliverable packet flooding attack. Packets with minimum hop count of 1 for delivery are not affected by non-deliverable packet flooding attack, but it

is still susceptible to identity impersonation attack.



(a) Haggle

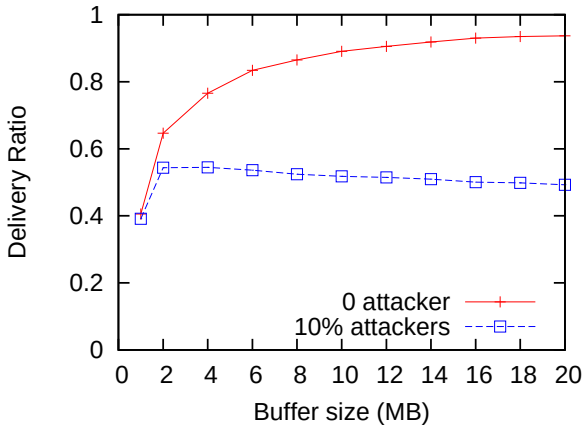


(b) DieselNet

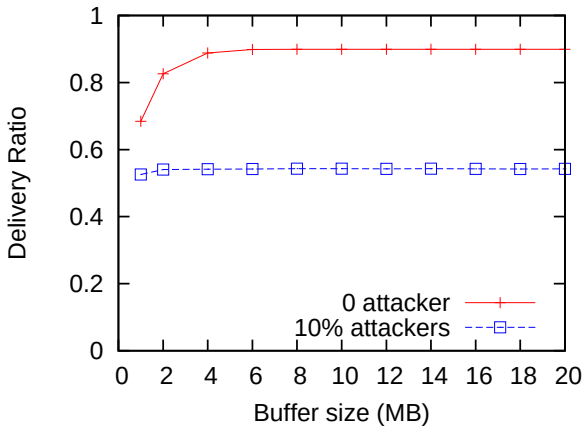
Figure 5.11: Delivery Ratio with different minimum hop count for delivery (10% attackers)

5.3.3 Study Impacts of Varying Buffer Sizes

We investigate whether increasing buffer size makes MaxProp more robust against non-deliverable packet flooding attack. Figure 5.12 shows that increasing buffer size does not help in making MaxProp more robust against non-deliverable packet flooding attack. Even with additional buffer, the attackers' packets quickly filled up the buffer, causing similar level of resource contention.



(a) Haggle



(b) DieselNet

Figure 5.12: Delivery Ratio varying buffer size

5.4 Attacks on Application-Aware Routing

We have seen that the increased effectiveness of our attacks comes from the exploitation of routing metadata. Routing metadata such as the contact history are flooded into the network so that relay nodes can make better decision in replicating/buffering packets. Given that our application-aware routing protocols also flood routing metadata (eg. dependency graphs) into the network, attackers can similarly exploit them to increase effectiveness in their attacks. In the following sections, we first extend MaxProp into an application-aware protocol. Next we show how attackers can exploit the application-aware dependency

graphs to increase effectiveness in their attacks.

5.4.1 Application-Aware MaxProp

Extending MaxProp to be application-aware requires us to change the cost computation for MaxProp routing. Figure 5.3 shows an example of how MaxProp computes cost to various destinations. This is done without regards to how likely the packets can help to complete the application task.

Similar to DSAR, we define *task contribution* for a task as the ratio of undelivered data blocks of the task in the node's buffer over the total number of (estimated) undelivered blocks of the task. The value of task contribution captures the utility of a set of data blocks for task completion (see section 3.2.3). The undelivered blocks of the task can be estimated easily due to the use of network wide acknowledgements in MaxProp.

In the *Application-Aware MaxProp* (*A-MaxProp*), nodes give higher priority to tasks (and their associated data blocks) that have higher chances of completion and delivery. Let the task contribution value of a task t be $TC(t)$. To deliver the task however comes with a path cost $c(t)$, which is the cost of the path from current node in consideration to task t's destination. (See section 5.1.3 on the computation of path cost in MaxProp). In A-MaxProp, we define the define the utility value of a data block i to be the sum of all its associated task's contribution value per unit of path cost:

$$U(i) \quad = \quad \sum_{t \in T} [\frac{TC(t)}{c(t)}] \tag{5.2}$$

where T is the set of tasks that data block i is associated with.

$U(i)$ is a utility function that takes into consideration on both the prospect of completing a task, and the cost of delivering the packet. Replication are performed first for data blocks having a larger utility value.

5.4.2 Application-Aware Attack

In this section, we show how flooding and acknowledgement counterfeiting can be extended to attack application-aware routing protocols.

Attackers can improve the effectiveness of flooding attack in application-aware protocols by making each attacker's flooding packet appear as if it contributes a lot to the completion of a task. A simple way to increase the effectiveness of flooding attack in A-MaxProp is to simply flood the network with many tasks that consist of one data block. From the relay node's point of view, this data block will have a task contribution factor of 1 since delivering this data block will be equivalent to completing a task in the network. This gives the attackers' flooding packets higher priority compared to others that may have a smaller task contribution factor.

To extend identity impersonation to attack application-aware routing protocols, an attacker can choose to counterfeit acknowledgements for only one data block in each task. The idea is to prevent the task from completing, yet allow other data blocks of the task to continue to consume resources in the network.

We will evaluate the effectiveness of these two variants of application-aware attacks in the following sections.

5.5 Evaluating Application-Aware Attacks

We evaluate the application-aware attacks using application-aware random flooding attacks (*arf*), application-aware non-deliverable packet flooding attacks (*andp*), and application-aware identity impersonation attack (*aimp*). These attacks are compared to their respective non-application-aware counterparts in the evaluation.

