TECHNICAL UNIVERSITY OF CATALONIA

Department of Telematics Engineering

Master in Telematics Thesis

Analysis and simulation of a deterministic routing model in a delay tolerant city bus network

Author: Javier Gálvez Guerrero

Supervisor: Josep Paradells Aspas

Barcelona, February 2011

TABLE OF CONTENTS

TABLE OF FIGURES

I. INTRODUCTION

Classical routing considers that nodes in the path from source to destination will all be available and in range during the data transfer session. This approach is known as store-and-forward, so every node in the path receives and stores the bulk of data, performs the route computation and, once a result has been found, the packet is forwarded to the next node in the path. Mobile Ad-hoc NETworks (MANETs), where some or all nodes are supposed to be mobile and wireless, have always implemented routing algorithms that follow the store-and-forward paradigm. However, some scenarios may not be well suited for this routing model.

Links in any network are supposed not to be everlasting and permanent; that's why routing protocols implement different methods for route recovery, loop control and fast convergence of routing data. A new level of route discontinuity is defined with Delay- or Disruption- Tolerant Networks (DTNs), where links may not be available during a specific or estimated period of time, thus the data transferred through the path, must be borne by the node until the link to the next node in the obtained route is available. Such novel approach is called store-carry-and-forward.

Some scenarios where DTNs are expected to offer better results than previous networking and routing proposals are space communications, military and disaster recovery operations or smart cities. Applications deployed over such scenarios include those that do not require fast delivery (thus delay tolerant) nor high throughput, but do prioritize a reliable transmission. So, the target of these networks is not real-time applications but general data transfer services like sensor monitoring or file distribution.

DTNs do not only involve a new routing paradigm, but also a redefinition of the general network architecture and suited application framework. Increasing interest has appeared in the networking research community since the first related proposals a few years ago, including the creation of an IRTF research group: the Delay-Tolerant Networking group, which is committed with the architecture and protocols specification.

This report presents a general overview of the definition, architecture, routing and applications of Delay- and Disruption- Tolerant Networks and focuses on the deterministic routing model, which is considered to offer the best results. As a potential scenario, such model is applied in a city bus network where relaying nodes are build on stations and buses act as mules which carry data packets from node to node.

II. DELAY TOLERANT NETWORKS

1. Background

1.1. Typical networking scenario

Current networking technologies are implemented in commercial products that both public and private network deployments use. Hardware enhancements and protocol amendments have allowed deploying multiple solutions for different environments. However, such scenarios coarsely shared the same assumptions regarding basic network requirements and features. In order to provide with these requirements and according to the capabilities of devices, several protocols were standardized, although some of them spread massively in the Internet and private networking implementations.

Some protocols of such stack, namely TCP/IP for the most representative protocols, implement specific techniques that, despite getting really good results in the typical networking scenarios which we are used to, are not really well suited for new emerging proposals where the assumptions previously made do not apply. One of these proposals of new scenario is *Delay- and Disruption- Tolerant Networking* (DTN).

In a typical networking scenario, where common services like web browsing, e-commerce, or video streaming access through a *Content Distribution Network* (CDN) are deployed, the implemented protocols assume and are designed to keep and take advantage of low end-to-end delays and high bandwidth. However, while some protocols can continue providing some of its functionalities when facing unfavorable conditions, others simply fail or even produce unexpected behaviors, which are likely to negatively affect the network performance.

Transmission Control Protocol (TCP) is one of such protocols which cannot be used off-the-shelf in a DTN because of its chatty handshaking procedure and the slow start behavior, amongst other controversial features. When establishing a TCP session between two nodes, a *three-way handshake* exchange of messages is produced prior to start sending data. If you consider a short contact time, it could be possible that the handshake procedure consumed such amount of time that little or no data could be exchanged, so wasting the contact opportunity. Moreover, once the connection is established, TCP implements a *slow start* transmission mechanism in order to provide reliable transmission of data in a proactive way. Such mechanism starts with a small transmission window and increments it after receiving the corresponding acknowledgement. This is neither suitable for deployment in DTNs, given that it would probably be much more efficient start with a less conservative transmission window. It must also be noted that TCP implements a general timeout of 2 minutes, meaning this

that if a server node does not receive any message from the client node after 2 minutes since the last transmission, the connection is closed and the related resources freed. This value is high enough for a scenario where IEEE 802.3 or IEEE 802.11 technologies are used and low latencies in the order of tens or hundreds of milliseconds are expected. However, when considering deep-space networks, a typical example of DTN, such timeout threshold is clearly not enough. While considering a new value for such timeout is not possible, new approaches regarding data transport and transmission control must be envisaged. As a direct consequence of such issues, all protocols and applications that layer on top of TCP would not work properly, thus including *Hypertext Transfer Protocol* (HTTP) (web browsing) for example.

On the other hand, despite delay and disruption were not considered when designed, *Internet Protocol* (IP) and *User Datagram Protocol* (UDP), in contrast to TCP which is a connection-oriented protocol, seem to require less or no amendments in order to be usable in DTNs. However, an upper layer in order to manage connectivity intermittency would be required to be added to the stack.

Additionally, given that links in DTN are constantly changing, new approaches must be envisaged in order to overcome the related routing issues, including connectivity disruption and variable delay. The obvious problem is that classic routing schemes assume that all links in computed paths from source to destination will be available during the data transmission once it has started. In DTN, such assumption is invalid, given that link availability is essentially time-dependent, so computing a path with current topology data and ignoring future link changes may produce longer delays or even packets to be drop. New routing algorithms for such scenarios remain as one of the most interesting research areas in DTNs, where several proposals have been emerging since the DTN concept arose.

1.2. Causes of delay and disruption

There are many possible causes for delay and disruption, but many of them are clearly scenario-dependent. All these causes, which will be depicted next, are the reasons why the classical networking model and protocols are not suitable for being deployed in DTN scenarios, as explained in the previous section.

Deep Space Networks (DSNs), the envisioned main scenario for DTNs, were a useful source for the definition of problems with disruption of connectivity. Thus speed of light, positioning accuracy and visibility were the main issues to be considered and which motivated the emergence of the concept of disruption tolerance in networking. Network relays in such scenarios, including spacecrafts and orbiters, move at *high speed*, but more a more important issue is that data cannot travel at a higher velocity than the

speed of light. This fact makes not possible using TCP as the transport protocol for such networks, so the 2 minutes timeout would be easily exceeded while the packet is still on its way to destination. As nodes move really fast, *position accuracy* requires estimation of future movements in space. This should not be a problem as spacecraft and orbiter's routes are scheduled ahead of time so any position is known in advance. However, some variations could take place during the flight and, given than data is sent with an important degree of anticipation, it could be possible that packets never reached their destination.

Also related with DSNs, but being a more general cause of disruption in connectivity, is the *visibility*, what produces intermittent links when obstacles appear in the middle of the path between source and destination. In a space scenario such events can emerge due to eclipses, while in an urban environment vehicles can break the line of sight between the two nodes while they are exchanging data.

In terrestrial applications, mobility patterns also introduce a high intermittency in link availability. So, depending on the scenario where the service is deployed, say using animals, vehicles, or personal mobile devices as network relay carriers, network topology can have a different change ratio, thus introducing connectivity disruption and, in the end, notably increasing the delay of the transmission.

Routing information data must be updated when a path to destination is required, so if nodes follow high mobility patterns and obstacles are constantly appearing in the line of sight, reachability and cost estimation for the routing metric of interest must be constantly evaluated and disseminated to all network nodes following the implemented routing algorithm. Current proactive routing protocols are not well suited for these scenarios as they do not consider the time dependency of routes. On the other hand, reactive approaches do not help in reducing the delay in data delivery, but increments it on purpose. Later, different proposals for routing in DTNs will be discussed.

In sensor monitoring deployments, another scenario where DTNs apply, nodes must be very power conservative in order to increase their lifetime and reduce the required non-remote maintenance (basically, battery replacement). In applications where nodes lack of this conservative behavior disruption events can appear if sensors run out of energy. The common mechanism to avoid such problem is to design specific *duty cycles*, so nodes periodically wake up the radio communication interfaces (which consumes a high percentage of the required energy resources, especially when transmitting) and searches for neighbor nodes which to transmit to in case it is expected. Obviously, such wake up mechanism does not help in guaranteeing the delivery of data, as nodes might be in range but the radio interface is off and when it wakes up, the neighbor has

gone away. As a result, this behavior introduces disruption, which once again leads to increasing the delivery delay, due to not having a proper opportunistic contact management. Such management could perform wake up procedures when the nodes are in range and save energy resources setting the network interfaces in sleep mode when not. RFID based and other wake up techniques are currently being studied and tested in real scenarios in order to check if they suit the energy saving requirements.

2. Applications

DTNs may be useful basically in applications where transmission latencies are high and connectivity disruption is likely to happen. Some of such scenarios, typically related to wireless environments, have been previously addressed by other network model proposals, such as *Mobile Ad-hoc NETworks* (MANETs). However, these approaches do not consider the basics of DTNs, these being delay and end-to-end disruption.

Several scenarios where such networks are expected to provide nice results and, actually, some are, are discussed in the next section. One important thing to note is that, although no killer application has been defined [1], several services using DTN deployments have been envisaged. As DTN research is actually in its infancy, new proposals are expected to arise as the momentum grows and the potential of such architecture features become widely known.

2.1. Wireless sensor networks

Wireless Sensor Networks (WSNs) are a classical example of network architecture where applications can clearly benefit from the delay tolerant approach. Such networks are deployed in order to, for example, monitor pollution, temperature, humidity and other parameters that may be of vital importance for specific studies or other services. Furthermore, the applicability of a set of nodes can be expanded to a wide number of services, including smart cities, earthquake and forest fire monitoring, amongst others.

All these applications share a common behavior: the spread nodes get data from the environment, thus sense it, and send it to an operational base where such data can be usefully managed. In the case of the environment monitoring application, it can be useful to deploy a WSN to track some parameters that can be important to ensure proper environment preservation. Lakes, rivers, forests or natural parks can benefit of such monitoring services. Furthermore, specific variations of such platforms could provide a more specific application, such as forest fire control support, earthquake

prevision or agriculture activities monitoring, including tracking of the quality of the land for example.

Smart cities are an emerging research field with many different proposals in the latest years. Several services have been envisioned and new ones are constantly appearing, but the ideal network architecture to be used has not been envisioned yet. However, DTNs seem to have attracted much of the interest regarding such applications. Giving intelligence to a city means using information technologies in order to increase the quality of life of citizens in a sustainable and efficient way. Some proposed services include providing free information concerning city services such as transport scheduling, traffic control, street lighting management and metering, amongst others.

Currently, when a streetlamp blows out, the area may remain without light some days or even weeks depending on the importance of the street and the amount of complaints the city council receives. Then, until it is replaced and the lightening system is recovered at this location, citizens may feel insecure or have problems for mobility. A sensor network able to monitor the light intensity or the electric consumption could warn the people in charge of the management and maintenance operations of the lightening system, thus allowing for a faster reaction to such issues. Metering is another application where WSNs in a smart city can help providing a low-cost and easy to deploy solution. Instead of requiring some personnel to walk around the city, door by door, checking the electric or water consumption, or asking consumers to make a call to update de value of the electricity meter, little sensor devices could be deployed to get such values and send them to the centralized system through a suitable network.

Sensors in the previous two scenarios are supposed to be stationary as they are committed to monitor a specific signal from a non-moving source. However, transmitting the gathered data to the corresponding server offers different alternatives. While using the wired telecommunications infrastructure or the cellular radio communications platforms could be admissible approaches, a DTN clearly introduces a more cost-effective solution for such delay tolerant services, given they do not require an instant reaction. Thus, vehicles from the different municipal services, including buses, squad cars or garbage collecting trucks, could perform as data mules in the DTN, then retrieving data from the spread sensors as they follow their ordinary service routes all around the city and get close enough to them to establish a communication session. Once the data collector has collected the information it can go to any data sink where it can be directly transmitted to the server or any other node in charge of managing all the gathered data. So, specific actions can be scheduled if required after the reports analysis.

Another interesting scenario where recent deployments have been performed is in zoological applications. The Zebranet project [2] aims at studying the natural behavior of zebras in Kenya, thus analyzing the feeding, moving and social interaction habits. Thus, the leader of each group of zebras had a collar with a sensor attached which logged their position acquired through a GPS module. Such information is spread all around the natural park in an epidemic way, thus such data is exchanged between the different zebra groups. Scientists, in order to get the information, only have to get closer to a zebra group and the data is retrieved for further analysis. This application presents a scenario where nodes must be more power-conservative than previous ones as they are wireless stand-alone devices. Moreover, it requires a routing scheme in order to spread the information and increase the probabilities of the scientists getting updated data.

As it has been seen with the previous examples of applications, DTNs are really well suited for numerous services where sensor data must be gathered and end-to-end connectivity cannot be guaranteed.

2.2. Deep space and undersea acoustic networking

Being the first scenario envisioned by people involved in the DTN research, *Deep Space Networking* (DSN) became a really interesting field, although somewhat futuristic and hard to prove with current technologies and resources. In such deep space missions, spacecrafts, orbiters, landers and rovers are the involved elements in charge of the different phases. Any data recollected by rovers, once the planet or moon has been explored, is logged and brought to the Earth at the end of the mission, where it can be properly studied. However, a group of engineers of the *Jet Propulsion Laboratory* (JPL) of the NASA [3], proposed a networking architecture which would allow having a somehow real-time transport of data and remote module management services through the space, considering delays caused by speed of light limitations and potential path obstruction. Initially, this concept was called *Interplanetary Internet* (IPN), although it ended up being approved by consensus among the different involved organisms that it should be called *Delay and Disruption Tolerant Networks*, according to the different scenarios proposed by NASA engineers, IRTF network researchers and DARPA scientists.

The depicted scenario does not only allow receiving data from the simultaneously deployed rovers at the surface of any planet or moon, but also performing remote management tasks. Space trips are scheduled in advance and any deviation on, for example, the spacecraft route must be predicted and any proper reaction to such possible issue must be implemented ahead of time. If near real-time communication

with the spacecraft would be possible, more precise decisions could be performed in case the scheduled behavior failed at getting good results. Thus, a DTN where mission elements can act as network nodes is expected to provide with the previous advantages over current solutions.

