UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Opportunistic Network Modelling and Algorithms

Jorge Visca

Postgraduate program PEDECIBA Informática
Instituto de Computación - Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay
October of 2023

UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Opportunistic Network Modelling and Algorithms

Jorge Visca

Doctorate's Thesis submitted to the Postgraduate Program PEDECIBA Informática, Instituto de Computación - Facultad de Ingeniería of the Universidad de la República, as part of the necessary requirements for obtaining the title of Doctor in PEDECIBA Informática.

Director:
Dr. Javier Baliosian

Academic director:
Dr. Javier Baliosian

Montevideo – Uruguay
October of 2023

Visca, Jorge

Opportunistic Network Modelling and Algorithms / Jorge Visca. - Montevideo: Universidad de la República, Instituto de Computación - Facultad de Ingeniería, 2023.

IX, 71 p. 29, 7cm.

Director:

Javier Baliosian

Academic director:

Javier Baliosian

Tesis de Doctorado – Universidad de la República, Program PEDECIBA Informática, 2023.

Bibliography: p. 22 – 25.

1. Redes Oportunistas, 2. Redes Tolerantes a Retrasos - DTN, 3. Modelos Estocásticos. I. Baliosian, Javier, . II. Universidad de la República, Postgraduate Program PEDECIBA Informática. III. Title

MEMBERS OF THE THESIS DEFENSE COURT

Ph.D. Prof. Luciana Bezerra Arantes

Ph.D. Prof. Esteban Mocskos

Ph.D. Prof. Alejandro Ribeiro

Ph.D. Prof. Gustavo Betarte

Ph.D. Prof. Álvaro Martín

Montevideo – Uruguay
October of 2023

RESUMEN

Las Redes Oportunistas son redes capaces de operar en algunos de los entornos más hostiles imaginables para una red de datos, cuando casi nada se sabe con certeza de la infraestructura. Los nodos pueden estarse moviendo saliendo de alcance continuamente, apagarse imprevistamente, o sufrir de interferencias; los usuarios de la red pueden querer acceder a datos sin saber dónde están alojados, o querer transmitirle a un nodo que está inaccesible por un tiempo indeterminado.

Esto causa que las Redes Oportunistas estén condicionadas por procesos estocásticos complejos. Son sistemas ciber-físicos, cuyo comportamiento resulta de la interacción entre sistemas de software y la realidad física. Por lo tanto, uno de los retos principales para los algoritmos de enrutamiento para Redes Oportunistas es gestionar esta aleatoriedad e imprevisibilidad.

Lo anterior resulta en que las Redes Oportunistas son difíciles de modelar y caracterizar. En esto radica uno de los mayores retos a la hora de diseñar y desplegar algoritmos de enrutamiento efectivos.

En este trabajo proponemos métodos de modelado de Redes Oportunistas y aplicamos esos modelos para construir algoritmos de enrutamiento eficientes y flexibles.

Desarrollamos dos clases de modelos. El primero es un modelo analítico-deductivo, que permite extraer conclusiones generales para redes ideales. El segundo modelo está basado en datos, y captura la dinámica y comportamientos de una red real, representándolos en una estructura de datos novedosa. Este modelo se aplica para desarrollar algoritmos basados en Aprendizaje Automático, siendo el resultado eficiente y generalizable.

Palabras clave:
Redes Oportunistas, Redes Tolerantes a Retrasos - DTN, Modelos Estocásticos.

ABSTRACT

Opportunistic Networking is a technology for sustaining a data network when the supporting infrastructure is as challenging as it can be. An Opportunistic Network can operate when almost nothing is known about the nodes and the environment: network nodes might be moving in unpredictable ways, be shut down or off range for long periods, or have severely limited computing resources; user applications might want to access data whose placement is unknown, or be producing data while there are no other nodes close to who transmit it to; and nodes might be subjected to unpredictable radio interference.

Because of this, Opportunistic Networks are subject to strongly stochastic behaviors. They also are cyber-physical systems whose behavior emerges from the interaction of software and the real world. The main challenge for Opportunistic Network routing algorithms is handling this randomness and unpredictability.

As a result of the above points, Opportunistic Networks are complex to describe and characterize. This difficulty in modeling the network is one of the main challenges when designing and deploying practical routing algorithms.

In this document, we compile a set of incremental works that study the underlying structures of Opportunistic Networks and propose modeling methods for analyzing their behavior. They also show how these models can be capitalized to build efficient and flexible algorithms.

We develop two classes of models. The first is an analytical one, built by deduction from axioms. It allows us to extract general conclusions on the behavior of an idealized network. The second is a data-driven model that builds a representation of a real network, capturing real-life behavior in a novel data structure. We show how this model can support Machine Learning algorithms to solve the routing problem efficiently and in a generalizable way.

Keywords:
Opportunistic Networks, Delay Tolerant Networks - DTN, Stochastic Models.

Acronyms

List of acronyms

CBR Content-Based Routing

CDN Content Delivery Network

DTN Delay Tolerant Network

ML Machine Learning

NN Neural Network

ONM Opportunistic Network Model

OppNet Opportunistic Network

P2P Peer-to-Peer Network

QoS Quality of Service

RL Reinforcement Learning

VANET Vehicular Ad-Hoc Network

Contents

Chapter 1

Introducction

Computer Networks are an essential part of modern life. At the same time, Computer Networks came to be synonymous with the Internet, and we became accustomed to expecting certain behavior: almost complete and redundant coverage in inhabited areas; spotty or no coverage in remote areas; huge bandwidths and 24×7 availability through infrastructure maintained and deployed by competent and resource-rich operators.

However, there are other computer networks. Networks that depend on cheap devices that move around, their number or position unknown, and are out of range for long periods; where datalinks are slower than a modem dial-up from 25 years ago, and every transmission is counted because it depletes the batteries. These networks are known as Opportunistic Networks (OppNets).

1.1. Motivation

The basic concept behind *opportunism* in Networking is the ability to use resources for which there is no deterministic knowledge about their properties or availability. This ability could be useful in networks where the exact placement of a file is not known beforehand, as in some Content Delivery Networks (CDNs); short-lived networks formed by mobile devices when meeting, like Vehicular Ad-Hoc Networks (VANETs); or networks that must deliver data using sporadic low-quality encounters between routers, like in Delay Tolerant Networks (DTNs).

This work concentrates on the scenario where the primary source of stochasticity is the movement of the nodes. In these networks, it can be necessary for

nodes to carry data in local storage as they move, waiting for an encounter to happen. Once an encounter happens, nodes must determine what data to transfer to the new carrier candidate.

As usual in networks, the behavior of a device can be described by two tasks. The first is *Routing*, the process by which nodes acquire information on the behavior of the network, information that will be used to make better decisions for the second task, *forwarding*. Forwarding determines what information should be copied and stored locally during an encounter.

Because of the uncertainty regarding the network topology and status, specific routing algorithms are developed. These algorithms vary significantly in complexity and behavior: from very simple, like flooding copies of information in the network [32], to highly sophisticated, like using Machine Learning techniques to predict future encounters.

This variety of techniques and algorithms makes deploying OppNets a daunting task. These networks can support a broad spectrum of applications with very different requirements. Selecting a good algorithm depends on the precise characterization of the network and a deep understanding of the properties of the different algorithms. At the same time, these algorithms tend to depend on the correct configuration of multiple parameters to adapt to a particular network and application.

The guiding observation for this thesis is that an OppNet is a Cyber-physical system [19], and its behavior emerges from the interaction between node dynamics and opportunistic algorithms. This emerging behavior makes these systems complex to model, partly because the tools available to model the different aspects of the system are very different. For example, the movement dynamics of the network nodes can be described with tools such as differential equations and stochastic processes. At the same time, the algorithmic behavior is usually analyzed with tools like computational complexity theory or finite state transducers.

1.2. Objectives

The common purpose of the works compiled in this thesis is to provide methods for analyzing and designing OppNets as a whole, combining network dynamics modeling and opportunistic algorithms in a single framework. The final objective is to design better performing OppNet algorithms, which are

supported by well-understood modeling techniques.

That main purpose is further decomposed into two specific objectives. These are outlined as follows:

Develop modeling tools: The different modeling approaches are discussed, both analytical and data-driven. In particular, models that integrate mobility and algorithmic behavior are proposed.

Design protocols that capitalize the new models: The proposed models offer a better representation of the OppNets as a system and thus are a good substrate for routing algorithms. A more informative representation of the networks allows more effective decision-making by the network agents.

1.3. Contribution

The contribution of this thesis can be described in two stages. The first consist of a set of models that can be applied to characterize an OppNet; the second consists of a set of opportunist routing algorithms that capitalize on said models. These contributions can be outlined as follows:

- An analytical model based on the deduction of the network's behavior from its fundamental properties.
- A data-based or empirical representation that allows capturing both the dynamics of the network and the effect of the algorithmic decision-making in a single data structure.
- A Machine Learning approach to opportunistic routing that uses the data-driven model to provide improved performance. Two alternatives are presented, offline and online. The former is based on processing recorded network traces; the latter learns through the network's lifetime.

1.4. Structure of document

Besides the articles compiled in this thesis, this document lays out an introductory text to frame the work presented in them. Chapter 2 presents the concept of Opportunistic Networking and its main research challenges. Then,

in Chapter 3, each article composing this thesis is summarized and placed in context. Finally, Chapter 4 provides some global concluding remarks.

Chapter 2

Opportunistic Networks

This Chapter presents a brief introduction to Opportunistic Networking as a research area, including the idea that, as a concept, it has a deep history and has been studied under multiple names and in different scenarios.

2.1. History

One of the concepts that separated Computer Networking from phone networks was connectionless packet switching, which appeared in the sixties. The basic idea is that data is split into discrete data units or packets, and each packet traverses the network independently. It was expected that a best-effort packet-switching network could quickly react to topological changes and minimize data losses. This concept led to the IP protocol and, thus, the modern Internet [13].

In the seventies, another scenario was explored with military applications. Packet Radio [11] was a technology to route data over radio links between mobile terminals without depending on some underlying infrastructure. Conceptually it was very similar to what is known today as a *wireless mesh networks*.

These examples can be described as following a fail-recovery approach based on restoring end-to-end connectivity as quickly as possible. The reason is that the typical network nodes (routers) have very little storage, and if a packet can not be dispatched fast enough, in the order of fractions of a second, it will have to be dropped, producing a loss. As a result, packet-switching networks are best adapted to networks where multiple possible paths exist.

However, in the pre-Internet era, other types of networks existed. When

long-distance links were established via dial-up over phone lines, local networks were isolated most of the time. Local networks would periodically establish short-lived links with an upstream network, usually at night when phone charges were lower. Then, these networks used application-layer protocols such as FidoNet or UUCP to upload and download pending data. A surviving example of an algorithm with a similar design is SMTP, the basis for the e-mail service. This functionality already can be seen as an OppNet in the modern sense, though the term was not used back then (they were called *Store-and-forward* networks).

OppNets were launched as a specific research topic in 2003 under the name Delay Tolerant Network (DTN) [10]. The proposal introduced the concept of *challenged network* as well as a specific architecture for interoperating networks over them.

A challenged network is a computer network that suffers from one or more of the following problems:

High Latency and Low Data Rate In structured IP networks, Gb/s speeds are usual, as are Mb/s in wireless links. However, there are applications where Kb/s or lower rates are usual, as are highly asymmetric or unidirectional links.

Disconnection There are networks where disconnection is not an abnormal state caused by a failure but the natural state where most of the time is spent. That might be because of mobility and range considerations or low-duty cycle operation (discussed later).

Long Queuing Times In most conventional networks queuing times are the main factor in the total network latency, but even then, they are measured in milliseconds. Networks exist where the total latency and queuing times are measured in hours or days.

Low Duty Cycle Operation. In many battery-powered applications, wireless transmissions are a significant part of power consumption. Nodes might try to keep their wireless interfaces powered down as long as possible.

Limited Resources Many applications depend on low-cost devices, which leads to low computational resources such as computing power, onboard storage, or wireless bandwidth.

The DTN architecture proposed to handle these networks will be described in Section 2.2.3.

In recent years computing platforms experienced a significant cost reduction, with a new class of battery-powered, wirelessly interfaced, microprocessor or microcontroller-based systems, which led to an interest in deploying them very widely. New use cases and deployment scenarios, such as Sensor Networks, the Internet of Things, and Smart Cities, were defined. Many of these deployments extended outside the umbrella of conventional networks, which led to an explosion in research on new supporting network technologies.

2.2. Applications

This Chapter presents some of the prominent use cases for OppNets. They cover a broad spectrum of requirements and functionality.

2.2.1. Sensor Networks

A Sensor Network is composed of widely distributed low-cost nodes that collect some kind of data. The nodes can be fixed or mobile and are frequently battery-powered. The network's task is to collect data from these sensors and transmit it to a collection point to be stored and processed [2].

When the nodes are fixed, opportunistic behavior can emerge from a low-duty cycle operation. Other times, the data collection is managed by a separate class of mobile devices moving through the deployment area and encountering the sensor nodes. An example of this use case is DEMOS [4], a network for collecting environmental data from sensors using school children's OLPC-XO laptops as data carriers.

In other networks, the sensor nodes themselves are mobile. In this case, the nodes may carry data packets and handle them over to other nodes until they reach the destination through several hops. These networks are frequently called *carry and forward* networks because nodes are expected to keep data in local storage for extended periods while moving around and waiting for a forwarding opportunity. This use case is the basis for some of the works collected in this thesis (see Appendices 4 and 5).

2.2.2. LTE *Device2Device* offloading

Opportunism can be advantageous even when a conventional network is available. Device2Device, or D2D, is a capability proposed for LTE cellular networks that would allow data exchange between close-by devices, bypassing the cellular base infrastructure [3, 12]. This communication can be either *in-band*, using the same radio transmitter used for cellular communication, or *out-of-band*, using a secondary interface such as WLAN or Bluetooth. This capability promises to reduce power consumption and network load for data-heavy applications such as OS updates or video streaming. This scenario is discussed for this thesis in Appendix 2.

2.2.3. Delay Tolerant Networks

A Delay Tolerant Network (DTN), sometimes called Disruption Tolerant Network, is a message-based overlay network that uses challenged networks to communicate traditional networks. The network's message is called a *bundle* and has been described through several RFCs [5] as the network itself [24]. The design is extensive, specifying features such as addressing schemes, message fragmentation, time synchronization, and security management. Routing and Forwarding are not specified and are left to be developed following the requirements of a particular use case. This omission is justified because the DTN framework is intended to be used in different environments, from underwater to interplanetary communications [22]. Nevertheless, many design choices are made with the understanding that the main task is to interoperate IP networks.

Sometimes the terms DTN and OppNet are used interchangeably, though this is incorrect. DTN is a specific architecture that has a place reserved for routing and forwarding behavior in its design. Research in routing for OppNets can be done in the context of the DTN framework, but this is not mandatory. Some of the DTNs design decisions, like how nodes are identified or what is the exact structure of a network message, impose artificial restrictions for some applications. Thus, basic research in routing algorithmics is better done outside the DTN general design.

2.2.4. Content Delivery Networks

The problem of routing data to a receiver can be seen from the other side: a receiver might want to access some data, while the number of copies of the data and where they are placed might be unknown to it. This scenario is typical of Content Delivery Networks (CDNs). Multiple technologies are involved in sustaining CDNs, such as P2P Networking and Content-Based Routing (CBR). Some CDNs also have a mobility component. This scenario is the base for one of this thesis' works; see Appendix 2.

2.2.5. Fleet coordination

Many vehicle fleets, crewed or uncrewed, can capitalize on the timely exchange of information between the participants. A group of robots might exchange their local views while building a global map of an area to be explored. Robots operating warehouse logistics might coordinate exchanging commands and assigning tasks. Cars moving down a highway might communicate between them to warn of dangerous situations.

The ability to communicate moving robots flexibly and robustly is critical for many robotic deployments, like drone swarms, agricultural service robots, or Search-and-Rescue robots for disaster mitigation.

Many of these applications impose extreme stress on the communication systems due to hostile radio environments, limited power supply, the coexistence of traffic of very different QoS requirements, and fast mobility. The works presented in Appendices 4 and 5 are of interest to this application.

2.3. Challenges

Opportunistic Networks are Cyber-physical systems [19]. That is, systems where there is a software component and a real-world component, and both components interact to produce the system's behavior. The interaction between the computational and physical components may be intentional, like in a control system where a software controller impacts the physical world through actuators and senses physical properties through sensors. The interaction can also be accidental: the physical world can create interference in communications systems or degrades the performance of actuators and sensors as they wear out.

2.3.1. Modeling

Different laws guide the computational and physical components: the physical world is driven by physics, mechanics, and thermodynamics. The software's behavior is described by concepts such as logic, algorithmic complexity, and computability. Following [20], we outline the main differences between the computational and physical worlds as follows:

Static vs. dynamic: Unlike in a computer program, things in the real world change *by themselves*, outside the control of the system's owner.

Real Time: physical processes run in an external, independent, thermodynamic time that can not be halted or slowed down.

Observability: not all real-world's information can be accessed to build an internal world representation. There is useful information that can not be acquired.

Discrete vs. Continuous Digital computing systems have a discrete representation of state and time. Physical systems tend to be continuous.

