PEON: PRIVACY-ENHANCED OPPORTUNISTIC NETWORKS WITH APPLICATIONS IN ASSISTIVE ENVIRONMENTS

by

GAURI VAKDE

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2010

To my parents Shailaja and Arun Vakde, and my grandmother Hemlata Khopkar.

ACKNOWLEDGEMENTS

I thank Dr. Matthew Wright for being a very motivating mentor, I've learnt a lot from working with him. Most importantly, I've learnt to not get daunted or discouraged by hurdles but overcome them through optimism, persistence, and intelligence.

I thank Dr. Donggang Liu and Mr. Mike O'Dell for serving on my thesis committee. I really appreciate their feedback and evaluation of my work, their interest was indeed very encouraging. I thank Dr. Zhengyi Le for her help and guidance. I thank Dr. Khalili for his valuable advices.

I thank the Graduate School at UTA for their resources as well as the thesis and dissertation seminars conducted by them. I thank Ginger Dickens for her patience in answering all my queries.

I must acknowledge that DTNSim2 and the DieselNet traces were most important for my simulation experiments.

I thank faculty, and staff of Computer Science Department at UTA for their help.

I thank all the members of iSEC lab at UTA for their support.

I thank my family and friends for their support and encouragement.

Above all, I would like to thank my parents, their love has been my biggest strength.

December 7, 2009

ABSTRACT

PEON: PRIVACY-ENHANCED OPPORTUNISTIC NETWORKS WITH APPLICATIONS IN ASSISTIVE ENVIRONMENTS

GAURI VAKDE, M.S.

The University of Texas at Arlington, 2010

Supervising Professor: Matthew Wright

Opportunistic Networking holds a great deal of potential for making communications easier and more flexible in pervasive assistive environments. However, security and privacy must be addressed to make these communications acceptable with respect to protecting patient privacy. We propose Privacy-Enhanced Opportunistic Networking (PEON), a system for using opportunistic networking in privacy-preserving way. PEON uses concepts from anonymous communications, re-routing messages through groups of peer nodes to hide the relation between the sources and destinations. We describe a set of protocols that explore a practical range of trade-offs between privacy and communication costs by modifying how closely the protocol adheres to the optimal predicted path. We also present the results of extensive trace-based simulation experiments that allow us to both compare between our proposed protocols and observe the costs of increasing the number of groups and intermediate nodes in a path.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Assistive environments using sensors, handheld devices, and other pervasive technology require communications between devices that is flexible and robust. The primary model for this kind of communications is *opportunistic networking (ON)*, a framework in which messages are passed between devices whenever the opportunity arises [1, 2]. For example, suppose that a nurse carries a PDA equipped with an 802.11 wireless receiver. The nurse passes by a set of sensors, equipped with 802.11 wireless transmitters that send their sensor readings to the PDA. The PDA then comes within the vicinity of a laptop, and both devices are Bluetooth enabled, so the PDA uploads the sensor readings to the laptop. Finally, the laptop is plugged into a wired network and uploads the sensor readings to a server for storage and processing. The communication provided by message-passing in this fashion saves the expense of putting in a complete networking infrastructure. This is particularly important to facilitate creating assitive environments in developing countries, sparsely populated rural areas, and for quick deployment in established facilities.

Assitive environments, however, have critical privacy requirements. In particular, data security and privacy must be maintained to protect patients from having their health information exposed. This makes the use of the kind of message passing used in ON an unacceptable privacy risk – most ON systems do not limit messages from being sent to only known and trusted parties [3]. In fact, if untrusted parties are never used to forward messages, messages may wait for long periods without being sent and may fail to be delivered. Encryption should be used to hide the contents of

messages. However, the fact that a source is contacting a particular destination may itself be sensitive. For example, if a patient sends a message to a specific doctor, then the doctor's speciality (e.g. cancer or heart disease) can reveal a likely set of diseases that the patient may have.

1.1 Contribution

In this thesis, we explore a method to hide the relationships between the source and destination in opportunistic networks for assitive environments. In particular, we propose Privacy-Enhanced Opportunistic Networking (PEON), a framework that applies the principles of anonymous communications to the specific challenges of opportunistic networking. The basic idea of PEON is to identify a series of intermediate receivers, called *pawns*, and send the message through this series of pawns towards the destination. The message will be encrypted in layers, following the basic idea of Chaum's mixes [4] and used in the popular Tor system [5], such that each pawn can see only the next pawn in the series. The series of pawns is effectively an overlay route and the system can use a variety of routing techniques to get the message from pawn to pawn, including those of [1, 2].

In ONs, however, communication is already uncertain and slow. If we were to route messages through a series of pre-specified nodes, too many messages may never be delivered or take unacceptably long to arrive. To provide a better tradeoff between providing privacy and keeping both failed deliveries and networking overhead low, PEON puts nodes into groups, with group members sharing public key pairs. This allows the source to select a series of intermediate *pawn groups* to forward the message. When a message is sent to a particular pawn group, it can be passed through the system until it reaches any member of the group. This reduces overhead by allowing the message to be routed to the nearest member of the pawn group, rather than to a

specific node. In this thesis, we shall discuss these tradeoffs and enhancements. We also present the results of extensive trace-based simulations which we have conducted to compare the performance of our protocols. We use the results to study the effect of increasing total number of groups in the system as well as increasing the number of intermediate pawn groups in the path.

1.2 Thesis Organisation

In the next chapter, we will provide some background on opportunistic networks with a focus on routing in delay tolerant netwroks and discuss related work. In Chapter 3, we describe a simple network model for ON. We also describe describe possible attacker types and the extent to which we seek to defend against them. These basic models guide the design of PEON.

In Chapter 4, we will describe the PEON system and ways to integrate the PEON groups with established ways to perform routing in opportunistic networks. We also discuss trade-off between privacy and performance with respect to average group size and the number of intermediate pawn groups in a path.

In Chapter 5 we describe our communication scenario and the overall simulation experiment. We present the results of our simulations in Chapter 6. Finally, our conclusions and future work are described in Chapter 7.

CHAPTER 2

BACKGROUND

Opportunistic networks (ONs or OppNets) are an evolution from the models of mobile ad-hoc networks (MANETs) and delay-tolerant (or disruption-tolerant) networks (DTNs). All three of these models share the idea that users, enabled with wireless devices, can communicate by using intermediate nodes — i.e. other users' devices — as routers. This allows the users to communicate in areas with little or no infrastructure. MANETs and their counterpart, vehicular ad-hoc networks (VANETs), are best suited to applications such as military operations and busy highways, in which users are densely packed and the network remains mostly well-connected.

DTNs, however, allow for the possibility that the users may move out of signal range for some time, leading to partitions of the network. By making robust data forwarding and queueing algorithms, DTNs can deliver messages even when users are spread apart and have only intermittent contact. ONs extend this model further. In ONs, connectivity is established and message delivery occurs at each contact between wireless devices. According to Pelusi et al., "Any possible node can opportunistically be used as next hop, provided it is likely to bring the message closer to the final destination" [6]. One application for ONs is *pocket-switched networks*, studied extensively in the HAGGLE project [7], in which users carry devices on their person and the devices exchange messages with Bluetooth [8]. The DakNet project [9] showed how rural villages in India could be connected using shared kiosks and mobile access points (MAPs) attached to busses, motercycles, and bicycles that go between the village and Internet-connected towns.

The literature, unfortunately, is unclear about the relationship between ONs and DTNs. Although we envision PEON as being useful for both ONs and DTNs, much of the related work from both routing and privacy is in DTNs. Thus, we focus in this section first on routing in DTNs and then on ways of providing privacy in other systems.

2.1 Routing in Delay Tolerant Networks

To get a broader and more general perspective of routing in ONs we shall now discuss routing in Delay Tolerant Networks (DTN). DTNs are a class of networks which are characterized to be tolerant of high latency and intermittent connectivity. DTNs are composed of nodes connected by links. The availability and capacity of links are time-varying. There can be multiple links between a pair of nodes. A contact in the context of DTNs is an opportunity to send data over a link. DTNs may be constrained by limited resources like contact opportunities, contact duration, data carrying capacities of contacts, storage at nodes, processing abilities of nodes, and the energy capacity of each node.

The main goal of DTN routing, unlike fully connected data networks, is to maximize the chance of delivery. However, there are not intuitive metrics that we can use directly for building routes with this goal in mind. However, minimizing delivery latency can reduce the chance that a message gets dropped. This approach was validated in simulation by Jain et al [1].

Routing algorithms in DTNs can be classified as either flooding-based or forwarding-based. Flooding-based algorithms replicate each message and send it to many nodes. The redundancy implies reliability in terms of eventual delivery of the message as it can be reasoned that at least one copy will reach the destination. These approaches also seem to reduce latency. However, these algorithms can be very expensive in

terms of consumption of resources. They are also not scalable. Epidemic routing [10] is a flooding-based routing approach. In this approach, each node stores its messages in its buffer and when the two nodes come in contact with each other they exchange summary vectors that list the IDs of the messages in their buffer. Then the nodes exchange messages to synchronize their buffers. Several variants have been suggested to make epidemic routing less expensive.

On the other hand, forwarding-based algorithms use knowledge of the network topology to determine the optimal path along which the message must be forwarded. Jain et al. describe algorithms that use knowledge oracles, abstract entities representing specific knowledge about the network [1]. Four different oracles are defined — the contact summary oracle, the contacts oracle, the queuing oracle, and the traffic demand oracle. The contact summary oracle gives time-invariant, aggregate statistics about the contact schedule. For example, it can provide the average waiting time until the next contact of an edge. The contacts oracle can give any information regarding contacts between two nodes at any point in time. The queuing oracle gives instantaneous buffer occupancies at any node at any time; it is most difficult oracle to realize. The traffic demand oracle gives the present or future traffic demand.

As the latency depends on the time a message arrives at a node, this time is taken into account to compute the cost of links emanating from that node. The paper presents four algorithms based on a time-varying version of Dijkstra's algorithms, these are: Minimum Expected Delay (MED), Earliest Delivery (ED), Earliest Delivery with Local Queues (EDLQ), and Earliest Delivery with All Queues (EDAQ). These algorithms use progressively more information from the oracles.

