Onion Routing in Predictable Delay Tolerant Networks

Adrian Antunez-Veas

Department of Information and Communications Engineering,
Universitat Autònoma de Barcelona (UAB), 08193 Cerdanyola, Spain
aantunez@deic.uab.cat

Abstract With the growth of Internet of Things (IoT) the privacy and security has become a challenging research topic. Using Delay Tolerant Networks (DTNs) along with onion routing a secure and anonymous independent network can be achieved. Predictable DTNs are networks where the behaviour is known in advance or where a repetitive action occurs over the time like in public transport networks. This project shows how the prior stage of path selection in onion routing can be achieved using the information provided by predictable networks. We also analyse the security of our proposed method.

Keywords: Onion Routing, Delay-tolerant Network, Anonymous networking, Privacy, Security

1 Introduction

Delay Tolerant Networks (DTNs) [18] are a class of networks aimed to provide end-to-end communication into environments lacking continuous connectivity or with long delays. To do so, a new network layer called bundle is used. It works on top of the transport layer to handle disruptions and long delays using the store-carry-and-forward principle. The received data, called bundle, is stored in persistent storage on each DTN node, forwarding it when a new opportunity of contact appears.

There are several scenarios where DTNs can be useful like sensing systems, deep space missions, disaster recovery, and rural area networks. Many of these applications need to preserve user information as they can share sensitive information. In traditional encryption the transmitted data is hidden from third parties. Nonetheless, the source and destination identities are shown, leading to potential information leaks about sender and receiver. Techniques can be used to preserve the anonymity too, e.g: in anonymous sensing systems a user may want to preserve his identity to avoid possible reprisal.

A popular technique to achieve anonymous communication in traditional networks is onion routing [6]. Onion routing provides source anonymity by encapsulating messages in layers of encryption. It is said that onion routing is not aplicable to DTNs because onion routing method needs to know the route in

advance to perform the cryptographic layers for each node [1]. Nevertheless, this protocol can be applicable in deterministic (predictable) DTNs. Deterministic DTNs have been studied before [2], [7]. Using predictable DTNs, the route to encrypt the message needed in onion routing can be obtained. At the same time, we solve the challenging routing problem in DTNs [11] because we decide to which nodes forward the message.

Previous research work on the use of onion routing in DTNs is very scarce. Possibly, the most relevant is ARNDEN [15]. ARDEN uses groups of nodes to perform the layering process and broadcast messages between these groups. The message's route is chosen randomly. To perform the cryptographic layers attribute based encryption (ABE) is used. Our method is easier to implement than ARDEN, and does not need complex protocol cryptography methods like ABE.

The remainder of the paper is organized as follows: Section 2 presents a deeper review of relevant concepts used in our proposal. Section 3 shows the process to use onion routing over predictable DTNs. A method to know the route in advance, given a set of restrictions, for the layering process is proposed. Section 4 presents an informal analysis of the security of our proposal from the point of view of passive and active attackers. Section 5 presents the results of our simulation over a public transport network (considered predictable networks). This results show the performance of our method in terms of different statistical metrics. Section 6 provides concluding remarks as well as possible future research work.

2 Relevant concepts and assumptions

In this section, we review relevant concepts like predictable DTNs, onion routing and key management that we consider essential to understand our proposal.

2.1 Predictable DTNs overview

Predictable DTNs are such networks where the behaviour is known in advance or where a repetitive action occurs over time.

These networks fit perfectly with the concept of oracle schemes. Oracle schemes are a subset of DTN routing protocols. In this schemes there is a set of nodes that have nearly full knowledge of the network and its planned evolution [4]. There are oracles that can answer any question regarding contacts between two nodes at any point in time, this oracles are called contacts oracle [11]. Contacts oracle schemes are used in deterministic scenarios where the oracle node have full details of occurred contacts as well as the future ones.