We use the same Haggle and DieselNet trace as in the previous sections. Each node has a 5MB buffer size and packets may be deleted before delivery

when the buffer is full. When a packet is to be dropped due to buffer full, a node will always drop packets originating from other nodes before considering dropping its own packets.

We use file transfer as the application in the evaluation (ie. all data blocks of the file must be transferred for the task to be considered as complete). All honest nodes generate a file every hour that is destined for other randomly chosen honest nodes. The number of data blocks for each generated file is randomly chosen to be in the range of [1,20] data blocks.

In *arf* and *andp*, attackers create many tasks with only one data block per task and flood them into the network. In the non-application-aware flooding (*rf* and *ndp*), attackers create many tasks with task size randomly chosen from the range of [1,20]. In addition, note that for *arf* and *rf*, the packets' destinations are randomly chosen among the honest nodes while for *andp* and *ndp*, the destinations are non-existent nodes.

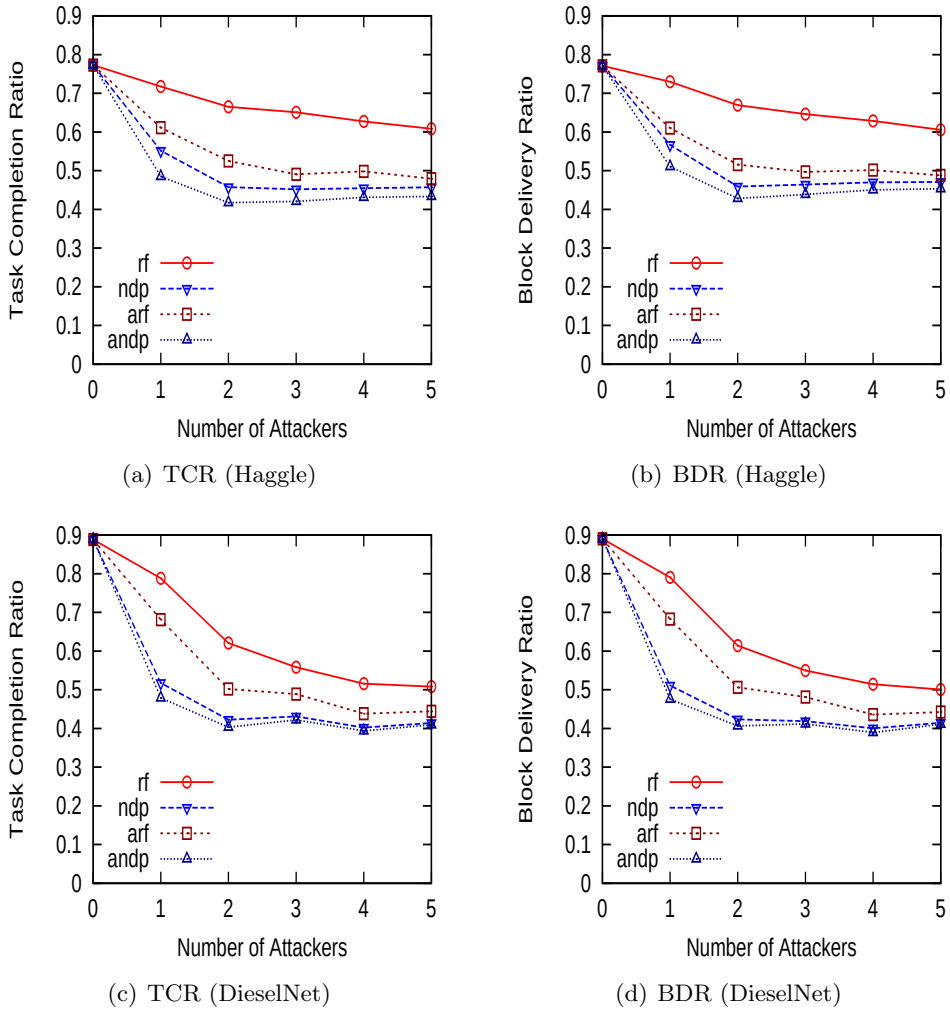
In *aimp*, attackers counterfeit acknowledgements for the first data block of each task. In the non-application-aware identity impersonation attack (*imp*), attackers counterfeit acknowledgements for all data blocks of each task.

The metrics for evaluation are Task Completion Ratio (TCR) and Block Delivery Ratio (BDR) (defined in section 3.3).

Figure 5.13 shows the results comparing the various flooding attacks on A-MaxProp. It can be seen that application-aware flooding attacks improve the effectiveness of both random and non-deliverable packet flooding attacks. The reason is that relay nodes tend to give data blocks from small tasks higher priority since delivering such data blocks is equivalent to completing tasks in the network.

When there are two attackers in the Haggle network, the TCR is 0.66 for *rf* and 0.53 for its application-aware counterpart (*arf*). For *ndp*, TCR is 0.46 and 0.42 for its application-aware counterpart (*ndp*).

