Onion Routing in Predictable Delay Tolerant Networks

Adrian Antunez-Veas and Guillermo Navarro-Arribas

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

Abstract With the growth of Internet of Things (IoT) the data as well as the anonymity preservation has become a challenging research topic. Using Delay Tolerant Network (DTN) with onion routing an alternative secure network that hides the source identity can be achieved. We consider as predictable DTN at such 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 choosing in the onion routing can be achieved using the information provided by predictable networks while the security of our proposed method is analysed.

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

1 Introduction

Delay Tolerant Networks (DTNs) [1] 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 layer called Bundle layer is used. It works on top of 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 occurs.

There are several scenarios where the DTNs can be useful like reporting systems, deep space missions, disaster recovery and rural area networks among other things. Many of these applications needs to preserve user information as they can share sensible information. In traditional encryption the transmitted data is hidden from thirds. Nonetheless the source and destination identities are shown, leading to possible attackers know information about the victim. Using onion routing we are able to preserve the anonymity of such nodes too, e.g: in anonymous reporting systems a user may want to preserve his identity to avoid possible reprisal.

We will focus in reporting systems where the data as well as the identity of the nodes need to be protected using the DTN as a communication alternative as DTN can be used as an alternative when the usual network can not be trusted.

A popular method to achieve sender anonymity as well as data preservation is onion routing [2]. Onion routing is used to protect the communications allowing to hide the source of a message as well as the data itself to the rest of the nodes that forwards the information. It is said that onion routing is not aplicable to DTNs [3]. Nevertheless, this protocol can be applicable if we use predictable (deterministic) DTNs. Deterministic DTNs has been studied before [4], [5]. A method to represent the network behaviour, as is used in our proposal, is trough a graph representation where nodes are the elements of the network and each edge of the graph is a contact between nodes.

As onion routing needs to know the route in advance to encrypt the message accordingly for each node in the path, we solve the challenging routing problem in DTNs [7].

There are previous research using DTN and onion routing together to achieve anonymous communications like in [12]. Explain why this algorithm is not as good as ours. Explain other methods that was researched like [11]...

The remainder of the paper is organized as follows: Section 2 presents a deeper review of relevant concepts used to design an algorithm to choose a path from a given set of restrictions. Allowing to perform the layering stage of onion routing. Section 3 presents an informal analysis of the security of our proposal from the point of view of passive and active attackers. Section 4 presents the results of our simulation over public transport networks, as they are considered predictable networks. This results show the performance of our method in terms of different statistical metrics. Section 5 provides concluding remarks as well as possible future research work.

2 Proposal

In this section, we review some relevant concepts like predictable DTNs, onion routing and key management. Later, we define a method to put together all contacts information. Finally, we propose an algorithm for onion routing path choosing process using the prior knowledge of the network.

2.1 Relevant concepts and assumptions

We define as predictable DTNs at such networks where the behaviour is known in advance or where a repetitive action occurs over the time.

These networks fits 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 has nearly full knowledge of the network and its planned evolution [6]. Contacts oracle is an oracle that can answer any question regarding contacts between two nodes at any point in time [7]. Contacts oracle schemes are used in deterministic scenarios where the oracle node will have full details of occurred contacts as well as the future ones, among other things.

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 a full knowledge of the behaviour of the whole network. Is important to note that despite these networks are nearly deterministic, an error needs to be assumed for exceptional situations.

As we explained previously, all network information will be known even by active adversaries, therefore a security analysis needs to be carried out. This analysis is discussed in section 3.

As onion routing understanding is essential to our proposal, we provide a high-level description of how onion routing works as is the essence of this proposal. The source node, wishing to send an anonymous message to another node, chooses a path and ciphers the message several times making "layers". Each layer of encryption uses a different pre-shared key between the source and a node on the path. When the message reaches a node, the layer is removed with the private key of the node and the packet gets modified. At the end of the process the destination will receive the fully decrypted message without knowing who sent it.