Deterministic vs. Stochastic: Computing systems are fully deterministic: given an internal state and an input, the resulting state is always the same. There are many stochastic processes in the physical world.

The stochastic behavior is of particular interest when working with OppNets. It is the main challenge, and the major design decisions are about how to handle this randomness. There are two primary sources of stochasticity in OppNets.

The first is link availability, which is typically due to node movement and radio interference. Though there are some OppNets composed of vehicles with precisely known trajectories, such as spaceships and satellites [22], most vehicles move with a degree of randomness. Nevertheless, most mobile nodes are not entirely random and follow some laws. People move following specific patterns through a city, and their encounters and popularity follow laws studied by sociologists and urbanists [18, 23]; Public transport buses move in circuits with approximate schedules; Reindeer move in herds following seasonal environmental changes [9], and so on.

The user's data production is the second source of stochasticity. In most computer networks, predicting users' data flows is very difficult. That is especially true when the network is seen as a *service* for applications, so the inner logic of the network's users is unknown.

These OppNets properties result in a set of hurdles when modeling their structure and behavior:

1. The system has a strong stochastic component, which limits the modeling tools available.

2. The behavior of the system depends on the behavior of both a computer and physical systems, which are usually modeled with different tools.

3. The system has many components and processes, which results in a high number of parameters.

The resulting models tend to be complex and hard to use [6].

2.3.2. Evaluating OppNets

Another effect is that it is challenging to compare different algorithms. There are two main difficulties.

In the first place, it is usual that solutions can not be tested on real deployments and must be verified through simulations. The simulations tend to be computationally expensive because the systems are very complex: There are multiple nodes, each running a complex networking stack; wireless propagation and interference depend on very complex equations; the system must be simulated for long runs, and because of stochasticity, the experiment must be repeated multiple times. As a result, most research is done on small to moderate-size networks [14].

The second difficulty is that system's performance is very sensitive to the deployment scenario, namely to the movement pattern of the nodes. As a result, different works, usually interested in different deployments, are hard to compare. At the same time, traces for real networks usable as good reference scenarios are scarce, though there is some work in this direction, like trace repositories [1].

2.3.3. Performance metrics

The final issue is measuring the performance of an OppNet. Its primary performance metric is the delivery rate or the proportion of emitted messages that reach a destination. Sadly, this metric is conditioned by network properties other than the routing algorithms. Namely, it depends on the mobility scenario where the algorithm was executed and frequently on the data flows present in the network. That makes it difficult to compare different network instances. Another issue with the *delivery rate* metric is that it does not consider the *fairness* or how the delivery rate varies across different emitters in the network. Fair sharing of the network's resources is a complex issue, rarely attacked in OppNet research [31].

Besides *how many* messages arrive, another important question is *how much it takes* for them to arrive. As mentioned above, OppNets are characterized by very long delivery times, and the distribution of latencies tends to be complex as messages following different opportunistic paths spend different amounts of time in the network. Another difficulty is that the latency interacts with the delivery rate: a simple method to reduce the latency is to drop messages that spend too much time in the network at the cost of reducing the delivery rate, as messages may be dropped before reaching the destination.

Finally, an important property is the *network overhead*. This overhead can come from messages following suboptimal trajectories or algorithms that produce multiple copies of messages to increase the delivery rate. Also, some algorithms exchange routing data or add metadata to messages to support routing decisions. The network overhead is important because, as the medium is usually wireless and shared, excessive transmissions degrade the performance of all the nodes in range, reducing the available bandwidth and increasing the power consumption.

Chapter 3

Articles Supporting this Ph.D. Thesis

The work presented in this Thesis is a continuation of the work done in a Master Thesis [26], in the context of a research project for developing a Sensor Network with low-cost environmental sensors [4]. One of the author's contributions was RON, a Publish-Subscribe OppNet routing algorithm used to route messages between sensors, collection points, and mobile carrier devices. The experience gained in this work motivated the continued exploration of routing techniques for OppNets.

This Chapter presents the publications of this Ph.D. thesis resulting from that exploration.

3.1. Articles Compiled in this Ph.D. Thesis

During the development of RON, several pathological mobility cases were identified [25]. Under some conditions, these cases affected an important class of algorithms called *gossiping-based* [8]. It was found that under some frequently used buffer management policies, some data flows can starve others. The basic phenomenon is that nodes prefer to carry "easy" messages to the detriment of "difficult" ones, even if the node is of critical importance for their delivery. The result was considerable disparities in the service provided to different clients, with some data flows collapsing completely.

[31] Jorge Visca et al. «Path sampling, a robust alternative to gossiping for opportunistic network routing». In: *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. 2016, pp. 1–8. DOI: 10.1109/WiMOB.2016.7763244

This analysis led to the creation of an alternative routing mechanism, called *Path Sampling* (this article is included in Apendix 1). The algorithm is of a class called *Publish-Subscribe*, where receiver nodes announce an interest in receiving some class of data (through Subscription messages), and then emitters serve these requests. This addressing scheme is a generalization of the usual *Destination-Routing*, where emitters specify the target node of messages. The main idea of the algorithm is to explore the possible message propagation paths while distributing the subscriptions and then use the best paths found when serving these subscriptions. Finally, nodes distribute buffer space among the participating data flows, independently of their position within a data flow. By doing this, they obviate the pathological buffer assignation as seen in Epidemic algorithms.

Besides Sensor Networks, other use cases for OppNets were explored. As mentioned in Section 2.2.2, direct delivery between mobile devices is a promising capability for next-generation LTE cellular networks. To understand its power, imagine the following scenario. Suppose a popular event, such as a sporting event or a political broadcast, is streamed to many users. All the people within a building, stadium, or train will get the same content over the air from a particular cellular base. In high-density areas, this can impose a massive load on the cell operator's infrastructure and available cellular bandwidth. Now, imagine that phones could download data from close-by neighbors that have already downloaded it. Doing this would have multiple advantages, like a power consumption reduction for the participating mobile devices, lesser load on the cell, and improved coverage [16].

[30] Jorge Visca et al. «Opportunistic media sharing for mobile networks». In: *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*. 2016, pp. 799–803. DOI: 10.1109/NOMS.2016.7502902

In this work (included in Appendix 2), an OppNet protocol is proposed for supporting opportunistic media sharing between mobile devices in a D2D scenario.

This protocol, called FLOP, is based on Path Sampling and takes advantage of two of its capabilities.

First, it uses the Publish-Subscribe infrastructure to request chunks of video content from other devices in the OppNet. This very flexible mechanism allows operating a network without a centralized registry of participants. Then, opportunistic routing allows it to operate in the evolving network topology formed by people moving around. It can also serve users while they are disconnected from the cellular network, as sometimes happens in dense urban areas when moving between cells or the cellular network is overloaded.

The works presented up to this moment follow a traditional approach in OppNet research [14, 21]. It consists of proposing an algorithm, defining a network scenario in a simulator, and comparing its performance against other well-known algorithms. This approach has drawbacks. Due to complex network dynamics, OppNet algorithms tend to depend on many configuration parameters that profoundly impact the algorithm's behavior. These parameters must be calibrated for the specific scenario, usually through extensive simulation. This step should be (but rarely is) performed for all the algorithms used for comparison. The scenarios themselves are highly configurable.

A difficulty is that scenario and algorithm parameters are usually uncoupled: it is hard to predict the optimal parameters for given network dynamics. The result is that the algorithms are hard to deploy on unknown networks effectively, and the simulated evaluations are of limited generalizing value. That is especially true for big networks, which are difficult to simulate [14].

The root of the inability to generalize is the lack of a model of the algorithm-network interaction. Having a model implies the ability to make predictions, something lacking in OppNet research.

Following [17], there are two main approaches for building models, the *Deductive method of theoretical or axiomatic modeling*, and the *Empirical method of experimental modeling*. The Deductive method builds from the bottom up, defining some fundamental axioms the system must respect and then applying mathematical manipulations to generate laws that predict the behavior of the real system. The power of this approach is that mathematical models are rigorous in tracking their applicability, so as long as the hypotheses and analytical tool's preconditions are respected, the conclusions are known to be valid and applicable. In this sense, the model is *general*.

[29] Jorge Visca, Matías Richart, and Javier Baliosian. «Stochastic Models for Opportunistic Networks». In: *2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW)*. 2019, pp. 1–6. DOI: 10.1109/WCNCW.2019.8902544

This article (included in Apendix 3) proposes an analytical model for OppNets. It is based on classic epidemiological models, making an analogy between contagious diseases and data: forwarding data between agents is like transmitting a disease. In this work, epidemiological models are extended to handle the main difference with OppNets: in a computer network, multiple messages compete for limited buffer space on the nodes, something that has no equivalent in biological diseases.

The model finds the expected lifetime and reach of a message in the network. The importance of an analytical model is that it gives a general insight into the system's response to the different network parameters, such as mobility attributes, node density, or traffic intensity.

A problem with analytical models is their limited expressive power to capture complex real-life processes. For example, two mathematical tool sets are applied in the previous work. First, a differential-equations-driven model is explored. In it, the magnitudes are treated as continuous and differentiable, so it is only applicable to very big networks to approximate their discrete nature. The differential model also ignores all stochastic phenomena, being completely deterministic. The second model uses Markov Processes, which capture the stochastic behavior but depends on the Markov property. This restriction limits the model's applicability, as most real networks are not memoryless: node movement is frequently correlated, as is data transmission.

The result is that many complex behaviors of the real world must be ignored. The problem is that some of these behaviors -such as the correlations in the node movements- may be critical for the network's behavior.

The limitations described in the previous work led to the use of Empirical or experimental models. These are models derived top-down from reality, trying to infer laws that can adjust the observed behavior.

[27] Jorge Visca and Javier Baliosian. «A Model for Route Learning in Opportunistic Networks». In: *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. May 2022, pp. 1–4. DOI: 10.1109/NOMS54207.2022.9789826

The presented article, included in Appendix 4, proposes a novel model for OppNets called Opportunistic Network Models (ONMs). It is based on *Spatio-temporal Graphs* [7] and captures the most relevant data for describing an OppNet. It expresses the temporal evolution of the encounters in the network and can naturally represent the concepts of data routing and forwarding.

The model allows us to describe an OppNet precisely, but its main power is that it allows exploring the solution space of data routing in the network. The model has straightforward semantics and uses an abstraction with many tools available: the model is a simple directed graph, though not a *connectivity graph* as usual in conventional networks. This simplicity compares favorably with other OppNet models [6], which depend on cumbersome and limiting abstractions.

As a result, the ONM is a practical support for applying ML techniques to derive good routing decisions. In effect, it supports a proposed routing algorithm called *BOW*. In this work, the ONM is built from recorded traces of a real network. This model's optimal routing behavior is computed by applying exact methods. Then, several ML techniques are used to train a predictor function that would make good decisions in other network instances.

Notice that this approach differs from the one used with Path Sampling and FLOP. There, the algorithm was defined independently from a network scenario. In this case, the network model and routing algorithm are linked. First, a network model is provided, and then the algorithm learns its optimal routing decisions. Because the model is generated and the learning is performed offline from recorded traces, this method can use sophisticated and computationally expensive ML methods such as Neural Networks (NNs). Also, the resulting policy does not depend on nodes exchanging routing data as conventional algorithms do, reducing the load on the network. As a disadvantage, the algorithm's learning is "frozen" at the point the model was captured and can not adapt to changes in the network dynamics.

[28] Jorge Visca and Javier Baliosian. «rl4dtn: Q-Learning for Opportunistic Networks». In: *Future Internet* 14.12 (2022). ISSN: 1999-5903. DOI: 10.3390/fi14120348. URL: https://www.mdpi.com/1999-5903/14/12/348

This article is included in Apendix 5 and proposes a different approach. Instead of building the ONM from recorded traces, it is built and maintained online during the network's lifetime. This management is performed in a distributed fashion by the network nodes themselves, where nodes build a local view of the whole model. At the same time, Reinforcement Learnings (RLs) techniques are used to learn the optimum routing decisions. The main difference of this work with other RL algorithms for OppNets (for an overview, see [15]) is that it learns over the ONM, which captures temporal information. Thus, the learning system is fed information on the timing of node encounters, allowing the system to make more informed decisions.

3.2. Statement of Authorship

The author of this thesis is the main author of the works presented in this chapter. A detailed description of the author's contribution to each piece can be found in the corresponding appendices.

None of the works in this compendium were included in other compendium-thesis documents, nor will they be included in such type of thesis in the future.

Chapter 4

Conclusions

The main goal of this Thesis was twofold. First, to produce new models capable of capturing the complex behavior of Opportunistic Networks. Secondly, use those models to design better routing algorithms.

4.1. Concluding remarks

In this Thesis, two models of very different natures and serving distinct purposes are proposed. They are not competing but complementary and showcase the advantages and limitations of both approaches.

The first model is analytical, derived from first principles, and describes the fundamental properties of an ideal network. It provides clear relationships between the parameters and preconditions. As a result, it gives a general insight into the network's behavior.

As a downside, this model is highly restrictive on the network properties under which the characterization is exact. Most real networks do not satisfy those preconditions nor behave as the model predicts.

However, the qualitative phenomena described in that analytical model helped us to understand the real behavior of an OppNet better, even when the predictions are not exact; understanding where the real networks diverge from the models is an important result in itself because it can lead to identifying critical properties of networks.

The data-driven model is proposed to handle the problem of describing a specific network. It captures its dynamics, representing all the processes (movement and transmissions) that affect its functioning as a data network.

The result is a single data structure abstracting the complete network's behavior through its lifetime. An advantage of this model, compared with previous work, is the simplicity of its representation, being just a labeled directed graph. Combined with straightforward semantics, this allows the application of powerful and well-known tools.

Another advantage of this data-driven model is that it is well-suited to support exploring the network's behavior algorithmically. This allowed the application of Machine Learning methods more effectively than in previous works. Learning temporal behavior is a particularly complex task to implement. Firstly, because it is critical for the algorithm's effectiveness, and secondly, because this information, by definition, is not immediately available and must be stored and represented somehow. The proposed model includes a method for codifying temporal data in a representation readily available for the learning mechanisms.

An interesting consequence of the proposed Machine Learning approach is that it is built over a network model derived from the real network. Thus, it does not depend on pre-deployment tuning, as usual in conventional algorithms. In a way, the tuning step is contained within the algorithm's training when the decision-making code is exposed to the details of the particular network it will run on. That dramatically simplifies the deployment of the network, as the usually labor-intensive and artisanal parameter tuning gets automated. The parameters controlling the learning process remain, but those are usually more robust and better understood.

4.2. Open Research Lines and Future Work

Three main research lines were identified as potential future work.

The first research line is aimed at improving the analytical model. Preliminary work has already been done to extract more information from the model, such as a deduction of the effective delivery rate of the described network. It will be helpful to analyze further the conditions under which the model is applicable and try to soften the dependency on the Markov property of the underlying network as much as possible.

The second area is to understand better the properties of the data-driven ONM model and how it adapts to Machine Learning processing. In particular, Graph Neural Networks are a promising technology that could adapt well to

the underlying model.

Finally, a variation of the data model is being developed. It is similar to the data-driven ONM model, but instead of representing a single instance of the network, where edges capture the observed behavior, in this representation, edges are probabilities. For this, it splits the time in time windows, as the rl4dtn algorithm does. As edges have associated probabilities, the label's product along a graph walk is the probability that a message would reach the destination following said path. The total probability of reaching a destination from a starting node through all possible paths is a well-known concept, the *reliability network*. Modeling a OppNet as a reliability network is a promising approach.

References

[1] *A Community Resource for Archiving Wireless Data At Dartmouth*. URL: https://crawdad.org.

[2] Tekenate Amah et al. «Preparing opportunistic networks for smart cities: Collecting sensed data with minimal knowledge». In: *Journal of Parallel and Distributed Computing* 135 (Jan. 2020), pp. 21–55. DOI: 10.1016/j.jpdc.2019.09.005.

[3] Rafay Iqbal Ansari et al. «5G D2D Networks: Techniques, Challenges, and Future Prospects». In: *IEEE Systems Journal* 12.4 (2018), pp. 3970–3984. DOI: 10.1109/JSYST.2017.2773633.

[4] Javier Baliosian et al. «Self-managed content-based routing for opportunistic networks». In: *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*. 2011, pp. 121–128. DOI: 10.1109/INM.2011.5990682.

[5] Scott Burleigh, Kevin Fall, and Edward J. Birrane. *Bundle Protocol Version 7*. RFC 9171. Jan. 2022. DOI: 10.17487/RFC9171. URL: https://www.rfc-editor.org/info/rfc9171.

[6] Richard Coombs. «Models for information propagation in opportunistic networks». PhD thesis. Cardiff University, UK, 2014. URL: http://orca.cf.ac.uk/68235/.

[7] Mark R.T. Dale. «Spatio-temporal Graphs». In: *Applying Graph Theory in Ecological Research*. Cambridge University Press, 2017, pp. 222–251. DOI: 10.1017/9781316105450.011.

[8] Anwitaman Datta, Silvia Quarteroni, and Karl Aberer. «Autonomous Gossiping: A self-organizing epidemic algorithm for selective information dissemination in mobile ad-hoc networks». In: *In International Con-*

ference on Semantics of a Networked World. 2004, pp. 126–143. DOI: 10.1.1.58.3986.