Figure 5.14 shows the result comparing the various identity impersonation



(a) TCR (Haggle) (b) BDR (Haggle)

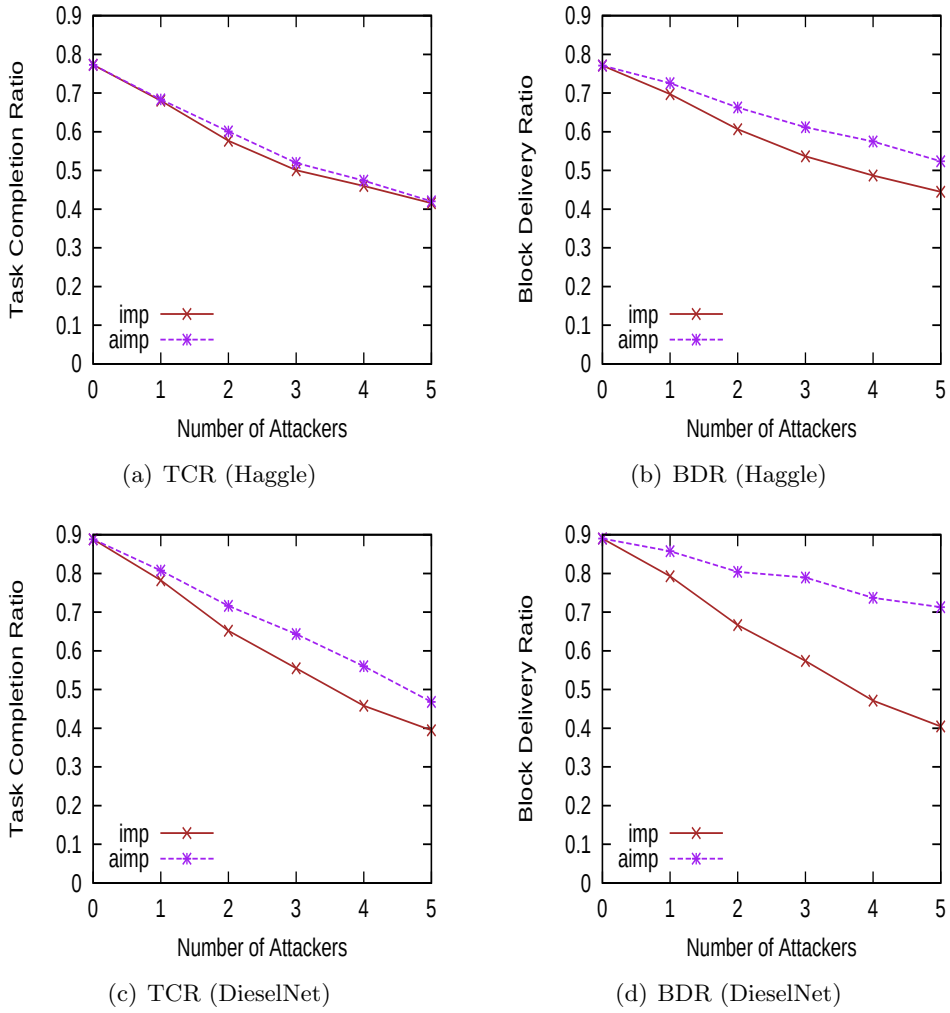
(c) TCR (DieselNet) (d) BDR (DieselNet)

Figure 5.13: Application-Aware Flooding

attacks on A-MaxProp. For application-aware identity impersonation attack (*aimp*), attackers only counterfeit acknowledgement for 1 data block per task. As a result, the BDR remains high compared to its non-application-aware counterparts (*imp*). For instance, when there are 5 attackers in the DieselNet, the BDR for *aimp* is 0.71 and the BDR for *imp* is only 0.40. This possibly makes *aimp* a more stealthy attack compared to *imp*.

The goal of *aimp* is to reduce the TCR, which it is able to do so effectively as shown in the figure 5.14. Note that *aimp* is however, still less effective compared to the non-application-aware *imp*. When there are 5 attackers in the DieselNet,

the TCR for *aimp* is 0.47 and 0.39 for *imp*. The reason is as long as there is one honest node in aimp that is still holding the targeted data block, aimp can still complete that task.



(a) TCR (Haggle)

(b) BDR (Haggle)

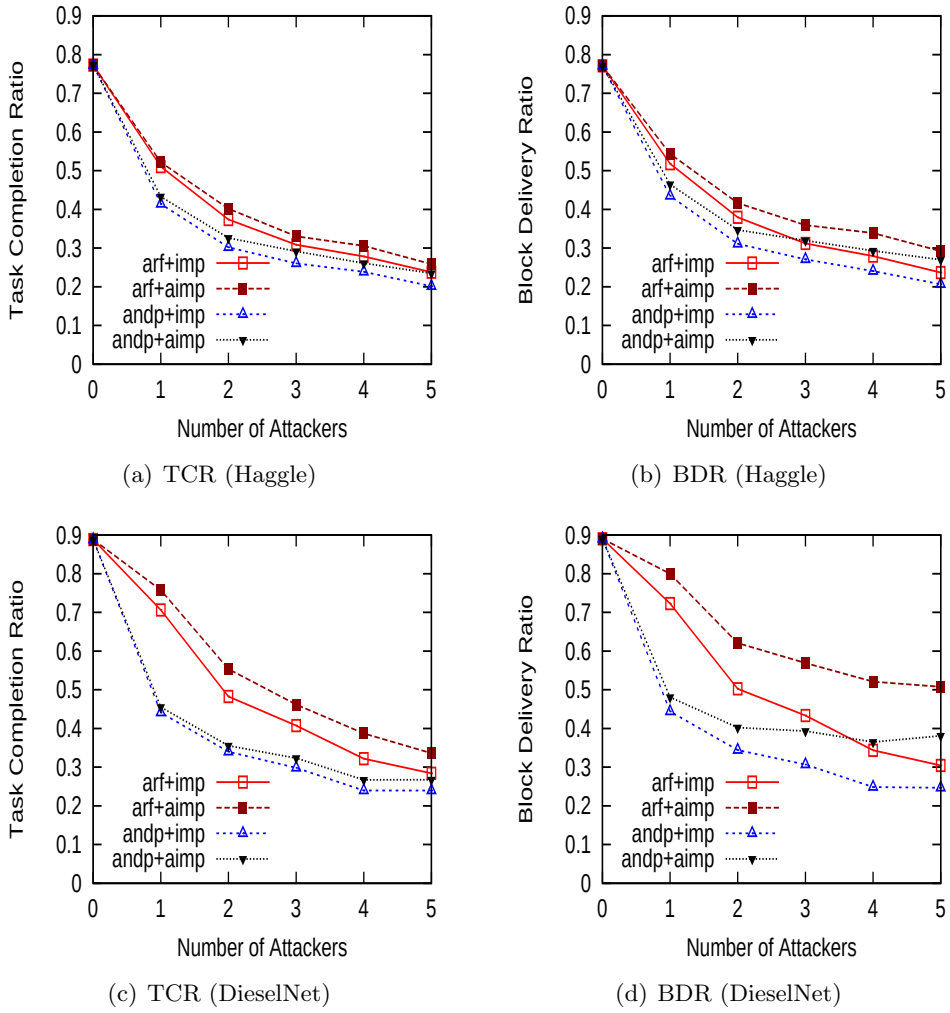
(c) TCR (DieselNet)