MED consults the contact summary oracle to get the average waiting time until the next contact is available. It therefore takes the edge cost to be sum of average waiting time, propagation delay, and transmission delay. Thus the edge-cost becomes

time-invariant which implies that same route will be obtained for messages with same source-destination pair and after the route has been computed, it will not re-compute the route to use a better contact. ED consults the contact oracle to determine the availability of contacts such that the computed route will get the message to the destination at the earliest time. Thus instead of taking an average waiting time it takes into account the minimum waiting time which is time-varying. However, it does not incorporate any queuing information.

EDLQ estimates the latency by taking into account the queuing at all edges out going from the current node. Therefore, route is recomputed at every hop (per-hop routing) in contrast to MED, ED and EDAQ which are all source-routing algorithms. But computing the route at every hop may lead to formation of loop. EDAQ uses the queuing oracle to determine the instantaneous queue sizes across the entire topology at any point in time. Once the route is computed at the source, link capacity must be reserved for that message for all links in the route. This is required to ensure that messages avoid missing contacts as well as to enable the queuing oracle to make accurate predictions. However, realizing link capacity reservation in DTNs is quite challenging.

Apart from Dijkstra-based algorithms the paper also describes First Contact (FC) and Linear Program (LP). FC assigns a message to the first available contact or randomly to one of the currently available contacts and therefore has a poor performance. On the other hand, the LP formulation of the DTN routing problem uses all four oracles to obtain the complete knowledge of the network to determine the optimal route. Jain et al. conclude that algorithms using more knowledge (EDLQ, EDAQ, LP) provide a significant benefit only when resources are limited which is less common. Therefore it may not be worth implementing the queuing oracle which is challenging to realize.

These approaches require that each node must have access to accurate schedule data for which the routing must be manually configured with the contact schedules. To make routing more practical, one can make a routing algorithm self-configuring and more suitable for imprecise or unpredictable schedules [2]. Jones, et al. propose the Minimum Estimated Expected Delay (MEED) protocol where the link-cost is based on the observed contact history over a sliding history window. Once the costs have been computed at each individual node, MEED assumes that each node has the complete network topology. A link-state routing algorithm was chosen for this because it is similar to epidemic routing in DTNs and is suitable for propagating updates in a single contact, resulting in an Epidemic Link State Protocol.

Simulation results indicate that performance of MEED approaches that of protocols having complete knowledge of network topology. Also MEED achieves 96% of epidemic routing's delivery ratio using only a single message, instead of a copy for every node.

In this paper, we have suggested using information from such routing protocols to improve the performance of the PEON System. Specifically, our scheme requires that the routing protocols can be queried by a node to find its distance to other nodes. The distance could be based on any relevant metric like bandwidth or latency. Our scheme relies on the information provided by these protocols and thus works only as well as the information they use. However, our scheme can leverage new advances in such protocols and thus become more efficient with improvements in them.

2.2 Related Work

The first anonymous communication solution for DTN proposed in [3] uses identity-based cryptography (IBC). It suggests making identities of the users more specific by combining them with geographical identifiers. Further, it recommends

hierarchical identity-based cryptography (HIBC) for greater scalability. The paper attempts to protect user identities from DTN routers (including kiosks) but it is assumed that DTN gateways are trusted and are aware of user identities. Protocols designed therefore use pseudonyms instead of user identities while routing messages in DTNs. For example, to send a message a user must compute a pseudonym through which a router can ensure that the user is valid but can't obtain the associated identity. Commonly in DTNs one-way authentication is required for which the authors have suggested a quick, non-interactive scheme. Additionally, an anonymous mutual authentication scheme which takes three flows is also given.

The advantage of this scheme is that it incurs no additional overhead for routing. However, the scheme is very tightly tied to the DakNet model, and it assumes a strongly trusted central authority. A more general approach is required for systems in which a trusted central authority cannot be assumed. This is especially true in assitive environments, where having a trusted authority presents a substantial privacy risk for patients.

CHAPTER 3

NETWORK AND ATTACKER MODELS

Before describing our proposals for protecting privacy, including personal medical information, while using opportunistic networks, we first outline the basic models we are using to guide our design. We describe a simple network model for opportunistic networks that is detailed enough to compare different algorithms. We also describe possible attacker types and the extent to which we seek to defend against them.

3.1 A Model for Opportunistic Networks

As described in Section 2, there are a variety of applications for opportunistic networking, from remote village Internet connectivity to pervasive computing environments. One example that has been used in a variety of experiments is a city bus system, with both stationary access points and moving busses with occasionally crossing routes [1]. Since there are network traces available for this model, we will also use it as the basis for our work. We expect that our results will apply well in general, but recognize that some applications may feature unique networking characteristics that will require more specific models.

In our model, we have a large number of nodes, of which some are stationary and others physically move in the local area. When two nodes are in proximity, within the range of their respective wireless transmitters and receivers, the nodes may exchange a number of messages depending on their transmission rates and the amount of time they are in proximity. While the exact transmission rate may depend on the nodes'

distance from each other and may vary from connection to connection, we model the transmission rate as a reasonable fixed value across the system and across time. In initial experiments, we may model the system even more simply by allowing each sender to send all of the messages it chooses based on other considerations like queue size and desired route. This will cause us to underestimate latency and message overhead, but will still allow us to make reasonable comparisons between algorithms. Because we are only concerned with whether two nodes are in contact, we extract connectivity information from traces and ignore the actual movement patterns of the nodes. To extend our experiments, we may generate additional connectivity information based on distributions extracted from the traces.

As we describe in Section 4, we put nodes together in pawn groups. We select nodes for each group at random from all nodes in the system. We group the nodes unequally by placing each node into a group (the total number of groups is pre-chosen) chosen at random (with replacement), leading to varying group populations. This is more realistic than having fixed group size, as groups are best selected as nodes that trust each other, at least in a limited way. Note that group members will not be assumed to trust each other to protect each other's privacy or otherwise provide any specific help to group members. However, group members will share keys and it is easier to establish shared keys among trusted group members. For example, with sensors or busses administered by the same administrator, the administrator can be in charge of generating the group key. Among people with strong trust relationships, a single trusted friend could be the key generation authority, or the authority could rotate in round-robin fashion through the group. If a semi-trusted PEON authority assigns nodes to groups, then the group should either elect a key-generation authority or rotate the authority among the nodes in a round-robin fashion.

3.1.1 Measuring Privacy

There have been a variety of ways to measure the privacy of anonymous communications system. We propose to use information theoretic metrics, as proposed by [11, 12]. Generally speaking, this will be the number of bits of entropy from the perspective of the attacker. Low entropy means that little information is hidden from the attacker, while high entropy means that the attacker can discern very little about the network.

3.2 Attacker Model

While the privacy of users is important, we cannot protect it absolutely against attackers of unlimited capability, at least not with reasonable cost. Thus, we must carefully consider the capabilities of the attacker we seek to protect against. We will now describe several such attacker models that we believe to be reasonable.

3.2.1 Local Eavesdropper

One of the more likely attacker types is an attacker in the vicinity of the user with the ability to eavesdrop on the user's wireless signals. This attacker could, for example, simply be a curious neighbor. Such an attacker requires little in the way of equipment — a simple wireless receiver is sufficient to observe messages in the system. Fortunately, the attacker is easily defeated. Simply encrypting messages and the destination with a key that the attacker doesn't have will prevent most leaks.

3.2.2 Curious Responders

Another likely attacker is the responder who receives the messages from the initiator. This may seem counter-intuitive, as the responder and initiator exchange messages. However, the responder knows the content of the messages and may be

more curious than any other party about who sent them. This attacker, like the local eavesdropper, is easily defeated. As long as the messages that reach the responder contain no information about the initiator, little can be revealed.

It is possible that, with detailed knowledge of the network layout and expected message delivery latencies, the responder could learn something from the delay between messages. For example, if a response message is expected immediately after receiving a message from the responder, the round-trip latency can be measured. Similar attacks have been explored for the Tor network [13]. However, without a realistic deployment, it is hard to perceive exactly what kind of information could be leaked. Further, the attack depends greatly on the application being used and does not work against messages whose sending start times are not highly predictable. Thus, we leave exploration of this line of attack to future work.

3.2.3 A Set of Corrupt Peers

Another attacker type that should be anticipated is a set of corrupt peers that work together to break the privacy of users. At the very least, a single corrupt peer should be defended against, representing an otherwise honest but curious member of the network. However, a more aggressive attacker could insert a set of corrupt peers into the network. For example, the administrator of a set of sensors or busses could use these nodes to monitor some of the activity in the network. Again, it is best if the whole set of nodes belongs to just one pawn group, and we will examine this as one possible attacker model. However, we cannot rely on this assumption for all scenarios, especially when group membership is established in another manner than by administrative control, such as randomly. Thus we will also consider an attacker with a set of peers spread randomly throughout the pawn groups.

An attacker with a set of peers can perform a variety of attacks. First, with a presence in multiple groups, he will have access to multiple private keys, enabling decryption of more messages than any single node. Second, the corrupt peers can record observations of which users send messages, the times that they are sent, and the recipients of messages during that time. If a user sends many messages to the same destination or set of destinations over the time, its statistical patterns will stand out and the attacker will be able to link sources and their destinations. Similar attacks have been described in anonymous communications for the Internet [14, 15, 16, 17]. We intend to apply similar analytical and simulation methodologies to evaluate our proposed approaches.

3.2.4 Partial Eavesdropper

Another important attack model is the *global eavesdropper*, an attacker with the ability to see, for all messages in the network, the sender, receiver, and sending time. We argue that a fully global eavesdropper is not appropriate for our study. Opportunistic networks are designed for environments in which regular connectivity is not available. Other wireless networks, such as sensor networks and connected ad-hoc networks, may be subject to global eavesdroppers due to the nodes being densely populated in a more compact area [18, 19]. In opportunistic networks, however, nodes are expected to be without connectivity due to large distances relative to their wireless transmission range. In such a network, the eavesdropper would need a very large ratio of wireless receivers to nodes in the network to cover the entire network's operations.

Nevertheless, a *partial eavesdropper* would be able to place wireless receivers in a few *hotspots* in the network where many nodes meet and exchange messages. From observing these hotspots, the attacker could learn much about what is happening

in the network. Similar attacks against Tor were explored in [20], which explores the threat of a compromised Internet exchange (IX) that can monitor a substantial fraction of the network's traffic. In such networks, the IX will observe many of the same communicating pairs over time. In opportunistic networks, however, the eavesdropper may see different communications and have different capabilities for attack. We plan to explore the power of these attacks to deanonymize users.