For instance, we could consider public transport networks as predictable (deterministic) networks because every node performs the same route periodically so this route can be known in advance. This information could be shared among all nodes in the network in order to let them be "oracle", i.e: let them have full knowledge of the whole network. It is important to note that, despite these

networks are nearly deterministic, an error needs to be assumed for exceptional situations.

As we explained previously, contact information will be known by everyone, therefore a security analysis needs to be carried out. This analysis is discussed in section 4.

2.2 Onion routing overview

We provide a high-level description of how onion routing works. The source node, wishing to send an anonymous message to another node, chooses a path ciphering the message several times making "layers". Onion routing uses asymmetric cryptographic protocols where the corresponding public key between the source and a node on the path is shared. Each layer of encryption is performed with the corresponding public key. When the message reaches a node, the layer is removed with the node's private key and the packet gets modified. At the end of the process, if every node in the path did the decryption correctly, the destination will receive the fully decrypted message without knowing who sent it.

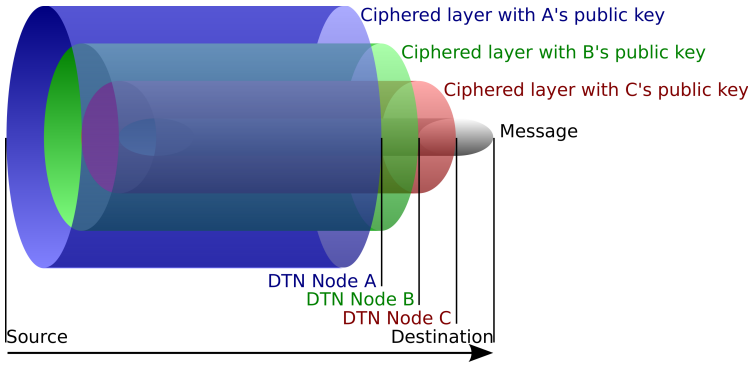


Figure 1: Onion routing graphical example of how layering works [19].

To clear out the main idea, an example can be seen in figure 1. The source node wants to send an anonymous message to a destination using a previously chosen path composed by the nodes A, B and C. The source ciphers the message with the public key of C, B and A respectively. When the message reaches every hop, each node removes the external layer using the node's private key. Finally, the destination will get the fully decrypted message.

2.3 Key management

In order to use onion routing a prior key exchange process needs to be performed. We consider that the loading of the network information behaviour as well as the keys sharing process will be done simultaneously at a previous stage, i.e: before

starting the periodical routing. For example, in public transport networks the data as well as the needed keys will be shared before the scheduled route starts.

3 Onion routing for predictible DTN

In this section, we define a model of predictable DTNs using dynamic graphs. After that, we propose an algorithm for onion routing path selection process using the contact knowledge of the network.

We need to perform the following steps in order to successfully send an anonymous message:

- **Step 0:** Key exchange process, load of contact knowledge and network setup.
- **Step 1:** The source s chooses a destination d, an instant of time t to send the message, the number of routers in the path n and the number of paths k.
- **Step 2:** The source, using the path selection method, obtains a set of paths and choose one path p randomly.
- **Step 3:** The message is sent from s to d at time t through p path using onion routing.

3.1 Contact representation

Clearly, we need to use a structure to represent the network activity. One method that seems to work is to use a graph structure as a way of network activity representation [2], [7], [9]. Each node of the network is represented as a graph vertex (or node) and each contact between nodes as a graph edge.