[9] Avri Doria, Maria Udé;n, and Durga Prasad Pandey. «Providing connectivity to the Saami nomadic community». In: *International Conference on Open Collaborative Design for Sustainable Innovation :* Godkänd; 2002; 20070416 (biem). 2002.

[10] Kevin Fall. «A delay-tolerant network architecture for challenged internets». In: *Sigcomm '03: proceedings of the 2003 conference on applications, technologies, architectures, and protocols for computer communications*. Karlsruhe, Germany: ACM, 2003, pp. 27–34. ISBN: 1-58113-735-4. DOI: 10.1145/863955.863960.

[11] R.E. Kahn et al. «Advances in packet ratio technology». In: *Proceedings of the IEEE* 66.11 (Nov. 1978), pp. 1468–1496. ISSN: 0018-9219. DOI: 10.1109/PROC.1978.11151.

[12] Udit Narayana Kar and Debarshi Kumar Sanyal. «An overview of device-to-device communication in cellular networks». In: *ICT Express* 4.4 (2018), pp. 203–208. ISSN: 2405-9595. DOI: https://doi.org/10.1016/j.icte.2017.08.002. URL: https://www.sciencedirect.com/science/article/pii/S2405959517301467.

[13] Peter T. Kirstein. «The Early International Activities in the Arpanet, Its Mutation into the Internet, and Some Further Regional Extensions». In: *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. MobiCom '18. New Delhi, India: Association for Computing Machinery, 2018, pp. 573–574. ISBN: 9781450359030. DOI: 10.1145/3241539.3241588. URL: https://doi.org/10.1145/3241539.3241588.

[14] Vishnupriya Kuppusamy et al. «Evaluating Forwarding Protocols in Opportunistic Networks: Trends, Advances, Challenges and Best Practices». In: *Future Internet* 11.5 (2019). ISSN: 1999-5903. DOI: 10.3390/fi11050113. URL: https://www.mdpi.com/1999-5903/11/5/113.

[15] Zoubir Mammeri. «Reinforcement Learning Based Routing in Networks: Review and Classification of Approaches». In: *IEEE Access* 7 (2019), pp. 55916–55950. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2913776.

[16] Aida-Ștefania Manole et al. «Opportunistic Network Algorithms for Internet Traffic Offloading in Music Festival Scenarios». In: *Sensors* 21.10 (2021). ISSN: 1424-8220. DOI: 10.3390/s21103315. URL: https://www.mdpi.com/1424-8220/21/10/3315.

[17] Dietmar PF Möller. «Guide to computing fundamentals in cyber-physical systems». In: *Computer Communications and Networks. Springer, Heidelberg* (2016).

[18] Mirco Musolesi and Cecilia Mascolo. «A Community Based Mobility Model for Ad Hoc Network Research». In: *Proceedings of the 2nd International Workshop on Multi-Hop Ad Hoc Networks: From Theory to Reality*. REALMAN '06. Florence, Italy: Association for Computing Machinery, 2006, pp. 31–38. ISBN: 1595933603. DOI: 10.1145/1132983.1132990. URL: https://doi.org/10.1145/1132983.1132990.

[19] Raj Rajkumar, Dionisio de Niz, and Mark Klein. *Cyber-Physical Systems*. 1st. Addison-Wesley Professional, 2017. ISBN: 9780321926968.

[20] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. USA: Prentice Hall Press, 2009. ISBN: 0136042597.

[21] Rahul Sachdeva and Amita Dev. «Review of opportunistic network: Assessing past, present, and future». In: *International Journal of Communication Systems* 34.11 (2021), e4860. DOI: https://doi.org/10.1002/dac.4860. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/dac.4860. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.4860.

[22] Adam Schlesinger et al. «Delay/Disruption Tolerant Networking for the International Space Station (ISS)». In: *2017 IEEE Aerospace Conference*. 2017, pp. 1–14. DOI: 10.1109/AERO.2017.7943857.

[23] Annalisa Socievole et al. «Routing in Mobile Opportunistic Social Networks with Selfish Nodes». In: *Wireless Communications and Mobile Computing* 2019 (Feb. 2019), p. 6359806.

[24] Leigh Torgerson et al. *Delay-Tolerant Networking Architecture*. RFC 4838. Apr. 2007. DOI: 10.17487/RFC4838. URL: https://www.rfc-editor.org/info/rfc4838.

[25] J. Visca et al. «Buffer Management in Opportunistic Networking». In: *Proceedings of the Latin America Networking Conference on LANC 2014*. LANC '14. Montevideo, Uruguay: Association for Computing Machinery, 2014. ISBN: 9781450332804. DOI: 10.1145/2684083.2684085. URL: https://doi.org/10.1145/2684083.2684085.

[26] Jorge Visca. «RON - Opportunistic Networks». MA thesis. Universidad de la República, 2014. URL: https://hdl.handle.net/20.500.12008/5174.

[27] Jorge Visca and Javier Baliosian. «A Model for Route Learning in Opportunistic Networks». In: *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. May 2022, pp. 1–4. DOI: 10.1109/NOMS54207.2022.9789826.

[28] Jorge Visca and Javier Baliosian. «rl4dtn: Q-Learning for Opportunistic Networks». In: *Future Internet* 14.12 (2022). ISSN: 1999-5903. DOI: 10.3390/fi14120348. URL: https://www.mdpi.com/1999-5903/14/12/348.

[29] Jorge Visca, Matías Richart, and Javier Baliosian. «Stochastic Models for Opportunistic Networks». In: *2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW)*. 2019, pp. 1–6. DOI: 10.1109/WCNCW.2019.8902544.

[30] Jorge Visca et al. «Opportunistic media sharing for mobile networks». In: *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*. 2016, pp. 799–803. DOI: 10.1109/NOMS.2016.7502902.

[31] Jorge Visca et al. «Path sampling, a robust alternative to gossiping for opportunistic network routing». In: *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. 2016, pp. 1–8. DOI: 10.1109/WiMOB.2016.7763244.

[32] Qingshan Wang and Zygmunt J. Haas. «Performance Analysis of Epidemic Routing for Delay-Tolerant Networks». In: *Resource Management in Mobile Computing Environments*. Cham: Springer International Publishing, 2014, pp. 579–594. ISBN: 978-3-319-06704-9. DOI: 10.1007/978-3-319-06704-9_26. URL: https://doi.org/10.1007/978-3-319-06704-9_26.

APPENDICES

Appendix 1

Path sampling, a robust alternative to gossiping for opportunistic network routing

Jorge Visca et al. «Path sampling, a robust alternative to gossiping for opportunistic network routing». In: *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. 2016, pp. 1–8. DOI: 10.1109/WiMOB.2016.7763244

Author's contribution The author provided the design and implementation of the Path Sampling protocol. He also implemented the tests and provided the analysis of the results.

2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)

Path Sampling, a Robust Alternative to Gossiping for Opportunistic Network Routing

Jorge Visca*, Javier Baliosian*, Raul A. Fuentes†, and Ana R. Cavalli†
* University of the Republic, Montevideo, Uruguay. E-mail: {jvisca, baliosian}@fing.edu.uy
† CNRS SAMOVAR-TELECOM Sudparis, Evry, France. E-mail: {fuentess, ana.cavalli}@telecom-sudparis.eu

***Abstract*—Opportunistic Networks have been designed for transmitting data in difficult environments, characterized by high mobility, sporadic connectivity, and constrained resources. To sustain these networks, the literature describes methods such as *Epidemic* and *Spray&Wait*, which do not learn from the network behaviour, and Gossiping-based algorithms that collect historical network data to improve efficiency. In this paper, we show that Gossiping-based solutions suffer from pathological behaviour in some simple network scenarios. Under certain conditions almost all the data transmitted by some nodes may get lost in the network, not reaching its destination. To address this problem we have proposed an algorithm that responds in a more robust manner while staying relatively simple. In this work, we show that our algorithm achieves delivery rates comparable to gossiping-based algorithms, while being more robust and providing better fairness. To illustrate this result, we test native implementations of our solution, *Path Sampling*, and related algorithms on a network simulator.**

I. Introduction

The proliferation of low-cost wireless devices has enabled the creation of novel applications, such as sensor networks. These are self-supported network deployments composed of low-cost devices with the purpose of collecting data for environmental monitoring, fleet tracking, etc. They are expected to operate without support from additional infrastructure, to be deployed in remote areas or provide services during natural disasters. To maintain these operations, various technologies have been developed.

One of such technologies is Opportunistic Networks (ON), that intends providing networking in harsh conditions and critical situations. In an ON some of the assumptions made when designing protocols for common computer networks are not satisfied: there may not exist an end-to-end path at any given moment; connectivity between two nodes may be highly asymmetrical or even unidirectional; nodes may spend most of the time in a low-power consumption state, enabling radio interfaces for short periods of time, etc. Additionally, it is usual for nodes to meet others peers sporadically, for short-lived sessions. Depending on the application this kind of networks is also named "Delay Tolerant Networks" and "Disruption Tolerant Networks" (DTN).

Because of those challenges, the techniques and algorithms developed for conventional networks do not apply in ON and different routing algorithms are needed. ON protocols are of the "Carry and Forward" type because nodes store data packets and move carrying them until a transmission opportunity appears. A packet may have to be carried by several nodes before it reaches destination. Opportunistic protocols determine when and for how long a node store a particular data packet. There are multiple ON protocols of varying complexity and based on different assumptions, designed for different use cases and performance requirements.

In a previous work we presented FLOP [1], a system for the efficient distribution of content in wireless heterogeneous networks, in particular networks with high mobility and radio interference. To support FLOP operations in such difficult conditions we sketched a ON methodology for data dissemination. In this work, we formalize this methodology under the name of *Path Sampling* protocol. *Path Sampling* belongs to the class of the simple opportunistic protocols, adapted for working on very low-cost and low-power equipment, usually mobile and battery-powered. In particular, this new algorithm takes advantage of the periodic probes used to detect nodes in the vicinity to collect data about the network connectivity patterns.

This paper is organized as follows. Popular ON routing protocols are discussed in Section II. The proposed new protocol, *Path Sampling*, is described in Section III, and a comparison with the existing protocols is made in Section IV. Finally, conclusions and future work are outlined in Section V.

II. Related Work

Following a classification proposed in [2] the work presented in this paper concerns infrastructure-less networks. These are networks where all the nodes are equals and have the same role in the data routing. This class of protocols is classified as either Dissemination- or Context-based.

Dissemination-based protocols typically do not use knowledge on the network, and are variations of controlled flooding of messages over the network. The best known protocols of this class are *Spray & Wait* and *Epidemic*. On contrast, context aware algorithms differ in the amount of the states they collect and the complexity of the process they apply. It has been shown that, while being a NP-Hard problem, algorithms that collect data on the network behaviour perform better than general algorithms, and there is a balance between the amount of data collected and the associated performance gains [3].

The *Epidemic* routing protocol is based on a controlled flood [4], [5] over the network. A node delivers all the messages in its buffer to all the nodes it meets. In this way, a message will "infect" nodes until it reaches the destination. The flooding of the messages is controlled by two parameters: the buffer space and the maximum hop count. The first

978-1-5090-0724-0/16/$31.00 ©2016 IEEE

parameter limits the number of messages that can be carried, with a FIFO buffer management strategy used to replace older messages with newer ones. The second parameter controls the maximum number of hops that a message will move away from the emitter. When the maximum hop count is set to 1, the protocol is equivalent to *direct delivery*, where a message will reach destination only if the emitter node comes in the range of the receiver. Conversely, a maximum hop count value bigger than the total number of nodes will imply a full flooding of the network. Between these extremes there is a balance between latency (how fast a message reaches destination) and resource utilization.

The *Spray & Wait* routing protocol attempts to reduce the resource utilization compared to *Epidemic*. It is based on two phases: *i*) *Spray* where a message is flooded to the first n different nodes it meets, and *ii*) *Wait* where the nodes with a copy hold the message until some of them finds the destination.

There are several variations of the *Spray & Wait* protocol. For example in *Randomized Spray & Wait*, during the flooding phase the replication is controlled by a configurable replication probability. In *Binary Spray & Wait* in the flooding phase there is a virtual number of copies of a message, and upon delivery each node handles over to the receiver half of the copies in its own inventory.

The *Spray & Wait* and *Epidemic* are considered stateless protocols because they do not learn anything about the status of the network for improving the delivery rates. Some of the simplest stateful protocols are based on gossiping [6], where nodes exchange vectors describing the local perceived quality associated to different nodes as potential destinations. Examples of such protocols are *PROPHET* [7], *CEPMF* [8] and *RON* [9].

RON is a content-based[1] opportunistic protocol that uses a variation of gossiping. As a routing protocol *RON* has two functionalities: routing, where data needed to make forwarding decisions is computed and distributed, and forwarding, the act of taking a decision once there is a data packet to process.

For supporting routing, nodes contain a *View*, a list of $\{$destination, quality$\}$ pairs where a delivery probability is associated to each known destination. The quality property represents how effective the node considers itself to deliver data to the corresponding destination. Notice that in *RON*, a destination is a content-based subscription and not just a node, but this does not affect the routing mechanism. Views are exchanged with other nodes as they meet, and this exchange is used to update the information in the local *View*. The qualities in the *View* are updated following two equations; in the first place, quality regularly decreases as the corresponding destination is not encountered as seen in equation 1 (ageing). This effect is driven by a configuration parameter $0 < \gamma < 1$. Then, the destinations that appear mentioned in a incoming *View* have its quality reinforced following equation 2, using a configuration parameter $0 < P < 1$. Typical values for γ and P are 0.9 and 0.01 respectively, but the exact values depend on the dynamics of networks.

$$q_{new} = q_{old} \times \gamma^{-(t-t_0)} \tag{1}$$

$$q_{new} = q_{old} + (1 - q_{old}) \times q_{incoming} \times P \tag{2}$$

The delivery probability attribute is then used to decide how to assign buffer space to different messages. Messages to destinations with higher delivery probability have better chance to be successfully delivered, so the node considers itself a better carrier for such messages.

When there is a tie on the computed quality, several possible buffer policies are possible. For a study on the impact of the different policies under various mobility patterns, see [10]. In this work, we use simple FIFO rule, where the oldest message is evicted first.

III. The Path Sampling protocol

In this section, we present the new protocol, *Path Sampling*, which is based on a new method for prioritizing messages to be stored by different nodes. This priority is based in a estimation of the end-to-end delivery probability of a message. Unlike traditional gossiping that learns a delivery quality from the encounter probability with neighbours, in *Path Sampling* the quality is learned by observing special probes sent from subscribers. These probes are included in the beacon sent by the nodes to detect neighbouring nodes, and thus add little overhead.

As subscription are broadcasted by the subscriber, stored, and then re-broadcasted by other nodes, they are flooded through the network. This process is used to collect data on the network topology to better route content from the data source to the subscribers.

For this purpose subscriptions have a sequence number and a $visited$ attribute, which is a list of nodes it is known they have traversed. The same subscription can reach a node following different paths. The sequence number allows a node to detect the first time a subscription of a given original emission is received. Whenever a node receives such a "first to arrive" subscription message, it adds itself to the $visited$ attribute. Thus, the $visited$ attribute of the subscription will contain the nodes participating of the fastest propagation path of the subscription instance, which can be interpreted as a opportunistic shortest path.

An example of the route sampling process can be seen in Figure 1. In the example, the node A subscribes to a content available in node F. In a) node A broadcasts its subscription with sequence number 1 and the $visited$ field only containing A itself. As this is a new subscription for B and C they store it, registering its sequence number and updating the $visited$ field adding themselves to it. In b) the node C broadcasts the subscription. Node A receives it, but as the sequence number is not higher than the one already registered the message is

[1]Content-based means that it uses a generalized addressing where, unlike in the usual destination-based routing, the receivers specify a filter requesting that matching messages must be delivered to them. Thus, destinations are not nodes but subscriptions.

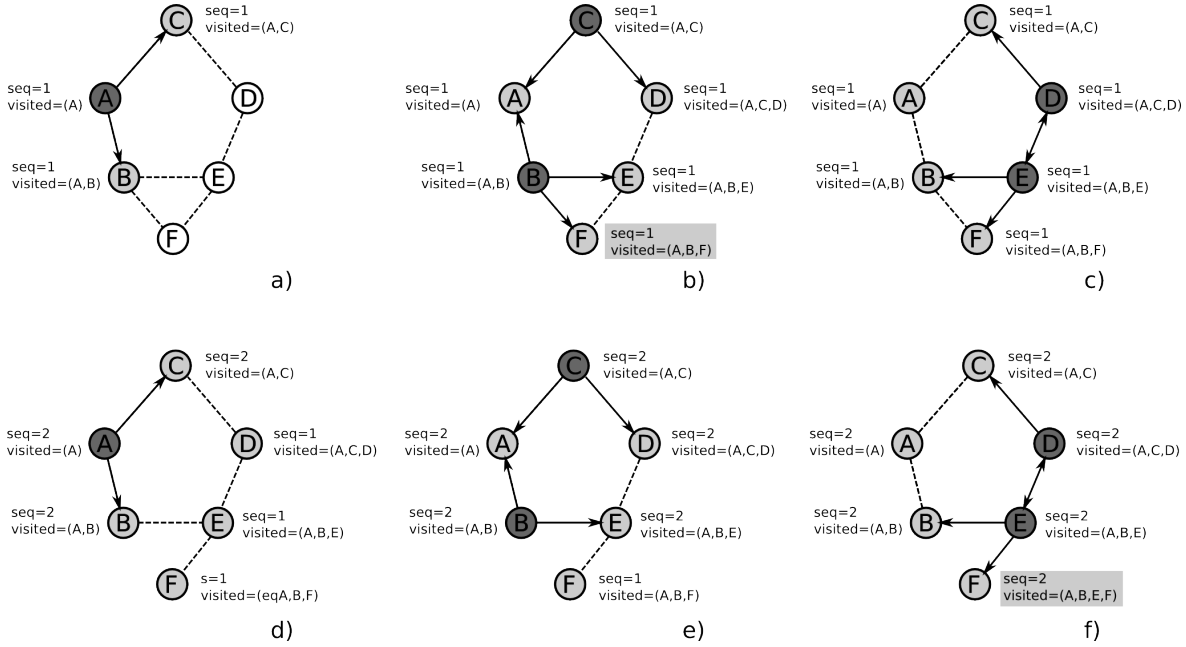


Fig. 1: Route sampling process.

ignored. The node D, in turn proceeds to store it as a new subscription.