(d) BDR (DieselNet)

Figure 5.14: Application-Aware Identity Impersonation

Finally, we combine the flooding and identity impersonation attacks and evaluate their effectiveness.

Figure 5.15 shows the result comparing the various combinations of attacks on A-MaxProp. The most effective combination of attacks is *andp+imp*. In this attack, attackers can remove many honest nodes' packets through identity impersonation and floods many non-deliverable packets to the honest nodes. In



(a) TCR (Haggle) (b) BDR (Haggle)

(c) TCR (DieselNet) (d) BDR (DieselNet)

Figure 5.15: Combining Application-Aware Attacks

andp+aimp, the flooding is not as successful since there are more honest nodes' packets (due to *aimp* minimally counterfeiting acknowledgements) to contend with the attackers' flooding packets.

5.6 Discussion

Previous sections discussed the attacks without any routing authentication. What if some form of authentication can be performed? We consider two levels of authentication here:

1. Authenticate the identity of the peer in an encounter

2. Authenticate the identity of the peer in an encounter, routing metadata and acknowledgements

In (1), nodes in the network authenticate the peer when there is an encounter. Nodes can easily authenticate each other based on public key in a certificate. Overhead is low but in this case, it only prevent against identity impersonation (partially). The attacker will not be able to impersonate as the destination of packets, but it will still be able to flood fake acknowledgements. Note that non-deliverable packet flooding attack is not affected by such authentication at all. As such, authentication at this level is not effective against our attacks.

In (2), besides authenticating the identity of the peer, it also authenticates the routing metadata and acknowledgements. For acknowledgements, they are signed by the destination node. The overhead involved is much higher compared to (1), but it can fully prevent identity impersonation. As for non-deliverable packet flooding attack, flooding to non-existent destination is still possible. However, the metadata falsification component is thwarted. Hence, non-deliverable packets will be correctly determined by relay nodes that it is unlikely to be delivered. In this case, the relays may choose to drop these packets in the event of buffer contention. As a result, non-deliverable packet flooding will not be effective. It should be noted however, *tailgating* can be launch with non-deliverable packet flooding attacks. The attacker can tailgate the target destination node for a sufficient period of time. This allows non-deliverable packets to be seen as more deliverable by relay nodes. The effectiveness of non-deliverable packet flooding is hence improved. For solutions to attacks with tailgating, reader is referred to the paper [86].

In addition, one should note that both levels of authentications alone cannot prevent application-aware flooding attacks. The main problem with our application-aware routing protocols is that it allows the node itself to use depen-

dency graphs to specify the importance of its packets in completing a task. This makes it impossible to know if the node is lying or cheating. A possible way to prevent resource misuse here is to perform admission control to limit the rate of task injection for each node in the network [87] [88] [89].

5.7 Conclusion

Routing metadata such as contact history, acknowledgements, dependency graphs and etc. that are employed in DTN routing to improve resource management can similarly be exploited by attackers to improve the effectiveness of attacks. We have presented two attacks - non-deliverable flooding and identity impersonation attacks that - that demonstrates how attackers can exploit routing metadata to improve the effectiveness of attacks.

Using non-deliverable flooding and identity impersonation attack, we evaluated MaxProp routing on Haggle and DieselNet which has earlier been thought to be robust against even a large number of attackers. We observe that the small number of hops needed to deliver a packet is one of the main reasons why they are more robust against routing attacks. Nevertheless, our non-deliverable flooding and identity impersonation attacks can still degrade the routing performance of Haggle and DieselNet considerably. The increased effectiveness of our attack mainly comes from the exploitation of routing metadata such as the contact history used in MaxProp.

Due to the fact that our proposed application-aware routing protocols send out dependency graphs as routing metadata, attackers can similarly exploit them to enhance their attacks. To demonstrate the attack, we extended MaxProp to application-aware (A-MaxProp), and show how attackers can modify non-deliverable flooding and identity impersonation to attack A-MaxProp.

Our work demonstrates the importance of authenticating routing metadata that may potentially be exploited by attackers to create highly effective attacks.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Unlike traditional networks, DTNs are characterized by intermittent connectivity. Nodes may experience frequent disconnections and long communication delay in DTNs. While there are existing works that focus on improving the performance of DTN, they do not look into how it may benefit or improve the applications running on the DTN. With the unique characteristic of DTN, we believe that it is beneficial to the applications if the DTN are application-aware.