3.2.5 Combined Attacks

Finally, we recognize that an attacker need not be limited to a single vector of attack. For example, a local eavesdropper and a curious responder would be able to combine their collected information to better expose the initiator. Such attacker types must also be considered in the system design.

CHAPTER 4

SYSTEM DESIGN

The PEON system applies the principles of anonymous communications to opportunistic networking, with the goal of enhancing privacy for applications like patient care. In place of mixes or onion routers, we propose pawn groups, a group of nodes of which any could serve as an intermediate pawn in the overlay path. We now describe the elements of this system: the basic cryptographic framework and the integration of routing with pawn group and pawn selection.

4.1 PEON Onions

We call the layered cryptography used to provide anonymity *PEON onions*. Consider an initiator I sending a message M_{IR} to a responder R, using pawn groups A and B. Suppose that for a given pawn group X, the pawn group shares a public key KU_X and a private key KR_X. Any member of the group can decrypt a message M encrypted with the public key as $E_{KU_X}[M]$. Let us assume that the public keys of all pawn groups are known to all nodes in the system, including I. Also, note that nodes are members of only one pawn group each.

To send its message, I will construct a PEON onion by progressively encrypting the message with the public keys of each of the pawn groups it wants on its path. Within each layer, it will put the ID of next pawn group to which the message should be sent. We now briefly show the sequence of messages.

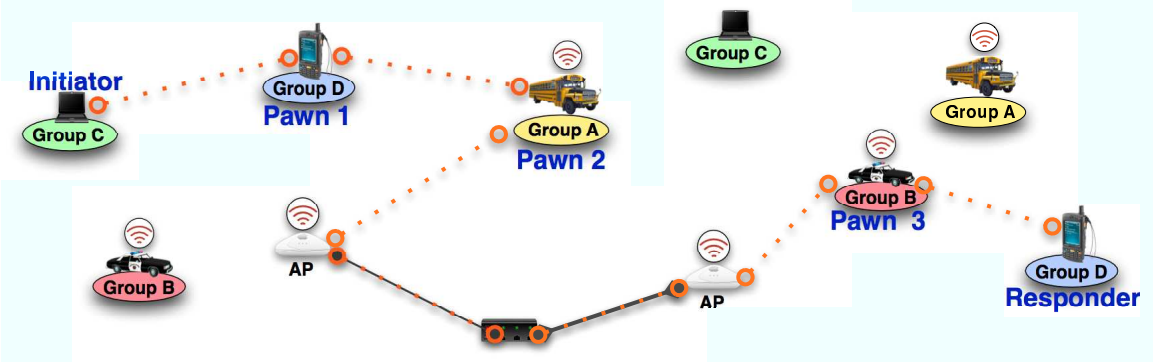

Figure 4.1. PEON Architecture. The dotted line shows the path for a message from the initiator to the responder.

First I sends M_A to pawn group A. Let us use the notation $X \sim\!\!\rightarrow Y$ to indicate that node X sends a message to a member of group Y. The selection of that member and the process of routing the message are described in Section 4.2:

$$I \sim\!\!\rightarrow A : M_A = E_{KU_A}[M_B, B]$$

Suppose that node A_0, a member of A, gets M_A and becomes the first pawn on the path. A_0 decrypts M_A to get M_B and the ID of group B. It then sends M_B to a member of group B.

$$A_0 \sim\!\!\rightarrow B : M_B = E_{KU_B}[M_{IR}, R]$$

Then suppose that B_0, member of B, gets M_B and becomes the second pawn on the path. B_0 similarly decrypts M_B to get M_{IR} and the ID of responder R. It then sends M_{IR} to R. From these messages, we see that M_A can be written as:

$$M_A = E_{KU_A}[E_{KU_B}[M_{IR}, R], B]$$

Through this process, the identity of the sender, I, has been hidden from any intermediaries except the first node to receive the encrypted message from I. Any

node with the ability to see M_{IR} (which may or may not be encrypted depending on the message or application) or the identity of the responder R will not know the identity of I.

4.2 Routing and Node Selection

To take advantage of the fact that PEON uses pawn groups instead of selecting individual nodes, we need to devise a scheme to route messages to a group of nodes. To clarify, this is different from multicasting, in which the message is sent to all nodes in the group. Rather, the message should go to any member of the group. In this section, we will describe a series of possible ways to route the message through the path of pawn groups and discuss the expected impacts of each choice on the performance and privacy of the system.

4.2.1 Random Selection

The simplest way to build the PEON system would be to not consider the way in which the underlying routing algorithm works, and to use it as a black box without modification. This provides a substantial advantage in keeping the system modular, allowing for different underlying routing algorithms to be used without any changes to PEON itself. To do this, PEON initiators would have to be agnostic to performance considerations when selecting pawns. Because of this choice, we would be assured that no privacy would leak to do biased selection of pawns due to performance considerations. Other research has examined ways in which privacy can be lost when performance is used to select onion routers in the Tor network [21, 13, 22]. However, we can also expect that ignoring the underlying routing algorithm will provide the worst performance, although significantly better than directly selecting individual pawns.

We now outline a simple algorithm that would operate without knowledge of the underlying routing algorithm. In this *Random Selection* scheme, the initiator first chooses each pawn group uniformly at random. Then it chooses the initial pawn uniformly at random from the first pawn group. As the message is forwarded to a member of each group, that member chooses the next pawn uniformly at random from the next pawn group. For example, if the initiator is using a path of two pawns, it will select the first pawn group G^1 and then select a second pawn group G^2 *without replacement*. In other words, we are careful not to select $G^2 = G^1$ so to ensure that the same key could not be used to open both layers of encryption and see the message. Having selected the groups, the initiator chooses a pawn, say G_i^1, from among the nodes in G^1 and sends the message to it using the standard routing algorithms. When G_i^1 gets the message, it decrypts it and sees that the next pawn group is G^2. It then selects the next pawn, say G_j^2, at random from G^2 and sends it the message. G_j^2 decrypts the message and sends it directly to the responder R.

Note that we allow each pawn to choose the next pawn in the path from among the next pawn group. This may seem undesirable if, for example, the first pawn is corrupt and has a corrupt partner in the second pawn group that it can choose. However, if there are such corrupt partners, they can also share their groups' private keys with each other, making it unnecessary for the first pawn to select the second pawn to see the responder's identity. The security of the system is thus based more on the selection of groups than individual nodes. Note that the pawns have no impact on the selection or order of pawn groups, as the initiator encrypts the message with the public keys of the pawn groups that it chooses.

4.2.2 Closest Pawn Routing

Although Random Selection has the benefits of being modular and not leaking any privacy, it also does not help to limit overheads. Since opportunistic networks can be quite slow, we would seek to provide better performance as long as not too much privacy is leaked. We now describe *Closest Pawn Routing (CPR)*, a scheme that uses information directly from the routing protocol to improve performance. For this scheme, we assume two things about the underlying routing algorithm: (1) that the routing algorithm maintains some notion of *distance* between nodes based on metrics such as latency or bandwidth; and (2) that we can query the routing algorithm to find out the distance between the query node and all other nodes in the system. Several existing routing algorithms for opportunistic networks meet these requirements, including ED and MED [1].

Given these two requirements, we can select pawns from each group more intelligently. We first assume that the initiator selects the pawn groups randomly, as with Random Selection. Then, each sender selects as the pawn the closest member of the next pawn group in the path. By doing this, we are making a greedy choice that ignores future pawn groups, but we expect to improve both message overhead and average path latency over Random Selection (which also ignores future pawn groups during pawn selection). In particular, we expect that if the distance from the current sender to the closest member of the next pawn group is half that to the average pawn group member, then the overall cost of sending the message will be cut in half. Note that the cost reflects the routing algorithm's cost metric and may not correspond exactly with the cost metric we seek to minimize. However, the cost metrics for both normal routing and PEON should be roughly the same for most systems.

The privacy lost due to Closest Pawn Routing should not be large. The main attack is to take advantage of the locality the choice of pawns provides. While random

selection should not have any locality, CPR will allow an attacker to observe that the user's messages are closer to the user than to other nodes in the system. For example, let us assume that the last pawn is always physically located within radius r of the initiator (even accounting for node movement). If the user's messages can all be identified as coming from the same initiator, then the attacker could take the intersection of all nodes with radius r of the last pawn for each observed message. Eventually, such intersection attacks can substantially limit the set of possible initiators [23].

This attack is overly simplistic, but it suggests the direction that an attacker could take to gain an advantage in CPR that random selection would not allow. One defense against this would be to use random selection for the final pawn; the locality-based attack would fail unless the attacker controlled the final two pawns for multiple messages. Another approach is simply to ensure that the ratio of groups to nodes is high enough so that most selected pawns will not be close to the initiator.

4.2.3 Directed Routing

In CPR, we seek to improve performance by having each sender intelligently select the closest member of the next group as the next pawn. The improvement in performance is achieved by lowering the overall cost of sending the message. However, no effort is made to direct the message towards the responder which may result in selection of a path that eventually doesn't lead to the responder or that has pawns which are closer to the sender but farther away from the responder.

Therefore, to further improve routing performance, we describe Directed Routing (DR), a scheme that directs the PEON routing along the optimum path obtained from the underlying routing algorithm for routing a message from an initiator to a responder. We expect that directing the PEON path along the optimum path will not only decrease the overall cost of sending a message but it will also result in more

messages being actually delivered to their final destination. To implement this scheme we require the underlying routing algorithm to support CPR and capable of being queried, at the source node, for the optimum path for routing a message. Again ED and MED [1], meet these requirements.

In DR, the initiator first queries the routing algorithm to obtain the optimum path for routing a message to the responder. Note that different routing algorithms have different ways of determining the optimum paths and thus the paths may differ for any two algorithms. From the path, the initiator gets the intermediate nodes which eventually lead to the responder. The initiator prepares the list of intermediate pawn groups for PEON from the groups to which these intermediate groups belong to. The group of the first intermediate node becomes the first intermediate pawn group for PEON, the group of the second node becomes the second group for PEON and so on and so forth till the initiator gets the required number of intermediate pawn groups.