Similarly, when B broadcasts the subscription it will be ignored by A and stored in E and F. Node F is provider of data matching the subscription, so it takes a note of the path A-B-F. In c), F receives the subscription through a longer path (via E), but does not take any action as the sequence number is no bigger than the one already registered.

Starting in d), a new broadcast is generated in the node A, with a sequence number 2. Let us suppose the link B-F disappears, for example due to movement of the node F. In this configuration the shortest propagation of the subscription to F is A-B-E-F. When the new subscription reaches F following this path, F will register it as a new sample for the path towards A as it has a higher sequence number (2) than the one registered (1).

The $visited$ attribute samples obtained can be used to detect the best nodes to handle traffic to a subscriber. For example, if a certain node repeatedly appears in the shortest path for a subscriber, this means it is a good candidate for serving data to it. Conversely, a node never or rarely mentioned is probably badly placed for this. The quality associated to nodes can be managed by a reinforcement/ageing mechanism, where a node's quality is increased when it appears in a $visited$ list, and decays if it is not.

Notice that this method assumes that paths are symmetrical: a good node to deliver data from the subscriber is expected to be a good node to deliver data to the subscriber.

The quality parameter allows to determine the favoured nodes for handling the data for a given subscription. In particular, data providers can tag messages with this list of nodes (the $path$ attribute). To limit the data transmitted, $path$ attribute is limited to a configurable number of top ranking nodes. This attribute is used by intermediate nodes to decide whether to hold a copy of the message or not: if a node find itself in the node list, the message will be selected for storing.

The pseudocode for *Path Sampling* can be seen in Listing 1. Line 1 is the *View* data structure, which is a set of Subscriptions with associated sequence number and $visited$ set. Line 5 is the data structure for holding the learned node quality for each subscription. The data structure at line 6 is the message buffer.

The loop at the line 7 periodically broadcasts the subscriptions, increasing the sequence number for the subscriptions it owns.

These broadcasts are received and processed by the loop on line 13. Only subscriptions with a higher sequence number than the last seen are accepted (line 17). The processing consists on the reinforcement of the quality associated to nodes participating in the subscription's propagation (lines 19 and 23), with configuration parameters γ and P, and the selection of matching messages for including in a response broadcast (line 25). These messages will be tagged with the best ranked nodes to participate in the propagation towards the subscriber ($path$ attribute at line 27).

The messages received are processed in the loop at the line 31. Messages that specify the node in the $path$ attribute are merged in the buffer M. The merging may imply dropping some other messages to make space in the buffer. Also, if the message matches some of the node's own subscriptions it will be handled by the application.

Listing 1 Path sampling protocol

```
1:  V : TABLE[Filter]→{
2:    seq: Number
3:    visited : SET OF Node
4:  }
5:  Q : TABLE[Filter]→(Table[Node]→Number)
6:  M : SET OF Message
7:  loop
8:    wait τ seconds
9:    for all f, view IN V do
10:     if is_own(f) then
11:       view.seq ← view.seq + 1
12:     broadcast_view({f, view.seq, view.visited} )
13: loop
14:   {f, seq, visited} ← wait for View broadcast
15:   response : SET OF Message
16:   add(visited, this_node)
17:   if NOT f IN V.keys OR seq > V[f].seq then
18:     V[f] ← {seq, visited})
19:     for all q IN Q[filter], m IN q.keys do
20:       q[m] ← q[m] × γ^Δt
21:       if m IN visited then
22:         q[m] ← q[m] + (1 − q[m]) × P
23:     for all m IN visited | NOT m IN Q[f].keys do
24:       Q[f][m] ← initial_q_value
25:     for all m IN matching(M, f) do
26:       if is_own(m) then
27:         m.path ← merge_best_nodes(Q[f])
28:       remove(d.path, this_node)
29:       response.add(d)
30:   broadcast_messages(response)
31: loop
32:   m ← wait for Message broadcast
33:   if NOT m IN M AND this_node IN m.path then
34:     merge(M, m)
```

IV. Experimental Evaluation

To compare the behaviour of *Epidemic*, *Spray & Wait*, *RON*, and *Path sampling* protocols we implemented them as part of the *Rong* package [11]. *RON* is used as an example of *Gossiping* protocol. The *Rong* package provides the implementations in the Lua language. The tested implementations use TCP (for *Epidemic* and *Spray & Wait*) and UDP (for *RON* and *Path sampling*).

Spray & Wait was configured with to a initial value of 32 copies, and *Epidemic* protocol had the maximum number of copies control disabled. The *Path sampling* was setup with a maximum path count of 8. For all protocols, the nodes where configured with a buffer size of 15 messages, and using FIFO buffer policy.

A. Simulation platform

We deployed *Rong* in the NS-3 simulator using the Direct Code Execution (DCE) module [12], which allows to run native code inside the simulated environment.

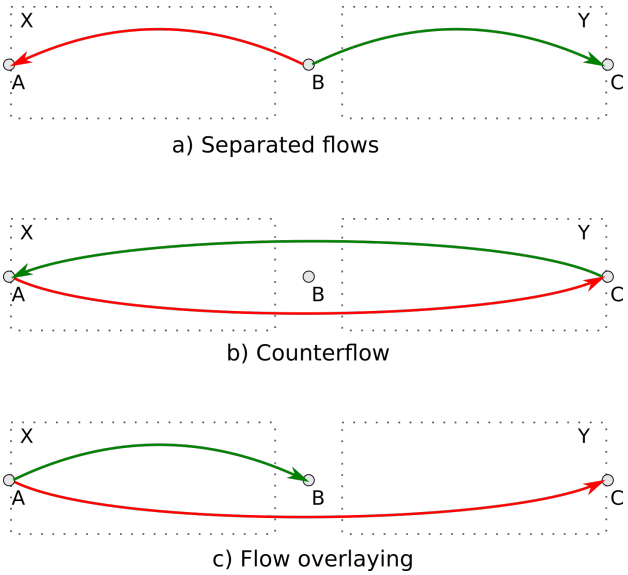


Fig. 2: The simulation scenario consists of two adjacent geographic areas. Nodes move randomly inside each area. Three data flow scenarios are tested.

NS-3 allows using a realistic network software stack and modelling the radio medium utilization, and takes into account physical layer phenomena, such as signal decay and radio interference. This is not possible when using a specialized opportunistic protocol simulator such as ONE [13], as it is done frequently in research works in the area. Another advantage of using NS-3 with DCE is that it enables us to run exactly the same codebase that is deployed on the real hardware, so it is possible to verify other characteristics such as resources utilization.

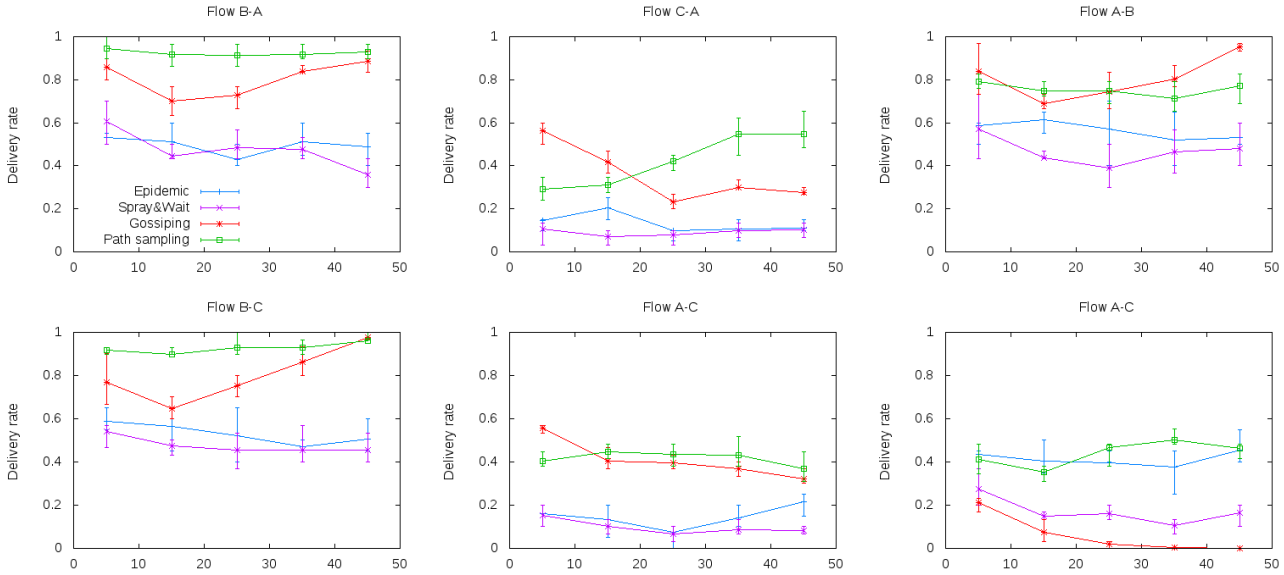
The radio interfaces are 802.11a, with a range of about 100m to 130m, depending on interference from other nodes.

B. Scenarios

The analysis of human mobility traces has shown that they adhere to the *small-world* network pattern [14][15]. This implies that the average shortest path is only a few hops long, and that relations tend to be symmetrical. Furthermore, strong modularity is detected, which means that nodes tend to form communities, where links within the community are stronger that with nodes outside. This behaviour is typical of social networks [15].

Accordingly to those findings, we explore the behavior of three possible dataflow arrangements shown in Figure 2 because they are representative of use cases that are expected to occur in many deployments. We focus on one of the simplest clustered topologies, with only two groups side-by-side. Dataflows in this topology can be intra- or inter-cluster. Inter-cluster traffic may have to compete with other inter-cluster or intra-cluster traffic.

In all three scenarios there are two zones of 400m×400m each (compressed vertically in the figure), separated by a



(a) Separated flows scenario (b) Counterflow scenario (c) Overlay scenario

Fig. 3: Delivery performance, averaged each 10 consecutive messages.

100m gap, and named X and Y. Within each zone, there are five nodes moving in a Random Waypoint mobility pattern, with a speed uniformly distributed between 1 m/s and 2 m/s. There are also three fixed nodes, one at each extreme and one between the two sectors. These nodes play the roles of emitters and receivers in the different scenarios. A message travelling between the extreme nodes has to be handled over between mobile nodes placed in different sectors because the node's radio-range is smaller than 500m.

1) Separated flows scenario: A single data source B has two separate data flows, one towards A (trough zone X) and the other towards C (trough zone Y). Notice that stateful algorithms must learn that nodes inside X are adequate to transport data towards A but not towards C and, conversely, nodes in Y are adequate to serve data towards C but not towards A.

2) Counterflow scenario: The nodes A and C exchange data. Both data flows must traverse the whole network in opposed directions. Besides that, both flows are equal, thus if the network usage is fair both flows should attain equal performance.

3) Flow overlaying scenario: In this scenario there are two flows from a single source node (A), with both flows sharing part of the path (trough sector X). This scenario is interesting because it forces two different flows to compete for the same resources. One of the flows has an easier task, as it is shorter and remains inside one cluster. The other must cross between two clusters. Of special interest is to study how the protocol assigns resources to both flows.

For the experiments, simulations were run for 5000s. During the first 500s no data is transmitted to allow protocol stabilization. Then, source nodes transmit 60 messages each, with 60s intervals between messages. There is time left to give opportunity for messages to reach destination. In the figures, $t = 0$ is set at the moment of the first message transmission. The simulations are run 30 times for statistical reasons.

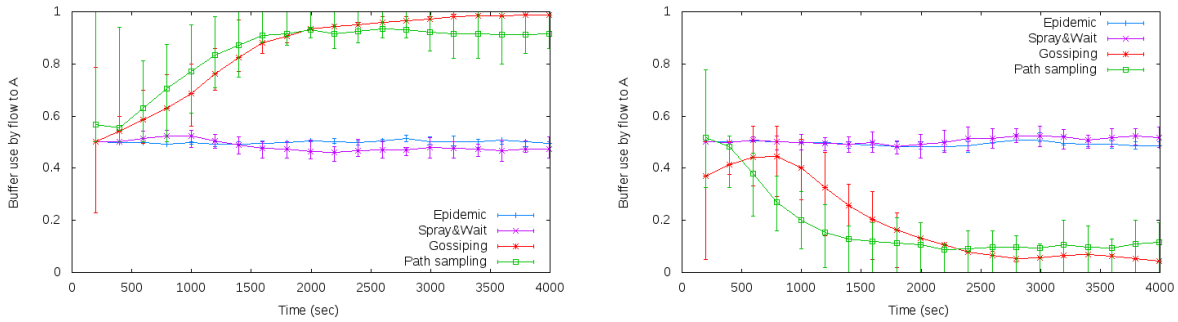
C. Results

The ON routing protocols are probabilistic protocols, and their main performance metric is delivery rate, i.e. the proportion of emitted messages that reach destination.

The delivery performance is presented in Figure 3. For each scenario, the delivery rate for both flows is shown. The horizontal axis represents the first 50 messages sent. The last 10 messages are excluded because they have different behavior due to lack of pressure for replacement in buffers from subsequent messages. Values are the average and interquartile range of the delivery rate.

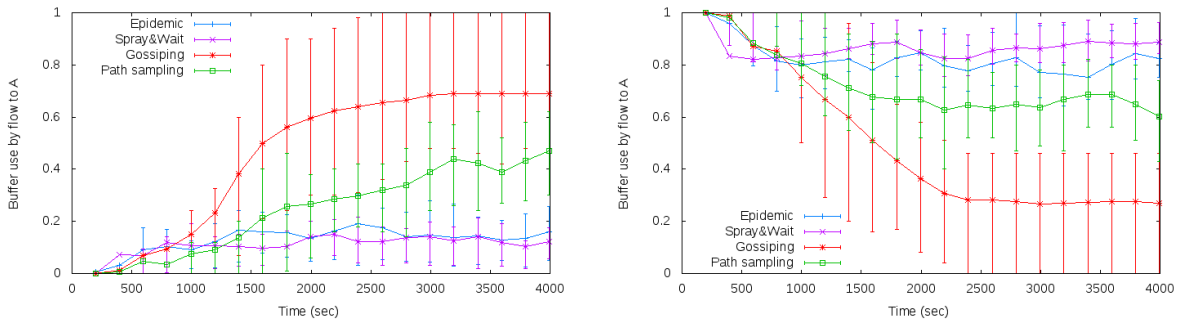
The main observation for the separated flows scenario (Figure 3a) is that stateful protocols achieve better performance than the stateless. The difference in their behaviour can be seen in Figure 4: stateful protocols learn the characteristics of the network and adapt the buffer utilization to the actual needs of the flows. This can be seen in the fact that as time passes buffer utilization in the sector X is skewed towards the messages that have A as destination. In the stateless protocols, conversely, all flows are equally represented in the buffers. This means that all messages compete for buffer space regardless of placement and destination, degrading network's performance. In this scenario the stateless protocols behave as if they had half of the available buffer space.

Figure 3b shows bad behavior from the *Gossiping* protocol in the counterflow scenario. Of the two flows, one achieves lower delivery rate than the other, despite being equivalent.



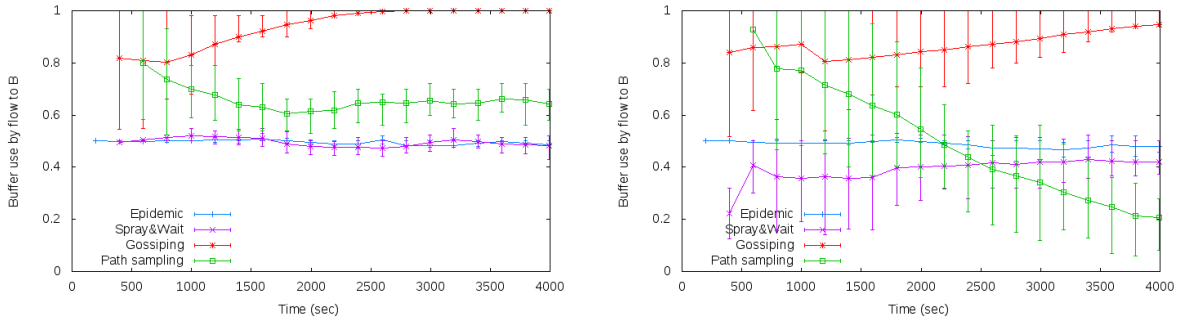
(a) Sector X (b) Sector Y

Fig. 4: Buffer distribution, separated flows scenario.