In our first work, we have shown that in the face of inherent intermittent connectivity, knowledge of application semantics or requirements can be exploited in routing to improve application performance in the network. We proposed a mechanism to capture application semantics into dependency relationships. The mechanism is general and can be used to model a large class of applications. Finally, we showed how to incorporate dependency relationships into existing DTNs routing algorithms to enhance application performance.

In our second work, we looked into a class of applications in which nodes in the DTN participate in completing application-related tasks in the system. We looked at a possible real life taxi scenario that falls into the described class of applications and proposed the Taxi Advisory Dispatch System (TADS). TADS

is a distributed system in which taxis collaboratively estimate the number of free taxis and clients in each region. Due to communication uncertainty in DTNs, instead of having all nearby taxis agreeing upon the region they should move to next, we proposed to use a leader approach in which a taxi is chosen to be the representative leader node for a region based on some criteria after some time of monitoring. If a region requires more free taxi, then the leader for that region sends out an appropriate number of advisory tokens to request for free taxis to move into the region. We evaluated TADS using real-life taxi traces that consist of over 15,000 taxis. Our evaluation results showed that TADS can reduce the number of clients with wait times longer than 60 minutes by over 30%.

Lastly, we studied attacks that affect the resource usage in the DTN. We revisited the Haggle and DieselNet DTNs that Burgess et al. [1] have previously reported that both the DTNs (with no authentication mechanisms) are robust against even a large number of attackers. We showed how techniques that are employed by many routing protocols to improve resource usage can similarly be exploited by attackers. Specifically, we showed how routing metadata such as contact history and acknowledgements can be exploited to improve the effectiveness of attacks and we identified the scenarios where DTNs are most vulnerable to such attacks. In addition, we showed how attackers can increase the effectiveness of attack in our application-aware routing protocols through the manipulation of dependency graphs. Finally, we gave a discussion on the level of authentication that is required to secure the attacks that we presented.

6.2 Future Work

In our current work, we have proposed a mechanism to capture application semantics based on dependency relationships. Future work can improve on the mechanism to capture more kinds of application semantics. For example, our current dependency graph does not capture “exclusion” relationship. Consider

an application such that the existence of some data blocks invalidates the usefulness of some other data blocks. In this case, relay nodes upon knowing the existence of the data blocks can drop those data blocks that are now considered as useless.

Another interesting area of research is on routing protocols with limited copies of replication. We have worked on protocols that are epidemic in nature (eg. MaxProp, SAR, DSAR), but there are other protocols that limit the number of replication allowed (such as Spray and Wait). We believe that for large network with many nodes, protocols that are purely epidemic in nature may be faced with severe resource constraints as each relay node may now be tasked to buffer replicate copies of data from many nodes in the network. Hence, it makes sense to limit the copies of replication for such large network. It is interesting to design application-aware routing protocols with adaptive limit on the copies of replication.

Limited copies of replication may require a smarter routing algorithm to improve delivery. However, a smarter routing algorithm may also be easier to be exploited by attackers. In addition, the robustness or fault tolerance of the routing algorithm is lower with limited copies of replication. For example, if every message can have only one replica, then dropping attack will be successful if the attackers manage to drop both the message. Researching into this area will be interesting.

Bibliography

[1] J. Burgess, G. D. Bissias, M. D. Corner, and B. N. Levine, "Surviving attacks on disruption-tolerant networks without authentication," in *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*. Montreal, Quebec, Canada: ACM Press, 2007, pp. 61–70.

[2] T. P. et. al., "Coolspots: Reducing the power consumption of wireless mobile devices with multiple radio interfaces," in *Mobisys*, 2006.

[3] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*. Kluwer Academic Publishers, 1996, pp. 153–181.

[4] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *IN PROCEEDINGS OF THE 2ND IEEE WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS*, 1997, pp. 90–100.

[5] P. Zhou, T. Nadeem, P. Kang, C. Borcea, and L. Iftode, "Ezcab: A cab booking application using short-range wireless communication," in *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, march 2005, pp. 27 –38.

[6] J.-P. Sheu, G.-Y. Chang, and C.-H. Chen, "A distributed taxi hailing protocol in vehicular ad-hoc networks," in *Vehicular Technology Conference (VTC 2010-Spring), 2010 IEEE 71st*, may 2010, pp. 1 –5.

[7] M. Y. S. Uddin, A. Khurshid, H. D. Jung, C. A. Gunter, M. Caesar, and T. F. Abdelzaher, "Making dtns robust against spoofing attacks with localized countermeasures." in *SECON*. IEEE, 2011, pp. 332–340.

[8] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *Lecture Notes in Computer Science*, vol. 3126, pp. 239–254, January 2004.

[9] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networking," in *Proceedings of IEEE Infocom 2006*, Barcelona, Spain, April 2006. [Online]. Available: http://prisms.cs.umass.edu/brian/pubs/burgess.infocom2006.pdf

[10] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "DTN Routing as a Resource Allocation Problem," in *Proc. ACM Sigcomm*, Kyoto, Japan, August 2007. [Online]. Available: http://www.sigcomm.org/ccr/drupal/?q=node/273

[11] J. R. Department and J. Reich, "Robot-sensor networks for search and rescue," in *In Proc. IEEE Intl Workshop on Safety, Security and Rescue Robotics*, 2006.

[12] H. SUGIYAMA, T. TSUJIOKA, and M. MURATA, "Qos routing in a multi-robot network system for urban search and rescue," in *Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 02*, ser. AINA '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 323–330.

[13] J. Burke, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava, "Participatory sensing," in *Workshop on World-Sensor-Web, co-located with ACM SenSys*, 2006.