Note that a group will appear only once in the list of intermediate pawn groups. Also, the order of appearance of groups in the list will be in accordance with the order of the corresponding nodes in the optimum path. If the number of nodes in the path is less and the initiator can't find the required number of intermediate pawn groups through these nodes then the extra groups are chosen randomly as with Random Selection and they are inserted in the list after the groups which are chosen through the optimum path. Again, if the routing algorithm does not find a path between the initiator and the responder then also we choose all the intermediate pawn groups randomly. In DR as with CPR, each sender selects as the pawn the closest member of the next pawn group in the path.

The improved performance for this scheme also means compromise with privacy. Clearly, it will aid in the locality-based attack that is applicable to CPR. Moreover, it will aid a curious responder in finding the initiator as anonymous communication

with PEON will up to an extend resemble the normal communication. In [24] we discuss some variations to this approach so that the PEON path is partially directed by the routing algorithm and there can be some improvement with respect to privacy.

4.3 Design Considerations

We shall now discuss some of the design considerations for PEON. PEON creates two ways to trade off between privacy and performance; we shall briefly discuss these tradeoffs. We shall also discuss the benefits and costs of enhancing PEON through the use of cover traffic.

The first way to trade off between privacy and performance is the size of the group. Large groups make it easy to find a member nearby, increasing the chance of quickly passing the message to the destination. However, with few groups, there are fewer possibilities for diverse paths needed for anonymity over the lifetime of the connection. Also, there is the possibility of a single node stealing private keys from a few other nodes and thus being able to open all the layers of communication for a set of messages. With more groups, the danger of this attack is reduced.

The second way to trade off between privacy and performance is with the number of pawn groups a sender selects for its path. Clearly, shorter the path, lower the overhead and delay. Very short path lengths, however, lead to lower privacy for the sender. For example, with a path length of only one pawn group, a pawn in that group will be in position to see the intended recipient as well as, possibly, the source node. With enough messages, the source will eventually be linked with the messages. Further, the messages will all be received by pawn nodes close to the position of the source node, helping to reveal the sender's location. If the source selects two pawn groups in its path, this will improve privacy by enforcing greater distances from the

source and clearly separating the message from the source node. In general, the more intermediate pawns on the path, the harder it is to link the source and recipient.

To enhance PEON further, cover traffic can be introduced to hinder a powerful attacker like the partial eavesdropper. When a node comes in contact with another node, it should send all available real messages first to ensure high delivery rates. If the contact is still available, it could then send dummy messages. This would require the overhead of the sending node generating and transmitting dummy messages, and the receiving node receiving and filter dummy messages. When communications costs are a prime consideration, these overheads may be too high. In some cases, however, they may be acceptable. This simple scheme provides variable costs and benefits, depending on connection times and traffic rates, among other parameters. Thus, the tradeoffs require further analysis.

CHAPTER 5

SIMULATION DESIGN

To evaluate the performance of PEON protocols we used DTNSim2 [25] [26], a discrete-event simulator for DTN that uses FIFO, reliable links with fixed bandwidth and delay. It is based on the DTN simulator developed for the DTN routing paper by Jain et al [1].

Our aim is to implement PEON protocol over forwarding-based routing algorithms for DTN to evaluate their performance regarding the delivery ratio, average delivery delay, and overhead in terms of actual bytes transmitted per each byte of the data. Moreover, we want to study the manner in which performance is influenced by the average size of the groups in the system, and the number of pawn groups an initiator selects for its path. For our purpose, it is sufficient to implement the protocols over ED and MED which are both forwarding-based DTN routing algorithm that require some form of schedule information as described in Section 2.

We shall now describe our communication scenario and the overall simulation.

5.1 Scenario: City Buses

Scenarios based on a network of buses equipped with wireless devices providing DTN have been previously discussed by Jain et al. [1] and Jones et al. [26]. In [26], Jones et al. note that the trends for all aspects that they had studied were similar for the city bus scenario and the wireless LAN scenario. However, for performance under ideal conditions it was noted that though the trends were same, the city bus scenario had a better performance.

We therefore use the DieselNet traces [27] that were compiled for buses running routes serviced by UMass Transit. UMass Transit had a total of 40 buses with DieselNet equipment. Out of these, some had to be operated daily. The buses that would operate on a day depended on which ones were available as opposed to those that had to be rested due to failure or service requirements. As a result, different buses operated on different routes on different days and hence the schedules for any two days are not identical. Every bus connected with other buses or APs when it would come in contact with them. The traces thus contain connection events between buses as well as between a bus and an AP. Connection events for a day are stored in a file matching that date. To make the simulations more tractable we eliminate all but ten most visible APs. Thus, we create a scenario with less infrastructure than in [1] and [26].

5.2 Traffic

In [26] it is mentioned that for a bus scenario, the schedules of five weekdays had been combined because these schedules were identical and statisctics for messages generated during the second day were recorded. Since the daily schedules obtained from DieselNet trace are not identical, we can not apply this scheme to our scenario. We instead concentrate on obtaining statistics for traffic sent over a single day that has high probability of being delivered on that same day. We do this for five consecutive days and calculate the final statistics as a suitable average.

For generating traffic for a day, we listed all ordered pairs of nodes in that schedule. The first node in the pair is the initiator and the second is the responder. Pairs having both the initiator and responder as APs were discarded as such a communication would happen via the Internet rather than the DTN. For each of the remaining ordered pairs, we select a random start time from the first quarter of the

day. At this time, the initiator will send a message to the responder for the first time. That initiator will continue sending a message to the same responder at an interval of 1 hour during the first half of that day. The size of each message is 10,000 bytes. This pattern ensures that each node initiates at least 6 messages to every other node which can be a valid responder in DTN. And since this traffic is generated in the first half of the day, it has high probability of being delivered on the same day.

5.3 Metrics

We evaluate our protocols in terms of delivery ratio, average delivery delay, and the overhead. Delivery ratio is the percentage of total messages delivered to their final destinations by the end of the simulation out of the total messages injected into the system. Delay for a delivered messages is the time between when that message is generated at the initiator and when it is received at it's final destination. By the end of the simulation we get the average delivery delay which is the average of the delay for all the delivered messages. The overhead of the protocol is defined as the ratio of total bytes transmitted during the simulation to the total bytes of actual data that have been delivered to their final destinations by the end of the simulation. Total number of bytes transmitted includes protocol bytes and data bytes and measures the total amount of bandwidth consumed by the protocol. The overhead indicates the cost incurred in terms of bytes transmitted while delivering each byte of the actual data that has been delivered to its final destination.

5.4 Experiment

Currently we only study the performance of the PEON protocols under ideal conditions of operation. This implies that we consider the DTN nodes to have un-

limited buffer capacity for holding in-transit data. Also the links have unlimited bandwidth and negligible latency. The conditions are not practically feasible but we assume their existance as it will enable us to get an unbiased evaluation of our protocols and validate our assumptions about their behavior. In the future, we will study PEON protocols under more realistic conditions.

For every combination of PEON protocol and routing algorithm, we set the context by varying the number of pawns in the path as well as the total number of groups in the system. For each context, we run the simulation a total of five times, once for each of the five consecutive days mentioned in our scenario. We calculate the final values for the delivery ratio, average delivery delay and the protocol overhead as an appropriate average of these five sets of results. The delivery ratio is calculated as the percentage of total messages delivered during the five days out of the total messages injected into the system in that time. Average delivery delay is calculated as the weighted mean of average delays for the five days, weighing them by the number of messages delivered on the corresponding day. The protocol overhead is calculated as the ratio of total bytes transmitted in five days to the total data bytes delivered to their final destinations in that time.

CHAPTER 6

SIMULATION RESULTS

In this section, we present the results of our simulations. All results are for unequal grouping of nodes and they are presented for PEON protocols in combination with different routing algorithms. DR/ED, CPR/ED and Random/ED are DR, CPR and Random Selection used with ED, respectively. Similarly, DR/MED, CPR/MED and Random/MED are DR, CPR and Random Selection used with MED, respectively.

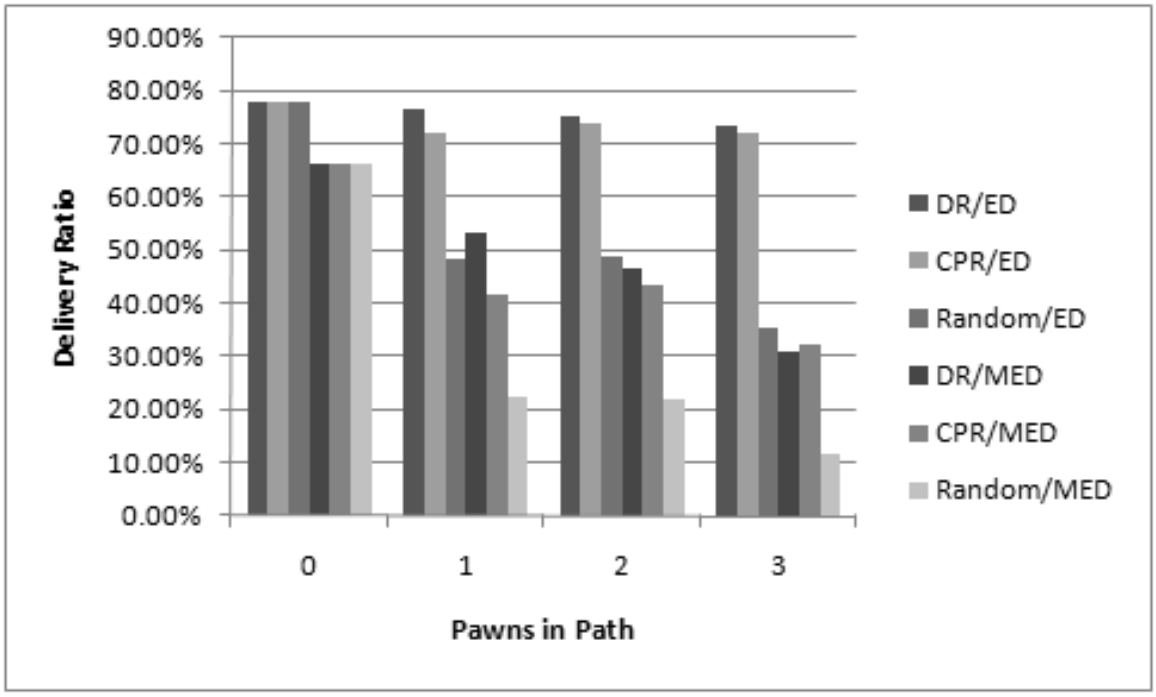

Figure 6.1. Delivery ratio for varying number of pawns in the path when the number of groups in the system is five.