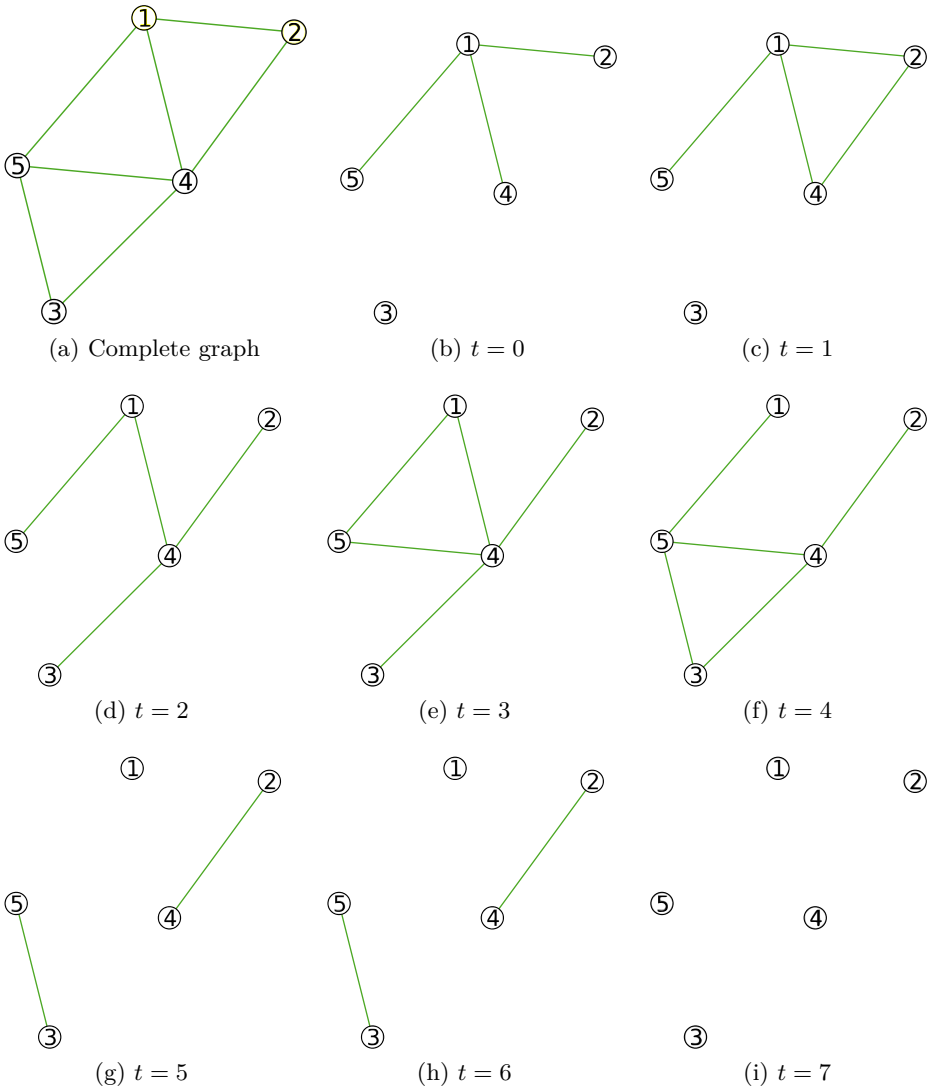
There is a subset of graphs in the graph theory called *dynamic graphs*. Dynamic graphs allow to represent the evolution of the network during time easily, i.e: nodes and edges may change positions and appear and/or disappear through time. To do so, we can set a pair of attributes that point out when a node appears and disappears.

The network is modelled as a dynamic graph $G = (V, E)$. The set of vertices V is the set of nodes, and the edges E are the contacts between nodes. Dynamic graphs provide a good way to store dynamic information about the network. In our case this information will be related to contacts like the instant of time when the connection opportunity occurs and how long this contact has been.

In figure 2 the evolution of a given graph representing different contacts between buses in a public transport network is shown. In this example we can see that at the beginning, no contacts have been produced yet. As the time passes, the contacts (*edges*) appear and disappear between buses (*nodes*).

3.2 Path selection

Imagine that we have a source node s willing to send a message to a destination d at time t using onion routing. In onion routing you need to choose the number of nodes n where the message will pass through. Node s obtains a path to perform



(a) Complete graph (b) $t = 0$ (c) $t = 1$

(d) $t = 2$ (e) $t = 3$ (f) $t = 4$

(g) $t = 5$ (h) $t = 6$ (i) $t = 7$

Figure 2: In (a) the accumulative graph $(t = [0, +\infty))$ can be seen. In $(b) - (i)$ different snapshots in given instants of time showing the evolution of the dynamic graph are depicted.

the layering process and send the message. The time required to exchange the message between nodes is defined as *tt*.