[14] M. Paxton and S. Benford, "Experiences of participatory sensing in the wild," in *Proceedings of the 11th international conference on Ubiquitous computing*, ser. Ubicomp '09. New York, NY, USA: ACM, 2009, pp. 265–274.

[15] R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. F. Abdelzaher, "Greengps: a participatory sensing fuel-efficient maps application," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 151–164.

[16] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe, "Parknet: drive-by sensing of road-side parking statistics," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 123–136.

[17] K. Fall, "A delay tolerant network architecture for challenged internets," in *Proc. of Annual Conf. of the Special Interest Group on Data Communication (ACM SIGCOMM'03)*, August 2003, pp. 27–34. [Online]. Available: http://citeseer.ist.psu.edu/728928.html

[18] W. Zhao, Y. Chen, M. Ammar, M. Corner, B. Levine, and E. Zegura, "Capacity enhancement using throwboxes in dtns," 2006.

[19] N. Banerjee, M. D. Corner, and B. Levine, "An Energy-Efficient Architecture for DTN Throwboxes," in *Proceedings of Infocom 2007)*, Anchorage, Alaska, May 2007. [Online]. Available: http://www.cs.umass.edu/~nilanb/papers/banerjee06-39.pdf

[20] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," August 2004.

[21] E. P. C. Jones, L. Li, and P. A. S. Ward, "Practical routing in delay-tolerant networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, Philadelphia, PA, USA, August 2005, pp. 237–243. [Online]. Available: http://www.acm.org/sigs/sigcomm/sigcomm2005/paper-JonLi.pdf

[22] J. Leguay, T. Friedman, and V. Conan, "Evaluating mobility pattern space routing for DTNs," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM)*, Barcelona, Spain, April 2006. [Online]. Available: http://www-rp.lip6.fr/site_npa/site_rp/_publications/669-infocom06.pdf

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

[24] Q. Yuan, I. Cardei, and J. Wu, "Predict and relay: an efficient routing in disruption-tolerant networks," in *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '09. New York, NY, USA: ACM, 2009, pp. 95–104. [Online]. Available: http://doi.acm.org/10.1145/1530748.1530762

[25] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Department of Computer Science, Duke University, Durham, NC, Tech. Rep., 2000.

[26] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *WDTN '05: Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. New York, NY, USA: ACM Press, 2005, pp. 252–259.

[27] ——, "Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility," in *Proceedings of the Fifth*

IEEE International Conference on Pervasive Computing and Communications Workshops, ser. PERCOMW '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 79–85. [Online]. Available: http://dx.doi.org/10.1109/PERCOMW.2007.108

[28] K. A. Harras, K. C. Almeroth, and E. M. Belding-Royer, "Delay tolerant mobile networks (dtmns): Controlled flooding in sparse mobile networks," in *In IFIP Networking*, 2005.

[29] Y. Wang, S. Jain, M. Martonosi, and K. Fall, "Erasure Coding Based Routing for Opportunistic Networks," in *Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking.* Philadelphia, PA: ACM Press, August 2005.

[30] J. Widmer and J.-Y. Le Boudec, "Network coding for efficient communication in extreme networks," in *WDTN '05: Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking.* New York, NY, USA: ACM Press, 2005, pp. 284–291.

[31] R. Ramanathan, R. Hansen, P. Basu, R. Rosales-Hain, and R. Krishnan, "Prioritized epidemic routing for opportunistic networks," in *Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking*, ser. MobiOpp '07. New York, NY, USA: ACM, 2007, pp. 62–66. [Online]. Available: http://doi.acm.org/10.1145/1247694.1247707

[32] B. Burns, O. Brock, and B. Levine, "Mv routing and capacity building in disruption tolerant networks," vol. 1, 2005, pp. 398–408 vol. 1.

[33] E. C. R. de Oliveira and C. V. N. de Albuquerque, "Nectar: a dtn routing protocol based on neighborhood contact history," in *Proceedings of the 2009 ACM symposium on Applied Computing*, ser. SAC '09. New York, NY, USA: ACM, 2009, pp. 40–46. [Online]. Available: http://doi.acm.org/10.1145/1529282.1529290

[34] J. Davis, A. Fagg, and B. Levine, "Wearable computers as packet transport mechanisms in highly–partitioned ad–hoc networks," in *Proceedings of IEEE Intl. Symposium on Wearable Computers.*, October 2001, pp. 141–148.

[35] M. Grossglauser and M. Vetterli, "Locating nodes with ease: last encounter routing in ad hoc networks through mobility diffusion," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, vol. 3, 2003, pp. 1954–1964 vol.3.

[36] M. Grossglauser and D. N. C. Tse, "Mobility increases the capacity of ad hoc wireless networks," *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, pp. 477–486, Aug. 2002. [Online]. Available: http://dx.doi.org/10.1109/TNET.2002.801403

[37] S. N. Diggavi, M. Grossglauser, D. Tse, and I. Summary, "Even one-dimensional mobility increases ad hoc wireless capacity," in *In Proc. of IEEE ISIT*, 2002.

[38] Q. Li and D. Rus, "Sending messages to mobile users in disconnected ad-hoc wireless networks," in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking.* New York, NY, USA: ACM Press, 2000, pp. 44–55.