(a) Sector X (b) Sector Y

Fig. 5: Buffer distribution, counterflow scenario.



(a) Sector X (b) Sector Y

Fig. 6: Buffer distribution, overlay scenario.

The reason lays in a weakness of the gossiping algorithm [16], and is illustrated in Figure 5: nodes in sector X compute a better quality for delivering to A than to C. This means that between the two flows in the networks these nodes will prefer the messages targeted to A over the ones targeted to C. Traffic from A gets starved: messages get evicted from buffers as nodes prefer handle messages close to destination over the ones close to the source. The result is high variation in buffer assignment seen in Figure 5 for the *Gossiping* protocol. The actual distribution of the delivery performance is shown in Figure 7. It can be seen that *Gossiping* does not converge as *Path sampling* does, resulting in a bimodal distribution: about half of the messages are delivered with rates clustered around 0.8, the other half with a rate of less than 0.2.

This starvation behaviour highlights a fundamental weakness of gossiping-based routing: it does not take into account the end-to-end delivery of messages. *Path Sampling* achieves robust performance because nodes have equal probability to participate in both flows, so they treat both flows equally.

The counterflow scenario also shows bad behavior from the stateless *Epidemic* and *Spray & Wait* protocols (Figure 5): buffer occupation is skewed towards younger messages, this is messages emitted in the closest node. The reason is that in FIFO policy, the probability of a message dropping from a buffer is independent from node's placement or message's content, and grows as time passes: the longer a message is in a buffer, the higher the chances of being replaced by another message. This results in a higher probability of losing

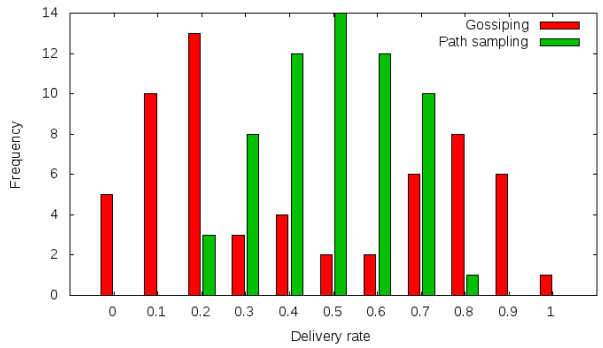


Fig. 7: Histogram of the attained delivery rates for the counterflow scenario.

a message in a longer path, as the message has to spend more time in buffers.

This same behavior results in equal buffer assignment to all messages in the overlay scenario (Figure 6). In this scenario all messages share the same emitter, the node A. This leads to higher delivery rate for flow $A \rightarrow B$ than $A \rightarrow C$, as there is a higher probability for losing a messages from the flow $A \rightarrow C$ because they spend more time traversing the network. Also, buffer space is wasted on sector Y on messages that already passed their destination (B).

As in the counterflow scenario, *Gossiping* also behaves pathologically in the flow overlaying scenario: there is a complete transmission failure for the longer link (Figure 3c). The reason is similar: bad buffer assignment. In the sector X the flow towards $A \rightarrow B$ starves the flow towards $A \rightarrow C$, because nodes prefer messages to the closer destination (Figure 6). The final result is that the Y sector is left without messages to C.

The *Path Sampling* protocol provides equal opportunities for both flows, even if they are of different length (Figure 6). The reason is that nodes in the X sector have the same probability to be on the path from A to B than from A to C, so both destinations are treated almost equally.

Although *Path Sampling* achieves higher delivery rate on the shorter flow (Figure 3c), it performs consistently better than *Epidemic* and *Spray & Wait*, and does not display the failure mode of *Gossiping*.

Even tough *Path Sampling* shows overall better performance, there are peculiarities in its behaviour. Figure 6 shows that the closest destination is somewhat favoured, converging to a value of about 60% of buffer share use. The reasons for this behaviour needs more study. One possible reason is the fact that *FIFO* is still used among the messages that are selected according to the $path$ attribute. Also, there is a probability than a node from "wrong" sector participates in a shortest path sample.

V. Conclusions and Future Work

In this paper, we have provided empirical evidence that gossiping-based protocols, while potentially performing better than stateless protocols, suffer from pathological behaviour in some simple scenarios. In gossiping protocols, nodes assign resources (buffer space) to messages deemed with the highest chance of reaching destination. We found that this property may cause starvation when a node is shared by several competing dataflows.

To overcome this, we have presented a protocol that collects information regarding data paths using token propagation. This protocol achieves similar delivery rate that a gossiping protocol without the performance degradation found in the counterflow and flow overlaying scenarios described in Section IV-B. The fairness of the protocol is found to be similar to a stateless protocol. This allows the *Path Sampling* algorithm to consistently outperform *Epidemic* and *Spray & Wait*, unlike *Gossiping* which displays some failure modes.

Path Sampling has three parameters: two dedicated to managing the reinforcement and aging of node quality for serving subscriptions, and the size of the $path$ vector. The last one is the most important. On one hand, it is the one that adds network overhead because it has to be attached to the messages. On the other hand, a bigger $path$ vector enables to include more node diversity. The optimal value for the number of nodes to include in a $path$ depends on the number of nodes participating in the different possibles propagation paths, and the expected path diversity level. Nevertheless, the typical *small-world* structure of networks implies that paths tend to be short, so it is expected that the list is not required to be too big.

Some of the data needed to optimize the $path$ parameter can be collected with the tokens as they propagate trough the network. For example, the path length and node's network centrality can be estimated from observing the $visited$ attribute. Additional data could be computed and transmitted with the $visited$ attribute, like the latency end-to-end and for each hop, or the pressure on the different buffers. Additional hints can be computed and distributed back with the $path$ attribute to help nodes when making a forwarding decision.

This suggest the possibility of creating methods for self management and automatic parameter adjustment. The impact of such techniques on performance on one hand, and network overhead on the other, should be studied.

References

[1] J. Visca, R. Fuentes-Samaniego, J. Baliosian, and A. Cavalli, "Opportunistic media sharing for mobile networks," in *IEEE/IFIP Network Operations and Management Symposium, 2016 NOMS*, April 2016.

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

[3] A. Balasubramanian, B. Levine, and A. Venkataramani, "DTN routing as a resource allocation problem," in *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, 2007, pp. 373–384.

[4] C. Baouche, A. Freitas, and M. Misson, "Routing mechanism for a DTN using WSN nodes for localization applications," in *IEEE International Conference on Networking, Sensing and Control*. IEEE, 2011, pp. 514–519.

[5] Agussalim and M. Tsuru, "Comparison of DTN Routing Protocols in Realistic Scenario," *2014 International Conference on Intelligent Networking and Collaborative Systems*, pp. 400–405, 2014.

[6] A. Datta, S. Quarteroni, and K. Aberer, "Autonomous Gossiping: A self-organizing epidemic algorithm for selective information dissemination in mobile ad-hoc networks," in *In International Conference on Semantics of a Networked World*, 2004, pp. 126–143.
[7] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," in *Service Assurance with Partial and Intermittent Resources*, 2004, pp. 239–254.
[8] L. Yazhi, N. Jianwei, and M. Jian, "Content encounter probability based message forwarding in opportunistic networks," in *Information Science and Engineering (ICISE), 2009 1st International Conference on*, dec. 2009, pp. 2589 –2594.
[9] J. Baliosian, J. Visca, M. Richart, G. Apollonia, L. Vidal, M. Giachino, and E. Grampin, "Self-managed content-based routing for opportunistic networks," in *IFIP/IEEE International Symposium onIntegrated Network Management*, 2011, pp. 121–128.
[10] J. Visca, M. Richart, J. Saavedra, J. Baliosian, and E. Grampin, "Buffer management in opportunistic networking," in *Proceedings of the Latin America Networking Conference LANC 2014*. ACM, 2014, pp. 2:1–2:7.
[11] J. Visca, "Rong." [Online]. Available: https://github.com/xopxe/rong
[12] H. Tazaki, F. Urbani, and T. Turletti, "DCE cradle: simulate network protocols with real stacks for better realism," in *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, 2013, pp. 153–158.
[13] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, 2009, pp. 55:1–55:10.
[14] T. Kathiravelu and A. Pears, "Benchmarking of opportunistic networking experiments," *Proceedings of The 5th Swedish National Computer Networking Workshop*, 2008.
[15] T. Hossmann, T. Spyropoulos, and F. Legendre, "A complex network analysis of human mobility," *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 876–881, 2011.
[16] J. Visca, "RON - opportunistic networks," Master's thesis, Universidad de la República, 2014. [Online]. Available: https://www.fing.edu.uy/inco/pedeciba/bibliote/tesis/tesism-visca.pdf

Appendix 2

Opportunistic media sharing for mobile networks

Jorge Visca et al. «Opportunistic media sharing for mobile networks». In: *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*. 2016, pp. 799–803. DOI: 10.1109/NOMS.2016.7502902

Author's contribution The author provided the design and implementation of the FLOP protocol. He also implemented the tests and provided the analysis of the results.

Opportunistic Media Sharing for Mobile Networks

Jorge Visca*, Raul Fuentes†, Ana R. Cavalli†, and Javier Baliosian*
* University of the Republic, Montevideo, Uruguay. E-mail: {jvisca, baliosian}@fing.edu.uy
†TELECOM Sudparis, Evry, France. E-mail: {fuentess, ana.cavalli}@telecom-sudparis.eu

***Abstract*—Nowadays there is a proliferation of smart devices used for media consumption. Usually, media contents are streamed from a Content Distribution Network (CDN), through a cellular network, which can get overloaded when there is a large number of active users per cell. At the same time the devices, i.e. smartphones, typically possess a set of wireless interfaces such as WiFi and Bluetooth that can be used to communicate with other devices in its proximity. In this work we propose a publish/subscribe, opportunistic network protocol aimed at offload infrastructure network nodes. It contributes to optimize the cellular network usage among mobile devices and to reduce the costs and energy consumption induced by the reproduction of media that are temporally popular.**

I. Introduction

The widespread use of LTE enabled smartphones, with high quality displays and high computing power, has resulted in a high use of streaming services for media consumption, typically video and music. This, when combined with a scenario with a high density of users such as a train station or wagon, concerts, campus, etc., can impose a very high load on the network. It is a known fact that this new user behaviour is pushing the cellular networks' capabilities to the limits of their current and foreseen possibilities [1]. Additionally, besides causing an increase in the costs for the provider and the users, this intensive data use is impacting on the lifetime of the mobile device's battery. It is also known that the social structure and interaction of the users of mobile devices is motivating the proposal of content distribution protocols that are based on *Device-to-Device* (D2D) communications [2][3]. The case of on-demand video, based on the transmission of independent chunks that can be stored and forwarded is particularly adequate for this type of data transmission.

In particular, the work presented in this paper is based in the Outband D2D scenario [2], where the direct communication between nodes is done over a secondary interface, such as WiFi or Bluetooth. Nevertheless, it is also applicable to the Inband D2D, where LTE is extended to provide direct communications.

The secondary network used for sharing content between devices is highly dynamic. This fact suggest the use of a Opportunistic Network (ON), or Delay Tolerant Network (DTN) as infrastructure. The main property of such networks is that is capable of operating even in the face of frequent network partitioning. This partitioning can be the result of a momentary disconnect of one or more nodes in a mesh network; or the normal state in a network where due the node mobility, the random encounter opportunities are used to deliver data, possibly in several hops. In either case, the fact that at a given moment no end-to-end path might exist means that the data must be kept at the nodes for potentially long lapses of time. This is referred as *store-and-forward*, a characteristic behavior of ONs, and can also be interpreted as a caching component of the network.

978-1-5090-0223-8/16/$31.00

We propose a protocol for enabling the efficient distribution of content in wireless networks, in particular when there is a mix of cellular and ad-hoc networks such as smartphones with WiFi support. The idea is to improve the network performance through efficient support of one-to-many dataflows and efficient cache placement. Of special interest are the networks where the user nodes have non-deterministic but highly organized behavior, such as spatial clustering and emerging data usage patterns (e.g. passengers on a train wagon). Also, the connectivity is provided by several network technologies of very different characteristics (structured vs. ad-hoc, licensed vs. unlicensed, short range/high throughput vs. long range/low throughput, etc.).

The paper is organized as follows. Section II presents some of the most relevant related work and positions our work in the body of prior art. Section III presents the proposed protocol itself and Section IV presents a simulation-based initial validation of the main idea. Finally, we conclude and depict future work in Section V.

II. Related Work

ONs typically have a high probability of intermittent connections, making the traditionally routing mechanisms not optimal [4]. In consequence ON routing protocols are based on the knowledge of the nodes about the network, (e.g PROPHET, Epidemic, Spray and Wait, MaxProp), or on adding additional static nodes (e.g. MULE, throwboxes) to collect packets and help to pass them to other nodes. *Epidemic* floods the network in a controlled way. The nodes store messages that are directed to other nodes, and "infect" other nodes by passing a copy of its messages without priority or limit [4], [5]. *MaxProp* is similar to *Epidemic*, however, messages are explicitly deleted when an acknowledge message sent by the destination node is received by a relay node. *PROPHET* controls the flooding using predictions based on the delivery probability for a specific destination. Finally, *Spray and Wait* is based on the phases that give name to the protocol, the former with the purpose of spraying n copies over the network and then wait until getting in contact with the destination.

The MULES are mobile nodes which can have a random path or predicted and can pass the messages of nodes between

```
subscriber_id=node_17
seq=157
visited={node_3,node_40}
filter={
  file_hash=0x102030
  start_ts>120
  bitrate>=512
  codec=H.264
}
```

(a) A Subscription.

```
file_hash=0x102030
file_name=ldca.avi
chunk=9
start_ts=212
bitrate=512
codec=H.264
path={node_2,node_40}
```

(b) A Descriptor.

Fig. 1: The messages used for Network Maintenance.

different areas, typically they moved to a central relay point. The Throwboxes are fixed nodes with better attributes than other nodes (power supply, storage capacity, computing power, etc.) working as relays for creating more contacts in the network.

In [6] the authors present a hybrid approach called Hybrid Routing System (HRS), which integrates infrastructure nodes into more classical DTNs. Their simulations show that exploiting infrastructure can boost the performance of the DTN protocols, no matter that they are centralized or decentralized. Yet, the work with HRS does not consider streaming traffic.

Of special interest is the work in [7] which focus on transmitting part of the load from the cellular companies between their licensed spectrum (e.g. 3g or 4g) and unlicensed specters for reducing the traffic in cellular networks, benefiting to the operators.

The two protocols that introduce the main concepts used by FLOP are RON and GrAnt. RON is a publish-subscribe opportunistic protocol in which a node that is interested in some type of messages issues a subscription that specifies that interest using a logic filter. Clients use the network through two type of messages, Subscriptions and Notifications.

The notifications are emitted by emitting nodes whenever there is data. These notifications will be delivered by the network to all the nodes with subscriptions whose filters that match the content. The emitter does not know who are the interested nodes, and the subscriber does not need to know the identity of the emitter nodes to be able to receive data of interest. This greatly simplifies the network's management as unlike in traditional destination-routed networks there is no need to maintain an inventory nor share well-known addresses.

FLOP uses the concept of publish-subscribe ON for high density networks with DTN characteristics.

The Greedy Ant protocol (GrANT) is a routing protocol able to adapt to the large variations suffered by a DTN topology, reducing latency in the message delivery.

GrAnt is based on Ant Colony Optimization (ACO), and is composed of two types of ants: the Forwards Ants (FA) and the Backward Ants (BA). The former are focused in discovering paths using greedy choices based on the social metrics from the nodes meanwhile the latest handles the placing and evaporation of pheromones in the paths traveled by the FAs.

FLOP adopts the idea of an ACO mechanism for routing. The mechanism itself is different, as GrAnt is destination-routed.

III. The FLOP protocol

FLOP aims to satisfy the following technical requirements:

- Distribution of files sized in several megabytes (e.g. a video)
- Files are divided in chunks, sized in the tens of kilobytes.
- Each chunk is characterized by a Descriptor, which contains the file ID, the place of the chunk in the file, and other attributes describing the content.
- Clients subscribe to content specifying a filter over the Descriptors of interest.

The protocol is conceptually aligned to MPEG-Dash [8] in how it handles files (chunking of media files, metadata for chunk description, chunk retrieval using HTTP), with a fundamental difference: in MPEG-Dash all content is obtained from a single server, while in FLOP the content may be distributed among several peer nodes, in addition to, or replacement of, a upstream server. The protocol has two functional areas. The first is the Network Maintenance, responsible of efficiently distributing information to nodes on what neighboring nodes contains chunks of interest. The second area is Content Handling, which consist on nodes selecting a neighboring node to request a chunk and transferring the file.