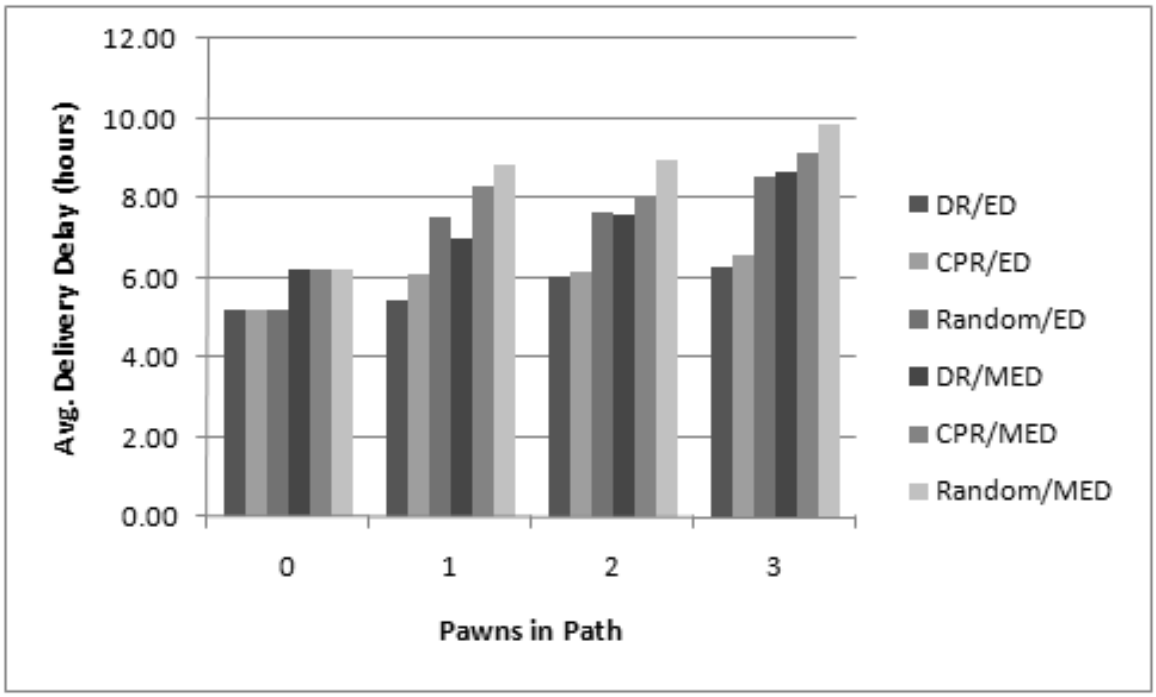

Figure 6.2. Average delivery delay for varying number of pawns in the path when the number of groups in the system is five.

6.1 Performance of PEON Protocols

We first compare the performance of PEON protocols, for presenting examples we consider the case where number of groups in system is five and number of pawns in path is two.

From Figures 6.1 and 6.4, we can see that over the same routing algorithm, the delivery ratio for random selection is much lesser than that of CPR. This is because firstly choosing a random pawn may result in a path which may not lead to the responder or incur very high delay so that the message can't be delivered on the same day. We also observe that the delivery ratio for DR is the highest because in DR the process of determination of the path is directed in such a way so as to most closely resemble the path computed by the underlying routing algorithm. However, the

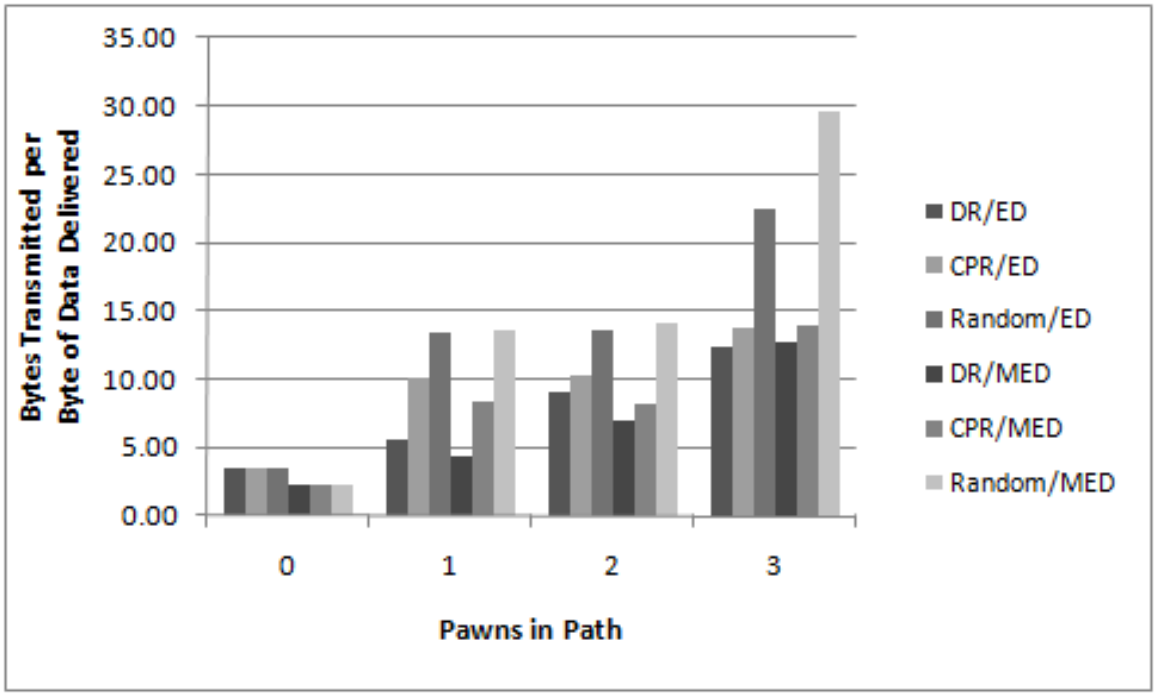

Figure 6.3. Overhead for varying number of pawns in the path when the number of groups in the system is five.

delivery ratio for DR is only slightly higher than that of CPR. The values of delivery ratios observed for Random/ED, CPR/ED, and DR/ED are 48.692%, 73.776%, and 75.148%, respectively. Notice that the delivery ratio for Random Selection much lesser than that of CPR, whereas, the delivery ratio for CPR is only slightly lesser than that of DR.

The results for average delay are shown in Figures 6.2 and 6.5. As expected, for the same routing algorithm, the delay for random selection is the highest. Note, that the expected value of delay in random selection is the population mean of cost, in terms of delay, for the nodes in the next pawn group. CPR has lesser delay than random selection because if the minimum delay from current sender to next pawn is half of the population mean then the overall delay will also be reduced by half.

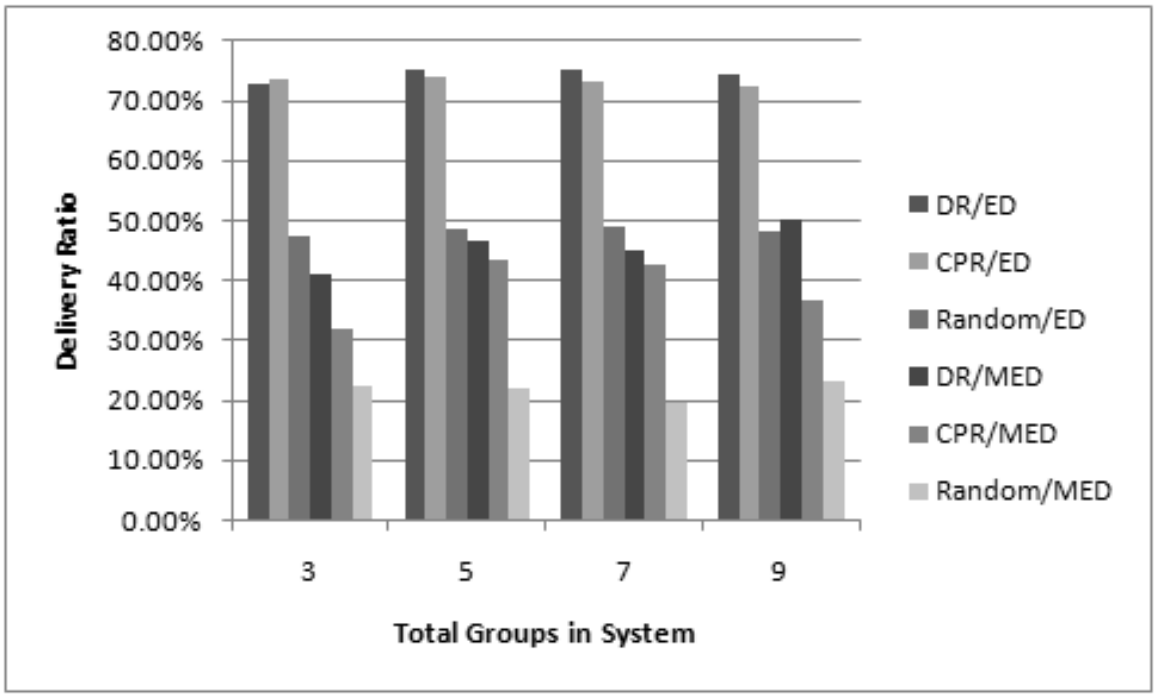

Figure 6.4. Delivery ratio for varying number of groups in the system when the number of pawns in the path is two.

Further, delay for DR is slightly less than that of CPR beacuse we not only try to choose the pawns closer to the sender but we try to find them in the groups which are most likely to have nodes that are also closer to the responder. The values of average delay observed for Random/ED, CPR/ED, and DR/ED are 7.671 hours, 6.137 hours, and 6.048 hours, respectively. Again notice that the average delay for Random Selection is considerably higher than that of CPR, whereas, the average delay for CPR is only slightly higher than that of DR.