[39] M. Shin, S. Hong, and I. Rhee, "Dtn routing strategies using optimal search patterns," in *Proceedings of the third ACM workshop on Challenged networks*, ser. CHANTS '08. New York, NY, USA: ACM, 2008, pp. 27–32. [Online]. Available: http://doi.acm.org/10.1145/1409985.1409992

[40] M. M. Bin Tariq, M. Ammar, and E. Zegura, "Message ferry route design for sparse ad hoc networks with mobile nodes," in *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*,

ser. MobiHoc '06. New York, NY, USA: ACM, 2006, pp. 37–48. [Online]. Available: http://doi.acm.org/10.1145/1132905.1132910

[41] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *MobiHoc 2004: Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. New York, NY, USA: ACM Press, May 2004, pp. 187–198.

[42] ——, "Controlling the mobility of multiple data transport ferries in a delay-tolerant network," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 2, 2005, pp. 1407–1418 vol. 2.

[43] A. Gil, K. Passino, S. Ganapathy, and A. Sparks, "Cooperative scheduling of tasks for networked uninhabited autonomous vehicles," in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 1, dec. 2003, pp. 522 – 527 Vol.1.

[44] A. Kansal, A. A. Somasundara, D. D. Jea, M. B. Srivastava, and D. Estrin, "Intelligent fluid infrastructure for embedded networks," in *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, ser. MobiSys '04. New York, NY, USA: ACM, 2004, pp. 111–124. [Online]. Available: http://doi.acm.org/10.1145/990064.990080

[45] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data mules: modeling a three-tier architecture for sparse sensor networks," in *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on*, may 2003, pp. 30 – 41.

[46] Y. Gu, D. Bozdag, E. Ekici, F. zgner, and C.-G. Lee, "Partitioning based mobile element scheduling in wireless sensor networks," in *IN. PROC. SECOND ANNUAL IEEE CONFERENCE ON SENSOR AND AD HOC COMMUNICATIONS AND NETWORKS (SECON*, 2005, pp. 386–395.

[47] Y. Gu, D. Bozdag, and E. Ekici, "Mobile element based differentiated message delivery in wireless sensor networks," in *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, ser. WOWMOM '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 83–92. [Online]. Available: http://dx.doi.org/10.1109/WOWMOM.2006.73

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

[49] R. Durst, "A infrastructure security model for delay tolerant networks," july 2002. [Online]. Available: http://www.dtnrg.org/docs/papers/dtn-sec-wp-v5.pdf

[50] A. Seth and S. Keshav, "Practical security for disconnected nodes," in *Proceedings of the 1st IEEE ICNP Workshop on Secure Network Protocols*, 2005.

[51] A. Kate, G. M. Zaverucha, and U. Hengartner, "Anonymity and security in delay tolerant networks," in *3rd International Conference on Security and Privacy in Communication Networks*, 2007.

[52] D. Djenouri and N. Badache, "Struggling against selfishness and black hole attacks in manets," *Wirel. Commun. Mob. Comput.*, vol. 8, no. 6, pp. 689–704, 2008.

[53] D. Hongmei, L. Wei, and A. Dharma P., "Routing security in wireless ad hoc networks," *IEEE Communications magazine*, October 2002.

[54] Y. C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: a defense against wormhole attacks in wireless networks," vol. 3, 2003, pp. 1976–1986 vol.3.

[55] Cristina and H. Rubens, "An on-demand secure routing protocol resilient to Byzantine failures," in *ACM Workshop on Wireless Security (WiSe)*, Atlanta, Georgia, September 2002.

[56] S. Yi, P. Naldurg, and R. Kravets, "A security-aware routing protocol for wireless ad hoc networks," in *in: Proceedings of ACM Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc)*, 2001, pp. 286–292.

[57] Y. chun Hu, "Rushing attacks and defense in wireless ad hoc network routing protocols," in *in ACM Workshop on Wireless Security (WiSe)*, 2003, pp. 30–40.

[58] 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, 2007.

[59] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: social-based forwarding in delay tolerant networks," in *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '08, New York, NY, USA, 2008.

[60] B. B. Chen and M. C. Chan, "Mobtorrent: A framework for mobile internet access from vehicles." in *INFOCOM'09*, 2009, pp. 1404–1412.

[61] *Single-copy routing in intermittently connected mobile networks*, 2004.

[62] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," CS-200006, Duke University, Tech. Rep., 2000.

[63] "apt-cache - APT package handling utility: cache manipulator," http://linux.die.net/man/8/apt-cache.

[64] M. Piorkowski, N. Sarafijanovoc-Djukic, and M. Grossglauser, "A parsimonious model of mobile partitioned networks with clustering,"

in *The First International Conference on COMmunication Systems and NETworkS (COMSNETS)*, January 2009. [Online]. Available: http://www.comsnets.org

[65] J. S. Jing Su, P. Hui, J. Crowcroft, E. de Lara, C. Diot, A. Goel, M. H. Lim, and E. Upton., "Haggle: Seamless Networking for Mobile Applications." in *Proceedings of the Ninth International Conference on Ubiquitous Computing (UbiComp 2007)*, Innsbruck, Austria, October 2007.

[66] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-Based decentralized auctions for robust task allocation," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 912–926, Aug. 2009.

[67] S. Ponda, J. Redding, H.-L. Choi, J. P. How, M. Vavrina, and J. Vian, "Decentralized planning for complex missions with dynamic communication constraints," in *American Control Conference (ACC)*, Jul. 2010, pp. 3998–4003.

[68] "Public Transport Monthly Figures for Taxi," Land Transport Authority of Singapore, 2011, http://www.lta.gov.sg/content/dam/lta/Corporate/doc/Taxi%20Info%20for%20LTA%20Website%202011.pdf.

[69] "Land Transport Masterplan," Land Transport Authority of Singapore, 2008, http://www.lta.gov.sg/content/lta/pdf/LTMP_Report.pdf.