A. Network Maintenance

Network Maintenance implements a distributed publish-subscribe ON. It allows nodes to collect information on availability of chunks, as well as support the efficient routing of content in a highly dynamic and heterogeneous network. As some portions of the network are expected to be very dense, special care is taken to minimize the amount of traffic generated. Also, as the network is subject to very fast changes, the protocol is optimized for a high speed of convergence. It also provides information to facilitate the efficient routing and caching of content, and the identification of well suited nodes for the gateway role.

The protocol uses two kinds of messages, Subscriptions and Descriptors. A Descriptor is the metadata needed to describe a chunk, presented as a list key/value pairs (Fig. 1(a)). A Subscription is used to express the interest of a node in receiving certain data, expressed as a list of predicates over the values of attributes of a Descriptor (Fig. 1(b)). The attributes seq, $visited$ and $path$ are used for routing and will be described in more detail in later sections.

1) Routing: Nodes periodically broadcast a beacon containing the subscriptions for which the node has an interest. These subscriptions can be from the node itself (i.e. files being played on the device) or coming from other nodes and received and accepted by the node for serving. The subscription has a unique subscription ID, the ID of the subscribing node and might be updated between broadcasts (for example, a starting timestamp be advanced, or a new bitrate be desired).

As subscription are broadcast by the subscriber, stored, and then rebroadcasted by other nodes, subscriptions are flooded through the network. This process is used to collect data on the network topology to better route content from the data source to the subscribers.

For this purpose subscriptions have a $sequence$ number and a $visited$ attribute, which is a list of nodes. As the subscriptions are flooded through the network, the same subscription can reach a node through several distinct intermediate holders. The sequence number allows a node to detect the first time a subscription of a given original emission reaches it. Whenever a node receives such a subscription message, it will add itself to the $visited$ attribute. Thus, the $visited$ attribute of the subscription will contain the nodes participating of the fastest propagation of the subscription instance, which can be interpreted as a opportunistic shortest path.

2) Forwarding: Whenever a node receives a subscription, it will answer with the matching descriptors for the chunks it has in its chunk buffer. This announces the node's ability to serve said chunks if requested. These descriptors are also broadcasted, and thus can be received and potentially stored by any node in range. Received descriptors are stored in a descriptor buffer.

The node that introduces the content to the network (the server) tags the notification with a $path$ attribute, which is a list of nodes considered as best suited to handle the subscription. This list is built by selecting the best ranked nodes according to qualities derived from the $visited$ attribute of the subscription that match said content.

Whenever a node listens a descriptor (a reply to a own beacon broadcast or perhaps overheard from other nodes answer) it can use the path attribute to help decide if it is advantageous to keep it: if it finds itself in the path, this means the node is a good potential carrier for the content. When a node decides to hold a copy of the descriptor, it will remove itself from the path attribute. This is done to reduce the chance of admitting a descriptor twice, if encountered again after handling to further nodes. Furthermore, as the same Descriptor can be received several times from different nodes, each transmitter is registered as a potential source for the described chunk.

B. Content Handling

Once a node received a Descriptor for a chunk, it can decide to download it. This can be because the node itself is interested in said chunk (he is the subscriber), or because the node considers itself a good carrier for the chunk (is listed in the 'path' attribute as described above). If a decision to download a chunk is made, one of the owning nodes is selected and an session is initiated If the download is cut short for some reason, the download will be restarted from some other owning node. When the download is complete the chunk is stored and, if the subscription is local, handled to the requesting application. Additionally, the node will start to answer with Descriptor broadcasts to Subscriptions matching the chunk as to offer it, until the chunk is dropped from the cache. The lifetime of the downloaded chunks in the buffer is handled by using caching policies, using criteria such as age and popularity.

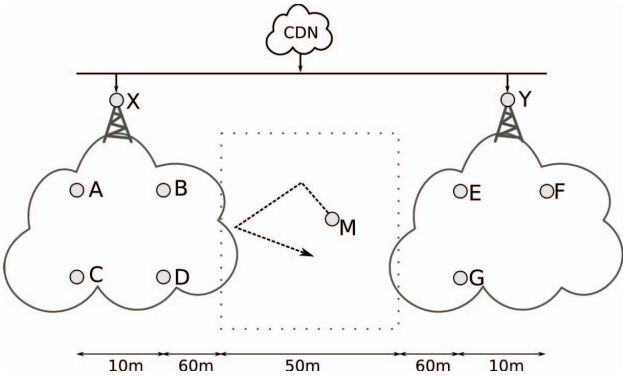


Fig. 2: Simulation Scenario.

IV. Validation

In this section we depict some initial experiments to show how FLOP reduces the cost of downloading media, saves mobile devices' energy and reduces the traffic forwarded by cellular radio bases in a hybrid WiFi-LTE scenario. For this validation we use the implementation of FLOP available at [9] deployed on a ns-3 simulator using the Direct Code Execution (DCE) module [10], which allows to run native code inside the simulated environment.

The validation scenario (see Fig. 2) contains, first, a fixed network node playing the role of the Content Distribution Network (CDN). This node is the source of a large video to be transferred. Connected to this node by fixed links, two cellular radio bases, X and Y, both covering disjoint geographic regions through LTE. A total of eight mobile nodes, equipped with LTE and 802.11a interfaces with 150m range. Of these nodes, seven are at fixed positions in close proximity in two groups, and one is moving in a random waypoint pattern in the area. All these nodes (fixed, mobile, and the radio bases) run the FLOP protocol. The mobile node (M) is outside the cellular coverage, but intermittently gets in range of the fixed nodes as it moves. The simulation is run for 1000 seconds.

Nodes play two video file stored initially in the CDN node. Files consists of 20 chunks of 250 KB each, each containing 10s of playtime. Nodes start playing the videos with 50s interval. Node A starts playing the first file at the 50s mark, D at the 200s mark. Then the mobile node M starts playing at the 250s mark, followed by nodes E through G. The second file is played in reverse order: the first node to play is G, and the last is A.

The nodes are configured with a inventory size of 60 messages, a time between beacons of 20s and a space for 3 nodes in *path* attribute. The results (averaged over 10 executions) are summarized in Table I. The $Protocol$ column shows the number of bytes broadcast as part of the protocol's routing mechanism. The $Rate$ column is the delivery rate, i.e. the fraction of the chunks that were actually delivered to the subscriber.

Studies show that LTE is considerably more energy consuming than WiFi [11] [12]. In particular [11] finds that even under ideal conditions LTE consumes no less power for receiving than WiFi for transmitting, with realistic traces on devices showing 23 times higher power consumption for LTE than for WiFi. It also shows that the power efficiency of LTE worsens as the data rate decreases, so more loaded cells result in higher power consumption for the terminal devices. This implies that our scenario, with high node density and heavy network traffic, puts a energy penalty on LTE traffic. This effect is offset partly by the fact that most of traffic is downlink (so the LTE interface is used mostly for receiving and not the much costlier transmitting) and bulk in nature (which allows TCP to improve the channel occupation given that the chunks are of considerable size).

For this work, we used power values of $0.8\,\mu\mathrm{J/bit}$ for transmitting and receiving over WiFi and $3\,\mu\mathrm{J/bit}$ for receiving over LTE. The column P/P_{ref} shows the ratio of the used power and a reference power consumption of a plain LTE download. The Op row is the impact on the cellular operator, this is, the ratio of the actual radio use and the one expected had all the nodes obtained the content from the radio bases. This also can be read as the reduction in the load on the licensed wireless medium.

The improvement for the cellular operator is clear. The reduction in the utilization of the LTE medium impacts not only the power consumption by the radio bases themselves, but also reduces the load on network and the provider's servers. Also, this means the wireless interfaces can achieve higher bandwidth and/or have more resources for other traffic, improving the user experience even for nodes not participating in the FLOP network.

The direct impact on the mobile nodes is twofold. In fist place, as node M shows a node can get content successfully even without cell coverage, only getting from their peers. It even helps other nodes, as seen in the WiFi upload value. In second place, most nodes get an effective power saving, spending less energy on the wireless communications. This means a longer battery life, or, alternately, playing bigger files at higher bitrates. It must be noted the energy consumption per node decreases as the number of nodes increases, as the content obtained through WiFi is spread over more nodes.

TABLE I: Performance simulation, traffic in Mb

	DwnLTE	DwnWifi	UpWiFi	Protocol	Rate	P/P_{ref}
A	4.96	5.04	5.45	0.12	1.00	0.78
B	3.84	6.16	8.26	0.12	1.00	0.77
C	2.98	7.02	5.13	0.12	1.00	0.63
D	2.25	7.75	9.13	0.12	1.00	0.68
E	4.60	5.40	1.02	0.12	1.00	0.64
F	6.31	3.66	4.28	0.12	1.00	0.85
G	5.05	4.95	6.64	0.12	1.00	0.82
M	0.00	10.00	10.06	0.12	1.00	0.54
Op						0.38

V. Conclusions and Future Work

In this work we propose FLOP, a publish/subscribe, opportunistic network protocol aimed at off-loading infrastructure network nodes and reduce the cellular network usage among mobile devices in order to reduce the costs and energy consumption induced by the reproduction of media that is temporally popular.

We show how FLOP can work also as a caching mechanism well suited for the common scenario in which some popular media is reproduced on mobile neighboring nodes, during a relatively short period of time.

The validation presented in this paper is just demonstrative of the protocol's potential. In a simplified but representative scenario, it shows how if there is more than a single file being played there is a net power saving for every node in the network and a dramatic traffic reduction at radio bases and fixed network.

So far we have worked on the "routing" part of the protocol, important research issues such as the caching strategies at each kind of node were not tested. However, we have very relevant insights from previous work [13] that point to caching policies that are highly related with the dominant mobility patterns.

FLOP must be validated against different network and mobility scenarios to study its behavior. In particular, the content handling component must be explored in depth. In the current implementation a node downloads a chunk if it finds itself on the shortest path to service some subscription, to satisfy the content distribution task. This information should interact meaningfully with the chunk popularity and download history to manage said chunk's lifetime in the buffer to offer a caching service for the network.

It is important to share the burden of downloading the content between the different nodes in the network. This presents a management problem, where tasks must be assigned to the different participant nodes to achieve an expected performance, optimize the resource utilization, etc. We plan on addressing this problem using autonomic management concepts, through a rule based distributed platform [14].

Acknowledgment

This work was partially funded by *Agencia Nacional de Investigación e Innovación* and the STIC AmSud project MOSAIC.

References

[1] Cisco, "Cisco visual networking index [or vni] global mobile data forecast, 2014–2019)," 2014. [Online]. Available: www.cisco.com/go/vni

[2] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 4, pp. 1801–1819, Fourthquarter 2014.
[3] ——, "A survey on device-to-device communication in cellular networks," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 4, pp. 1801–1819, Fourthquarter 2014.
[4] C. Baouche, A. Freitas, and M. Misson, "Routing mechanism for a DTN using WSN nodes for localization applications," in *2011 International Conference on Networking, Sensing and Control*. IEEE, Apr. 2011, pp. 514–519.
[5] Agussalim and M. Tsuru, "Comparison of DTN Routing Protocols in Realistic Scenario," *2014 International Conference on Intelligent Networking and Collaborative Systems*, pp. 400–405, 2014. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7057122
[6] C. P. Mayer and O. P. Waldhorst, "Routing in hybrid Delay Tolerant Networks," *Computer Communications*, vol. 48, pp. 44–55, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2014.03.018
[7] S. Dimatteo, P. Hui, B. Han, and V. Li, "Cellular traffic offloading through wifi networks," in *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, Oct 2011, pp. 192–201.
[8] "ISO/IEC 23009-1:2014 - Dynamic adaptive streaming over HTTP (DASH)," 2014.
[9] [Online]. Available: https://github.com/xopxe/rong
[10] H. Tazaki, F. Urbani, and T. Turletti, "Dce cradle: simulate network protocols with real stacks for better realism," in *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013, pp. 153–158.
[11] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4g lte networks," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '12. New York, NY, USA: ACM, 2012, pp. 225–238. [Online]. Available: http://doi.acm.org/10.1145/2307636.2307658
[12] A. Garcia-Saavedra, P. Serrano, A. Banchs, and G. Bianchi, "Energy consumption anatomy of 802.11 devices and its implication on modeling and design," in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '12. New York, NY, USA: ACM, 2012, pp. 169–180. [Online]. Available: http://doi.acm.org/10.1145/2413176.2413197
[13] J. Visca, M. Richart, J. Saavedra, J. Baliosian, and E. Grampin, "Buffer management in opportunistic networking," in *Proceedings of the Latin America Networking Conference on LANC 2014*, ser. LANC '14. New York, NY, USA: ACM, 2014, pp. 2:1–2:7. [Online]. Available: http://doi.acm.org/10.1145/2684083.2684085
[14] J. Baliosian, J. Visca, E. Grampin, L. Vidal, and M. Giachino, "A rule-based distributed system for self-optimization of constrained devices," in *IFIP/IEEE International Symposium on Integrated Network Management, 2009. IM'09.*, June 2009, pp. 41–48.

Appendix 3

Stochastic Models for Opportunistic Networks

Jorge Visca, Matías Richart, and Javier Baliosian. «Stochastic Models for Opportunistic Networks». In: *2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW)*. 2019, pp. 1–6. DOI: 10.1109/WCNCW.2019.8902544

Author's contribution The author provided the analytical model presented in the article and its evaluation.

Stochastic Models for Opportunistic Networks

Jorge Visca*, Matías Richart* and Javier Baliosian*†
*University of the Republic, Uruguay †Universitat Politècnica de Catalunya, Spain {jvisca, mrichart}@fing.edu.uy, javier.baliosian@upc.edu

***Abstract*—Opportunistic Networks are networks in which data delivery is achieved taking advantage of fleeting and random encounters between mobile nodes. To study such networks, their models must take into account the stochastic nature of the processes involved. In this work we show how results from epidemiology can be used to study the behavior of opportunistic algorithms. In particular, we apply a Markov model for a logistic birth/death process to an epidemic networking deployment. The method is based on analyzing the expected lifetime of messages in the network, and allows to model networks were nodes have a limited buffer capacity.**

I. Introduction

There is a class of mobile networks known as Opportunistic Networks (OppNets), where most of the time there is no end-to-end path between a data source and its destination. Actually, nodes move and encounter each other only sporadically, and they must carry data for prolonged spans of time until a data exchange opportunity arises. For this reason, algorithms supporting this class of networks are called *carry and forward* algorithms. Routing algorithms must make the best possible use of the storage and bandwidth available as to maximize the delivery rate, or the proportion of successfully delivered messages. For a brief history of the concept and a description of the various approaches used see our previous work [1].

OppNets show very complex interactions between mobility, data patterns and routing algorithms. Designing and tuning them is a complex process, highly dependant on the network's behavior. To tackle this complexity, this work develops a theoretical model of an OppNet protocol. In particular, the model allows to predict the life-cycle of messages in the nodes' buffers and through the network. This allows to predict basic network metrics, such as delivery rates and latencies.

The proposed model is developed using results and methods of epidemiology, this is, the study of the propagation of diseases through susceptible populations. This is an approach widely used to study the behavior of OppNets, making an analogy between a disease and a message that propagates through a network. However, a common limitation of this approach is that the interaction between multiple biological diseases is weak and rarely studied. On the contrary, an OppNet must usually serve multiple messages simultaneously. These messages compete for a shared set of resources: airtime, buffer space, computing power, etc. In this work, we improve existent epidemiological models in order to represent more accurately OppNets behavior, where buffer space is a scarce resource and several independent message-flows must be routed.

The paper is structured as follows. In Section II previous work related to OppNet modelling is presented. The network to be studied is described in Section III, and Section IV studies the modelling of mobility related attributes. Section V presents the epidemic model and applies it to extract some network metrics. Finally, conclusions and future work are presented in Section VI.

II. Related work

Given the wide variety of protocols and network topologies there is a considerable body of work studying the dynamics of OppNets. Some of the most representative and closely related works are described bellow.

The authors of [2] perform a model-driven analysis of a Gossiping-based protocol used for disseminating information in a mesh network. In a gossiping protocol nodes select randomly a destination node with which exchange a random subset of packets. The work builds a differential equation model for the dissemination process. Though neither the protocol nor the network model is the same as in this work, several analytical tools are shared.

In [3] a framework for modelling OppNets is proposed. The model is based on the construction of systems of ordinary differential equations (ODEs). The network model assumes that the contacts and Inter-packet times follow an exponential distribution of known rate, and there is a fixed set of data flows, with each node being the source and destination of a single flow.

The authors of [4] use a different tool for analyzing the performance of an OppNet, a Markov chain where each state represents the number of nodes a message is available on. The protocols studied are *two-hop multicopy* and *unrestricted multicopy* used to propagate a single message. No interaction with other messages in the network and associated competition for buffer space is considered. In fact, the number of nodes with a copy of a message is modelled as strictly growing until the message reaches destination. Despite this limitation, the approach used is of interest.

Another work that studies the stochastic behavior of a OppNets is [5]. It uses a similar Markov model, and has an insightful description of the similarities between an opportunistic protocol and an epidemic. It has similar limitations to the previous work, as it does not take into account the possible interaction with other data flows.

Most works make use of two main techniques: a deterministic model based on differential equations, and stochastic analysis based on a Markov chain representation. Both are used in this work, though our stochastic model is more realistic in that it models a practical OppNet protocol (Epidemic Network), and considers the fact that the buffer space is limited.