To make even harder the path guessing from third parties, as the network knowledge is shared among all nodes, the node s obtains a set of paths k in order to choose one of them randomly. To sum up, we need a method *f(s, d, t, n, k, tt)*

that will retrieve a set of paths to finally choose one following the t, n, k and tt requirements.

The procedures shown in Algorithm 1 and Algorithm 2 are an example of deterministic choosing for onion routing. Both of them inherit values to filter out unneeded paths. Specifically, in Algorithm 1 from a given node *host* at time *currentTime* we return a set of neighbors that are available or will be available to forward the message. In Algorithm 2 we perform a recursive search to get up to k paths of length n from *source* to *destination*.

In figure 2 there is an example of contact data representation using dynamic graphs. Applying our path selection method with the following parameters: source node *s=1*, destination node *d=5*, starting time *t=0*, number of paths *k=4*, number of nodes of each path *n=5* and transmission time *tt=1*, we get the following paths:

Path: 0
(1:0) → (2:0) → (4:1) → (1:2) → (5:3). Arrival time: 4
Path: 1
(1:0) → (2:0) → (4:1) → (3:2) → (5:4). Arrival time: 5
Path: 2
(1:0) → (4:0) → (1:1) → (4:2) → (5:3). Arrival time: 4
Path: 3
(1:0) → (4:0) → (2:1) → (4:2) → (5:3). Arrival time: 4

(node: t): Means forward the message to node at time *t*.

So, we get 4 different paths composed by 5 nodes each, i.e: origin and destination plus 3 onion routers, so 3 layers will be performed. From the given set of paths, the source node s will need to choose one in order to perform the onion routing itself. To make the guessing of the source node difficult, the path to send the message will be chosen randomly from the given set.

Algorithm 1 Procedure to get valid neighbors of a given node

INHERIT: (1) *source*, source node. (2) *destination*, destination node. (3) t, when the message will be sent. (4) k, maximum number of paths. (5) n, number of nodes for each path (including source and destination). (6) tt, transmission time required to forward the message.
INPUT: (1) *host*, host node. (2) *currentTime*, current time.
OUTPUT: (1) *validNeighbors*, a set of valid neighbors to whom forward the message.

```
1: procedure GetAvailableNeighbors (host, currentTime)
2:     for each nbr ∈ host.neighbors() do
3:         if (nbr.activationTime + nbr.contactDuration − tt) ≥ currentTime then
4:             validNeighbors.add(nbr)
5:     return validNeighbors
```

Algorithm 2 Procedure to get paths that follows the inherited requirements

INHERIT: (1) *source*, source node. (2) *destination*, destination node. (3) t, when the message will be sent. (4) k, maximum number of paths. (5) n, number of nodes for each path (including source and destination). (6) tt, transmission time required to forward the message.
INPUT: (1) *currentPath*, current path. (2) *currentTime*, current time.
OUTPUT: (1) *paths*, a set of paths that follows the inherited requirements.

```
1: procedure GetPaths(currentPath, currentTime)
2:     if currentPath.size() ≠ n and paths.size() < k then
3:         node ← currentPath.last()
4:         validNeighbors ← GetAvailableNeighbors(node, currentTime)
5:         for each nbr ∈ validNeighbors do
6:             oldPath ← currentPath
7:             nbr.sendingTime ← max(nbr.activationTime, currentTime) + tt
8:             currentPath ← nbr
9:             if nbr = destination and currentPath.size() = n and currentPath ∉ paths
   then
10:                validNeighbors.add(nbr)
11:            else
12:                GetPaths(currentPath, nbr.sendingTime)          ▷ Recursive part
13:            currentPath ← oldPath
14:    return paths
```

4 Security analysis

In this section, we evaluate our proposed scheme from a security perspective. First, we define the threat model, defining important considerations regarding the scenario under study. Later, we define different kinds of adversaries explaining what they can do and what we do to preserve the sender anonymity as well as the privacy of exchanged information.