[70] "Average Hourly Passenger Waiting Time," Land Transport Authority of Singapore, 2011, http://www.lta.gov.sg/content/dam/lta/publictransport/doc/Website-Nov11.pdf.

[71] H. Wang, D.-H. Lee, and R. Cheu, "PDPTW based taxi dispatch modeling for booking service," in *Fifth International Conference on Natural Computation*. IEEE, Aug. 2009, pp. 242–247.

[72] K. T. Seow, N. H. Dang, and D.-H. Lee, "A Collaborative Multiagent Taxi-Dispatch System," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 3, pp. 607–616, Jul. 2010.

[73] A. Boukerche and K. Abrougui, "An efficient leader election protocol for mobile networks," in *Proceedings of the 2006 international conference on Wireless communications and mobile computing*, ser. IWCMC '06. New York, NY, USA: ACM, 2006, pp. 1129–1134. [Online]. Available: http://doi.acm.org/10.1145/1143549.1143775

[74] N. Malpani, J. L. Welch, and N. Vaidya, "Leader election algorithms for mobile ad hoc networks," in *Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications*, ser. DIALM '00. New York, NY, USA: ACM, 2000, pp. 96–103. [Online]. Available: http://doi.acm.org/10.1145/345848.345871

[75] R. Ingram, P. Shields, J. Walter, and J. Welch, "An asynchronous leader election algorithm for dynamic networks," in *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, may 2009, pp. 1 –12.

[76] P. Parvathipuram, V. Kumar, and G.-C. Yang, "An efficient leader election algorithm for mobile ad hoc networks," in *Proceedings of the First international conference on Distributed Computing and Internet Technology*, ser. ICDCIT'04. Berlin, Heidelberg: Springer-Verlag, 2004, pp. 32–41. [Online]. Available: http://dx.doi.org.libproxy1.nus.edu.sg/10.1007/978-3-540-30555-2_5

[77] S. Vasudevan, J. Kurose, and D. Towsley, "Design and analysis of a leader election algorithm for mobile ad hoc networks," in *Network Protocols, 2004. ICNP 2004. Proceedings of the 12th IEEE International Conference on*, oct. 2004, pp. 350 – 360.

[78] S. Masum, A. Ali, and M.-y. Bhuiyan, "Asynchronous leader election in mobile ad hoc networks," in *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on*, vol. 2, april 2006, p. 5 pp.

[79] F. Kuhn, N. Lynch, and C. Newport, "The abstract mac layer," in *Proceedings of the 23rd international conference on Distributed computing*, ser. DISC'09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 48–62. [Online]. Available: http://dl.acm.org/citation.cfm?id=1813164.1813176

[80] K. P. Hatzis, G. P. Pentaris, P. G. Spirakis, V. T. Tampakas, and R. B. Tan, "Fundamental control algorithms in mobile networks," in *Proceedings of the eleventh annual ACM symposium on Parallel algorithms and architectures*, ser. SPAA '99. New York, NY, USA: ACM, 1999, pp. 251–260. [Online]. Available: http://doi.acm.org/10.1145/305619.305649

[81] H. C. Chung, P. Robinson, and J. L. Welch, "Regional consecutive leader election in mobile ad-hoc networks," in *Proceedings of the 6th International Workshop on Foundations of Mobile Computing*, ser. DIALM-POMC '10. New York, NY, USA: ACM, 2010, pp. 81–90. [Online]. Available: http://doi.acm.org/10.1145/1860684.1860701

[82] Z. Xu, Y. Jin, W. Shu, X. Liu, and J. Luo, "Sred: A secure reputation-based dynamic window scheme for disruption-tolerant networks," in *Military Communications Conference, 2009. MILCOM 2009. IEEE*, 2009, pp. 1–7.

[83] M. Azer, S. El-Kassas, A. Hassan, and M. El-Soudani, "A survey on trust and reputation schemes in ad hoc networks," in *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, 2008, pp. 881–886.

[84] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Pocket switched networks and human mobility in conference environments," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, Philadelphia, PA, USA, August 2005, pp. 244–251. [Online]. Available: http://www.acm.org/sigs/sigcomm/sigcomm2005/paper-HuiCha.pdf

[85] A. Keränen and J. Ott, "Increasing reality for dtn protocol simulations," Helsinki University of Technology, Tech. Rep., July 2007. [Online]. Available: http://www.netlab.hut.fi/~jo/papers/2007-ONE-DTN-mobility-simulator.pdf

[86] F. Li, A. Srinivasan, and J. Wu, "Thwarting Blackhole Attacks in Distruption-Tolerant Networks using Encounter Tickets," in *Proceedings of IEEE Infocom 2009*, 2009.

[87] M. Demmer, "Dtnserv: A case for service classes in delay tolerant networks," in *Intelligent Computer Communication and Processing, 2008. ICCP 2008. 4th International Conference on*, aug. 2008, pp. 177 –184.

[88] E. Johnson, H. Cruickshank, and Z. Sun, "Managing access control in delay/disruption tolerant networking (dtn) environment," in *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*, feb. 2011, pp. 1 –5.

[89] E. Johnson, G. Ansa, H. Cruickshank, and Z. Sun, "Access control framework for delay/disruption tolerant networks," in *Personal Satellite Services*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, K. Sithamparanathan, M. Marchese, M. Ruggieri, I. Bisio, O. Akan, P. Bellavista, J. Cao, F. Dressler, D. Ferrari, M. Gerla, H. Kobayashi, S. Palazzo, S. Sahni, X. S. Shen, M. Stan,

J. Xiaohua, A. Zomaya, and G. Coulson, Eds. Springer Berlin Heidelberg, 2010, vol. 43, pp. 249–264.