A scenario where similar transmission delays issues arise is *Undersea Acoustic Networking*, where radio-frequency (RF) communications generally do not work and sonar perform low data rates when used to carry data signals. However, the same physical layer issues as in RF communications arise, including fading, interference and multipath reception, while signal propagation is limited by the speed of sound, as the previous DSN scenario must face light-trip times. Such networking model presents a great opportunity for deploying a wide range of applications, like submarine communication and traffic control, cable monitoring or detecting fish movements. The *Seaweb* project [4], deployed by *United States Navy*, has performed some tests where autonomous underwater vehicles provide network connectivity using repeater nodes and its own *medium access control* (MAC) layer protocol.

2.3. New technologies in the developing world

WSNs and DSNs are two different scenarios where DTNs can provide with a network architecture suitable to application requirements in an environment that is already connected, in the former case, or there is not an immediate need for deployment, as in the later. However, there are some countries or locations where the technology has not arrived yet, mainly because of the high costs of network deployment. Some examples can be rural villages far away from metropolitan areas or developing countries where there is little opportunity to get Internet access.

Actually, one of the classical examples of use cases for DTNs is the mail delivering courier service [5]. Such example depicts a scenario where every computer in a rural area with no connectivity at all locally stores the mails written and ready to be sent. As the courier passes through the town, it retrieves all mails from each computer. Then it goes back to the city where a sink node, which is connected to the Internet, performs the corresponding forwarding tasks once it receives the mails from the courier. At the same time, the courier gets all mails which destination is a computer in the town, thus restarting its route and delivering the mails to the corresponding nodes. It is a clear example of delay tolerant network as mails cannot be forwarded until the courier does its round in the town.

3. Network architecture

3.1. Overview

In the classical Internet architecture, some assumptions are taken for granted:

- An end-to-end path between source and destination exists for the duration of a communication session.
- Transmission errors can be effectively repaired through retransmissions based on timely and stable feedback from data receivers.
- End-to-end loss is relatively low.
- TCP/IP protocols are supported through the entire path.
- Applications need not worry about communications performance.
- Endpoint-based security mechanisms are sufficient.
- Packet switching is the most appropriate abstraction for interoperability and performance.
- A single route between sender and receiver is sufficient for achieving acceptable communication performance.

Many of such assumptions do not apply to Delay Tolerant Networking, so a new proposal for an architecture which suits the features and requirements of this new approach has emerged and been revised during the last years. This proposal considers:

- Use variable-length messages instead of streams or limited-size packets as the communication abstraction.
- Naming syntax that supports a wide range of naming and addressing conventions to enhance interoperability.
- Do not require end-to-end reliability, use storage within the network to support store-and-forward operation over multiple paths and over potentially long timescales.
- Provide security mechanisms that protect the infrastructure form unauthorized use.
- Provide coarse-grained classes of service, delivery option, and a way to express the useful lifetime of data.

- Applications should minimize the number of round-trip exchanges, cope with restarts after failure and inform the network about of the useful life and importance of data.

Which layer delay and disruption tolerance should be faced in has motivated an intense debate during the last years in the community. Different approaches have been studied, considering mainly link layer, transport and application layer proposals. However, three alternatives to the architecture, the transport protocol and routing algorithm in such scenarios are seen as the most well-suited ones; actually, they have been proposed for future standardization as they are currently experimental RFCs.

First, the Bundle Protocol [6, 7] presents an end-to-end transport protocol while defining the general architecture of a DTN. The Licklider Transport Protocol [8] is another solution for the data transference between nodes but in a single link scenario. Details in both proposals make these protocols proper approaches for the applications expected to run in DTNs deployments. Finally, the PRoPHET [9] routing protocol introduces the new paradigm of store-carry-and-forward into probability-based routing algorithms, which seems the best solution for DTNs in terms of flexibility and performance.

As said before, although there are different approaches for architecture, transport and, specially, routing in DTNs, the previous three proposals have the support of the IRTF group behind the DTNs research, the DTNRG and they are constantly being revised in order to become as suitable and complete as possible. Next, the three of these proposals are discussed. Additionally, further details of other routing approaches will be given as it comprises one of the most interesting challenges in DTNs and is the core issue of this report.

3.2. Bundle protocol

The Bundle Protocol presents the overlay approach for DTN networking, which is the mainstream proposal (Figure 1). It is described in two different documents, one focused on the overlay network architecture for DTNs [6] and the other on the bundle protocol specification itself [7] Moreover, there are additional documents defining security extensions for the bundle protocol. In this section different concepts defined in these specifications are commented in order to have a general idea about how it works. However, it must be noted that most of these definitions are currently under discussion and can vary in next revisions.

Every device in a DTN network is called a DTN node, which runs an instance of the bundle protocol. Such nodes are identified with endpoint identifiers (EIDs), which are syntactically uniform resource identifiers (URIs) using the “dtn:” scheme. Each of these EIDs can refer to one or more DTN nodes, depending of the transmission system, being unicast or multicast respectively. Such identifiers are considered to be mapped to lower layer addresses in order to identify them in these layers and perform the required routing and forwarding procedures. Such mapping should be resolved as close to the final destination as possible, which is known as late-binding. The protocol also defines the extended concept of contact, which refers to the time that two nodes have the opportunity to establish a connection and exchange data in, at least, one direction. Related to this contact, two new concepts are introduced, namely bundle expiry and custody. The bundle expiry refers to the time that a bundle may exist in the network before being discarded in case it is not delivered to its ultimate destination. This value, along with the creation time, is set by the packet creator. Synchronized clocks would help in the task of managing such bundle aging but this broad synchronization cannot be assumed in any scenario. The main proposal of the bundle protocol to provide end-to-end reliability to some extent is to give some custody responsibilities to specific nodes in the path from source to destination. Thus, nodes with proper capabilities

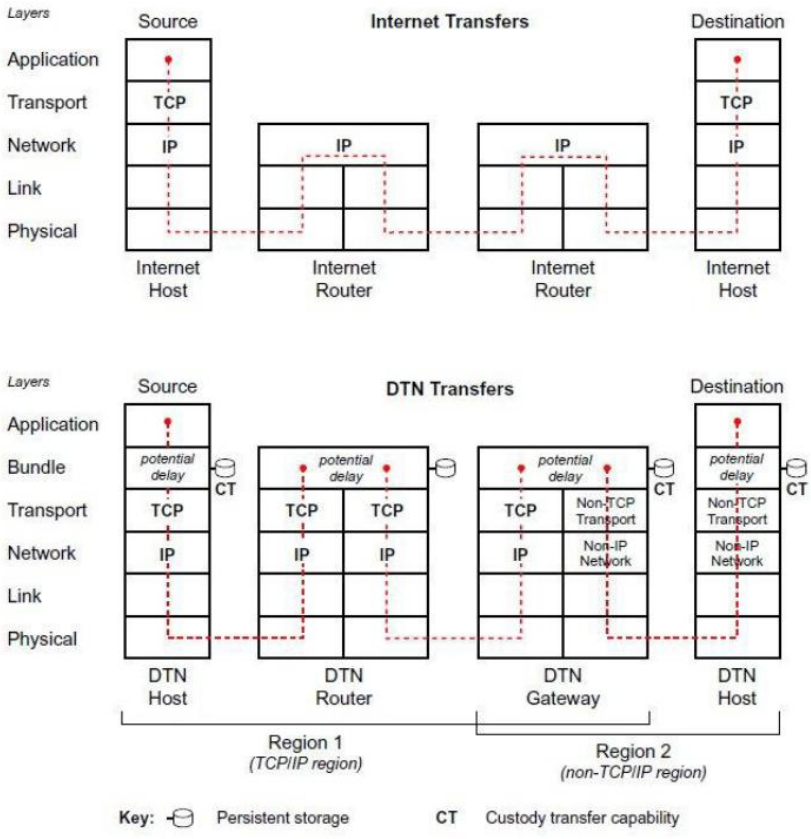


Figure 1: Transfers with the bundle protocol

(extended lifetime, high storage capabilities, etc) are better suited for being bundle custodians. The tasks they must commit include acknowledging bundle reception, storing bundles until receiving an acknowledgement of the next custodian and retransmitting when necessary. It must be noted that custodians are assigned on a bundle basis, which means that the custodians assigned to different bundles may be also different. This is a first approach to handle DTN network congestion, thus allowing nodes to discard bundles which is not custodian of in order to free resources to handle bundles which custody is assigned. However, more sophisticated and proper congestion mechanisms are being studied to be included in future revisions. Reliability can be reinforced through the use of classes of service (COS), currently specified as “bulk”, “normal” and “expedited”, intended to prioritize some traffic over others where a coarse QoS mechanism is required.

All the related signaling to control this and other features are sent using new bundles with no answer requirement in order to avoid many round trip times prior to the data exchange. However, piggybacking is being considered to be a new amendment in future protocol revisions according to recent specification extensions that make possible to add signaling to data bundles.

Another issue regarding congestion avoidance is fragmentation. Two different approaches are devised in the bundle protocol specification. First, proactive fragmentation should be performed when the transfer of the whole bundle is not guaranteed to succeed, thus smaller chunks of data are sent in different contact opportunities. On the other side, reactive fragmentation reacts to lower layer indications that bundles have not been successfully transferred, meaning this that, first, the original whole bundle has been sent but failed; then the sender node reacts splitting it into a proper number of chunks and send them independently. In both proactive and reactive fragmentation, reassembly is done at the destination node, so each fragment can be routed straight to the end node.

The bundle structure defines the numerous fields included in the primary and extension headers. Specifications for such fields can be found in the commented RFC documents. However, there is a general feature of the bundle structure that is worth mentioning here, namely the *self-delimiting numeric value* (SDNV) type used to encode fields in such packets. This is a simple, flexible an efficient method of encoding non-negative numeric values which allows avoiding limiting restrictions. This method is based on the *basic encoding rules* (BER) of ASN.1, which encodes a positive integer value using 7 value bits per octet and with the most significant bit of each octet being set, except for the last octet, which has the most significant bit unset to indicate the end of the SDNV, thus the end of the field. Another important feature of the bundle structure

is the *dictionary* field of the primary header, which consists of a list of all EIDs required by the bundle in terms of source, destination, custodian and report-to values. In order to point to such values of the dictionary, offsets are used in the corresponding fields. So, in case that source, custodian and report-to EIDs were all the same, instead of including the EID three times, an offset would be used to point to the proper entry in the dictionary, thus saving bits.

Bundle primary header also includes fields used to manage bundle fragmentation, custody transfer and COS, which are specified in the corresponding documents. Extension headers requirements are under discussion, while the only specification agreed considers an extension header for including security parameters.

Security issues in DTNs, according to the bundle protocol security extension specifications include the threats of classical wireless networking, such as resource consumption, denial of service (DoS), confidentiality and integrity, traffic storms, etc. For example, an unauthorized entity can insert fake bundles into the network or deliberately generate and send non-requested reports. Other common threats include forging a bundle's source, change the intended destination, or replaying bundles, thus affecting the confidentiality and integrity of the DTN deployment. The implementation of mechanisms to protect data and nodes against these issues considers the limitations of DTNs and the specific requirements, like the lack of a practical way of deploying a public key infrastructure (PKI) due to restricted node capabilities or unavailability of an authorization server.

In order to face the previous issues the bundle security specification defines three header types: the bundle authentication header (BAH) that is used for data integrity in a hop-by-hop basis, the payload security header (PSH) that is used for data integrity in an end-to-end basis, and the confidentiality header (CH) used for data confidentiality. The specification defines a different ciphersuite for being used with each of these headers. For BAH a shared secret Message Authentication Coding (MAC) scheme using the HMAC-SHA1 algorithm is proposed, while for the PSH RSA digital signatures and the SHA-256 hash algorithm is assigned. CH uses an RSA algorithm for key transport and the AES algorithm for bulk data and symmetric key encryption. Obviously, the different security mechanisms might be combined in order to provide stronger security protection depending on the application. However, such specifications and considerations are not supposed to be definitive, so, until final standardization, new proposals may emerge and schemes be proposed.

A last consideration about the bundle protocol should be noted here, which is the so called convergence layer. This term is used in the bundle protocol specification to

denote the layer just below the bundle layer, which can be TCP, UDP, SCTP or LTP, described in the next section. Again, combination with such protocols is established according to application and scenario requirements.

3.3. Licklider Transmission Protocol

While the bundle protocol is designed to manage data transmission in an end-to-end basis, the *Licklider Transmission Protocol* (LTP) [8] handles delay tolerance and disconnection in a point-to-point link. Actually, it is the specific proposal for bundle protocol's convergence layer by the DTNRG, while UDP is also devised as a candidate upper layer. Its canonical use is a single hop deep space link, where long delays are expected, given the original motivation for DTNs, although it is supposed to be also useful in terrestrial applications. This protocol is described in three different documents, which discuss motivation, protocol base specification and extensions.

Being a point-to-point protocol for DTNs, there is no negotiation between nodes like in TCP because of the many problems stated in previous sections. In fact, parameters required for the LTP session must be agreed before contacts occur, thus being a stateful protocol. As a result, transmission opportunities are scheduled according to previous and upcoming events. There is no message exchanging due to the opportunistic feature of contacts, which is managed by timeouts that consider *punctuated time* instead of typical elapsed time, meaning this that it computes the continually-being-interrupted duration of the scheduled contact.

Partial reliability is provided through the *segmentation* of upper layer data blocks into smaller chunks by the LTP. At the same time, the upper layer can specify the size of the initial bytes in the data block that must be prioritized and, thus, may be sent reliably. Segments from this first part of the original block are tagged as *red segments*, while the remaining bytes till the end are tagged as *green segments*. Red segments must be retransmitted in case of transference error or timeout exceeded when waiting for a report from the receiving node. Green segments only must be acknowledged through a report so timeouts can be reset. If any green segment is lost it cannot be retransmitted. In case of receiving many errors or timeouts have occurred, cancellation segments are sent in order to stop the session.

Further details of this protocol can be found on the specific documents already mentioned.

3.4. Routing

There are different DTN routing protocols classifications [10, 11]. However, most of them usually divide the different proposals in flooding/epidemic-based, probabilistic or history-based and knowledge-based.

If future topology network and/or the mobility patterns of nodes are deterministic and known, the transmission can be scheduled ahead of time. This case represents the optimal routing scenario, the knowledge-based one. If nodes know nothing about the network topology evolution in time they can a) randomly forward packets to their neighbors (epidemic-based) or b) estimate the forwarding probability of its neighbors, thus enhancing the routing decision (history-based). Minority approaches are those where the movement of nodes can be remotely controlled, or coding-based routing protocols, where intermediate nodes can combine some of the bundles received so far and send them out as a new bundle.

