Onion Routing in Deterministic 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@deic.uab.cat, guillermo.navarro@uab.cat

Abstract. Deterministic DTNs are networks where the behavior is known in advance or where a repetitive action occurs over time like in public transportation networks. This work proposes the application of an onion routing approach to deterministic DTNs to achieve anonymous communications. We show how the prior stage of path selection in onion routing can be achieved using the information provided by deterministic networks.

1 Introduction

In this paper we consider how to provide anonymity in Delay Tolerant Networks (DTN). DTNs [6] provide end-to-end communication in environments lacking continuous connectivity or presenting long delays. We focus our work in a class of DTNs called deterministic DTNs, which are such networks where the behavior is known in advance or where a repetitive action occurs over time. We will show that in such scenarios anonymous communications can be established using an onion routing [8] approach which takes into account the particularities of deterministic DTNs.

Deterministic DTNs [2, 9] 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 [6]. There are oracles that can answer any question regarding contacts between 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 can have full details of occurred contacts as well as the future ones.

We consider public transportation networks as deterministic because every node performs the same route periodically so this route can be known in advance. This information can be shared among all nodes in the network in order to let them be an *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.

Providing anonymous communications in DTNs is not easy and has not been deeply considered. It is said that onion routing is not applicable to DTNs because it needs to know the route in advance to construct the onion structure

of cryptographic layers for each node [1]. Nevertheless, this protocol can be applicable in deterministic DTNs, where source routing can be possible. Previous research work on the use of onion routing in DTNs is very scarce. One of the most relevant is [16], which uses attribute based encryption to per perform the layering process between groups of nodes. Our approach is simpler since we apply the onion routing directly as in conventional networks and does not require the formation of groups.

In Section 2 we introduce our proposal, simulations and provided in Section 3, and Section 4 discusses the proposal. Finally, Section 5 concludes the paper.

2 Onion routing for deterministic DTN

We describe our approach to provide anonymous communication in deterministic DTNs by using an onion routing approach. We first need to obtain a model of the network, and then provide means for the sender to establish the onion routing circuit (the path to the destination).

2.1 Deterministic DTN model from a public transportation network

We model the network as a dynamic graph $G = (V, E)$. The set of vertices V is the set of DTN 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 we associate two attributes to each edge, the instant of time when the connection opportunity occurs and how long this contact has been.

As an example of deterministic DTN we use a urban public transportation network. More precisely for illustration purposes we will focus in a small public bus network from a university campus. The public transportation network that operates inside the Autonomous University of Barcelona (UAB) campus is composed by 5 buses that make different routes around the campus. It is important to note that every single bus makes the same route daily. This scenario can be seen as a good example of deterministic networks, where each bus can be equipped with a DTN node with wireless connectivity.

To obtain the mobility model, first, we export the campus road map from OpenStreetMap into SUMO (Simulation of Urban MObility) [4], filtering some unnecessary items like buildings and railways with the JOSM (Java OpenStreetMap) editor tool [12].

Once the campus roads were imported in SUMO, we recreate the buses movements taking into consideration the official bus schedule of the UAB public transportation network. In addition, we can tune some bus characteristics like acceleration and deceleration parameters in order to get coherent travel times.