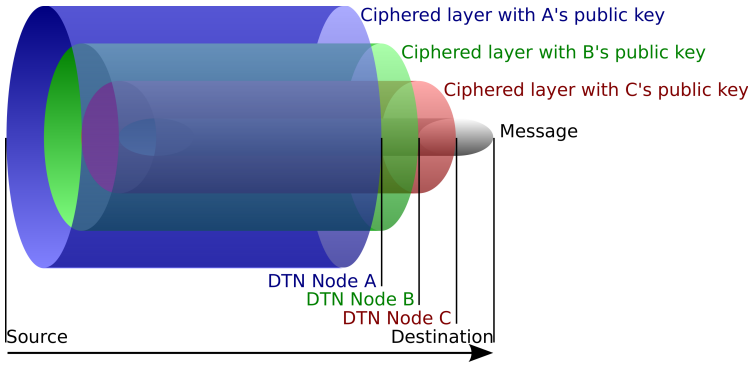


Figure 1: Onion routing graphical example of how layering works.

An example can be seen in figure 1 where 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 private key.

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.

2.2 Contact data representation

Is it clear that 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 to represent network activity [4], [5], [8]. Where each node of the network is represented as graph vertices and each contact between nodes as graph edges.

A graph $G = (V, E)$ consists of two sets V and E [9]. The elements of V are called *vertices* or *nodes* while the elements of E are called *edges*. Each edge is composed by two vertices. Edges and vertices can have *attributes*. An attribute is a value associated either to edges and vertices that permits to store information.

There is a subset of graphs in the graph theory called *dynamic graphs*. Dynamic graphs permit to represent the evolution of the network during the 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.

Dynamic graph provides a good way to store 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*).

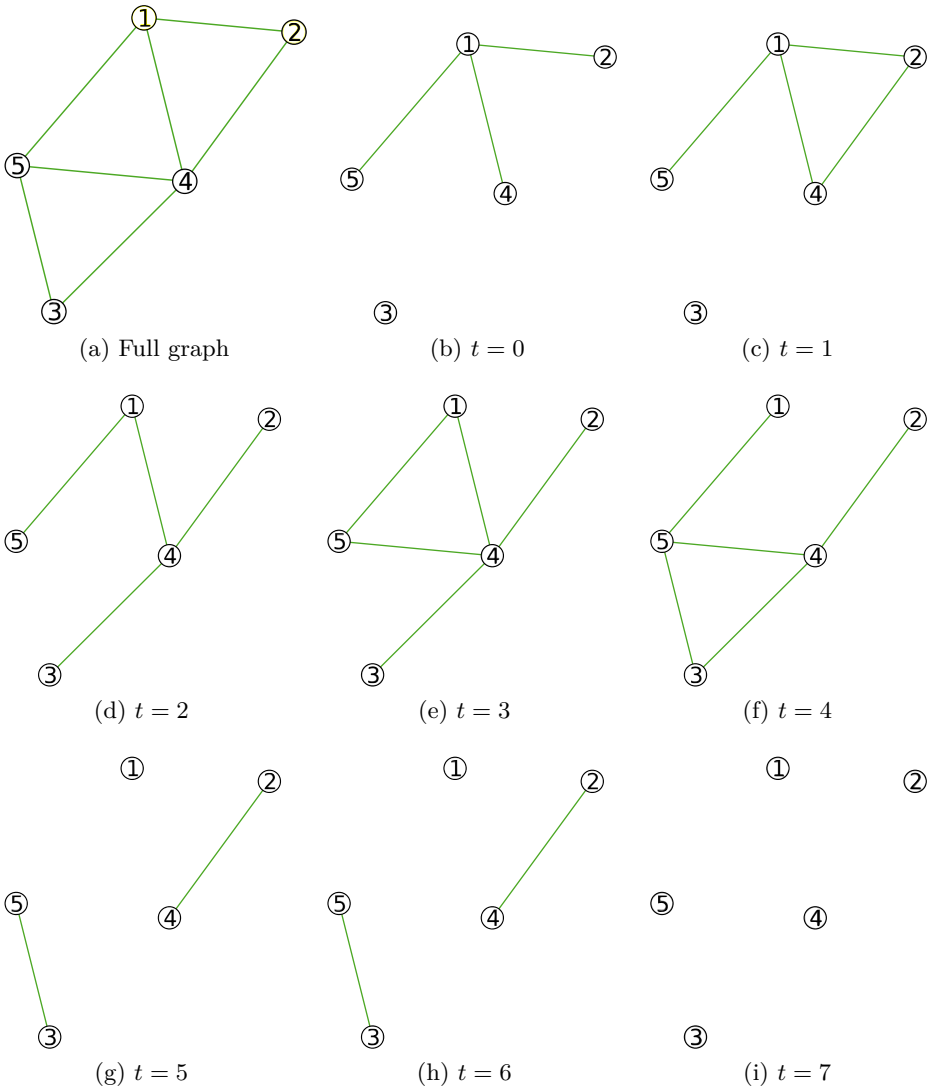
2.3 Path choosing

Imagine 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 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 thirds, 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 choosing 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:



(a) Full 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 full (static) graph can be seen. In $(b) - (h)$ different snapshots in given instants of time showing the evolution of the dynamic graph are depicted.