Knowledge-based routing protocols are not suitable in dynamic scenarios, given that distributing topology structure information in a timely manner can be difficult or even impossible. On the other hand, although the epidemic approach is supposed to provide the best results in terms of delivery rate and delivery delay when the future topology changes are unknown, it must be noted that real scenarios introduce important tradeoffs related with energy efficiency and storage requirements. Thus, probability- or history-based routing protocols are the obvious alternative to such cases.

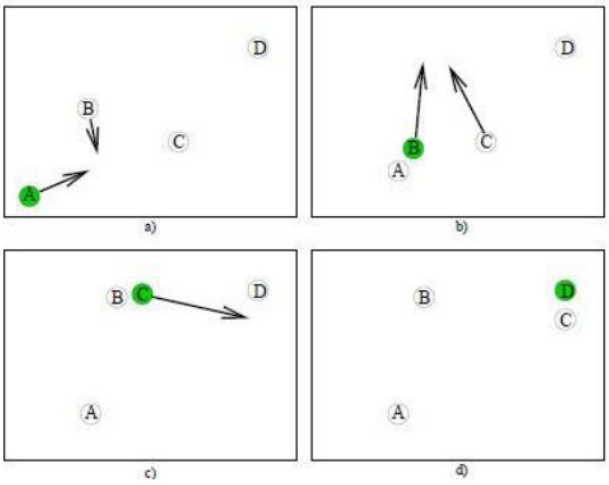


Figure 1: Store-carry-and-forward model

3.4.1. Knowledge-based routing

Classic routing algorithms have always been based on knowledge of costs associated to links, being these somehow distributed over all the nodes of the network or only in a

neighborhood base. This approach is the obvious one for static networks, where nodes are not expected to move to different places and costs have a null or low variability.

In DTNs, this knowledge concept is extended considering the time variance property of costs. Thus, links can be available or not depending on the instant of time and the mobility patterns, as well as the costs associated to such links might be also time-variable. The main difference between routing in ad hoc wireless networks and DTNs is that the information that the algorithm uses to compute the route in the former case is assumed to be effective in the moment of computation, while in the later the knowledge concerns future topology changes and links availability.

Deterministic routing [12] represents the best case, as the algorithm has all the information about the links availability and the computed route will be the optimal solution. However, the requirements can be really excessive, including the storage capacity necessary to host the information regarding all nodes availability and the processing load required to keep this information updated. Moreover, this routing model, generally used in source routing mode, assumes that all links in the path will be available as the data transference session remains active. In case any of the links would not be available as expected, data delivery could be even more delayed or just lost.

3.4.2. Replication-based routing

The replication-based schemes are based in an epidemic distribution of data. A node that must forward a packet just forwards a copy of such packet to each other link different from the one which it received the data from. Eventually, when all nodes in the network have been flooded with the data packets, they are supposed to have been received by destination, so the delivery is guaranteed. However, this model imposes high requirements regarding data buffering (all nodes will receive all sent packets, at least one copy) and energy resources (continuous network interface activity upon reception and forwarding of packets).

In order to reduce such requirements, different proposals try to perform efficient bundle replication instead of issuing a simple flooding distribution scheme. Two examples are discussed next.

3.4.2.1. Spray and Wait

Spray and Wait is a replication-based routing protocol that uses a controlled epidemic behavior, thus leveraging the typical resource constraints of DTN nodes. According to [13] *Spray and Wait* is highly scalable and, under low load, results in fewer

transmissions and comparable or smaller delays that standard flooding schemes, while under high load these improvements are even more remarkable. Moreover, it is claimed to obtain comparable delays to knowledge-based models when using only a handful of copies per message. However, these results were extracted from a simulation environment where the used mobility pattern was the *random waypoint mobility* (RWM) model, which does not fit real scenarios. Anyway, given that the vast majority of proposals of DTN routing protocols have not been deployed yet in real scenarios, we will keep in mind the general approach of them.

Spray and Wait consists of two phases, namely *spray* and *wait*. During the former, L copies of the original message are initially spread by the source and possibly other nodes receiving a copy. In the *wait* phase, if the destination is not found in the *spray* phase, each of the L nodes carrying a message copy will perform direct transmission to the destination node once it is found. Note that, while taking about *copies* of the data message, it refers to only *one* copy and the right to spread it a number of times.

This protocol is quite simple; however, the number of copies of the original message (L) and the spraying algorithm remain unspecified for the sake of scenario adequacy. The designers of the protocol devised two basic approaches. In the *Source Spray and Wait*, the simplest model, the source node forward all L copies to the first L distinct nodes it encounters. In the other alternative, the *Binary Spray and Wait*, any node that has $n > 1$ copies of the message and encounters another node with no copies, hands over it n/2 and keeps n/2 for itself until only 1 copy is left, which is kept for direct transmission to the destination node. This last mode has the minimum expected delay among all *Spray and Wait* algorithms when all nodes move in an IID (*Independent and Identically Distributed*) manner.

As said before, this protocol has not been tested in real scenarios or simulated with realistic mobility patterns, thus not providing reliable results when thinking of a real implementation. The main disadvantage of this approach appears when taking this issue into account, as the packet could be retained in a specific network cluster if the selected number of message copies does not allow for wider spreading in order to reach further destinations. On the other side, while this protocol was designed with constrained requirements of storage capacity in mind, it would be possible to choose a high number of packet copies in order to cope with the previous drawback, but then it would behave like an ordinary epidemic scheme, thus losing its *raison d'être*.

3.4.2.2. RAPID

In [14], *Resource Allocation Protocol for Intentional DTN* (RAPID) is shown as a routing protocol designed to explicitly optimize an administrator-specified routing

metric. Such optimizations are defined as *minimize the average delay*, *minimize missed deadlines* and *minimize maximum delay*. Packet replication is performed taking into account how useful the new copy becomes to such metrics. As a result of this prioritized replication, the designers claim to take into account the bandwidth occupancy, thus avoiding unnecessary data replication.

The RAPID protocol is divided in three components: a *selection* algorithm, which determines which packets to replicate at a transfer opportunity given their utilities; an *inference* algorithm, in charge of estimating the utility of a packet given the routing metric; and the *control channel*, which propagates the necessary metadata required by the inference algorithm.

In order to measure the delay, what is required for the three metrics considered by the protocol designers, they use two different models for estimating the contact probability. The first one aims at considering every node movement independently, which they recognize as not being realistic but simple, which helps in protocol simulation. The second model performs the delivery delay estimation according to previous encounters information, thus becoming a probabilistic-based routing scheme. Such information includes average time of past transfer opportunities and expected meeting times with nodes, and is transferred through the control channel when two DTN nodes become in contact, along with acknowledgements for delivered packets.

In fact, this protocol could have been conceived as an example of probabilistic-based routing, discussed in the next section, but as it was originally thought as an enhancement of classic flooding schemes and the stochastic information treatment is just an implementation option, it is considered a replication-based routing example. However, it introduces a simple but interesting queue prioritization scheme based on the replication usability, although the concept of resource saving through such combination is just left as an obvious effect the algorithm instead of one of its main goals, so no thorough analysis is provided.

3.4.3. Probability-based routing

While the deterministic model provides with low delivery delay and the flooding scheme allows for reliable delivery, both introduce high resource consumption requirements regarding route processing, network interface activity and storage capacity. Despite these issues, they can be proper methods in specific DTN scenarios.

However, a more general approach with lower requirements would better suit the vast majority of scenarios where DTNs could be deployed. In probabilistic routing, algorithms perform their decisions according to previous node contacts, thus

considering that two nodes that have been in contact many times in the past will also have new contact opportunities in the future. As a result, the data forwarding is prioritized for these links that have a higher probability of appearance. Additionally, many schemes can be implemented according to the previous situation, with different forwarding strategies and queuing policies.

Probability-based models can be single-copy (forwarding) or multiple-copy (replication) based, depending on how many copies of the original packet are spread in the network [REF]. While the former helps saving resources, the multiple-copy increases the delivery probability and reduces the delay. In the end, such approach is just a replication model where complex forwarding and queuing techniques are introduce for the sake of resource saving and performance enhancement.

3.4.3.1. MaxProp

The authors of [15] introduce a probabilistic routing protocol which aims at appropriately managing the buffering queue of each node in the DTN, selecting the next packet to send wisely and using a network-wide acknowledgement system. Such protocol, called MaxProp, assumes that each node has an unlimited buffer for local messages (messages that the own node generates) while having a fixed-size buffer for packets from other nodes in the network. As a result, data packets will be removed from nodes' buffers when a) they time out, b) an acknowledgement for such packet is received, or c) the buffer is already full and the queuing policy prioritizes other packets. These features depict a model more suitable for realistic scenarios.

The queue management algorithm considers parameters like the delivery likelihood and the packet hop count. As depicted in Figure 2, a hop threshold splits the buffer in two, giving priority to newer packets, thus having a lower hop count value. Newer packets are sorted by hop count, while packets in the second half of the buffer are sorted by delivery likelihood. As a result, older packets with low delivery probability will be the first to be dropped when necessary.

The so-called *delivery likelihood* is computed by nodes when receiving neighbor information regarding past encounters, so considering contact probability since the more encounters have taken place in the past, the higher probability of a new contact

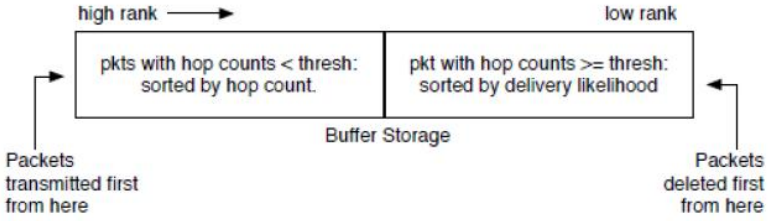


Figure 2: MaxProp queue priorization

opportunity is supposed to appear.

3.4.3.2. PRoPHET

Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET) [9, 16] is another routing protocol which decisions are based on stochastic data gathered from the past contact events of the nodes in the DTN. This protocol is one of the oldest proposals in this field and has been recognized by the IRTF as a candidate to be standardized, since it is an experimental Internet-Draft submitted by the DTN Research Group. Continuous revisions have been added to this proposal, which make of this protocol one of the most important.

The vast majority of proposals of routing in DTN, especially those being replication- or probability-based include in the corresponding papers a comparison with the basic PRoPHET algorithm. Many of such proposals are just a protocol with the same foundations of PRoPHET but with a new amendment that makes the new proposal suitable for a specific scenario. However, PRoPHET includes in its draft several proposals regarding forwarding strategies and queuing policies, so aiming at being a highly customizable protocol according to the different scenarios and applications where a DTN can be deployed.

Given its importance in the research community and its standardization probabilities, this protocol will be further analyzed in this section.

There are 4 phases in the PRoPHET routing protocol, namely *peer link setup*, *information exchange*, *routing decision* and *bundle transmission* (Figure 3).

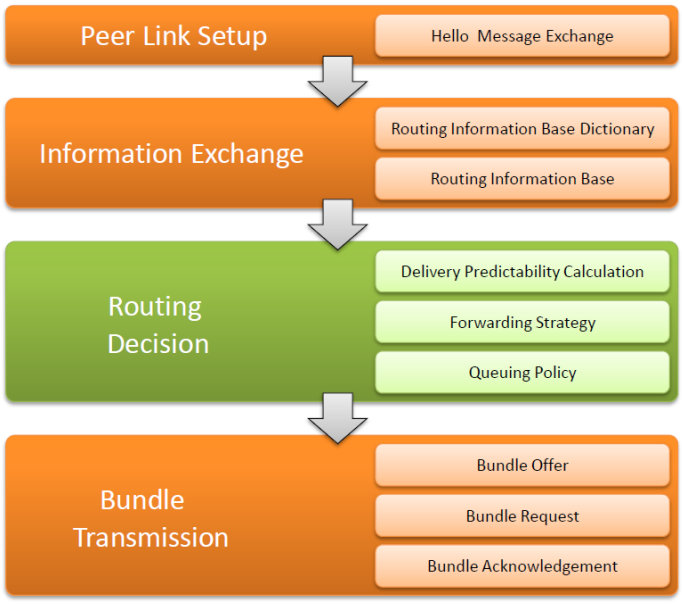


Figure 3: PRoPHET routing protocol phases

The protocol may be run on top of several different network technologies, many of them having their own neighbor discovery mechanism. As a result, PRoPHET does not include any functionality to support this feature, so it relies on underlying layers, thus being notified when neighbor nodes appear and disappear. Once a new neighbor is detected, the protocol sets up a link with that node through an exchange of *hello* messages.

After the link has been set up, the protocol proceeds to the information exchange phase. First, a *Routing Information Base Dictionary* message is sent, which contains all EIDs that will be listed in the *Routing Information Base* message. This message contains the delivery probabilities of all nodes that the sender has knowledge of, so the receiver can proceed to the update of its delivery predictability table. Once this is done, the *routing decision* is performed, thus using its forwarding strategy to decide which of its stored bundles have to be offered to the peering node. After making this decision, a list of all the bundle identifiers and their destinations it wishes to offer is included in the *Bundle Offer* message. Acknowledgements of already received packets are also included in such message. Finally, the receiver of the *Bundle Offer* message evaluates which bundles it is willing to store for future forwarding, so local policies must be considered in order to take a wise decision. Every bundle the receiver accepts to get is included in the *Bundle Request* message, which is sent to the other peer. Next, agreed bundles are sent.

The routing decision is taken according to the delivery predictability calculation and the implemented forwarding strategy. All delivery predictabilities are initially configured to be 0. Then, if nodes A and B meet each other, the probability of contact is updated according the equation 1, where *P(a,b)* is the probability of peer A meeting peer B. However, if a pair of nodes does not meet each other during a period of time the delivery predictability values must age according to equation 2. Finally, from the *Routing Information Base* messages nodes can also update the probability values of their tables applying the transitive property, considering that if node A frequently encounters node B, and node B frequently encounters node C, then node B is a good node to forward bundles destined for node C to. Such computation is depicted in equation 3. P_{init}, B and γ are parameters used to configure how initialization, aging and transitivity updates must affect the delivery predictabilities, respectively. The proposed values for such parameters are 0.75, 0.25 and 0.999, also respectively. However, the designers recommend network operators to configure them according to the requirements that better suit specific scenarios or deployed services.

$$P(a,b) = P(a,b)old + (1 - P(a,b)old) \times Pinit \text{ with } Pinit \in [0, 1] \quad (1)$$

$$P(a,b) = P(a,b)old \times \gamma \text{ with } \gamma \in [0, 1] \quad (2)$$

$$P(a,c) = P(a,c)old + (1 - P(a,c)old) \times P(a,b) \times P(b,c) \times \beta \text{ with } \beta \in [0, 1] \quad (3)$$

PRoPHET considers two different parameters when defining a forwarding strategy (Figure 4), these being a) who forward the message to and b) how many copies of the message to forward. The combination of both parameters allows for different approaches according to many scenarios and applications, thus considering delivery and delay rates and energy efficiency and storage requirements tradeoffs.