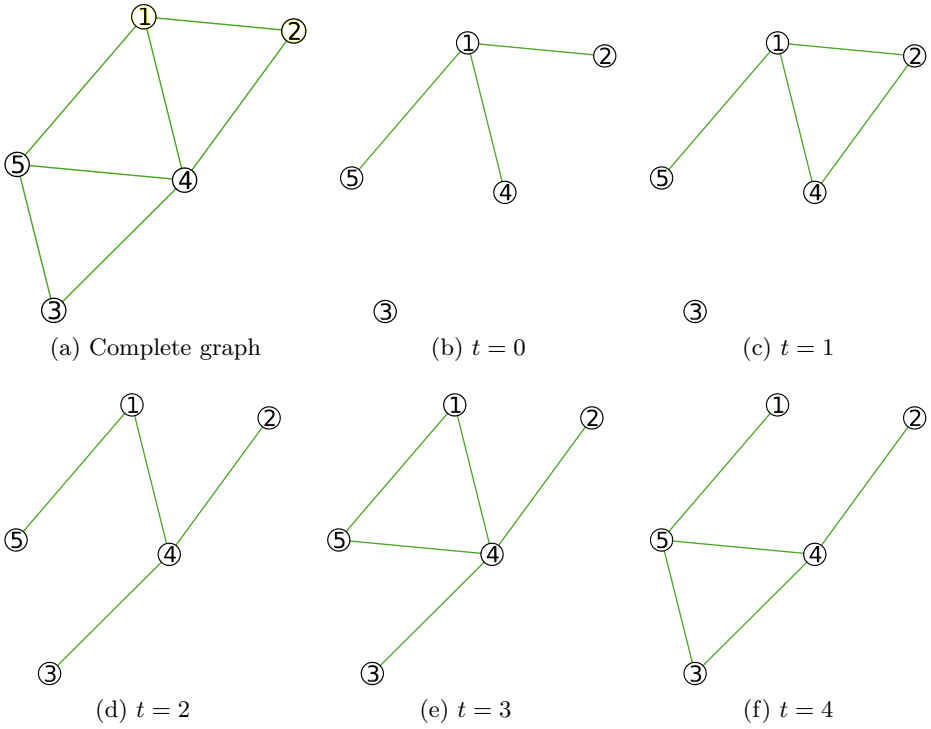
Finally, we convert the bus model to a NS-2 mobility trace [14], which can be used in NS-3. We used the NS-3 simulator to obtain the contact related data of the campus network, i.e: information about the duration of the contacts as well as the instant of time when they occurred. This contact information is used to build the dynamic graph that serves as the base model of the deterministic DTN.

2.2 Path selection

We have a source node s willing to send a message to a destination d at time t using onion routing. In onion routing we 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 third parties, as the network knowledge is shared among all nodes, the node s obtains a set of k paths in order to choose one of them randomly.

To sum up, we define a method *get_paths(s, d, t, n, k, tt)* that will retrieve a set of up to k paths given the t, n, and tt parameters.

The procedures shown in Algorithm 1 and Algorithm 2 are an example of deterministic selection of paths. Both of them inherit values to filter out unneeded paths. Specifically, in Algorithm 1, from a given *node* 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*.



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

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

Fig. 1: In (a) the accumulative graph $(t = [0, +\infty))$ can be seen. In $(b) - (f)$ different snapshots in given instants of time showing the evolution of the dynamic graph are depicted.

Figure 1 shows an example of contact data representation using a dynamic graph. 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 4 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 guessing the source node more 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 Simulations

As explained in Section 2.1, we use NS-3 [13] to model and simulate the public transportation deterministic DTN. The DTN modules are under current development in NS-3 [19], so 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. With this simulation environment we are able to test our proposed method (cf. Section 2) on the small campus transportation network.

As an example, in Figure 2, after fixing *s=1; d=5; t=0* and *k=10*, we consider the time required to route a message with the n value from *n=5* to *n=40*. We