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

(n: t): Means message sent to node n at time t.

So we get 4 different paths composed by 5 nodes each, i.e: 3 layers will be done. 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
```

3 Security analysis

In this section, we evaluate our proposed scheme from the aspect of security. First, we define the threat model, defining important considerations regarding the scenario under study. After, 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.

3.1 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,i.e: eavesdropping messages or take an active paper in this scenario, i.e: make 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.

3.2 Passive adversaries

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

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
```

As is explained in [10] 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 sending start time, known as t parameter in our path choosing method, is not highly predictable [11].

Another attacker model is a set of compromised nodes that works together to retrieve information leading to break the users privacy. We have two different situations to deal with: the multiple decryption and the sending node periodicity. First, the 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. Second, 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 overcome such attacks the n value can be increased, i.e: the number of nodes that has a single path to send the message. As much nodes, much layers a message will have. There is a trade-off between efficiency and security deciding n value. To solve the guessing issue the creation of dummy packets when ingress throughput drops below a certain threshold [12] may help to prevent such attacks.

Is important to note that an attacker can combine previously explained attacks to increase their chance of guessing.

To decrease the probability of guess the path, different paths are retrieved using our path choosing method and one of them is chosen randomly.

3.3 Active adversaries

Active adversaries are those that performs actions against other nodes or modifies information that cross 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 [12]. There are several possible attacks to do against onion routing by malign node in the network [13], [14], [10].

An attacker who has a control of a node of the network can attacking non-observed nodes to shut them down. Prohibiting the communication between the source and the destination if that node was implied in involved in the chosen path. 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 in discussed in conclusions and future work section 5.

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.

4 Evaluation

In this section 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 the research usage [15]. We explain as well 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.

4.1 Scenario: 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 makes different routes around the UAB campus. Is important to note that every single bus makes the same route daily. By this way, this example can be seen as a good example of deterministic networks.

Each bus has a DTN node that allows to achieve secret communications as well as source anonymity using onion routing protocol. There are several applications that can take profit of such networks like anonymous reporting systems.

4.2 Mobility Model

We obtained the mobility model going through different stages. First, we exported the UAB Campus map from OpenStreetMaps into SUMO software [16], filtering some unnecessary items like buildings and railways with the Java OpenStreetMap editor tool [17].

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 [18]. In addition, we tuned some bus characteristics like acceleration and deceleration parameters in order to get coherent travel times.

Finally, we exported the model to a NS-2 mobility trace as is explained in [19]. 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.

4.3 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 beacons 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.

As can be seen in table 1, we also tuned different wireless parameters to be suitable with resource-constrained computers like those that could be used in each bus.

4.4 Simulation results

In figure 3 we have an overall view of the network activity. There is a group that have really short contact times, these contacts can be suitable for those applications that does not require a huge amount of data to send. There is another group that has long contact times (nearly 7 minutes), these group is able to perform complex communications sending higher amount of data. The average contact time is near 1 minute indeed is suitable for several applications like anonymous reporting systems. A summary of contacts related information during the simulation is shown in table 2. With this simple network characterization 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.

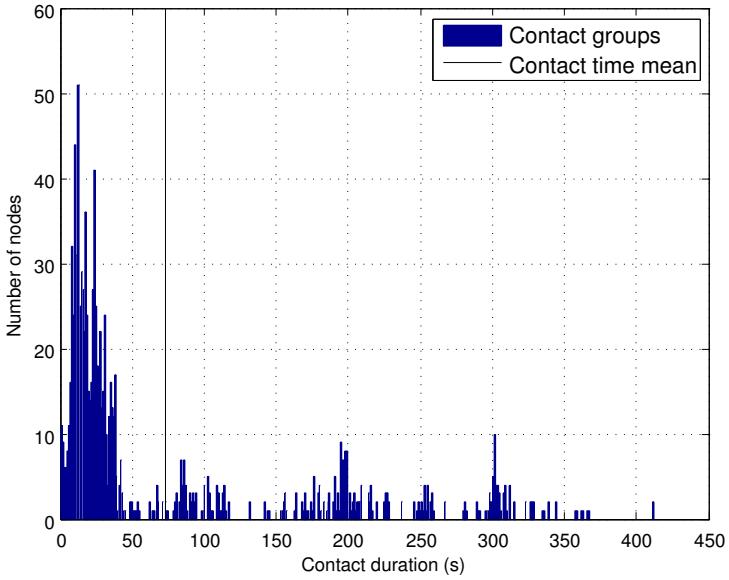


Figure 3: Number of contacts with the same duration.