Silmilar trend is observed in the case of overhead. Figures 6.3 and 6.6 indicate that bytes transmitted per byte of data delivered is least for DR, slightly higher for CPR, and highest for random selection. The values of overhead for DR/ED, CPR/ED, and Random/ED are 9.136 bytes/byte, 10.349 bytes/byte, and 13.658 bytes/byte,

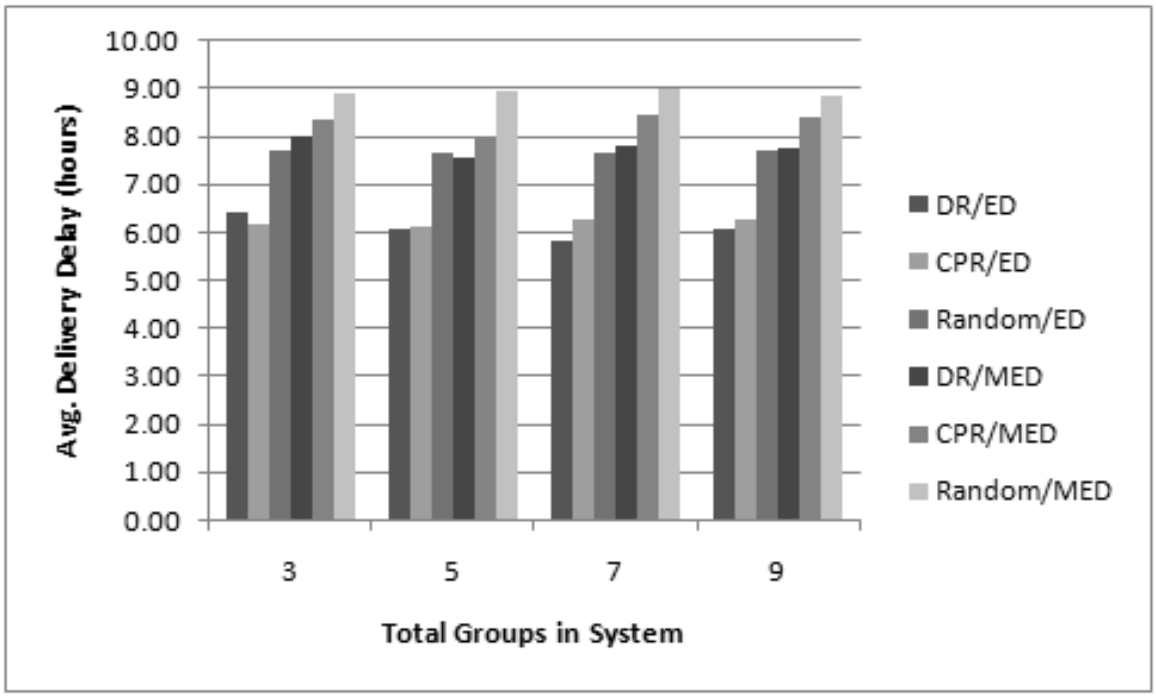

Figure 6.5. Average delivery delay for varying number of groups in the system when the number of pawns in the path is two.

resepectively. In general, from our observations of delivery ratios and average delay we can say that the performance of CPR is much better than that of Random Selection and the performance of DR is slightly better than that of CPR.

6.2 Choice of Routing Algorithm

We now attempt to understand the effect of choice of underlying routing algorithm on the performance of PEON protocols. For our examples we consider the case where number of groups in system is five and number of pawns in path is two.

From Figures 6.1 and 6.4 we can infer that the delivery ratios for any PEON protocol when used with ED is much higher than when used with MED. For example,

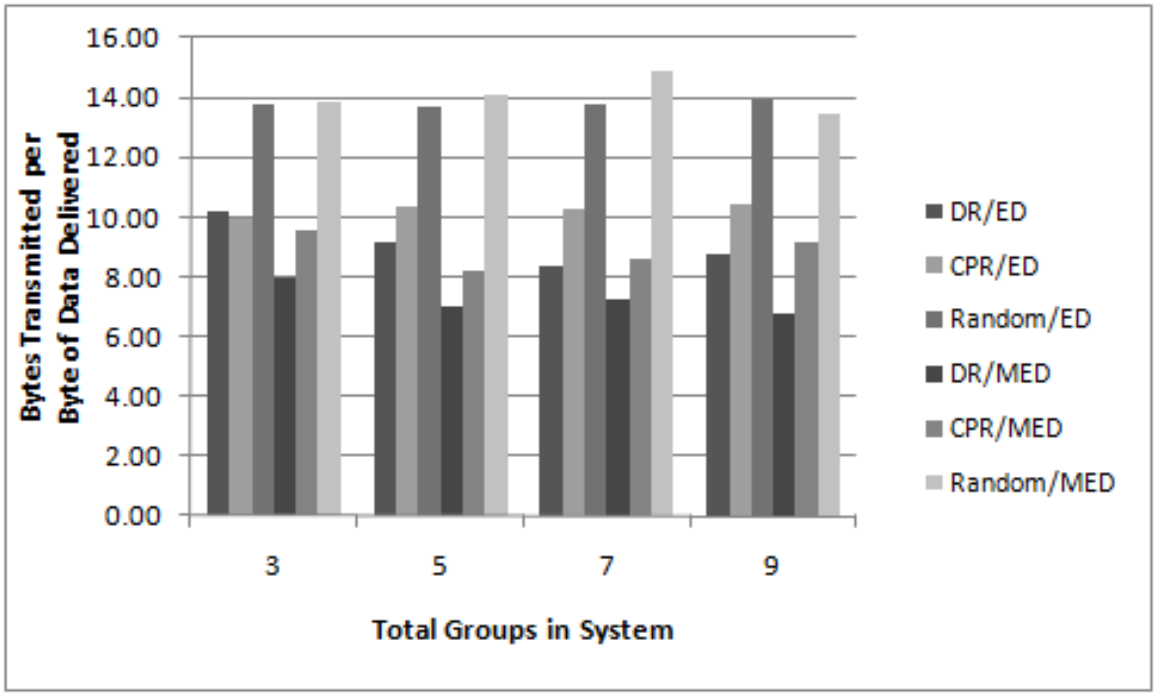

Figure 6.6. Overhead for varying number of groups in the system when the number of pawns in the path is two.

DR/ED has delivery ratio 75.148% which is much higher than 46.544% which is the delivery ratio observed for DR/MED.

From Figures 6.2 and 6.5 we can observe that the average delay for delivered messages is lower for a PEON protocol when the underlying protocol is ED rather than MED. For example, DR/ED has a lesser delay of 6.048 hours compared to the delay for DR/MED which is 7.554 hours.

However, from Figures 6.3 and 6.6 we see that when DR and CPR are used in combination with ED then they have a higher overhead than when they are used with MED. On comparing DR/ED and DR/MED, we see that the former has a higher overhead of 9.136 bytes/byte and the latter has a lower overhead of 6.998 bytes/byte. This trend is not observed in the case of random slection.

In general for any PEON protocol, the performance is better when the underlying routing algorithm has more accurate knowledge of the schedule although a slightly higher overhead may be incurred.

6.3 Effect of Number of Pawns in Path

To study the effect of varying number of pawns in the path we consider the case where total number of groups in the system is five.

Note that when the number of pawns in path is 0, PEON does not come into effect and we instead observe the behaviour of the underlying routing algorithm. We can see this case in Figures 6.1, 6.2, and 6.3 and infer that for ED the value of delivery ratio is 77.944%, delay is 5.195 hours, and overhead is 3.547 bytes/byte. Similarly, we can also infer that for MED the value of delivery ratio is 66.273%, delay is 6.203 hours, and overhead is 2.297 bytes/byte.

Figure 6.1 indicates that in each of the six cases of DR/ED, CPR/ED, Random/ED, DR/MED, CPR/MED, and Random/MED; there is a slight decrease in delivery ratio as the number of pawns in path increases. For example, for DR/ED the delivery ratio gradually decreases from 77.944% to 73.444% as the number of pawns increase from zero to three. Similarly, for DR/MED the delivery ratio steadily decreases from 66.273% to 30.788% as the number of pawns increase from zero to three. As an exception, for CPR/ED, Random/ED, and CPR/MED we see that the delivery ratio is lesser when the number of pawns is two than when it is one. However, the difference between the corresponding delivery ratios in these cases is negligible ($< 1.7\%$) and hence the two values can be considered almost equal. In general, we can say that delivery ratio decreases with an increase in number of pawns.

From Figure 6.2, we can see that there is a gradual increase in average delivery delay with an increase in number of pawns. For example, when the number of pawns

increases from zero to three, the delay increases slowly from 5.195 hours to 6.284 hours for DR/ED and it increases clearly from 6.203 hours to 8.647 hours for DR/MED . Note that for most combinations of PEON protocols and routing algorithms, there is a very small difference in delay values when number of pawns is one and when it is two.

In Figure 6.3, we can see that clearly more bytes will be transmitted per byte of data delivered as the number of pawns included in a path increases. Again if we consider the case of DR/ED and DR/MED, for the former the overhead increases from 3.547 bytes/byte to 12.336 bytes/byte, and for the latter the overhead increases from 2.297 bytes/byte to 12.737 bytes/byte.

Therefore, we infer that the performance will suffer if number of pawns is increased.