978-1-7281-0922-0/19/$31.00 ©2019 IEEE

III. An Epidemic Protocol Model

To build a complete model for the network it is necessary to model two behaviors, the encounters between pairs of nodes, and the forwarding decisions made by the routing algorithm.

Our network model is composed of N mobile nodes, moving in a square region of h meters wide. The nodes move using a *random direction* mobility pattern (RD model), with an average speed of $\bar{v}$ meters per second, and a expected epoch length (the distance traveled in a straight segment) of $\bar{L}$ meters. New data is produced on all nodes equally, with a constant total rate of $r_{arrival}$ new messages per second. Nodes are considered to be *in range* if they are closer than R meters and, for simplicity, we assume that encounters are long enough to exchange all the requested messages.

The nodes implement epidemic routing protocol, with buffers with M message slots, and random replacement (RR) buffer policy, where random slots are selected for eviction. We use it instead of the more usual FIFO because it is stateless, which simplifies modeling. It has to be noted that the expected lifetime of a message under RR and FIFO are equal, namely, M new message arrivals. The probability distributions for lifetimes are different, though.

The model parameters are summarized in Table I.

TABLE I: Model parameters

N	Number of mobile nodes
$h \times h$	Size of the area where nodes move
$\bar{v}$	Average speed of nodes
$\bar{L}$	Expected distance travelled before changing direction
M	Buffer size in messages
$r_{arrival}$	Rate of new message arrival in the network
R	Wireless range of nodes

IV. Forwarding probability

As we assumed that encounters are long enough to exchange all the messages requested, the main characteristic of interest is the probability of two nodes meeting inside a time window.

It has been shown that the relative speed between two mobile nodes under RD mobility is $\overline{v}' = \hat{v}_{rd}\overline{v}$, with $\hat{v}_{rd} = 1.27$ [6]. The exact value is $\frac{4}{\pi}$, tough we have not included the computation in this work for brevity. This means that the meeting time (the time between encounters for two mobile nodes) is equivalent to the hit time (the time to encounter a fixed node) where nodes move faster by a factor of 1.27. This is useful because the hit time is easier to compute than the meeting time.

In the RD mobility model, nodes homogeneously cover all the area over time [7]. As a node with radio range R moves with velocity $\bar{v}'$, it covers an additional area of $2R\bar{v}'$ per unit of time. The probability that a given point is contained in that said area is:

$$\lambda = \frac{2R\bar{v}'}{h^2} \tag{1}$$

Thus, we model the process of two mobile nodes meeting as a Poisson process with rate λ (for a similar approach, see [4]). Notice that after changing the direction of movement, a node can cover some area twice due to overlap in the swept areas. This effect is minor if $\bar{L} >> R$.

The expression in Equation (1) corresponds to the rate of encounters between a pair of nodes. When there are N nodes, we must sum the rates of encounter carefully as not to count encounters twice from each of participating nodes. Thus, as long as $NR^2 << h^2$, the rate of encounters among N nodes is:

$$\lambda_N = \sum_{i=1}^{N-1} i \times \lambda = \lambda \frac{N(N-1)}{2} \tag{2}$$

V. Analytic model for a simple OppNet protocol

In this section we develop a method for obtaining metrics of interest for a Epidemic Protocol when the rate of encounters is known. In particular, we are interested in characterizing two behaviors: how messages are copied between nodes; and how messages are evicted from buffers.

To study epidemic routing in more detail, it is useful to resort to methods developed in epidemiology [8], on which the OppNet algorithm itself is inspired. The analogy is that a message that is copied from one node to another is akin to a contagion of a disease occurring between two individuals. A healing, in turn, corresponds to a message being evicted from a node. Notice that unlike real healing, evictions happen only during encounters, when new messages may have to be stored in the buffer.

First, we apply the standard approach used in epidemiology of finding the equilibrium point between the replication and healing [9] to approximate the fraction of infected population. Let A and B be two nodes participating in an encounter, where data is copied from A to B. Let the messages be homogeneously distributed, with a_x the probability the message x is found on a given node (the *availability* of x). This implies $\sum_{x \in S} a_x = M$, where S is the collection of all messages available in the network at the moment of the encounter, and M is the buffer size.

Let define the *expected availability* metric Q as

$$Q = \mathbb{E}(a_x) = \frac{M}{|S|} \tag{3}$$

Notice that Q depends on M which is a parameter of the network, and $|S|$, which is a unknown value dependant on the dynamics of the network. Assuming that all buffers are full, the maximum possible amount of distinct messages $|S|$ is MN (each message appears only once through the network), and the minimum is M (when all nodes' buffers have identical content.) This results in the bounds $\frac{1}{N} \leq Q \leq 1$.

We compute the copy and eviction rates r_{copy} and r_{evict}. For this purpose we will compute the probabilities of copy and evict events, and then obtain the expected number of said events per node encounter.

The epidemic algorithm copy a message from A only if it is not found on B, so the probability a message $x \in S$ is copied from A to B is

$$p_{copy}(x, A, B) = p_{x \in A} p_{x \notin B} = a_x(1 - a_x) \tag{4}$$

We consider $x \in A$ and $x \notin B$ as independent events. This assumes that the content of the buffers are not correlated. This assumption depends on two properties. First, the network must be big enough that the impact of conditioning a node to have a message has negligible impact on said message's availability; second, the mobility model must mix nodes fast

enough as to not introduce time dependence. Because in RD mobility nodes move in straight segments, it is improbable that a node can be encountered twice in a short period of time. This might not be true on other mobility models, like for example in a Random Walk. Accepting independence allows us to take expected values:

$$\mathbb{E}(p_{copy}(x, A, B)) = \mathbb{E}(p_{x \in A})\mathbb{E}(p_{x \notin B}) = Q\,(1 - Q) \tag{5}$$

The number of copied messages during an encounter $\#C$ and its expected value are:

$$\#C(A, B) = \sum_{x \in S} p_{copy}(x, A, B) \tag{6}$$

$$\mathbb{E}(\#C(A, B)) = \sum_{x \in S} \mathbb{E}(p_{copy}(x, A, B)) \tag{7}$$

$$= \sum_{x \in S} Q(1 - Q) = M\,(1 - Q) \tag{8}$$

To study the behavior of replacements in a buffer, we will first observe that the stable state for a node is to have its buffer full. In fact, there is no advantage for a node in dropping data before finding a more valuable piece of information to fill its slot. Also, because we consider buffer space a limited resource, the transient state while buffers are filling is short compared to the lifetime of the system.

When using *random replacement*, a message x is discarded when its slot is randomly selected for placing an incoming message y (x is *evicted*). The probability a message x is evicted from B during an encounter is:

$$p_{evict}(x, A, B) = p_{x \in B} \frac{1}{M} \#C(A, B) \tag{9}$$

$$\approx a_x(1 - Q) \tag{10}$$

The approximation above is based on the observation that $\#C(A, B)$ is independent from x, so we can replace $\#C$ with its expectancy.

Using again the independence of events argument, the expected number of evictions per encounter $\#E$ can be computed as:

$$\#E(A, B) = \sum_{x \in S} p_{evict}(x, A, B) \tag{11}$$

$$= \frac{1}{M} \sum_{x \in S} \#C(A, B) a_x \tag{12}$$

$$\mathbb{E}(\#E(A, B)) = \frac{1}{M} |S| \mathbb{E}(\#C(A, B)) \mathbb{E}(a_x) \tag{13}$$

$$= M\,(1 - Q) \tag{14}$$

Each copy event triggers a replacement so the expected number of both must be equal, as seen in the fact that $\mathbb{E}(\#C) = \mathbb{E}(\#E) = M\,(1 - Q)$.

A single meeting can trigger multiple copies and evictions, so while node encounters are a Poisson process, copies and evictions are not (in Poisson processes no simultaneous events are allowed). Nevertheless, we will approximate the copy and eviction processes using an exponential distribution with rates:

$$r_{copy} = r_{eviction} = \lambda_N M\,(1 - Q) \tag{15}$$

The quality of this approximation improves as the number of copied messages per encounter decreases. This is related to the mixing ability of the network's movement pattern, as mentioned when discussing the possible correlation between the content of the nodes' buffers.

After the initial warmup, newly generated messages must be placed in already full buffers. A natural solution is to place them using the same mechanism than used during the node encounters. This way, each new message triggers an eviction, so the effective eviction rate is $r^*_{eviction} = r_{eviction} + r_{arrival}$.

The model was verified against a simulation, available at [10]. We simulated nodes moving using Random Direction pattern on a square region $1000\mathrm{m}$ wide, with an epoch length of $500\mathrm{m}$ and with a velocity of $1\mathrm{m/s}$.

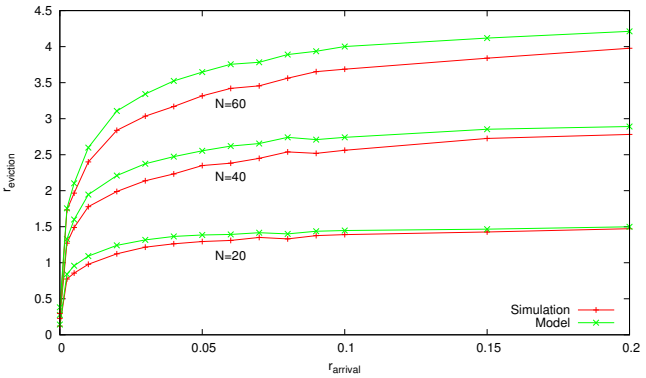


Fig. 1: Eviction rate modelling.

Figure 1 compares the eviction rates measured on the simulation, corrected for the arrival rate, with the $r_{eviction}$ approximation. The simulation is configured with 40 mobile nodes, and varies the buffer sizes. As outlined for Equation (15) the behavior of the copy rate r_{copy} is similar.

Preliminary analysis suggests that the discrepancies between the model and the simulation are rooted in the Poisson assumption for the copy event. As mentioned before, messages are copied not independently, but in batches when nodes meet. This explains why the model degrades as the buffer size increases.

A. Deterministic model

During the encounter of two nodes, a message x stored in one of them can be copied to the other node, or be evicted by another message with probabilities $p_{copy}(x)$ and $p_{evict}(x)$ respectively. If we assimilate the message x to a disease, and copy and eviction events as contagions and healings, we can describe the process as a *SIS* (susceptible-infective-susceptible) epidemic [9] in the sense that nodes do not develop immunity after an infection, or, in other words, a node can acquire a message again after evicting it.

The copy and eviction probabilities from Equations 4 and 10, considered as contagions and healings, correspond to a *stochastic logistic epidemic* process [11]. This model is used to represent the growth of a population in a limited environment. It is found in the literature with several different notations and parametrizations, so we proceed to define the parameters used here.

Let I be the number of nodes *infected*, out of a population of N individuals. The infection and healing rates $\alpha(n)$ and $\beta(n)$ depend on the number of infected nodes n. Both can be

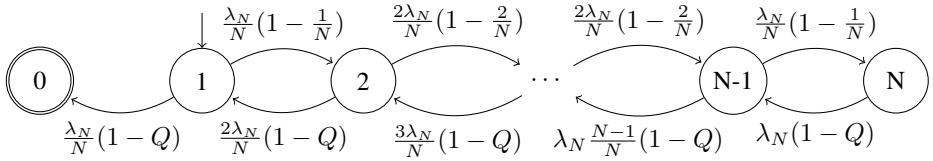


Fig. 2: Markov chain for the logistic process.

obtained from Equations 4 and 10, respectively. The resulting rates are:

$$\begin{cases}\alpha(n) & = \alpha\frac{n}{N}(1-\frac{n}{N}) \\ \beta(n) & = \beta\frac{n}{N}\end{cases} \tag{16}$$

The process is controlled by the parameters α and β. In our model these constants have values:

$$\begin{cases}\alpha & = \lambda_N \\ \beta & = \lambda_N(1-Q)\end{cases} \tag{17}$$

Notice that we do not deal with a single epidemy, but with multiple messages competing for limited buffer space, something not represented in the standard SIS model. This effect is represented indirectly by the Q parameter, which sums up the impact that the buffer composition has on the epidemic behavior. It is clear that this parameter does not affect the contagions, but only healings. Also notice that the value of Q changes as the network evolves, so β is not strictly constant. As it is discussed later, Q converges to a constant value, depending on the constant value of the arrival rate $r_{arrival}$.

The main parameter of an epidemic system is the *basic reproductive ratio* R_0, or the expected number of infections that can be caused by an infective through its lifetime, when the rest of the population is susceptible. Because in our case

$$R_0 = \frac{\alpha}{\beta} = \frac{1}{1-Q} > 1, \tag{18}$$

it means the disease becomes endemic. In other words, after starting with a few cases it establish itself on a stable portion of the population. We show this result next.

The first approach to study the evolution of $a_x = I/N$ is to approximate the system as a continuous deterministic model with infinite population, driven by the following nonlinear differential equation:

$$\frac{\mathrm{d}a_x}{\mathrm{d}t} = \lambda_N p_{copy} - \lambda_N p_{evict} = \lambda_N a_x(Q - a_x) \tag{19}$$

This equation can be solved explicitly, giving rise to a logistic curve. For our purpose this is not necessary, and it is enough for us to study its fixed points, which are its roots:

$$\frac{\mathrm{d}a_x}{\mathrm{d}t} = 0 \Rightarrow a_x = \begin{cases}0, & \text{repulsive} \\ 1-\frac{1}{R_0} = Q, & \text{attractive}\end{cases} \tag{20}$$

This means that all messages tend to have a common availability Q, the expected value for the availability. This is also the solution to the combinatorial problem for the fraction of nodes with a given message when all messages are equiprobable, which is $\binom{S-1}{B-1}/\binom{S}{B}$.

The p_{evict} rate from Equation (10) is strictly positive until $M = |S|$. This means that a network where there are no new messages introduced will converge to a configuration where there are M distinct messages in the network and all nodes' buffers are identical. Under more general conditions, when there are new messages arriving, $|S|$ is greater than M, and Q is strictly smaller than 1. Notice that when a message is just published it has a small availability of $1/N$, but it tends to increase (Equation (19)).

We are interested on the value of p_{evict}, which depends on the unknown expected availability Q. Our conjecture is that the value of Q depends on the dynamics of the network, this is, on the new-message arrival rate $r_{arrival}$. The structure of Q suggests that when $r_{arrival}$ is small, Q is close to 1, and as $r_{arrival}$ grows Q approaches $1/N$.

Intuitively, under a stable state the amount of active messages in the network must remain constant. In other words, the rate at which new messages are generated must match the rate at which messages are being dropped from the network. Dropping a message from the network implies its availability reached 0. The deterministic model, as described, does not provide an explanation on how this may happen, as the model predicts that the infection just increases until it settles to a fixed proportion of the population. This model ignores the stochastic behavior of the network, where the population varies due to randomness in buffers' content, encounter dynamics, etc. This stochastic behavior is studied in Section V-B.

B. Stochastic model

Notice that during an $A \Rightarrow B$ encounter a message x is copied only if $x \in A$ and $x \notin B$, while it can be evicted only if $x \notin A$ and $x \in B$. This means that p_{copy} and p_{evict} events are disjoint. Also, during a copy event the number of nodes possessing the message increases by one, and during an eviction decreases by one. This suggests using a Markov chain model, where states represent the number of nodes with a copy of a message. This is similar to the Markov chain used in [4], extended to support evictions, or links to "previous" states.

We define a continuous time Markov chain with $N+1$ states $s : \{0, ..., N\}$. The availability a corresponds to the state $s_a = \lceil Na \rceil$. Thus, the state associated to the fixed point of the deterministic model is $s_Q = \lceil NQ \rceil$. The state 0 is absorbing, this is, once a message is lost from all nodes, it can not appear again. The state 1 is the entry state, because all processes start with a single copy of a message being published by a node.

The transition rates are obtained from copy and eviction rates at the availability corresponding to each state, and are shown in Equation (21) and Figure 2. Notice that i/N is the discrete equivalent for a_x when message x is found on i nodes from a finite population of N. The eviction rates of a message increase with its availability. This is intuitive as the more copies of a message are available in the network, the higher the chances of being selected for an eviction. Copy rates have

a symmetrical distribution; if the availability of a message is small it is unlikely it will be selected to be copied. Conversely, if the availability of a message is high, it is unlikely a node without the message will be found. The highest copy rates are found when the message is on approximately half the nodes.

$$q_{i,j} = \begin{cases} \lambda_N \frac{i}{N}(1 - \frac{i}{N}), & \text{for } 0 < i < N, j = i + 1 \\ \lambda_N \frac{i}{N}(1 - Q), & \text{for } 0 < i \leq N, j = i - 1 \end{cases} \quad (21)$$

This is a finite birth-death Markov process with an absorbing barrier at state 0. Because the process has an absorbing state, and the Markov chain resulting from excluding said absorbing state is irreducible, it is sure that the process is eventually absorbed. Notice that this property does not appear in the deterministic epidemic model.

We are interested in the behavior of the chain between the moment it starts (at state 1) and the first time it visits the absorbing state 0, this is, the behavior of the network conditioned to the existence of at least one message. This is called the *quasi stationary distribution* (QSD) [12]. It can be shown that this matrix posses a QSD, which implies that the hit times for the absorbing state, or times to absorption, follow an exponential law, so, asymptotically there is a constant rate of absorption [13].