4.1 Threat model

Alice (A) wants to communicate to Bob (B) without revealing information about her. Using onion routing we intend to improve A's anonymity as well as data privacy against adversaries that can be passive (eavesdropping messages) or take an active role in this scenario (make message modifications or attack to other nodes of the network). In general, security threats can be divided into passive and active threats. We consider that nodes in our scenario use strong cryptographic algorithms with enough key lengths to prevent practical cryptanalysis attacks to discover the source, the destination or the contents of a message.

4.2 Passive adversaries

Passive attacks are those that perform guessing simply observing user traffic patterns from nodes.

As is explained in [8] if the attacker is the destination of the message, he can learn something from the delay between messages. This kind of attacks does not work when the sending start time, known as t in our path selection method, is not highly predictable [17].

Another attacker model is a set of compromised nodes that work together to retrieve information leading to break the users privacy. One possible situation is the multiple decryption. An attacker will be able to decrypt more layers, or messages if one of the nodes is the destination, because they have their corresponding private keys. To overcome such attacks the n value can be increased, i.e: the number of nodes that will have to pass through the message. A message will have as much layers as nodes are in the path. There is a trade-off between efficiency and security when deciding the value of n that needs to be studied in future researches.

Another possible scenario where a set of compromised nodes work together is the sending node periodicity analysis. There are scenarios where a node or a set of them rarely transmit information to others, discarding this nodes from the probable sending set, a globally passive adversary can correctly guess the source of a message. To solve the guessing issue the creation of dummy packets when ingress throughput drops below a certain threshold [15] may help to prevent such attacks.

To decrease the probability of guessing the path when several nodes are compromised and one of them is the destination, different paths are retrieved using our path selection method and one of them is chosen randomly. By this way, we pretend to make harder to guess the sender of the message simply analysing the origin and the destination of every compromised node in the path as well as the sending time.

It is important to note that an attacker can combine previously explained attacks to increase its chance of guessing.

4.3 Active adversaries

Active adversaries are those that perform actions against other nodes or modify information transmitted through them. As is the case of passive nodes, an attacker controlling a single node will be unable to extract the source of a message because of the use of multiple layers of encryption [15]. There are several possible attacks against onion routing by malicious nodes in the network like congestion attacks [3], location based attacks [5], and attacks based on latency analysis [8].

An attacker who has control of a node of the network can attack other nodes to shut them down leading to undelivered messages. This kind of attacks are called Denial of Service DoS attacks and can be addressed improving the robustness of the nodes as well as with reputation systems as are discussed in section 6: conclusions and future work.

Message modifications by attackers are easily detected using cryptographic hash methods. Other attacks like masquerading (nodes pretending to be different nodes) are solved as layers of encryption check the node identity. The key management process is safely done in a prior stage.

5 Evaluation

We conduct simulation based on realistic scenario trace file to verify the feasibility of our proposal. We decided to recreate the small public transport network of the UAB campus using the Network Simulator 3 (NS-3). NS-3 is a well-known discrete-event simulator targeted primarily for research [12]. Moreover, we explain how the mobility model was obtained as well as what parameters were used during the simulation. Finally, we will test our proposed method in the simulation to evaluate how it performs.

5.1 Scenario: UAB campus buses

In order to test our proposal we considered a very small public transportation network that works inside the Autonomous University of Barcelona (UAB) composed by 5 buses that make different routes around the UAB campus. It is important to note that every single bus makes the same route daily. By this way, this scenario can be seen as a good example of deterministic networks.

Each bus has a DTN node that allows users to achieve secret communications as well as source anonymity using onion routing protocol.