A general and obvious approach is to select a fixed threshold and forward a message to nodes that have a delivery predictability value over that threshold for the destination of the message. However, the default strategy proposed by the PRoPHET Internet-Draft, GRTR, aims at forwarding the message if the delivery predictability of the destination d of the bundle is higher at the other node b than the one taking the decision a. The Internet-Draft proposes many other strategies, many of which are based on this first approach. Variations introduced in each proposal follow, where node *a* and *b* are nodes that encounter each other and *d* is the destination node.

GRTR: $P_{(b,\,d)} > P_{(a,\,d)}$

GTMX: $P_{(b,\,d)} > P_{(a,\,d)}$ AND $NF < NF_{max}$

GTHR: $P_{(b,\,d)} > P_{(a,\,d)}$ OR $P_{(b,\,d)} > FORW_{thres}$

GRTR+: $P_{(b,\,d)} > P_{(a,\,d)}$ AND $P_{(b,\,d)} > P_{max}$

GTMX+: $P_{(b,\,d)} > P_{(a,\,d)}$ AND $P_{(b,\,d)} > P_{max}$ AND $NF < NF_{max}$

GRTRSort: DESC($P_{(b,d)} - P_{(a,\,d)}$) $P_{(b,\,d)} > P_{(a,\,d)}$

GRTRMax: DESC($P_{(b,\,d)}$) $P_{(b,\,d)} > P_{(a,\,d)}$

COIN: $X > 0.5$ where $X \in U(0,1)$

Figure 4: Forwarding strategies

GTMX adds a threshold (*NFmax*) that bounds the number of forwards that the node is allowed to make for the specific bundle. GTHR also spreads epidemically bundles to nodes with high delivery predictability (*FORWthres* is a threshold value above which a bundle should always be given to the node). GRTR+ keeps track of the largest delivery predictability of any node it has forwarded this bundle to and only forwards the bundle again if a node with higher delivery predictability is found. GTMX+ mixes GRTR+ and GTMX approaches, thus considering the maximum number of times a bundle can be forwarded and the higher delivery predictability of previously contacted nodes that the bundle was forwarded to. GRTRSort applies GRTR strategy after putting delivery probabilities in descending order according to the difference between the two nodes. GRTRMax also applies ordered GRTR taking into account $P_{(b,d)}$ instead of the difference. Last, COIN is a strategy similar to the ordinary Epidemic Routing, but instead of forwarding to every neighbor node, it only does if a random computation of the variable X (performed each contact time) results in a number higher than 0.5.

Like the previous forwarding strategies, queuing policies (Figure 5) are also scenario-dependent. Then, numerous approaches are proposed in the PRoPHET draft. FIFO is the classical policy where the first bundle to be dropped is the first that was stored in the buffer. MOFO attempts to maximize the delivery rate of bundles, so the most forwarded bundle is the first to be dropped. MOPR and linear MOPR drop those bundles which were forwarded to nodes with high delivery predictability, thus less favorably forwarded bundles will be kept in the queue larger times. The variable FAV depicts such favorability. While MOPR increases such favorability in a weighted way,

FIFO:	First in first out.
MOFO:	Evict most forwarded first
MOPR:	Evict most favorably forwarded first $FAV_{new} = FAV_{old} + (1 - FAV_{old}) \times P$
Linear MOPR:	Evict most favorably forwarded first, linear increase $FAV_{new} = FAV_{old} + P$
SHLI:	Evict shortest life time first
LEPR:	Evict least probable first

Figure 5: Queuing policies

linear MOPR does it linearly, where P is the delivery predictability to the destination of the node who the bundle is forwarded to. SHLI drops from the queue bundles with the shortest life time first, considering that these will be dropped soon anyway. LEPR simply drops the messages with the lower delivery predictability, but taking into account a minimum number of forwards.

Different simulation results [18] depict the protocol versatility and how they can achieve great results in different scenarios. However, further refinements can be done, and actually, are being done given it is an experimental RFC which goal is to become one of the standard routing protocols for DTNs.

This first part of the report introduces the concept of DTN and summarizes its different features, namely the definition, possible applications and services to be deployed and the network architecture, while focusing in the different routing alternatives. Once it has been concluded that DTNs can offer a really useful networking alternative to current standards for specific environments, the knowledge- and probability-based routing schemes are the motivation for the next chapters. As it has been said, knowledge-based routing proposals are not practical since they depict ideal and unrealistic scenarios. However, they can be taken as an upper bound to consider when designing and implementing realistic routing protocols; this is the main reason for this work. Next sections explain the set of requirements considered in the Dijkstra-based route calculator for a smart city data transport service deployment and analyzes the obtained results in different configurations.

III. DETERMINISTIC ROUTING IN A CITY BUS NETWORK

1. Scenario and requirements specification

Smart cities are a general scenario where many applications can be deployed in order to offer innovative services to citizens and public or private institutions. Such services may include real time information about metropolitan transport systems and road traffic, short news broadcasting, general data transport, monitoring systems, and many others.

On this report we focus on a service of data transport using a city bus network where routing nodes are the stations and network vehicles (buses) carry the data from station to station following a scheduled and known path. Data are supposed to be originated by fixed wireless sensors spread all around the city. For the sake of simplicity regarding data generation, it has been assumed that data is created in a physical point in the middle of two bus stations. Then, as the bus passes by the generation place, it gets the packet and computes the path that the packet will follow depending on its destination according to the bus network scheduling. As a result, the packet will start its way to destination, being carried by different buses and transferred on required stations to change from one line to another (Figure 6).

It is worth to mention here how the sensor nodes transmit their data to the bus. For this scenario, it has been supposed that a wake up mechanism [18] is implemented in the implied nodes, both sensor and data collectors at buses. The sensor has two wireless interfaces: one aimed to send data and other to look for close data collectors (buses), so the later requiring less energy to perform its functionality. On the other side, the data collector at the bus periodically broadcasts a wake up signal, which is assumed to be received by nearby sensor nodes. Such method helps saving energy resources and allowing constrained devices such as sensor nodes to extend their lifetime.

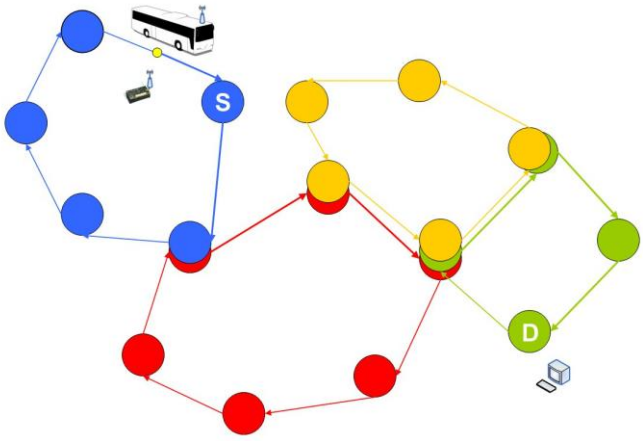


Figure 6: Bus network map

Some real deployment tests have recently been performed to quantify the distance between emitter and receiver of a wake up signal and the performance depending on the height and the speed of the devices. The extreme situation according to the results of the wake up system performance tests allow for 4 meters of distance with the emitter moving at 40 km /h. However, this situation makes sense only in situations when the bus is not stopping at the bus station. Moreover, buses are not expected to reach such speed limit in urban routes, so the performance of the wake up system is granted to be really high.

Although the main routing scheme studied here is the one where vehicles behave as *mules* and stations as relay nodes, two different approaches are also taken into account in order to analyze the network behavior and perform some interesting comparisons. Moreover, different applications may require different routing schemes. These two other methods imply there is no real routing within the bus network, but a) the bus sends the data to destination as soon as it receives the packet on creation location or b) the nearest station sends the data upon receiving it from the bus that got it.

As these schemes do not have any implications on routing, they will only be compared in order to study the differences scheme can be useful in. Moreover, while not defining any storage constraint in nodes for the simulation environment, the required buffer size will be depicted, so it can help with the routed approach in terms of delay and infrastructure deployment.

2. Routing algorithm design

In the bus network scenario, the location of relay nodes (bus stations) is fixed, but their availability is disruptive, meaning this that they are reachable at some punctual instants of time by means of the buses that go around the network, thus acting as data mules. As a result, it is obvious that the metric to be chosen to represent the cost of moving from one node to another is delay, representing the time that a bus requires for moving from one station to the next one. So, the routing algorithm must consider both the instants when the buses stop at the stations and the time between two subsequent stations when computing the best route from source node S to destination node D. Note that the delay metric can be obtained from stop times as a simple subtraction.

Generally, city bus networks are decomposed in different lines, each with a different route in order to reach sparse destinations. Then, given that source and destination may not be in the same line, transference between one or more lines might be required

to be performed. This feature adds a new level of complexity in route calculation as the network graph must include for each station as many relay nodes as bus lines the stop belongs to. Transferences are represented with two edges for each pair of nodes, representing the delay between bus arrivals to the stop for each line (Figure 7). This model is based in the approach devised in [19], thus having a network graph where the classic Dijkstra routing algorithm can be used to look for the path with the lower delivery delay among all the available options.

The transference event is produced at the exact moment the bus arrives at the station, calculated taking into account the bus lines schedule provided by the village council, where all required information is available. It must be noted that each routing simulation takes into account a random pair of source and a destination nodes, corresponding to specific stations of the city bus network. In order to compute the path and the related delay, the instants of data generation and data arrival must be clearly defined. Data arrival time is simply defined as the time when the bus carrying the bulk of data reaches the destination station. Data generation time corresponds to the time when the packet or packets are created by a sensor node, which is not part of the city bus network. In order to compute the time when the packet is received by the bus, the algorithm considers the mid time when the bus goes from the station previous to the source one and the source station itself. All mid times are computed and the earliest one that occurs after the packet generation is considered as the instant time when the packet gets into the bus, thus entering the bus network.

A specific case concerns the link between the origin and end stations of each line in case of circular lines, implying this that both stops are the same physical node, so there

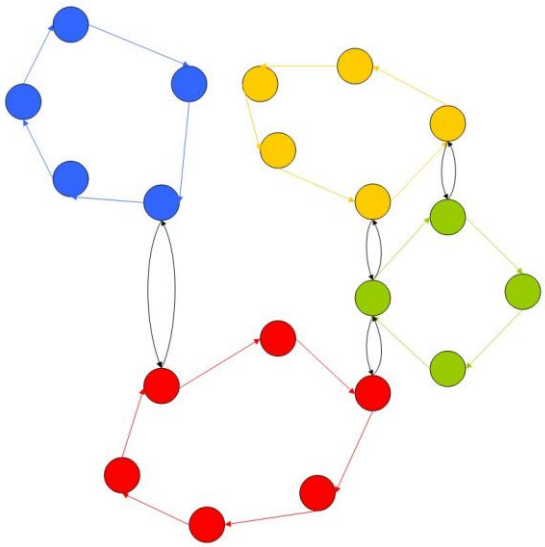


Figure 7: Bus network graph

is no real time between them. In the graph, these nodes are represented by two different stations linked by an edge with a cost equal to the time between the arrival and the next departure of the bus from the station. In case the randomly obtained data generation time is in this range, the *time to bus* (TTB) and *time to station* (TTS) will be both equal 0.

We call TTB the difference between the time when the packet arrives at the bus and the time when the packet is generated. The TTS reflects the difference between the instant when the bus that gets the packet reaches the source station and the time of data generation. There is no great difference between TTB and TTS, but it is interesting to compare both values and the *time to destination* (TTD) as they represent different data delivery schemes depending on the network infrastructure and service model. Then, the TTD is defined as the difference between the time when the data reaches the destination station through the bus network and the generation time.

IV. CASE STUDY: Ideal city bus network

1. Network configuration

The city of Barcelona has a notably deficient bus service due to a suboptimal bus network deployment. However, despite having an important bus fleet deployed, the vast majority of the lines focus their service on the city center (Figure 8), thus having a huge number of transference opportunities at some places and only a few on sparse locations.

In order to improve the efficiency of such bus network, the Council of Barcelona has recently started a research project in collaboration with CENIT and TMB. This project aims at designing and deploying a *Bus Rapid Transit* (BRT) service, which should eventually replace the current one. The network map conceived for being used in this service, depicted in Figure 9 is based in a hybrid model of mesh and radial network distribution, proposed by the professor Carlos Daganzo [20]. Such model diminishes the time users have to wait for the next bus and the number of required vehicles while increasing the transit speed.

The proposal of this report is closer to the standard mesh model, based in an ideal city of similar size to Sant Vicenç dels Horts, which will be studied in the next section, where all streets are orthogonal, so 10 lines are divided in 2 groups, transversal and longitudinal, each with 5 lines equally distributed in the city map as shown in Figure 100. Note the bus stops are also distributed following a symmetric and proportional distribution, considering transference stops in many line crossing locations. Additionally, in order to enhance the reachability to the city center (the commonly most crowded zone of the entire city), the density of stops, thus the density of transference stops, is higher in the central locations.

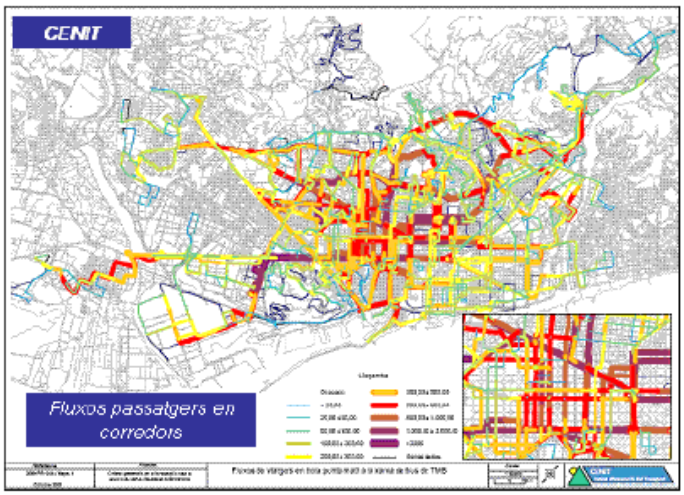


Figure 8: Current bus map at Barcelona

Figure 9: RETBUS map

Throwboxes [21, 22, 23] are additional fixed relay nodes that are added to a DTN in order to increase the number of available paths or even create new ones to decrease the delivery delay. Improvement in delay time through the deployment of throwboxes in specific locations has been studied with the ideal city bus network, and will be compared with a real city in the next section. Two different cases have been considered: first, locating only 8 throwboxes, each placed in one of the two possibilities of the crossing lines locations where there were no transference stops (Figure 111). Second, 8 additional throwboxes were placed in the remaining possibilities (Figure 12). In this last case, all line crossing locations have either a transference stop or a throwbox, which allows for data transferring from line to line in such places.