6.4 Effect of Group Size

Table 6.1. Average and standard deviation of delivery ratio for varying number of groups in the system when the number of pawns in the path is two

PEON Protocol / Routing Algorithm	Delivery Ratio	
	Average (%)	Std. Dev. (%)
DR/ED	74.796	0.933
CPR/ED	72.193	1.682
Random/ED	48.373	0.742
DR/MED	44.897	3.192
CPR/MED	39.761	4.040
Random/MED	22.221	1.283

For our data set, we use number of groups in the system between three and ten. From Figure 6.4, we can infer that the delivery ratio does not change with varying

Table 6.2. Average and standard deviation of delay for varying number of groups in the system when the number of pawns in the path is two

PEON Protocol / Routing Algorithm	Delay	
	Average (hours)	Std. Dev.(hours)
DR/ED	6.078	0.187
CPR/ED	6.313	0.221
Random/ED	7.708	0.075
DR/MED	7.774	0.216
CPR/MED	8.193	0.185
Random/MED	8.904	0.062

Table 6.3. Average and standard deviation of overhead for varying number of groups in the system when the number of pawns in the path is two

PEON Protocol / Routing Algorithm	Bytes Transmitted per Byte of Data Delivered	
	Average (bytes/byte)	Std. Dev. (bytes/byte)
DR/ED	8.992	0.684
CPR/ED	10.509	0.463
Random/ED	13.821	0.138
DR/MED	7.358	0.436
CPR/MED	8.710	0.499
Random/MED	13.845	0.550

number of groups in the system. The values of delivery ratio remain more or less constant for all values of number of groups and this is true for all six cases of DR/ED, CPR/ED, Random/ED, DR/MED, CPR/MED, and Random/MED. Similarly, from Figures 6.5 and 6.6 we observe that average delivery delay and overhead, respectively, are also not affected by the number of groups in the system. The table in

Tables 6.1, 6.2, and 6.3 show the average and standard deviation of the values for delivery ratio, delay, and overhead, respectively, over the range of number of groups. Small standard deviations imply that the actual values for delivery ratio, delay, and overhead will be close to the average which further implies a consistent performance. For example, from the tables we can say that while using DR/ED for

any number of groups, the value of delivery ratio, delay, and overhead will be around 74.796%, 6.078 hours, and 8.992 bytes/byte respectivey. We can validate this by observing Figures 6.4, 6.5, and 6.6; we see that the actual values of delivery ratio, delay, and overhead are in the range 72.688% to 75.812%, 5.820 hours to 6.395 hours, and 8.138 bytes/byte to 10.224 bytes/byte, respectively.

Thus we can conclude that for our dataset, the group size has minimal effect on the performance.

CHAPTER 7

CONCLUSIONS

In this paper, we proposed PEON, an architecture for privacy in message routing in opportunistic networks. The PEON design extends prior work in Internet anonymity with pawn groups, through which routing can happen more efficiently than with standard anonymity approaches based on random node selection. We described three ways of routing messages in the PEON framework and the trade-offs available in each method. Finally, we describe the design and results of simulations based on traces of busses in the DieselNet framework. We showed that our DR protcol provides the best delivery ratio and delay, but our CPR protocol may provide the best trade-off of communication costs with privacy. Further, we found that the number of groups has little influence on the costs of our approach, while the number of intermediate nodes makes a big difference in all protocols.

7.1 Future Work

For future work, we intend to investigate the privacy of our approaches. We believe that most attacks, given a realistic attacker model, will be based on long-term observations of the user's patterns. Approaches like those of Troncoso and Danezis [28] may be useful for this work. In [24], we have described the key distribution framework and PEON protocls based on partially directed routing approach, we shall further explore these ideas, implement them and analyze the associated trade-offs. With these studies, we believe that we can design a complete PEON system that provides reasonable privacy and performance in a variety of scenarios.

APPENDIX A

DTN SIMULATOR

In this appendix, we present the DTN Simulator

A.1 SIMULATOR COMMAND LINE OPTIONS

`-verbose` *`number filename`*

- *`number`* indicates logging level; values between 1 and 11 (including them) [see util.Verbose for details]
- *`filename`* name of the file with parameters and schedule for the simulation

Other options include -stdin, -run_line, -rl, -rnl. However, we have not used these for our experiments.

NOTE: Messages will be logged in log.txt; this is different from the original DTNSim2 which prints these messages on console.

A.2 EXAMPLES PROVIDED WITH THE SIMULATOR

The examples discussed here are included in the 'examples' folder of the dtnsim2.

A.2.1 simple

This scenario is explained in the 'README' included with this example.

Nodes

A, B, C, D

Input files

simple_ED	ED
simple_MED	MED
simple_MEED	MEED
simple_Ideal	Ideal node (ED + infinite buffer space)
use_Ideal	Forces ED/MED/MEED/Ideal Node to use the Ideal contact
simple_Times	Contact schedule

Output files

simple_stats

The results can be verified by comparing them with the statistics for those tests in the file proper_stats.

A.2.2 dartmouth simple

This is a small scenario for basic tests, using one access point and four mobile devices. A pair of mobile devices is considered to be connected when the two devices are associated with the same access point at the same time. The contact schedule is kept separate from the data traffic in the network which is included in a separate file.

Nodes

Access point: AcadBldg21AP1

Mobile devices: c1103, c409, c2949, c1169

Input files

test_ED	ED
test_MED	MED
test_MEED	MEED
Common	Common configuration / traffic
dartmouth_simple	Contact schedule

Output files

stats

A.2.3 myexample

We have created this example to test our understanding of the configurations and input files required for running dtnsim2. This scenario consists of a base station and three buses. The base and buses come in contact with each other. Randomly

generated discrete messages sent from one node to another are embedded in the schedule. The perl script generate.pl is used to generate this scenario.

Nodes

Base station: base

Buses: bus1, bus2, bus3

Input files

simple_ED	ED
simple_MED	MED
simple_MEED	MEED
simple_Times	Contact schedule

Output files

simple_stats

A.3 SIMULATOR INPUT: CONFIGURATIONS AND FILES

A.3.1 Input File

This file is named in accordance with the routing algorithm being tested and has the configuration for the node, contact, and algorithms. Following are some of the important commands:

`@time` All the events which follow this line in the file are queued to be fired at this @*time*.

`/` Appears within a command at the end of line, it indicates that the next line in the file must be appended to the current command.

`interpreter run_file =` *`filename`*

Set *filename* to any other file that must be included at that point for being interpreted. The file included could have contact schedule, traffic configuration, or common configuration between different inputs.

`default_contact bandwidth =` *bandwidth* `latency =` *latency*

- *bandwidth*: Set its value to normal bandwidth for real contact. Set its value to INF for infinite bandwidth in case of ideal contact.
- *latency*: Set its value to >0 for real contact. Set its value to 0 for ideal contact.

`network_element node =` *node*

- *node*: Set to implementations.GlobalKnowledgeNode for ED and MED, set to implementations.MEEDNode for MEED, and set to implementations.EpidemicNode for Epidemic routing.

`default_node buffer_size =` *buffer_size* `msg_size =` *msg_size*

`algo =` *algo* `read_schedule =` *read_schedule*

- *buffer_size*: Buffer space for holding in-transit data. Set to -1 for infinite capapcity.
- *msg_size*: Set only if discrete messages are included in the contact schedule.
- *algo*: Set only for MED and ED. For MED set its value to MED. For ED set its value to ED.
- *read_schedule*: Set only for MED and ED to the file having contact schedule.

`stats prints file=`*filename* `field=`*field_caption_1*`=`*field_value_1*

...

...

...

`field=`*field_caption_n*`=`*field_value_n*

This will write the field values into a file named *filename*.

`@end` Indicates end of simulation. Use it to give a command to be performed at the end of simulation

For example,

`@end stats print ...`

will write to a file when the end of the simulation is reached.

A.3.2 Contact Schedule

The contact schedule is a set of records each of which either indicates when a contact between two nodes becomes available or unavailable. Following record indicates that at time *time*, the contact between *node1* and *node2* is available:

`@`*time* *node1* `<->` *node2* `up`

Following record indicates that at time time, the contact between node1 and node2 becomes unavailable:

`@`*time* *node1* `<->` *node2* `down`

A.3.3 Discrete Messages Included in Contact Schedule

Discrete messages each indicating the sending of a message from *node1* to *node2* at time *time* may be included in the contact schedule. Default_node msg_size must be set to specify the size of these messages.

`@`*time* `$`*node1* `send=`*node2*

A.3.4 Traffic

Traffic pertains to a series of messages being sent from one node to another. It may be included in a separate file other than the contact schedule. In our example we shall include it in a separate file which is generated using the perl script diesel-

nettraffic.pl. This script randomly picks up the start time (in the range 0 to half of the end time of contact schedule) for each pair of source and destination which is a combination of every node included in the contact schedule chosen as source with every other node chosen as destination. Following parameters have to be set.

`default_traffic start =` *`start`* `stop =` *`stop`* `interval =` *`interval`*

`msgs_per_fire =` *`msgs_per_fire`* `msg_size =` *`msg_size`*

- *`start`*: Default time to inject traffic for the first time in the system.
- *`stop`*: Default time to stop injecting traffic into the system.
- *`interval`*: Default interval of time at which an event is fired to inject a subsequent round of messages.
- *`msgs_per_fire`*: Default number of messages to be sent from any source to any valid destination when the event to send messages is fired.
- *`msg_size`*: Default size for each message injected into the system.

`traffic start =` *`start`* `src =` *`src`* `dest =` *`dest`*

- *`start`*: Specific time to send messages from the *`src`* node to the *`dest`* node for the first time.
- *`src`*: Particular originator of the messages.
- *`dest`*: Particular final recepient of the the messages.

A.3.5 Common Configuration

The configurations common to all tests can be included in a separate file. This is helpful in avoiding repetition as well as having to change the values of the common parameters in only one file rather than all input files corresponding to each test.

Note: Specific value of a property in X will overwrite general value of that property in default_X. For example, the value of 'traffic start' will overwrite the value set in 'default_traffic start'.

APPENDIX B

DERIVING CONTACT SCHEDULE

In this appendix, we discuss the derivation of contact schedule required for DTNSim2 from the DieselNet traces.

B.1 DIESELNET TRACES

We shall derive our contact schedule from DieselNet traces [29]. To obtain this trace Umass Transit's buses were equipped with DieselNet equipment. Every bus connects with other buses or access points (APs) when it comes in a range of contact with it. This trace contains details of contacts between:

- One bus and another bus
- A bus and an AP

All contacts occurring in a day are stored in a file which has that date (yyyy-mm-dd) as its name. A file for each date is stored in two directories - bus-bus and bus-ap. The bus-bus directory contains details of contact between two buses. The bus-ap directory contains details of contact between a bus and AP.

For each record in a file we have the following interpretation relevant to our experiment.

Column	Meaning
1	bus ID
2	bus ID (files in bus-bus) / MAC address of AP (files in bus-ap)
3	time of contact (in absolute minutes and seconds after midnight)
5	total duration of meeting (in seconds)

Entries where the first and/or second columns are null must not be considered.