5.2 Mobility Model

We obtained the mobility model going through different stages. First, we exported the UAB campus road map from OpenStreetMap into SUMO software [10], filtering some unnecessary items like buildings and railways with the Java OpenStreetMap editor tool [13].

Once the campus roads were imported in SUMO, we recreated the bus movements of each bus taking into consideration the official bus schedule of the UAB public transportation network [16]. In addition, we tuned some bus characteristics like acceleration and deceleration parameters in order to get coherent travel times.

Finally, we converted the model to a NS-2 mobility trace as is explained in [14]. The NS-2 mobility trace can be used in NS-3. We used the simulator to obtain important contact related data of the campus network, i.e: information about the duration of the contacts as well as the instant of time when they occurred.

5.3 Simulation setup

The DTN modules are under current development in NS-3 [20]. For this reason we decided to implement a neighbour discovery in the application layer. This application broadcasts beacon messages periodically looking for new contact opportunities. The interval time is the time to wait between beacons while the expiration time is the time a neighbour is considered valid, these parameters can be set up manually.

Parameter	Value
Number of nodes	5
Wi-Fi range	100 meters
Interval time	1 second
Expiration time	2 seconds
Simulation time	15 hours
DSS Rate	1 Mbps
IEEE 802.11 specification	802.11b

Table 1: Simulation setup.

In table 1 we show different parameters for the neighbour discovery application as well as different wireless parameters to be suitable with resource-constrained computers.

5.4 Simulation results

In this section we show some data related to the simulations. We provide a general overview of the network and measures related to the contact behavior. We also analyze our proposal over the real scenario of the UAB campus.

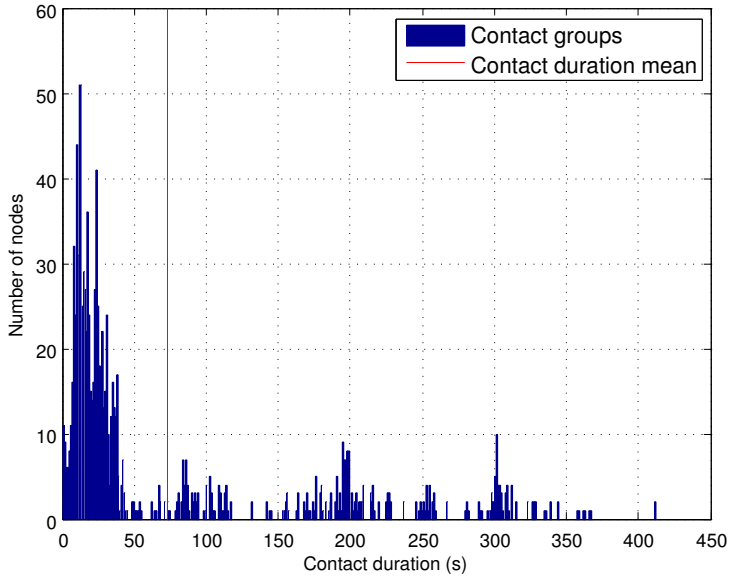


Figure 3: Number of contacts with the same duration.

In figure 3 we have an overall view of the network activity. There is a group that has really short contact times, these contacts can be suitable for those

applications that do not require a huge amount of data to be sent. There is another group that has long contact times (nearly 7 minutes), this group is able to perform complex communications sending higher amount of data. The average contact time is near 1 minute, suitable for several applications like anonymous sensing systems. A summary of contacts related information during the simulation is shown in table 2. With this simple network analysis we show that our evaluation model can be used for several applications.

Metric	Value
Number of contacts	1161
Average contact time	72.72 seconds
Maximum contact time	412 seconds
Minimum contact time	1 second

Table 2: Contact information.

In figure 4 after fixing *s=1; d=5; t=0* and *k=10* we set the *n* value from *n=5* to *n=40*. We computed the average delivery time mean of the *k=10* given paths. The minimum number of nodes allowed in onion routing is 5, i.e: at least 3 routers plus source and destination nodes. Unlike in traditional networks, in DTNs not always the shortest paths are the quickest ones, but even so we can see in the figure an increasing tendency.