A scheduling required for calculating the bus frequency is proposed with the following parameters. A round is started every 5 minutes from the line origin station where the bus departures from. This value requires that 5 buses are concurrently providing a line service, thus having 50 concurrent buses during the day. The time between two stations is set to 1 minute, as it is considered as a realistic average value. When a throwbox is added between 2 stops, this is located at 30 seconds from the previous and next stations. Finally, the service start time for all bus lines is set to 7:30 AM. All these parameters are shared by all bus lines for the sake of simplicity.

2. Simulation results

In the studied scenario we will suppose that, if a route from source to destination is available at any time, the data will traverse it hop by hop, so reliability will not be a parameter to study. Although being a Delay Tolerant Network, quantifying the delivery delay is important in order to consider which applications and scenarios this routing

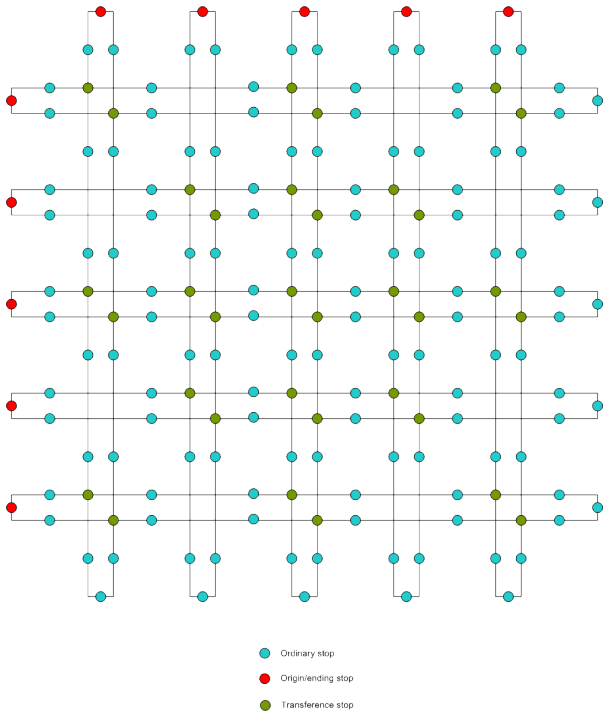


Figure 11: Ideal city bus network graph

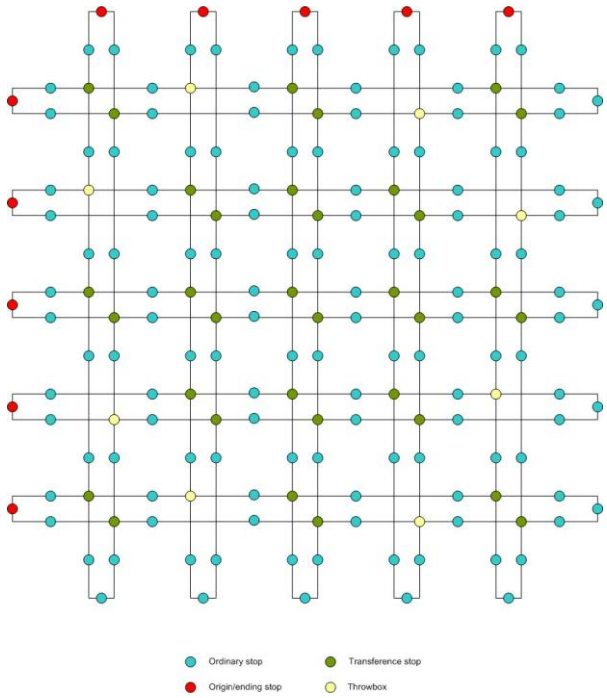


Figure 10: Ideal city bus network graph with 8 throwboxes

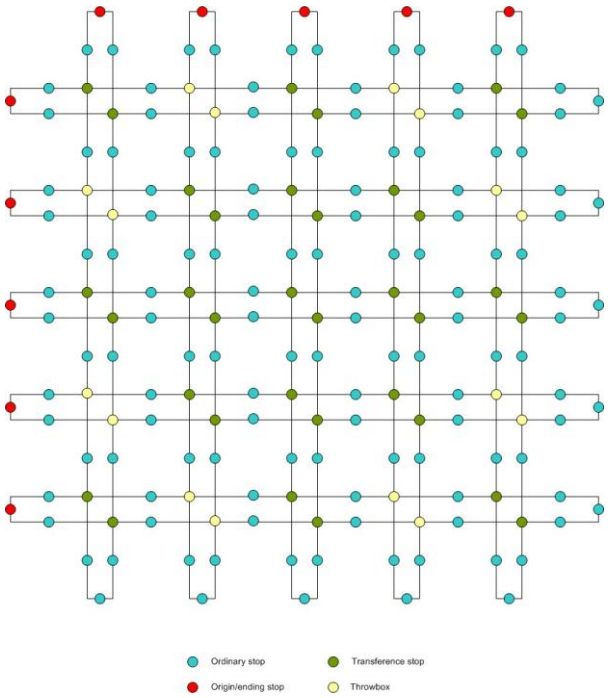


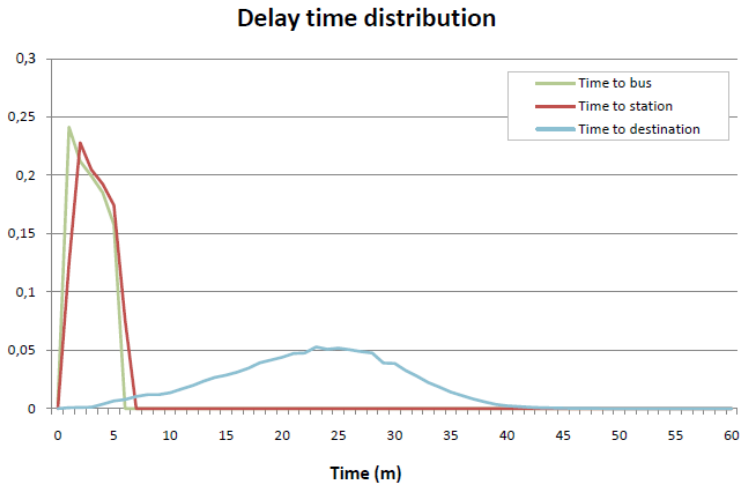
Figure 12: Ideal city bus network graph with 16 throwboxes

scheme can be useful in. Moreover, while not defining any storage constraint in nodes for the simulation environment, the required buffer size will be depicted, so it can help in extracting conclusions related to usability and required modifications to the routing protocol.

2.1. Delivery delay

Figure 13 depicts the results of delivery delay in the three different proposed schemes, including how long data takes to reach the bus (*Time To Bus*, TTB), the stop (*Time To Stop*, TTS) and the destination (*Time To Destination*, TTD).

The first conclusion that can be extracted from these data is that the maximum and average delay times are really low. Actually, no packet remains in the network more than an hour when buses are used as data mules (TTD scheme), which is the most interesting case. Maximum TTB is under 5 minutes as this is the time between two consecutive arrivals of a bus to any station. The TTS is nearly the same as, once the bus gets the packet, it is on its way to the next station. Given that time between two consecutive stations is 1 minute and data is supposed to be generated in the middle of them, the maximum TTS is under 5,5 minutes. Finally, regarding TTD, the average time is 22 minutes while the 95-th percentile is 34 minutes. The upper bound is 50 minutes, which is really good for environment monitoring applications for example.



	SECONDS	MINUTES
MAX TTB	299,0000	4,9833
MIN TTB	0,0000	0,0000
AVERAGE	138,0886	2,3015
DEVIATION	85,9109	1,4318
PER(95)	280,0000	4,6667
PER(85)	243,0000	4,0500
PER(75)	211,0000	3,5167
PER(50)	133,0000	2,2167

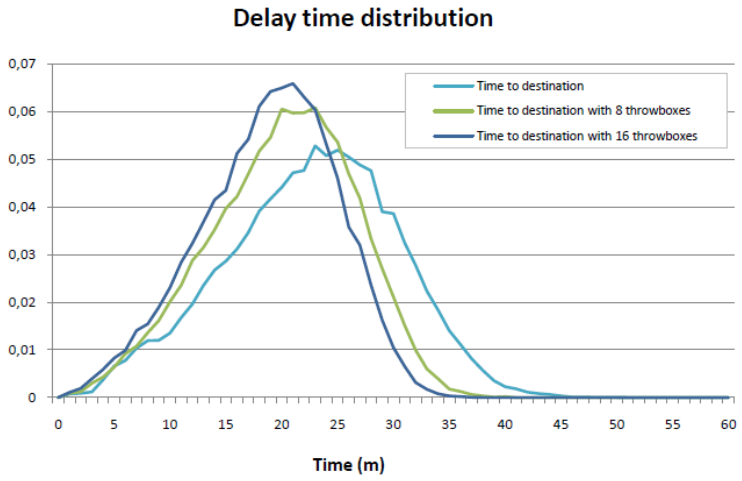
	SECONDS	MINUTES
MAX TTS	329,0000	5,4833
MIN TTS	30,0000	0,5000
AVERAGE	168,0886	2,8015
DEVIATION	85,9109	1,4318
PER(95)	310,0000	5,1667
PER(85)	273,0000	4,5500
PER(75)	241,0000	4,0167
PER(50)	163,0000	2,7167

	SECONDS	MINUTES
MAX TTD	2992,0000	49,8667
MIN TTD	30,0000	0,5000
AVERAGE	1328,3315	22,1389
DEVIATION	457,0700	7,6178
PER(95)	2039,0000	33,9833
PER(85)	1802,0000	30,0333
PER(75)	1653,0000	27,5500
PER(50)	1356,0000	22,6000

Figure 13: Delivery delay results in an ideal city bus network

2.1.1. Throwboxes enhancement

When introducing throwboxes in the city bus map, the delay time is decreased as shown in Figure 14. The more throwboxes are introduced, the faster the data reaches its destination, which is an obvious consequence as more route intersections appear between different lines, so more data handover opportunities appear. However, such delay decrease becomes slighter as more throwboxes are added. This a consequence specific to the studied scenario as the second set of throwboxes added to the bus network just adds them in intersections where the same lines already had a throwbox placed, although in a different location (the picture of the previous section clearly depicts this situation).



NO TB	SECONDS	MINUTES
MAX TTD	2992,0000	49,8667
MIN TTD	30,0000	0,5000
AVERAGE	1328,3315	22,1389
DEVIATION	457,0700	7,6178
PER(95)	2039,0000	33,9833
PER(85)	1802,0000	30,0333
PER(75)	1653,0000	27,5500
PER(50)	1356,0000	22,6000

8 TB	SECONDS	MINUTES
MAX TTD	2399,0000	39,9833
MIN TTD	18,0000	0,3000
AVERAGE	1172,5883	19,5431
DEVIATION	388,7189	6,4786
PER(95)	1768,0000	29,4667
PER(85)	1579,0000	26,3167
PER(75)	1455,0000	24,2500
PER(50)	1200,0000	20,0000

16 TB	SECONDS	MINUTES
MAX TTD	2166,0000	36,1000
MIN TTD	17,0000	0,2833
AVERAGE	1096,5510	18,2758
DEVIATION	364,5527	6,0759
PER(95)	1652,0000	27,5333
PER(85)	1475,0000	24,5833
PER(75)	1360,0000	22,6667
PER(50)	1125,0000	18,7500

Figure 14: Delivery delay results in an ideal city bus network with throwboxes

2.2. Queue size

In Figures Figure 15, Figure 16 and Figure 17 it can be seen how the amount of buffered packets change with time in three different nodes of the network, including one exterior intersection, one interior intersection and one throwbox. Note that more nodes were also monitored but, as they shown the same behavior, only three figures have been included. 16 bundles is the maximum capacity a network node is required to have in order to store packets as they are transferred from one bus line to another. Such capacity is really low and any node with low capabilities can accomplish them.

The addition of throwboxes does not produce any great improvement given the low storage requirements for this scenario.