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
```

computed the average delivery time 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 it is the common case 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.

4 Discussion

Unlike traditional onion routing, our approach could suffer from the predictability of the routing path. This is why the proposed method in Section 2.2 returns a set of k paths, and the final path is selected at random from the set. This lowers the probability for an attacker to guess the source of a communication by predicting the path. The exact probability will depend in the concrete scenario and setup. Besides this point, our approach presents the same security problems and benefits of traditional onion routing.

We can then consider threats imposed by passive adversaries (observe traffic patterns) and active adversaries (perform actions against other nodes or modify information in transit).

We assume that there is a prior secure key distribution procedure. The required network information as well as the cryptographic keys are distributed to the nodes at a previous stage, i.e: before starting the periodical routing. The public keys of the nodes are known to each other and used to establish each onion

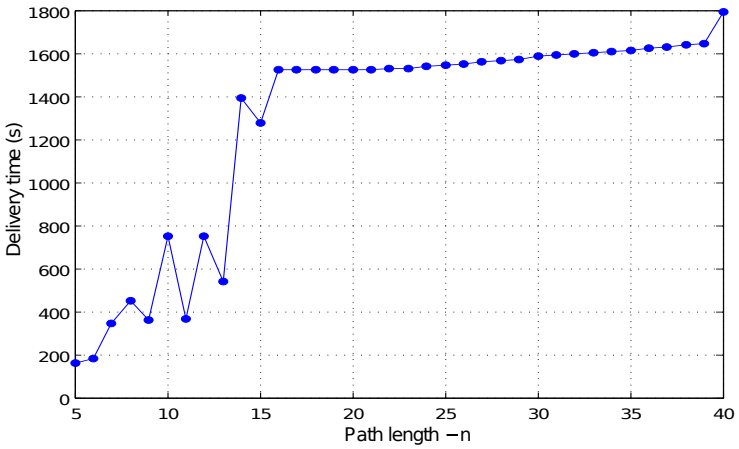


Fig. 2: Average delivery time considering the variation of the path length.

layer. Unlike current popular implementations of onion routing such as Tor [3] we cannot use Diffie-Hellman or similar interactive approaches to derive session keys in a DTN scenario. Symmetric session keys can be directly established or generated from a seed provided by each node to the successor node as in [8, 15]. At the moment we use directly public key encryption since most applications over DTN only need to send small quantities of data in one direction.

4.1 Passive adversaries

As explained in [10], if the attacker is the destination of the message, he can obtain information from the delay between messages. This kind of attack does not work when the sending start time, known as t in our path selection method, is not highly deterministic [18].

Another attacker model is a set of compromised nodes that work together to retrieve information breaking 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 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 research.

Another possible scenario where a set of compromised nodes work together is the source 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 source set, a globally passive adversary can improve guessing the source of a message. To solve this issue the creation of dummy packets when ingress throughput drops below a certain threshold may help to prevent such attacks [16].

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.

4.2 Active adversaries

As is the previous case, an attacker controlling a single node will be unable to extract the source of a message due to the use of multiple layers of encryption [16]. There are several possible attacks against onion routing by malicious nodes in the network like congestion attacks [5], location based attacks [7], and attacks based on latency analysis [10].

An attacker who has control of a node of the network can attack other nodes to shut them down leading to undelivered messages. This Denial of Service attacks can be addressed improving the robustness of the nodes.

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.

5 Conclusions

We have presented an onion routing based approach for deterministic DTNs. We also provide the means to easily model the concrete case of DTN networks deployed on urban public transportation systems. This enables to some extent establishing anonymous communications in these class of networks using a well known approach.

The presented idea is an initial step and will require further development. We will search and analyze efficient path selection mechanisms, as currently our method is non efficient because it explores every possible path. The efficiency comes crucial when resource-constrained computers are involved like tiny computers used in IoT, that could fit perfectly in each bus in our scenario, involving a very low deployment cost. Another future research branch can be to decrease the impact of active attacks like black holes using a reputation system. Path selection take this value into consideration. This is not an easy approach, since it can lead to security vulnerabilities by easing the path selection predictability. The more reputation a node has, the most probable it is for the node to be chosen nodes in the path.

We also note that contrary to more common mobile adhoc networks, deterministic DTNs are specially suited to support big delays and disruptions. This is interesting in our approach since it helps in providing a more stochastic path selection (a note can force a big delay to avoid path prediction through best path selection).

Acknowledgments

Support by projects MINECO project TIN2014–55243-P, and Catalan AGAUR 2014-SGR-691, is acknowledged.

References

1. Bhutta, N., Ansa, G., Johnson, E., Ahmad, N., Alsiyabi, M., Cruickshank, H.: Security analysis for delay/disruption tolerant satellite and sensor networks, International Workshop on Satellite and Space Communications, IWSSC 2009., 385–389, (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. Dingledine, R., Mathewson, N., Syverson, P., Tor: The Second-Generation Onion Router. In USENIX Security Symposium. USENIX Association, (2004)
4. DLR: SUMO – Simulation of Urban MObility, Insitute if Transportation Systems, Deutsches Zentrum für Luft- und Raumfahrt (DLR)
5. Evans, N.S., Dingledine, R., Grothoff, C.: A practical congestion attack on tor using long paths. In: USENIX Security Symposium, pp. 33–50 (2009)
6. Farrell, S., Cahill, V.: Delay- and Disruption-tolerant Networking. Artech House (2006)
7. 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)
8. Goldschlag, D., Reed, M., Syverson, P, Hiding Routing Information. In Proceedings of the First International Workshop on Information Hiding, 137-150. (1996)
9. 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)
10. 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 (2010)
11. Jain, S., Fall, K., Patra, R.: Routing in a delay tolerant network, vol. 34. ACM (2004)
12. OSM: JOSM, OpenStreetMap (OSM) `https://josm.openstreetmap.de`
13. Nsnam: NS-3 webpage, `https://www.nsnam.org`
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. Reed, M.G., Syverson, P.F., Goldschlag, D.M., Anonymous connections and onion routing, IEEE Journal on Selected Areas in Communications, 16(4), 482–494, (1998)
16. 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)
17. Tor Project, `https://www.torproject.org/`
18. 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)
19. Zhou, D.: DTN Bundle Protocol, `https://www.nsnam.org/wiki/Current_Development`