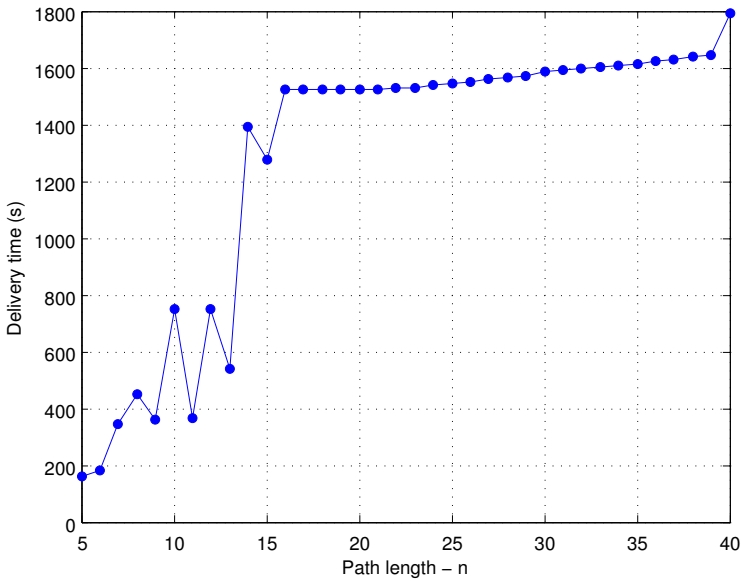


Figure 4: Average delivery time considering the variation of the path length.

In figure 5 after fixing *s=1; d=5; t=0* and *n=4* we set the *k* value from *k=1* to *k=15*. We can see that the order of appearance of each new path is not intermediately correlated to time (a new path can be quicker or slower than the previous ones). By this way, it is even more difficult to guess the path chosen. In this example, there is a maximum number of paths for the given set of attributes of 13. After that, as we can see, no more paths will be gotten.

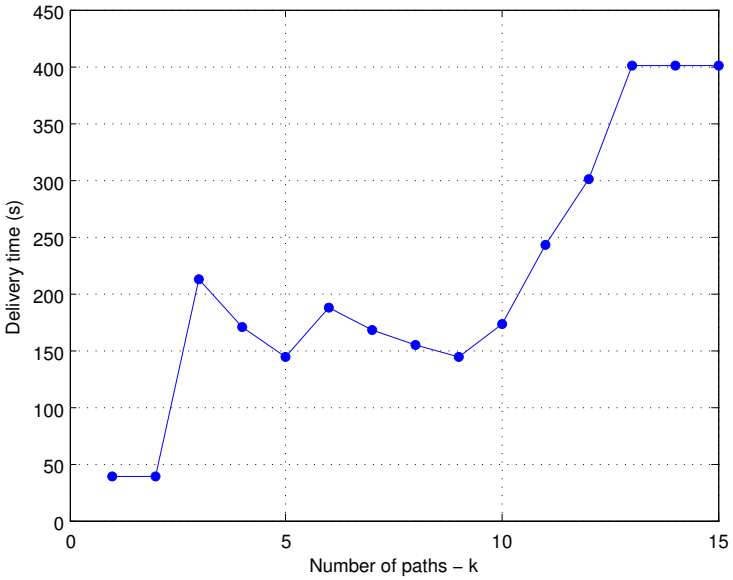


Figure 5: Average delivery time considering varying number of paths.

6 Conclusions and future work

In this paper, we proposed a method to exchange anonymous and private information using onion routing over a DTN network applying dynamic graphs as a way of representing periodical contact information.

An important conclusion is that unlike in traditional networks, in DTNs not always the shortest paths are the quickest ones. In addition, we can conclude that using our algorithm we can obtain several paths that are not directly related to time. By this way, we make the guessing of the chosen path more difficult.