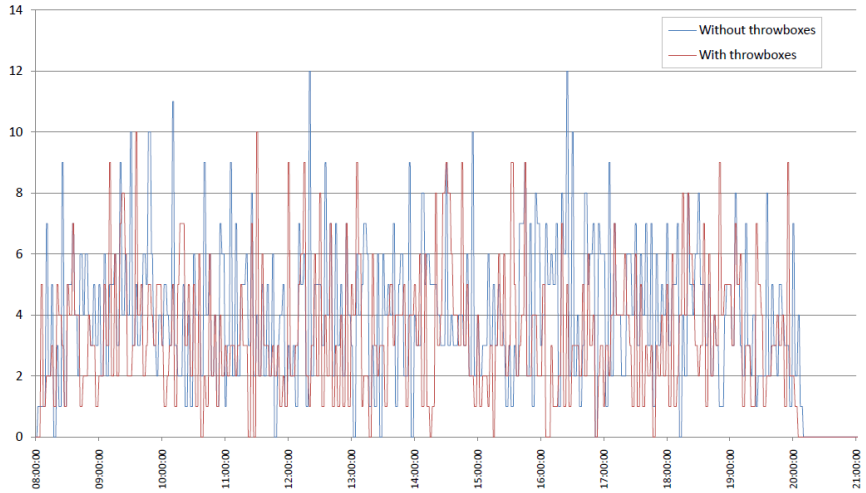


Figure 15: Buffered packets in stop V303 –H116

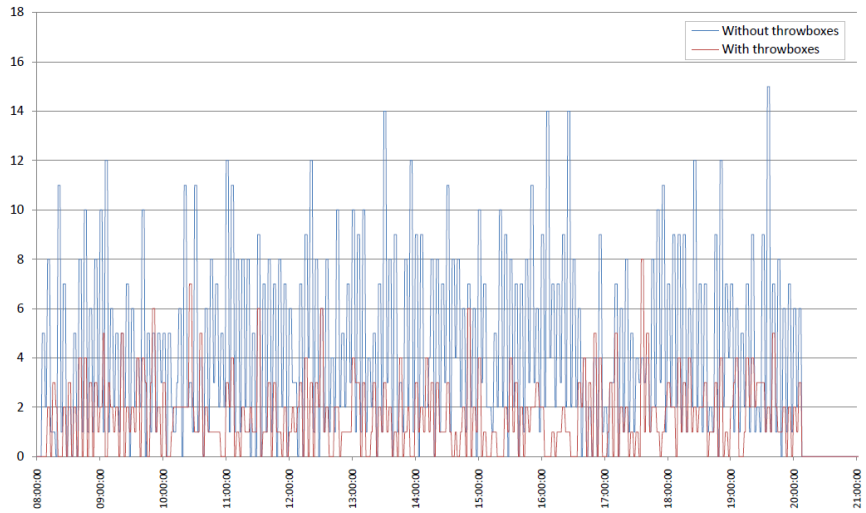


Figure 16: Buffered packets in stop V309 –H416

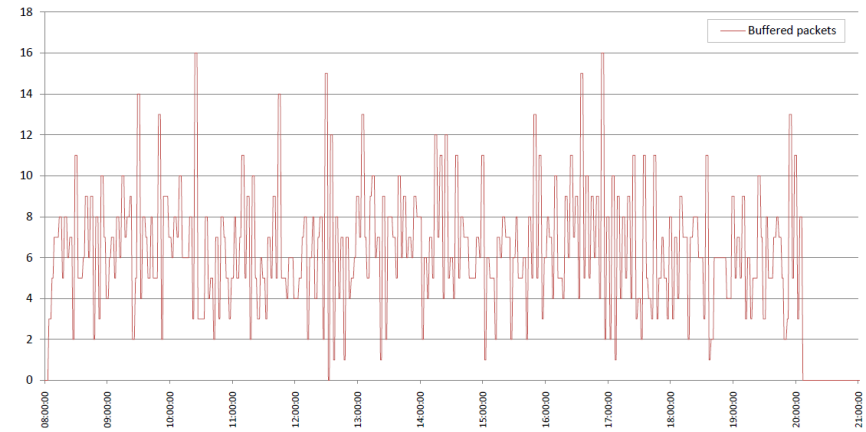


Figure 17: Buffered packets in stop T3

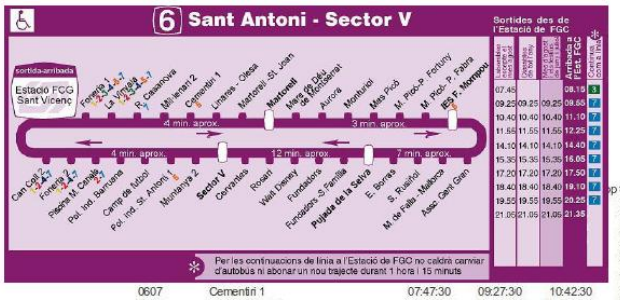
V. CASE STUDY: Sant Vicenç dels Horts

1. Network configuration

Sant Vicenç dels Horts is village with a bus network formed by 10 circle lines (Figure 19). Almost all paths are provided with redundancy, so many stations belong to more than one line. However, some parts of different lines provide individual reachability to sparse zones of the town.

This configuration allows data carried through the bus network to be properly transferred between lines as needed in order to reach sparse locations or just reach concurred places faster due to a line with a higher service frequency. Despite this advantage, such service frequencies are not as higher as desired, given that all line routes are performed by only 3 buses, thus introducing delay. As a result, this scenario reflects a proper environment for the study of routing in DTNs.

Throwboxes distribution is not supposed to provide a great improvement in routing over the bus network of this city, because of the already existing concurrency in many stations. Despite of that, 3 extra nodes have been located in specific map locations where new transference opportunities are expected to appear. These locations represent places where different bus lines cross their routes and no station allow the transference between them, so the throwboxes provide it.



L06

		Stop time 4	Stop time 5	Stop time 6	Stop time 7	Stop time 8	Stop time 9	Stop time 10
0607	Cementiri 1	11:55:00	14:10:00	15:35:00	17:20:00	18:40:00	19:55:00	21:05:00
0608	Linares - Olsa	11:55:30	14:10:30	15:35:30	17:20:30	18:40:30	19:55:30	21:05:30
0609	Martorell - St. Joan	11:56:00	14:11:00	15:36:00	17:21:00	18:41:00	19:56:00	21:06:00
0610	Martorell	11:56:30	14:11:30	15:36:30	17:21:30	18:41:30	19:56:30	21:06:30
0611	Mare de Déu de Montserrat	11:56:45	14:11:45	15:36:45	17:21:45	18:41:45	19:56:45	21:06:45
0612	Aurora	11:57:00	14:12:00	15:37:00	17:22:00	18:42:00	19:57:00	21:07:00
0613	Monturiol	11:57:30	14:12:30	15:37:30	17:22:30	18:42:30	19:57:30	21:07:30
0614	Mas Picó	11:58:00	14:13:00	15:38:00	17:23:00	18:43:00	19:58:00	21:08:00
0615	M. Picó - P. Fortuny	11:58:30	14:13:30	15:38:30	17:23:30	18:43:30	19:58:30	21:08:30
0616	M. Picó - P. Fabra	11:59:00	14:14:00	15:39:00	17:24:00	18:44:00	19:59:00	21:09:00
0617	IES Frederic Mompou	11:59:26	14:14:26	15:39:26	17:24:26	18:44:26	19:59:26	21:09:26
0618	Ass. Gent Gran	11:59:52	14:14:52	15:39:52	17:24:52	18:44:52	19:59:52	21:09:52
0619	M. de Falla - Mallorca	12:00:18	14:15:18	15:40:18	17:25:18	18:45:18	20:00:18	21:10:18
0620	S. Rusiñol	12:00:44	14:15:44	15:40:44	17:25:44	18:45:44	20:00:44	21:10:44
0621	E. Borras	12:01:10	14:16:10	15:41:10	17:26:10	18:46:10	20:01:10	21:11:10
0622	Pujada de la Selva	12:01:36	14:16:36	15:41:36	17:26:36	18:46:36	20:01:36	21:11:36
0623	Fundadors - S. Familia	12:02:02	14:17:02	15:42:02	17:27:02	18:47:02	20:02:02	21:12:02
0624	Fundadors	12:03:26	14:18:26	15:43:26	17:28:26	18:48:26	20:03:26	21:13:26
0625	Walt Disney	12:04:50	14:19:50	15:44:50	17:29:50	18:49:50	20:04:50	21:14:50
0626	Rosari	12:06:14	14:21:14	15:46:14	17:31:14	18:51:14	20:06:14	21:16:14
0627	Cervantes	12:07:38	14:22:38	15:47:38	17:32:38	18:52:38	20:07:38	21:17:38
0628	Sector V	12:09:02	14:24:02	15:49:02	17:34:02	18:54:02	20:09:02	21:19:02
0629	Muntanya 2	12:11:02	14:26:02	15:51:02	17:36:02	18:56:02	20:11:02	21:21:02
0630	Pol. Ind. St. Antoni 1	12:13:02	14:28:02	15:53:02	17:38:02	18:58:02	20:13:02	21:23:02
0631	Camp de futbol	12:15:02	14:30:02	15:55:02	17:40:02	19:00:02	20:15:02	21:25:02
0632	Pol. Ind. Barruana	12:17:02	14:32:02	15:57:02	17:42:02	19:02:02	20:17:02	21:27:02
0633	T3	12:19:02	14:34:02	15:59:02	17:44:02	19:04:02	20:19:02	21:29:02
0634	Piscina M. Canals	12:21:02	14:36:02	16:01:02	17:46:02	19:06:02	20:21:02	21:31:02
0635	Foneria 2	12:21:32	14:36:32	16:01:32	17:46:32	19:06:32	20:21:32	21:31:32
0636	Can Coll 2	12:22:02	14:37:02	16:02:02	17:47:02	19:07:02	20:22:02	21:32:02
0637	FCG	12:22:32	14:37:32	16:02:32	17:47:32	19:07:32	20:22:32	21:32:32
		12:23:02	14:38:02	16:03:02	17:48:02	19:08:02	20:23:02	21:33:02
		12:23:17	14:38:17	16:03:17	17:48:17	19:08:17	20:23:17	21:33:17
		12:23:32	14:38:32	16:03:32	17:48:32	19:08:32	20:23:32	21:33:32
		12:24:02	14:39:02	16:04:02	17:49:02	19:09:02	20:24:02	21:34:02
		12:24:32	14:39:32	16:04:32	17:49:32	19:09:32	20:24:32	21:34:32
		12:25:02	14:40:02	16:05:02	17:50:02	19:10:02	20:25:02	21:35:02

Figure 18: Scheduling data extraction

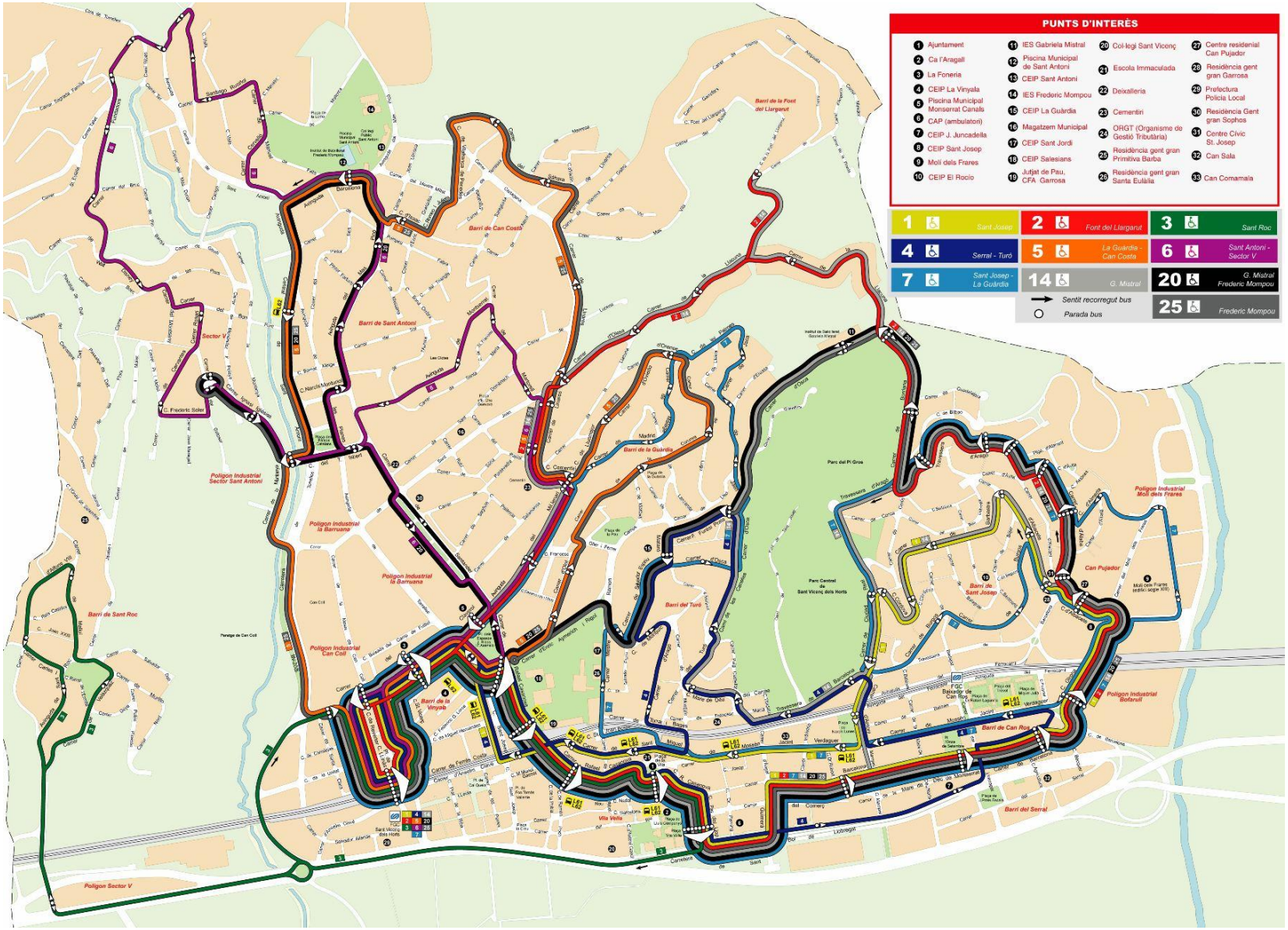


Figure 19: Sant Vicenç dels Horts bus network map

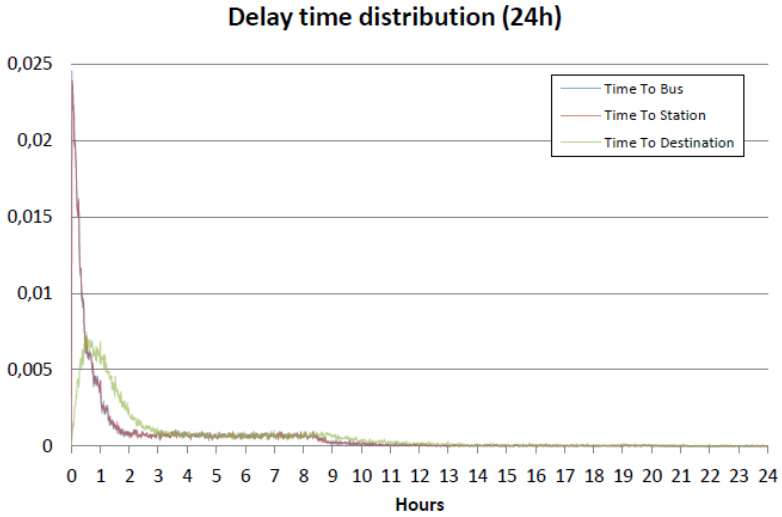
2. Simulation results

Next, simulation results in a real scenario will be discussed, thus showing how low delivery delay results and low buffering load from the previous ideal city network are no valid with current real deployments, such as the one in Sant Vicenç dels Horts.

2.1. Delivery delay

In the real city scenario, the delivery delay has also been studied in the case where data is generated at any time during the day, including when the bus service is not active. Such case can be interesting for specific applications and, thus, analyzing how long it would take to receive the data through the different schemes may be important.