Create a ratio between time and probability to guess the source. Normalize the value between 0 and 1. Do a 3-D graphic modifying k and n. Normalize like https://docs.tibco.com/pub/spotfire/5.5.0-march-2013/UsersGuide/norm/norm_scale_between_0_and_1.htm

5 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.

Think relevant conclusions!

In our future work, we will search and analyse efficient ways of path choosing, being able to use the algorithm in resource-constrained computers as our proposed method complexity is high as it explores every possible path. Another future branch of research can be decrease the impact of active attacks like black holes

using a reputation system. By this way the reputation value will be shared among all nodes in the network. The path choosing algorithm should be modified too to take this value into consideration in the choosing process. Finally this will lead into another security analysis because the 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 the enough information to decrease the number of failed contacts due to a bad contact prediction.

Acknowledgements Acknowledgements go here.

References

1. F. Warthman, "Delay tolerant networks (dtns) a tutorial, 2003," *Warthman Associates*.
2. D. Goldschlag, M. Reed, and P. Syverson, "Onion routing," *Communications of the ACM*, vol. 42, no. 2, pp. 39–41, 1999.
3. N. Bhutta, G. Ansa, E. Johnson, N. Ahmad, M. Alsiyabi, and H. Cruickshank, "Security analysis for delay/disruption tolerant satellite and sensor networks," 2009.
4. I. Cardei, C. Liu, J. Wu, and Q. Yuan, "Dtn routing with probabilistic trajectory prediction," in *Wireless Algorithms, Systems, and Applications*, pp. 40–51, Springer, 2008.
5. D. Hay and P. Giaccone, "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.
6. S. Farrell and V. Cahill, *Delay- and Disruption-tolerant Networking*. Artech House telecommunications library, Artech House, 2006.
7. S. Jain, K. Fall, and R. Patra, *Routing in a delay tolerant network*, vol. 34. ACM, 2004.
8. T. Hossmann, F. Legendre, and T. Spyropoulos, "From contacts to graphs: pitfalls in using complex network analysis for dtn routing," in *INFOCOM Workshops 2009, IEEE*, pp. 1–6, IEEE, 2009.
9. J. Gross, J. Yellen, and P. Zhang, *Handbook of Graph Theory, Second Edition*. Discrete Mathematics and Its Applications, Taylor & Francis, 2013.
10. N. Hopper, E. Y. Vasserman, and E. Chan-TIN, "How much anonymity does network latency leak?," *ACM Trans. Inf. Syst. Secur.*, vol. 13, pp. 13:1–13:28, Mar. 2010.
11. G. Vakde, R. Bibikar, Z. Le, and M. Wright, "Enpassant: anonymous routing for disruption-tolerant networks with applications in assistive environments," *Security and Communication Networks*, vol. 4, no. 11, pp. 1243–1256, 2011.
12. C. Shi, X. Luo, P. Traynor, M. H. Ammar, and E. W. Zegura, "Arden: Anonymous networking in delay tolerant networks," *Ad Hoc Networks*, vol. 10, no. 6, pp. 918 – 930, 2012.
13. N. S. Evans, R. Dingledine, and C. Grothoff, "A practical congestion attack on tor using long paths.," in *USENIX Security Symposium*, pp. 33–50, 2009.
14. N. Feamster and R. Dingledine, "Location diversity in anonymity networks," in *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pp. 66–76, ACM, 2004.

15. Nsnam, *NS-3 webpage*.
16. Institute of Transportation Systems, *SUMO - Simulation of Urban MObility*.
17. Open Street Map, *JOSM*.
18. UAB, *Horaris d'autobus 2015*.
19. C. Raj, U. Upadhayaya, T. Makwana, and P. Mahida, "Simulation of vanet using ns-3 and sumo," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, pp. 563–569, 2014.
20. D. Zhou, *DTN Bundle Protocol*.

All links were last followed on June 22, 2015.