B.2 OBTAINING CONTACT SCHEDULE FROM DIESELNET TRACES

The format of available dieselnet trace is very different from the desired contact schedule. The perl script dieselnetwithoutsend.pl takes as inputs the file for the same day from bus-bus and bus-ap directories, and generates the contact schedule having all valid contacts for that day. The contact schedule generated thus was large and must be reduced for using it with dtnsim2. Maximizing delivery ratio requires:

- Increasing bandwidth
- Decreasing latency
- Sending messages in the beginning of the schedule (say the first-half of the total period of the schedule)

The number of total nodes in the system needs to be reduced to avoid Out-OfMemory exception for maximum flexibility in increasing bandwidth and decreasing latency. Therefore, the schedule obtained from dieselnet traces has to be reduced by eliminating all but few access points. Note, that the number of access points is much larger than the number of buses. Therefore, many access points can be removed. Also the frequency of occurrence of access points in the schedule differ over a large range (800 to 2) so it is best to keep only the most frequent nodes.

The perl script dieselnetapcount.pl takes as input the complete contact schedule obtained from DieselNet traces and generates a list of 10 most frequent access points followed by all the buses in the schedule that are connected to either another bus or one of the 10 access points. In other words, less frequent APs and buses which connect to only these less frequent APs are not included in this list. This script also generates traffic between the nodes in this list.

Finally, the perl script dieselnetfilterschedule.pl takes as input this list of most frequent APs and buses as well as the complete contact schedule. It generates a reduced contact schedule by only retaining contacts wherein either of the nodes are

only from this list. Again each contact is either between a bus and another bus or between a bus and an AP. Corresponding to each date, yyyy-mm-dd, we obtain a reduced contact schedule in a file with that date as its name, we obtain a file named nodes_yyyy-mm-dd containing the list of APs and buses included in that schedule, and we also obtain a file named traffic_yyyy-mm-dd containing traffic for that schedule.

Note: Combining schedules of two days did not have much effect in increasing delivery ratio. The maximum delivery ratio may not be obtained for ideal contact (infinite bandwidth and zero latency).

APPENDIX C

PEON IMPLEMENTATION

C.1 PEON PACKAGE

In DTNSim2, a separate package has been created called peon which contains classes written to implement PEON. Following are the classes included in this package.

- PeonProperties: Properties which indicate whether PEON framework is enabled and define the grouping type, the number of groups in the system, the number of pawns that must be included in the path, the PEON protocol to be used, and the list of all nodes in the system.
- PeonOnion: This is a subclass of protocolStack.ProtocolStackMessage, it wraps up a DTN message and has other data required for defining a PEON onion such as the list of intermediate pawn groups, the initiator, responder, and the next pawn.
- GlobalKnowledgePawn: It is a sub-class of implementations.GlobalKnowledgeNode, it thus derives from a node which has the ability to use the ED and MED routing algorithms. Moreover, it has the knowledge of PEON protocols.
- PawnGroup: This class denotes a pawn group in the system, it contains a list of names of all pawns in that group.
- PawnGroupList: It encapsulates an ArrayList of PawnGroup. The list is populated in the begining of the execution before the simulation starts, only non-empty groups are included.
- PawnList: It encapsulates a HashMap of all the nodes in the system as the keys and the names of associated PawnGroups as their values. This map is also populated at the begining before the actual simulation begins.

C.2 CHANGES TO EXISTING CODE

These classes in the existing code for DTNSim2 have small changes required for PEON:

simulator.Main

- Added code for grouping nodes in the system when PEON is used

implementations.GlobalKnowledgeNode

- Added method getCost
- Added method getRoute

protocolStack.ProtocolStackNode

- Modified method acceptMessage - to allow data message to be routed back to previous immediate sender
- Added method forwardMessage

simulator.Node

- Modified method messageReceived - for a data message if this node is the destination and if PEON is enabled then it will call peonOnionReceived first to process that message
- Added (abstract) method forwardMessage

simulator.BasicNode

- Added method forwardMessage

simple.SimpleNode

- Added method forwardMessage

protocolStack.globalKnowledge.GKnowledgeTopologyHandler

- Added method getCost

protocolStack.EventHandler

- Added method getCost

C.3 CONFIGURATION FOR USING PEON

We now discuss the configuration required to use PEON. After configuring the system, run the simulator as usual for either ED or MED.

C.3.1 test_MED OR test_ED

`network_element node =` ***`node`***

Set the value of ***`node`*** as GlobalKnowledgePawn.

`default_node read_schedule =` ***`read_schedule`***

Set the value of ***`read_schedule`*** to the name of the file containing the conact schedule. In the DieselNet example, this will be the date of the schedule, for example, this could be 2007-10-26.

C.3.2 common

`interpreter run_file =` ***`filename`***

For the first occurance of this command, set the value of ***`filename`*** to the file containing the traffic for that schedule. For our example, it will be traffic_2007-10-26.

For the second occurance of this command, set the value of ***`filename`*** to the file containing the contact schedule itself. For our example, it will be set to 2007-10-26.

C.3.3 peon.properties

Set the value of `enablePeon` to true and the value of `nodeFile` to the name of the file containing the list of nodes corresponding to the contact schedule being used, for our example it will be nodes_2007-10-26. Set the values of the other parameters appropriately to indicate the grouping type, the number of groups in the system, the number of pawns that must be included in the path, and the PEON protocol to be used.

REFERENCES

[1] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," in *Proc. ACM SIGCOMM*, Oct. 2004.

[2] E. P. C. Jones, L. Li, and P. A. S. Ward, "Practical routing in delay-tolerant networks," in *Proc. ACM SIGCOMM*, Aug. 2005.

[3] A. Kate, G. M. Zaverucha, and U. Hengartner, "Anonymity and security in delay tolerant networks," in *Proc. SecureComm*, Sep. 2007.

[4] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–88, Feb 1981.

[5] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The next-generation Onion Router," in *Proc. USENIX Security Symposium*, August 2004.

[6] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic networking: Data forwarding in disconnected mobile ad hoc networks," *IEEE Communications Magazine*, vol. 44, no. 11, pp. 134–141, November 2006.

[7] HAGGLE Project. [Online]. Available: http://www.haggleproject.org

[8] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Impact of human mobility on the design of opportunistic forwarding algorithms," in *Proc. IEEE INFOCOM 2006*, April 2006.

[9] A. Pentland, R. Fletcher, and A. Hasson, "Daknet: Rethinking connectivity in developing nations," *IEEE Computer*, vol. 37, no. 1, pp. 78–83, Jan. 2004.

[10] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," Duke University, Tech. Rep. CS-2000-06, Jul. 2000.

[11] C. Díaz, S. Seys, J. Claessens, and B. Preneel, "Towards measuring anonymity," in *Proc. Privacy Enhancing Technologies workshop (PET)*, April 2002.

[12] A. Serjantov and G. Danezis, "Towards an information theoretic metric for anonymity," in *Proc. Privacy Enhancing Technologies Workshop (PET 2002)*, April 2002.

[13] N. Hopper, E. Y. Vasserman, and E. Chan-Tin, "How much anonymity does network latency leak?" in *Proceedings of CCS 2007*, October 2007.

[14] G. Danezis, "Statistical disclosure attacks: Traffic confirmation in open environments," in *Proc. of Security and Privacy in the Age of Uncertainty, (SEC)*, May 2003, pp. 421–426.

[15] N. Mathewson and R. Dingledine, "Practical traffic analysis: extending and resisting statistical disclosure," in *Proc. Privacy Enhancing Technologies workshop (PET 2004)*, May 2004.

[16] M. Wright, M. Adler, B. Levine, and C. Shields, "The predecessor attack: An analysis of a threat to anonymous communications systems," *ACM Transactions on Information and Systems Security (TISSEC)*, vol. 7, no. 4, 2004.

[17] M. Wright, M. Adler, B. N. Levine, and C. Shields, "Passive logging attacks against anonymous communications," *ACM Transactions on Information and Systems Security (TISSEC)*, vol. 11, no. 2, 2008.

[18] Y. Zhu and R. Bettati, "Compromising location privacy in wireless networks using sensors with limited information," in *Proc. Intl. Conf. on Distributed Computing Systems (ICDCS 2007)*, Jun. 2007.

[19] K. Mehta, D. Liu, and M. Wright, "Location privacy in sensor networks against a global eavesdropper," in *Proc. IEEE International Conference on Network Protocols (ICNP)*, Oct. 2007.

[20] S. J. Murdoch and P. Zieliński, "Sampled traffic analysis by internet-exchange-level adversaries," in *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*, June 2007.

[21] R. Snader and N. Borisov, "A tune-up for Tor: Improving security and performance in the Tor network," in *Proc. Network and Distributed Security Symposium (NDSS '08)*, Feb. 2008.

[22] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-resource routing attacks against TOR," in *ACM WPES*, 2007.

[23] M. Wright, M. Adler, B. Levine, and C. Shields, "Defending anonymous communications against passive logging attacks," in *Proc. IEEE Sym. on Security and Privacy*, May 2003.

[24] Z. Le, G. Vakde, and M. Wright, "Peon: Privacy-enhanced opportunistic networks with applications in assistive environments," in *Proc. Workshop on Privacy and Security in Pervasive e-Health and Assistive Environments (PSPAE)*, Jun. 2009.

[25] (2009) DTN Simulator: DTNSim2. dtnsim2_20060624.tar.bz2. [Online]. Available: http://watwire.uwaterloo.ca/DTN/sim/

[26] E. P. Jones, L. Li, J. K. Schmidtke, and P. A. Ward, "Practical routing in delay-tolerant networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 8, pp. 943–959, Aug. 2007.

[27] A. Balasubramanianm, B. N. Levine, and A. Venkataramani, "Enabling interactive applications for hybrid networks," in *Proc. ACM Mobicom*, September 2008.

[28] C. Troncoso and G. Danezis, "The bayesian traffic analysis of mix networks," in *Proc. ACM Conference on Computer and Communications Security (CCS '09)*, Nov. 2009.

[29] (2007) DieselNet Fall 2007 - AP Connectivity. mobicom-traces.tar.gz. [Online]. Available: http://traces.cs.umass.edu/index.php/Network/Network

BIOGRAPHICAL STATEMENT

Gauri Vakde received her M.S. degree with a Thesis from The University of Texas at Arlington in 2010 in Computer Science.