As depicted in Figure 20, three different ranges can be established. First, data that takes less than 3 hours to reach destination, which corresponds to places with regular bus frequency or favorable scheduling to isolated stops. Next, there is a range where data remains in the network between 3 and 8 hours, which depicts the case where data is generated at night and no bus gets the bundle until the bus service starts in the morning. Last, data generated at night in isolated stops or unfavorable scheduling to such isolated stops take a long time to get delivered to destination, more than 8 hours.



	SECONDS	HOURS
MAX TTB	86379,0000	23,9942
MIN TTB	0,0000	0,0000
AVERAGE	8692,9673	2,4147
DEVIATION	12534,6299	3,4818
PER(95)	31309,1000	8,6970
PER(85)	22193,3000	6,1648
PER(75)	13461,2500	3,7392
PER(50)	2478,5000	0,6885

	SECONDS	HOURS
MAX TTS	86394,0000	23,9983
MIN TTS	13,0000	0,0036
AVERAGE	8734,9334	2,4264
DEVIATION	12541,9601	3,4839
PER(95)	31337,2000	8,7048
PER(85)	22271,1500	6,1864
PER(75)	13544,0000	3,7622
PER(50)	2512,0000	0,6978

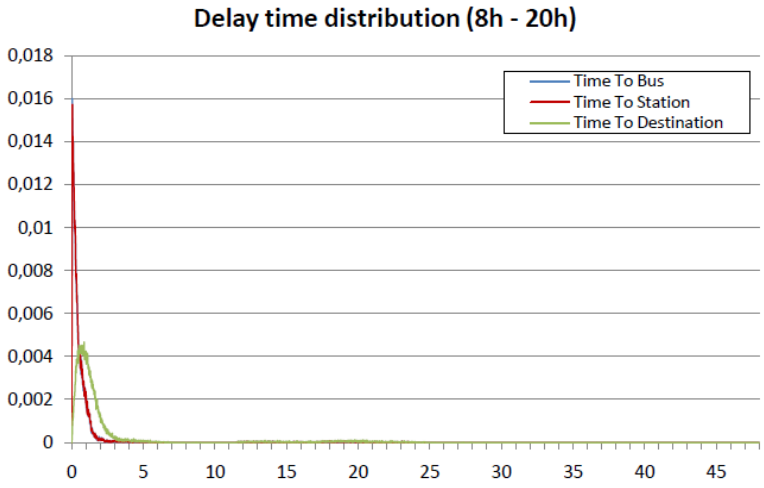
	SECONDS	HOURS
MAX TTD	171869,000	47,7414
MIN TTD	17,0000	0,0047
AVERAGE	13367,3128	3,7131
DEVIATION	15816,8712	4,3936
PER(95)	43228,6500	12,0080
PER(85)	29010,3000	8,0584
PER(75)	20239,2500	5,6220
PER(50)	6057,5000	1,6826

Figure 20: Delivery delay results in SVH

The maximum values depict the extreme cases where the data is generated in isolated stops and it must delivered to another isolated stop, thus waiting for a specific bus line to offer the service in such places, which are sometimes performed once in a day. In the case of TTD, such scheduling issues can produce a bundle to remain in the network until 2 days. Although the average values can be pretty good for many applications, the deviation is so high that some bundles may remain in the network too much time.

2.1.1. Bounded data generation time

Next, the delivery delay has been studied in a scenario where the data generation time is bounded between 8:00 and 20:00 hours, thus ensuring that the bus service is active when the bundle is created. As it can be seen in Figure 21, the range from 3 to 8 hours "disappears" and the overall delivery delay is decreased as results in the figure depicts. New average and deviation values also show better results in this scenario, which may suit applications where no data transport is required during the night and only produce data during the daytime, when the bus service is active.

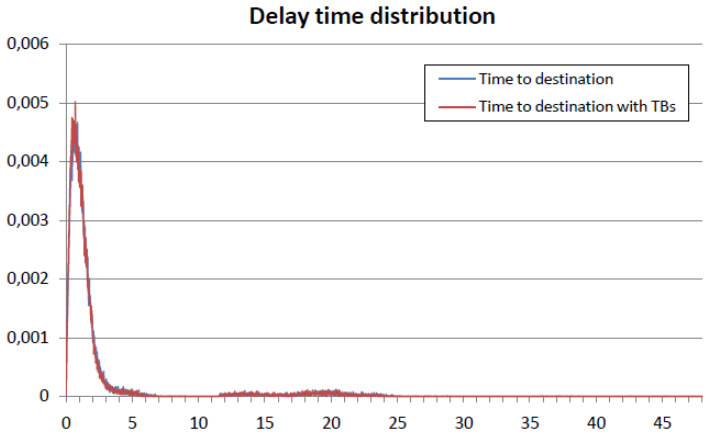


	SECONDS	HOURS		SECONDS	HOURS		SECONDS	HOURS
MAX TTB	84961,0000	23,6003	MAX TTS	84979,0000	23,6053	MAX TTD	153124,0000	42,5344
MIN TTB	0,0000	0,0000	MIN TTS	13,0000	0,0036	MIN TTD	24,0000	0,0067
AVERAGE	3501,7593	0,9727	AVERAGE	3530,2214	0,9806	AVERAGE	7786,1225	2,1628
DEVIATION	10654,8620	2,9597	DEVIATION	10654,3003	2,9595	DEVIATION	14983,4670	4,1621
PER(95)	7599,0500	2,1108	PER(95)	7644,1500	2,1234	PER(95)	49258,6000	13,6829
PER(85)	3483,0000	0,9675	PER(85)	3511,0000	0,9753	PER(85)	7746,0000	2,1517
PER(75)	2494,2500	0,6928	PER(75)	2527,0000	0,7019	PER(75)	5892,0000	1,6367
PER(50)	1117,0000	0,3103	PER(50)	1148,0000	0,3189	PER(50)	3717,5000	1,0326

Figure 21: Delivery delay results in SVH (bounded data creation)

2.1.2. Throwboxes enhancement

As said before, only 3 locations in the bus network of the city have been envisaged as potential places where throwboxes could enhance the performance of the DTN in terms of delivery delay. Such addition of throwboxes at strategic positions does not reflect any important improvement as shown in Figure 22, although actually there is a slight decrease in the delivery delay. Note that the data generation time model selected for this simulation has been the bounded one, which seems a more logical scenario given the characteristics of the bus service and, thus, the DTN itself. Note also that only the TTD scheme is depicted in figure 24, given the addition of throwboxes does not introduce any change in TTB and TTS schemes, as their functionality is to perform as forwarding nodes which only make sense in the TTD scheme.



	SECONDS	HOURS
MAX TTD	162890,0000	45,2472
MIN TTD	21,0000	0,0058
AVERAGE	7433,9391	2,0650
DEVIATION	14516,8952	4,0325
PER(95)	47690,3500	13,2473
PER(85)	7352,0000	2,0422
PER(75)	5688,0000	1,5800
PER(50)	3565,0000	0,9903

Figure 22: Delivery delay results in SVH with throwboxes

2.2. Queue size

Figure 23, Figure 24, and Figure 25 show the evolution of data buffering in some nodes. The different stops show also different load evolutions given their location in the network. For example, the *FGC* stop requires capacity to store up to 1500 bundles, although it has a fast variation. This is due to the location of the stop, which corresponds to the origin and ending of all bus lines of the network. This means that all lines share this stop, so, in case no transference is available in any other stop, *FGC* will be the stop to host bundles. The stop *Associació Gent Gran* shows a different behavior; given the bus lines which this stop belongs have a low frequency, so bundles remain in the stop during a longer time. However, lower storage requirements can be envisioned due to the low number of lines that share this stop. The stop Ajuntament has low load during all the time (less than 50 bundles) except for some specifics points in time, which corresponds to the transferences of lines with high load. In such instants,

a great amount of data is hosted during a short time, as it is quickly transferred given the high frequency of buses at this stop

Regarding the effect of throwboxes, it can be seen as they, despite representing a small number of all nodes in the network, allow decreasing the storage requirements for the three studied stops, although only slightly. Also, in Figure 26, Figure 27 and Figure 28 storage evolution of the three throwboxes is depicted. Note the similarity with the nodes previously analyzed, sharing the same behavior. Thus, *FGC* and T1 show a location with high bus frequency of many lines, *Associació de Gent Gran* and T2 depicts a confluence of low load lines with low bus frequency and *Ajuntament* and T3 show a low load transference point where high frequency lines converge.