In our future work, we will search and analyse efficient ways of path selection, as currently our path selection method is non efficient because it explores every possible path. The efficiency comes crucial when resource-constrained computers are involved like the tiny computers used in IoT, that could fit perfectly in each bus in our scenario. Another future research branch can be to decrease

the impact of active attacks like black holes using a reputation system. As, the reputation value will be shared among all nodes in the network. The path selection algorithm should be modified too to take this value into consideration in the path selection process. Finally, this will lead into another security analysis because more reputation a node has, most probable is to be one of the chosen nodes in the path.

We also noted that the contact data representation can be dynamic, i.e: the simulation model can be adapted to consider traffic modifications, generating enough information to decrease the number of failed contacts due to a bad contact prediction.

Acknowledgements The author would like to thank Guillermo Navarro Arribas for his excellent advice in this research work.

References

1. Bhutta, N., Ansa, G., Johnson, E., Ahmad, N., Alsiyabi, M., Cruickshank, H.: Security analysis for delay/disruption tolerant satellite and sensor networks (2009)
2. Cardei, I., Liu, C., Wu, J., Yuan, Q.: Dtn routing with probabilistic trajectory prediction. In: Wireless Algorithms, Systems, and Applications, pp. 40–51. Springer (2008)
3. Evans, N.S., Dingledine, R., Grothoff, C.: A practical congestion attack on tor using long paths. In: USENIX Security Symposium. pp. 33–50 (2009)
4. Farrell, S., Cahill, V.: Delay- and Disruption-tolerant Networking. Artech House telecommunications library, Artech House (2006)
5. Feamster, N., Dingledine, R.: Location diversity in anonymity networks. In: Proceedings of the 2004 ACM workshop on Privacy in the electronic society. pp. 66–76. ACM (2004)
6. Goldschlag, D., Reed, M., Syverson, P.: Onion routing. Communications of the ACM 42(2), 39–41 (1999)
7. Hay, D., Giaccone, P.: Optimal routing and scheduling for deterministic delay tolerant networks. In: Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on. pp. 27–34. IEEE (2009)
8. Hopper, N., Vasserman, E.Y., Chan-TIN, E.: How much anonymity does network latency leak? ACM Trans. Inf. Syst. Secur. 13(2), 13:1–13:28 (Mar 2010)
9. Hossmann, T., Legendre, F., Spyropoulos, T.: From contacts to graphs: pitfalls in using complex network analysis for dtn routing. In: INFOCOM Workshops 2009, IEEE. pp. 1–6. IEEE (2009)
10. Institute of Transportation Systems: SUMO - Simulation of Urban MObility, http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000
11. Jain, S., Fall, K., Patra, R.: Routing in a delay tolerant network, vol. 34. ACM (2004)
12. Nsnam: NS-3 webpage, https://www.nsnam.org
13. Open Street Map: JOSM, https://josm.openstreetmap.de
14. Raj, C., Upadhayaya, U., Makwana, T., Mahida, P.: Simulation of vanet using ns-3 and sumo. International Journal of Advanced Research in Computer Science and Software Engineering 4, 563–569 (2014)

15. Shi, C., Luo, X., Traynor, P., Ammar, M.H., Zegura, E.W.: Arden: Anonymous networking in delay tolerant networks. Ad Hoc Networks 10(6), 918 – 930 (2012)
16. UAB: Horaris d'autobus 2015, http://www.uab.cat/doc/horaris_busUAB_2015
17. Vakde, G., Bibikar, R., Le, Z., Wright, M.: Enpassant: anonymous routing for disruption-tolerant networks with applications in assistive environments. Security and Communication Networks 4(11), 1243–1256 (2011)
18. Warthman, F.: Delay tolerant networks (dtns) a tutorial, 2003. Warthman Associates
19. Wikipedia: Onion routing (2015), https://en.wikipedia.org/wiki/Onion_routing
20. Zhou, D.: DTN Bundle Protocol, https://www.nsnam.org/wiki/Current_Development

All links were last accessed on June 29, 2015.