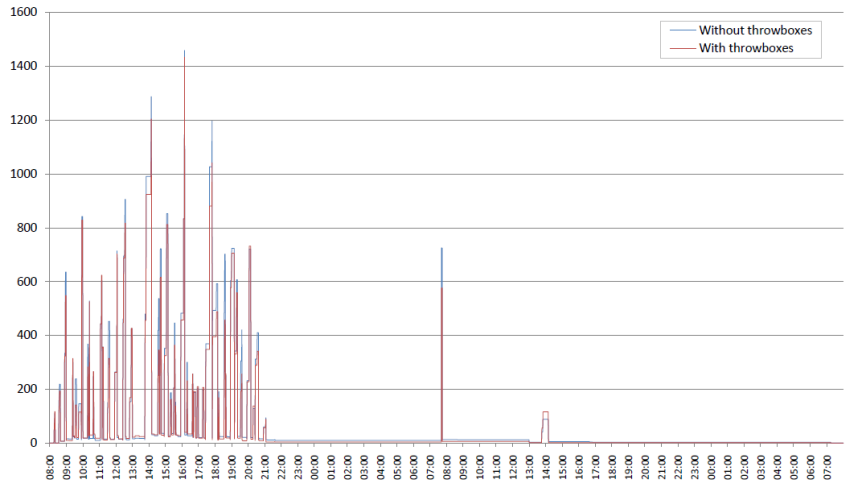


Figure 23: Buffered packets in stop FGC

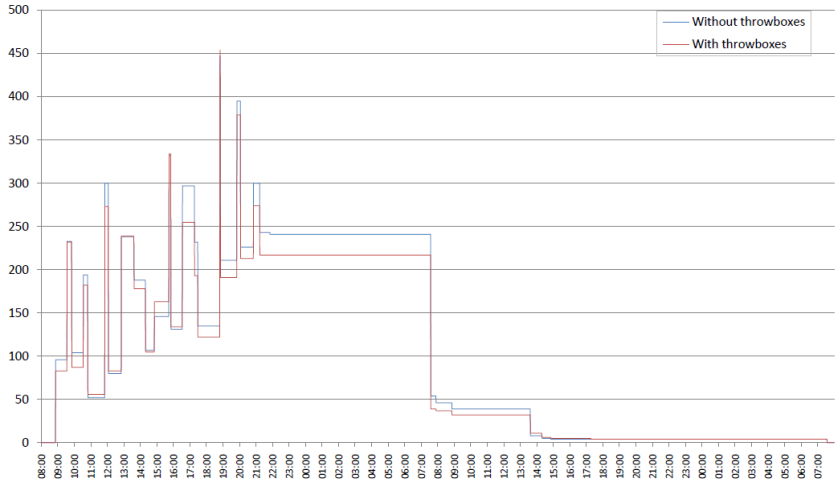


Figure 24: Buffered packets in stop Associació Gent Gran

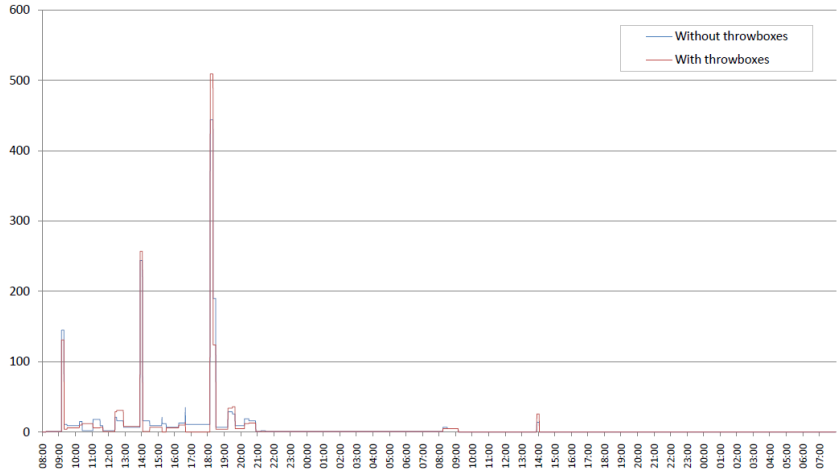


Figure 25: Buffered packets in stop Ajuntament

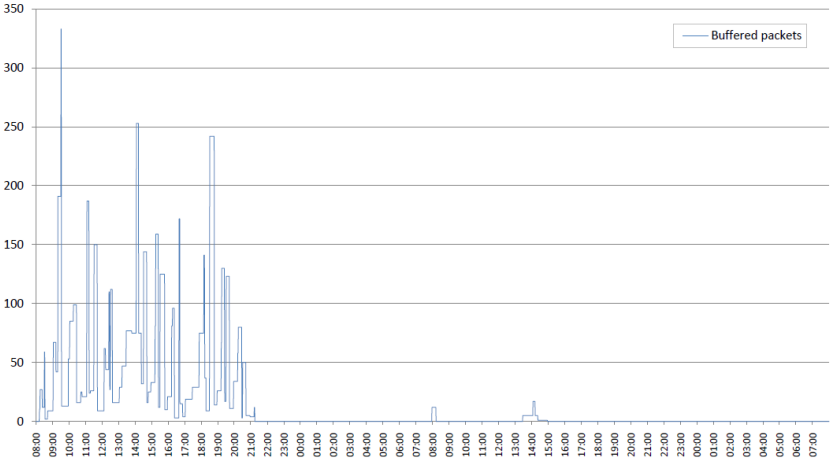


Figure 26: Buffered packets in throwbox T1

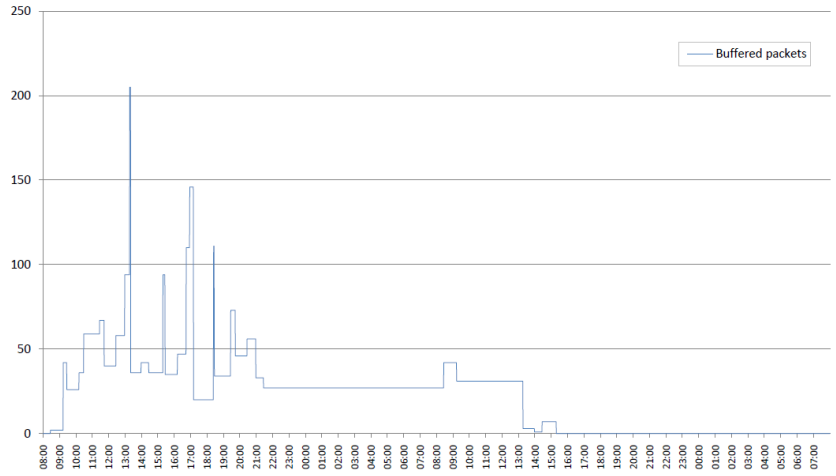


Figure 27: Buffered packets in throwbox T2

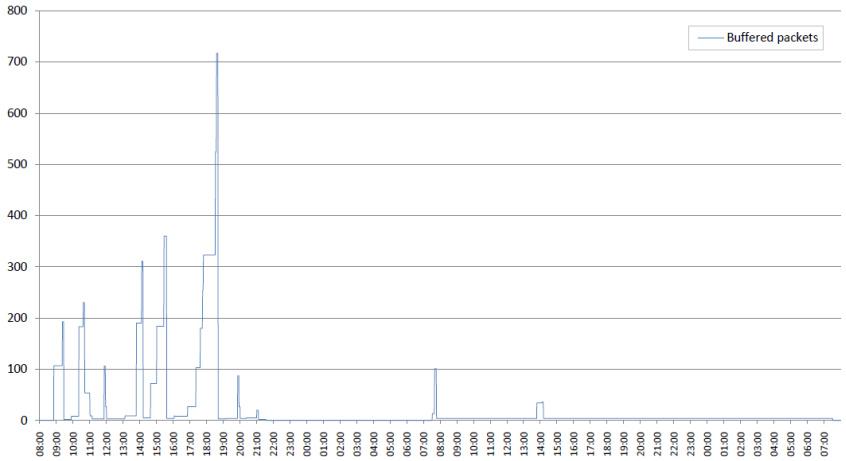


Figure 28: Buffered packets in throwbox T3

VI. CONCLUSIONS AND FUTURE WORK

Delay Tolerant Networks introduce a new paradigm of networking somehow similar to MANETs and VANETs where nodes are interconnected through highly time-variable links, so introducing new challenges that claim for new protocols. As a result, the *Delay Tolerant Networking Research Group*, an IRTF initiative, is defining a common DTN network architecture, including the *Bundle Protocol*, the *Licklider Transport Protocol* and the PRoPHET routing protocol, amongst other standardization efforts and initiatives. Several services and applications are envisioned to be potential deployment scenarios, including wireless sensor networks and deep space missions, thus, although no killer application has been identified yet, these networks can be really useful in specific scenarios where other networking proposals do not fit the requirements.

Routing in DTNs is one of the most interesting issues in this field, so many approaches have emerged considering knowledge-, replication- and probability-based alternatives. While PRoPHET is envisioned as the future main routing protocol for the devised scenarios, it is still in definition process by the corresponding research bodies. However, it is expected to not perform better than the defined as optimal cases where mobility patterns of involved nodes are known in advance and they are powered with unlimited resources. This situation presents an ideal deterministic routing model and this report explains how a route calculator to model such threshold has been implemented and how such results can be useful in future research tasks.

The designed route calculator is based on extended Dijkstra and its routing results for both ideal and real cases of city bus networks have been analyzed in terms of delivery delay and buffering requirements. The results show that different delivery approaches can better suit each scenario. While a pure DTN model shows reasonable results in a bus network with a regular topology distribution, a model where buses do not act as mules would be a better choice for irregular maps, like the one in Sant Vicenç dels Horts.

However, the routing scheme to be deployed depends on the application or service to be offered and its requirements. For example, environmental monitoring using a WSN where humidity and temperature data from some locations must be gathered with no urgency, the results of the DTN scheme where buses act as mules in SVH might be fully suitable. On the other side, non-critical alarm triggering service (i.e. alarms do not require to be triggered at real time), like city lighting incidences, could use schemes where delivery delay times are lower. So, data could be sent through cellular technologies once the bus gets it or as it arrives at the next bus stop.

Locating additional relay nodes, the so-called *throwboxes*, in specific network places has also been tested. The results show that if their location is strategically selected, linking two or more bus lines previously not linked, it can really improve the results, thus reducing the delivery delay and even the buffering requirements. The concept of throwboxes helps in considering a hybrid network where not only a DTN must be in charge of data transport, but adding specific nodes like throwboxes and even a classic fixed network can clearly enhance the overall service performance. For example, in isolated city locations, like those bus lines where only once a day a bus round is performed, instead of relying in the DTN or deploying throwboxes, independent fixed nodes powered with a cellular communication interface would better suit the requirements of the service. Thus, combining fixed network architecture with an opportunistic proposal like the one offered by DTN and throwboxes can help in offering a new set of applications with great performance results.

The general interest, as in any other field, is to get the best performance requiring the lowest investment, so achieving the highest efficiency. DTNs aim at providing a networking architecture that suits such requirements. However, as it has been seen in this report, lots of efforts need to be made in order to improve current proposals, especially in routing protocols where many alternatives have appeared and PRoPHET is the only one which has reached an status important enough to become a standard in a near future given its flexibility. Such proposal offers a wide set of options that allow tweaking the protocol for specific applications and scenarios.

Future work will be focused in studying the PRoPHET proposal, including network simulation and testbed deployment in order to quantify basic parameters such as those analyzed in this report and others which may be of interest. It will be compared with the upper bound presented here with the knowledge-based scheme where all the information regarding the bus network scheduling is known in advance, so the goal will be to achieve the highest routing efficiency. The different forwarding strategies and queuing algorithms will also be studied and new proposals are expected to appear, thus contributing in the protocol definition at possible standardization stages.

Further refinements in the different simulation scenarios are also expected to be added. For example, the used data generation model simply creates a packet in an instant of time randomly chosen in the specified range. Specific models should be implemented when considering real applications, thus defining data creation rate and bundle size.

Regarding real deployment of a DTN in order to evaluate the routing protocol behavior, some methods to save storage requirements and energy consumption should be

included in the platform design and implementation. Such enhancements can include a different coding scheme for bus stop names and stop times, for example, and further data compression techniques.

Regarding energy consumption efficiency, it makes sense that nodes only had to use the radio communications when a link is established, so no intense radio hardware use should be performed in any other case. To accomplish this, radio wake-up or RFID systems would be useful, as mentioned in Chapter IV. This solution will be further evaluated in a real DTN deployment as future work, so new variations and enhancements could be added.

Delay Tolerant Networking is a research field that has produced a considerably big community of people interested in providing a solution to provide an efficient communication architecture in scenarios where previous proposals do not fulfill the specific requirements. Network congestion, routing algorithms, security issues, architectural challenges, killer application quests… Several ideas are constantly appearing to face the different problems DTNs must face to arise as a really solid solution. Some of these ideas have been discussed through this report and a first step to further research in routing algorithm design for such networks has been introduced. Future work is expected to somehow contribute in the adoption of DTN as a real solution to specific services and scenarios where they may become really useful.

REFERENCES

[1] A. Lindgren, P. Hui; *The quest for a killer app for opportunistic and delay tolerant networks (invited paper)*; CHANTS '09, Proceedings of the 4th ACM workshop on Challenged networks; New York, USA, 2009.

[2] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.S. Peh, D. Rubenstein; *Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet*, ASPLO-X, Proceedings of the 10th international conference on Architectural support for programming languages and operating systems; New York, USA; 2002.

[3] http://www.jpl.nasa.gov

[4] J. Rice; *Seaweb Acoustic Communication and Navigation Networks*; Proc. International conference on Underwater Acoustic Measurements: Technologies & Results, Heraklion, Crete, Greece, 2005.

[5] F. Warthman, *Delay-Tolerant Networks (DTNs): A Tutorial v1.1*, Mar 2003. http://www.dtnrg.org/docs/tutorials/warthman-1.1.pdf

[6] V. Cerf, et al.; *Delay-Tolerant Network Architecture*; Internet draft, draft-irtf-dtnrg-arch; March 2006, work-in-progress.

[7] K. Scott, and S. Burleigh; *Bundle Protocol Specification*; Internet draft, draft-irtf-dtnrg-bundle-spec; May 2006; work-in-progress.

[8] M. Ramadas; *Licklider Transmission Protocol – Specification*; Internet draft, draft-irtf-dtnrg-ltp, March 2006, work-in-progress.

[9] A. Lindgren, A. Doria, E. Davies, S. Grasic; *Probabilistic Routing Protocol for Intermittently Connected Networks*, draft-irtf-dtnrg-prophet-08, October 2010.

[10] Z. Zhang; *Routing in Intermittently Connected Mobile Ad Hoc Networks and Delay Tolerant Networks: Overview and Challenges*; IEEE Communications Surveys and Tutorials. January 2006.

[11] I. Psaras, Ll. Wood, R. Tazafolli; *Delay-/Disruption-Tolerant Networking: State of the Art and Future Challenges*; http://www.ee.ucl.ac.uk/~uceeips/dtn-srv-ipsaras.pdf

[12] S. Jain, K. Fall, R. Patra, *Routing in a Delay Tolerant Networking,* SIGCOMM, Aug/Sep 2004.

[13] T. Spyropoulos, K. Psounis, C. S. Raghavendra; *Spray and wait: an efficient routing scheme for intermittently connected mobile networks*; WDTN '05, Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, New York, USA, 2005.

[14] A. Balasubramanian, B. Levine, A. Venkataramani; *DTN routing as a resource allocation problem;* SIGCOMM '07, Proceedings of the 2007 conference on Applications, technologies, architectures and protocols for computer communications; New York, USA, 2007.

[15] J. Burgess, B. Gallagher, D. Jensen, B. N. Levine; *MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks*; INFOCOM 2006, 25th IEEE International Conference on Computer Communications. April 2006, pp 1 – 11.

[16] A. Lindgren, A. Doria, O. Schelen*, Probabilistic Routing in Intermittently Connected Networks,* In Proceedings of the The First International Workshop on Service Assurance with Partial and Intermittent Resources (SAPIR 2004), August 2004, Fortaleza, Brazil.

[17] A. Lindgren, K. Phanse; *Evaluation of Queuing Policies and Forwarding Strategies for Routing in Intermittently Connected Networks*. In: Proceedings of Comsware '06. First International Conference on Communication System Software and Middleware. (2006) 1 – 10.

[18] J. Ansari, D. Pankin, P. Mähönen; *Radio-triggered wake-ups with addressing capabilities for extremely low power sensor network applications*; PIMRC 2009, IEEE 19th International Symposium on Personal, Indoor, and Mobile Radio Communications. September 2008. Pp 1 – 5.

[19] S. Y. Choi, J. Turner; *Configuring sessions in programmable networks with capacity constraints*; ICC '03, IEEE International Conference on Communications; pp 823 – 829, vol. 2; May 2003.

[20] http://w3.bcn.es/fitxers/mobilitat/pacte/noumodeldexarxadautobusos.312.pdf

[21] W. Zhao, Y. Chen, M. Ammar, M. Corner, B. Levine, E. Zegura; *Capacity enhancement using throwboxes in DTNs*; 2006 IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS); Vancouver, USA; 2006; pp 31 – 40.

[22] M. Ibrahim, P. Nain, I. Carreras; *Analysis of relay protocols for throwbox-equipped DTN*; WiOPTt 2009, 7th International Symposium on Modely and Optimization in Mobile, Ad Hoc, and Wireless Networks; Seoul, June 2009; pp 1 – 9.

[23] N. Banerjee, M. D. Corner, B. Levine; *Design and field experimentation of an energy-efficient architecture for DTN throwboxes*; IEEE/ACM Transactions on Networking, Volume 18, Issue 2, April 2010.

[24] S. Farrell, V. Cahill; *Delay- and Disruption-Tolerant Networking*; Artech House Publishers; September 2006.

[25] T. Spyropoulos, T. Turletti, K. Obraczka; *Routing in Delay-Tolerant Networks Comprising Heterogeneous Node Populations*; IEEE Transactions on Mobile Computing; August 2009; Volume 8, Issue 8, pp 1132 – 1147.

[26] M. Musolesi, C. Mascolo; *CAR: Context-Aware Adaptive Routing for Delay-Tolerant Mobile Networks;* IEEE Transactions on Mobile Computing 8; 2009; pp 246 – 260.

[27] K. Fall, S. Farrel; *DTN: An Architectural Retrospective. Selected Areas in Communications*, IEEE Journal on 26; 2008; 828 – 836.

[28] L. Wood, W. Eddy, P. Holliday; *A Bundle of Problems*; IEEE Aerospace Conference; 2009; pp 1 – 17.

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

[30] Y. Wang, H. Wu; *DFT-MSN: The Delay/Fault-Tolerant Mobile Sensor Network for Pervasive Information Gathering*; IEEE INFOCOM '06; 2006.

[31] T. Matsuda, T. Takine; *(p; q)-Epidemic Routing for Sparsely Populated Mobile Ad Hoc Networks;* Selected Areas in Communications, IEEE Journal on 26; 2008; pp 783 – 793.

[32] K. Fall; *A Delay-Tolerant Network Architecture for Challenged Internets*; Proceedings of SIGCOMM '03; New York, USA; 2003; 27 – 34.

[33] J. Scott, P. Hui, J. Crowcroft, C. Diot; *Haggle: A Networking Architecture Designed Around Mobile Users*; Proceedings of the Third Annual IFIP Conference on Wireless On-demand Network Systems and Services (WONS 2006), Invited paper; Les Menuires, France; 2006.

[34] J. Crowcroft, E. Yoneki, P. Hui, T. Henderson; *Promoting Tolerance for Delay Tolerant Network Research*; SIGCOMM Comput. Commun. Rev. 38; 2008; pp 63 – 68.

[35] I. Psaras, N. Wang, R. Tafazolli; *Six years since first DTN papers. Is there a clear target?;* 1st Extreme Workshop on Communication (ExtremeCom 2009); Laponia, Sweden; 2009.

[36] K.A. Harras, K.C. Almeroth; *Transport Layer Issues in Delay Tolerant Mobile Networks*; Proceedings of IFIP Networking; (2006).

[37] W. Zhao, M. Ammar, E. Zegura*; A message ferrying approach for data delivery in sparse mobile ad hoc networks*; Proceedings of MobiHoc '04; New York, USA;2004; pp 187 – 198.

[38] T. Small, Z. J. Haas; *Resource and performance tradeoffs in delay-tolerant wireless networks*; Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking (WDTN '05); New York, USA; 2005; pp 260 – 267.