As the absorption follows an exponential law, there is a constant absorption rate which can be computed as the inverse of the expected time to absorption. On the other hand, under stable behavior the rate at which new messages are introduced in the network ($r_{arrival}$) must be equal to the rate at which they are being dropped from the network (r_{drop}). Thus, if the time to absorption depends on Q (a parameter of our Markov process), we expect to be able to determine the value of Q that is needed to provide a drop rate equal to $r_{arrival}$.

Let us suppose that we know the expected time to absorption $\tau_1(Q)$, this is the average life time of a message in the network as function of Q. Using this definition and equaling arrival and dropping rates, we can affirm that

$$|S| = \frac{B}{Q} = r_{arrival} \times \tau_1(Q) \quad (22)$$

The Equation (22) can be interpreted as stating that multiplying the message lifetime by the new messages' rate of arrival is equal to the number of messages being alive simultaneously.

Our approach to estimate the eviction rate as $\lambda_N \times p_{evict}$ can be summed up as follows:

1) Find the relationship between Q and τ_1, the expected lifetime of a message.
2) Find the stable Q that satisfies Equation (22)
3) Use the found Q to obtain p_{evict} from Equation (10)

1) Life-cycle of messages: The time to absorption τ_i, defined as the expected time needed to reach the absorbing state starting from state i can be found applying standard Markov theory. This allows to numerically compute the absorption times given the transition probabilities. Nevertheless, this result is not directly applicable because our transition probabilities and lingering times depend on the unknown parameter Q, which we are trying to calculate.

For a logistic process, an explicit expression for τ_i is found by [14] as

$$\tau_i = \frac{N}{\beta} \sum_{j=1}^{i} \sum_{k=j}^{N} \frac{1}{k} \frac{(N-j)!}{(N-k)!} \left(\frac{R_0}{N}\right)^{k-j} \quad (23)$$

As we are interested in absorption time starting from a single infection, and using α and β from Equation (17), we obtain the following expression for the expected lifetime of a message in the network:

$$\tau_1 = \frac{N}{\lambda_N} \sum_{k=1}^{N} \frac{1}{k} \frac{N!}{(N-k)!} \frac{1}{N^k} \frac{1}{(1-Q)^k} \quad (24)$$

This gives an exact solution from which the value of Q needed to provide a certain lifetime can be computed.

2) Markov model validation: To validate the solution from Equation (24), we implemented a Markov chain simulator [10] as outlined in [14]. The model is configured with $N = 100$ and $\lambda_N = 0.1$, and the results are shown in Figure 3.

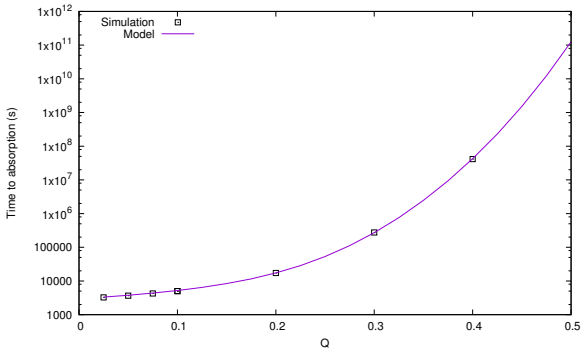


Fig. 3: Time to absorption model.

In Figure 4 the average time spent at different infection levels is displayed. The endemic states are computed as $\lceil NQ \rceil$, and it can be seen that in the more dynamic networks, with smaller Q, the process does not have enough time to settle on a stable state. As Q increases the infective times become long enough for the endemic behavior to become manifest.

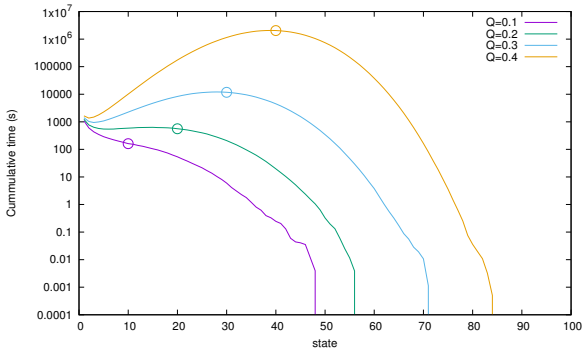


Fig. 4: Time spent in each availability level, and the corresponding endemic state.

3) Expected availability estimation: From Equation (22) and for a given $r_{arrival}$, Q can be computed as the unique solution of

$$\tau_1(Q) = \frac{B}{r_{arrival} Q} \quad (25)$$

Notice that when using the expression for τ_1 from Equation (24) both terms are monotonous and the equation has

a single solution that can be simply found numerically by bipartition in the domain $\frac{1}{N} < Q < 1$.

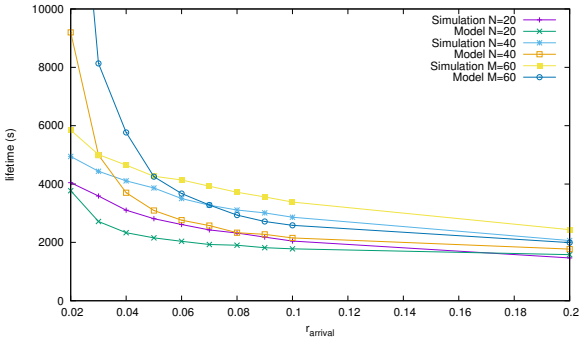


Fig. 5: Message lifetime modelling.

In Figure 5 the predicted lifetime of a message is compared with the results of the mobility simulation from Section V (setup with $N = 40$). Asymptotic behavior is correct, and a good match is achieved for bigger values of $r_{arrival}$. As mentioned before, with very small arrival rates the model predicts very high times to absorption, which are hard to confirm through simulation. An observation can be made that the lifetimes of messages are very sensible to the arrival rate of new messages.

4) Message survival rates: The rate at which messages are evicted from buffers is $\lambda_N M(1-Q)$ (Equation (15)), where Q can be computed following Equation (25). Remembering that the evictions are caused by both messages that arrive from other nodes and messages produced locally, this is $r^*_{eviction} = r_{eviction} + r_{arrival}$, the eviction rate that suffers each buffer slot is:

$$r_e = \frac{1}{N}\left(\lambda_N(1-Q) + \frac{r_{arrival}}{M}\right) \tag{26}$$

Then, the probability a message survives in a buffer during a time window T is $p_{store}(T) = e^{-r_e T}$.

Once we compute Q, the Markov chain model from Equation (21) is completely determined, and we can use it to obtain metrics of interest on the behavior of the network.

VI. Conclusion and Future Work

In this paper, we show that it is possible to develop a theoretical model for the behavior of an OppNet that takes into account the effect of multiple dataflows. This allows us to analytically study the metrics that impact on the network's performance.

We apply epidemic analysis to a fundamental opportunistic algorithm, epidemic routing, to model its behavior. Unlike previous works, we study a fully functional implementation that takes into account limited buffer capacities. We present a stochastic model, based on a Markov Chain, and a method to obtain the needed properties from the *Time to Absorption*, or expected duration of a process.

The model can be adapted to different mobility patterns or OppNet algorithms. In particular, variations of the epidemic routing protocol are of interest. Such extensions as disallowing message re-admittance, limited forwarding (were messages are copied a limited number of times), and different buffer policies (most notably FIFO) are of note.

In this work we analytically characterize the behavior of a synthetic model. A complementary and promising approach is to obtain the characterizing parameters empirically, for example from recorded traces from real networks, while using the same modelling strategy. This would allow to capture the behavior of complex real network deployments, which are hard to mimic with analytical models. Nevertheless, synthetic models remain useful as a tool to test the protocol's behavior under various conditions. This is important to validate the robustness of the protocols, and avoid overfitting the analysis to the available empirical data.

References

[1] J. Visca, "RON - opportunistic networks," Master's thesis, Universidad de la República, 2014.

[2] R. Bakhshi, D. Gavidia, W. Fokkink, and M. van Steen, "An analytical model of information dissemination for a gossip-based protocol," *CoRR*, vol. abs/0810.1571, 2008.

[3] X. Zhang, G. Neglia, J. Kurose, and D. Towsley, "Performance modeling of epidemic routing," *Comput. Netw.*, vol. 51, pp. 2867–2891, July 2007.

[4] R. Groenevelt, P. Nain, and G. Koole, "Message delay in mobile ad hoc networks," *Performance Evaluation*, vol. 62, pp. 210–228, 2005.

[5] M. de Souza Dias G. and M. S. R., "Epidemic sir model applied to delay-tolerant networks," in *30th Brazilian Telecomunications Symposium*, 2012.

[6] T. Spyropoulos, A. Jindal, and K. Psounis, "An analytical study of fundamental mobility properties for encounter-based protocols," *IJAACS*, vol. 1, no. 1, pp. 4–40, 2008.

[7] P. Nain, D. Towsley, B. Liu, and Z. Liu, "Properties of random direction models," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3, pp. 1897–1907 vol. 3, March 2005.

[8] R. M. Anderson, R. M. May, and B. Anderson, *Infectious diseases of humans: dynamics and control*, vol. 28. Wiley Online Library, 1992.

[9] M. J. Keeling and K. T. D. Eames, "Networks and epidemic models," *J. R. Soc. Interface*, vol. 2, p. 295, 2005.

[10] J. Visca, "Random Waypoint Epidemic Network Simulator." https://doi.org/10.5281/zenodo.1343585, 2018.

[11] H. Andersson and D. Boualem, "A Threshold Limit Theorem for the Stochastic Logistic Epidemic," *Journal of Applied Probability*, vol. 35, no. 3, pp. 662–670, 1998.

[12] S. Méléard and D. Villemonais, "Quasi-stationary distributions and population processes," *ArXiv e-prints*, dec 2011. http://adsabs.harvard.edu/abs/2011arXiv1112.4732M.

[13] D. Clancy and P. K. Pollett, "A note on quasi-stationary distributions of birth–death processes and the sis logistic epidemic.," *Journal of Applied Probability*, vol. 40, no. 3, p. 821, 2003.

[14] R. H. Norden, "On the distribution of the time to extinction in the stochastic logistic population model," *Advances in Applied Probability*, vol. 14, no. 4, pp. 687–708, 1982.

Appendix 4

A Model for Route Learning in Opportunistic Networks

Jorge Visca and Javier Baliosian. «A Model for Route Learning in Opportunistic Networks». In: *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. May 2022, pp. 1–4. DOI: 10.1109/NOMS54207.2022.9789826

Author's contribution The author provided the ONM model presented in the article, as well as the design and implementation of the BOW algorithm. He also designed and implemented the test scenario and provided the analysis of the results.

A Model for Route Learning in Opportunistic Networks

Jorge Visca* and Javier Baliosian*
*Universidad de la República, Uruguay
{jvisca,baliosian}@fing.edu.uy

***Abstract*—Opportunistic Networks (ONs) are complex systems where nodes move and meet stochastically. For data to be transmitted, it must be copied between nodes and then carried around through multiple hops. Because the network is continuously evolving and can be highly fragmented, the usual connectivity graph representation, typically used to compute routes or doing offline traffic engineering, is not suitable.**

This article proposes a model specially designed for route management on ONs, and we show its usefulness by implementing an ML-based route management algorithm for those networks. This algorithm is trained offline over the model built from historical traces of a particular network setup and then deployed to make fast, real-time forwarding decisions.

I. Introduction

An *Opportunistic Network (ON)* is a mobile network where there might be no continuous end-to-end path available, and data delivery must rely on nodes opportunistically exchanging packets as they move around and meet. Thus, nodes must store received data in local buffers for prolonged periods, and a message might have to be stored and handled over by several nodes on its path to a destination. Depending on the network this period might be hours, days, or even more. Thus, the algorithms that support this kind of network are called *carry-and-forward* protocols, and the networks themselves are sometimes called *Delay-* or *Disruption-Tolerant Networks (DTNs)*.

In this work, we present two complementary proposals. First, we present a method for modeling ONs. This model is based on well-known abstractions and allows the use of powerful existing tools. Namely, the model consists of a graph that captures both the networking aspect (which nodes meet) and the temporal behavior (how the network connectivity evolves in time). We then use this model to build route management mechanism based on routing information extracted through machine learning techniques.

II. ON Network model

The usual model for data networks is a graph where the nodes are the network devices, and the edges are data links that interconnect them. A message can be routed through the network if a path connects the source with the destination on the corresponding modeling graph. This abstraction is not readily applicable to ONs because the nodes are mobile, and the network topology can be very disconnected and continuously changing. Several proposals were made to overcome that problem, such as *temporal networks* [1], where a time-process is associated to each graph component, or representing the network as a set of graphs, each graph representing the network in a given moment in time. All these methods are difficult to manipulate and have limited and hard to use analytical tools available.

We propose applying *spatio-temporal graphs*, where nodes have an associated place in space and time, that can be used to capture the evolution of physical processes [2]. Using this representation, we propose building a single static graph to represent the whole network's behavior. This graph has straightforward semantics and easily applies many of the tools available to study and manipulate graphs. In our model, a graph node does not represent network devices as in usual networks; thus, to avoid confusion, we will refer to the mobile devices as *agents*.

The graph used in our model is composed of two classes of edges. The first class is a *delivery* edge, representing an encounter between two agents at a given time. This class of edges has an associated time of occurrence and can have more data attached, such as duration, bandwidth, etc. The two end-nodes represent the agents participating in the encounter and their corresponding state. Thus, an agent can have multiple nodes associated, one for each encounter in which it participated. A message traverses a delivery edge when it is copied between two agents during an encounter.

All the nodes modeling a single agent through time are chained together in order by the second class of edges, called *survival* edges. These edges connect the state of a given agent during an encounter with the same agent in the subsequent encounter. The naming comes from the idea that

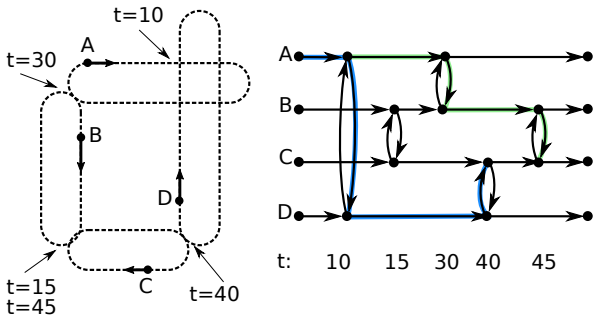


Fig. 1: Model construction

978-1-6654-0601-7/22/$31.00

when traversing a survival edge, a data unit is "surviving" in an agent's buffer in the time that passes between two encounters.

We outline the construction of this graph with an example in Fig 1. The ON consists of four agents A, B, C, and D. The agents move in closed trajectories (the dotted lines), meeting as they move in the points indicated with arrows.

A path in this graph represents a possible propagation history for a message. This path consists of a sequence of transmission and survival edges. To be considered as arriving at an agent, a message must have reached any of the agent's nodes, with the associated time being the arrival time.

A. Application

As an application example, we use the model to capture the behavior from the *RioBuses* dataset [3]. The dataset consists of mobility traces for the buses in Rio de Janeiro city in Brazil. It contains traces for over 12,000 buses moving on 725 bus lines, taken through October 2014, containing GPS data (time, position, velocity), the Bus unique identifier, and the Line being served at the moment.

We assume we want to collect data from the buses using an ON. The idea is that buses periodically produce data (e.g., ticket sales, occupancy level measures) and possess a short-range radio interface (e.g., WLAN) that allows exchanging data between buses when meeting on the road. The collected data is to be delivered to one or several Internet-connected fixed collection points using an ON.

The resulting graph for a single day has in the order of 20.000.000 nodes and 40.000.000 edges, split roughly equally between survival and transmission edges. Most of the transmission edges are paired in bidirectional links, except those that lead to a collection agent.

B. Routing protocols

Some ON algorithms do not keep any knowledge and are variations of controlled flooding (also known as *dissemination* algorithms). Other algorithms collect data on the network's behavior and attempt to make good decisions intelligently (*Context based* algorithms). This is because many real-life networks exhibit data and mobility patterns that can be taken advantage of, as in our application.

Binary Spray and Wait (BSW) is a dissemination-type protocol based on Epidemic Routing. In Epidemic protocols, data propagates through the network akin to an infectious disease: when two agents meet, the data "infects" the yet non-exposed agent. BSW implements a propagation control mechanism in which a data packet is assigned a number of virtual copies at emission time, which will be the maximum number of instances of the packet through the network. In each encounter, an infecting node handles half of its copies to the infectee.

Another well-known ON protocol is Prophet, a Context-based protocol that tries to learn a delivery predictability parameter for each agent. For this purpose, agents maintain a quality metric assigned to every other agent in the network that measures how good an agent considers itself for delivering data to that agent as a destination. The basic idea is that the delivery probability increases as agents meet and decreases as time passes without contact. Then, data is handled only to nodes with a delivery probability greater than their own when forwarding.

C. Simulating protocols

ON protocol simulation on the proposed model is made differently than on a standard ON simulator such as ONE. In ONE, agents are simulated as they move through their trajectories, encounter other agents, and the protocol reacts to network events. Routing messaging and data transmission are run together as part of the same simulation. This approach has historically limited the size of the studied networks to the order of a thousand nodes [4]. To improve the scalability, we observe that independent processes produce messages associated with routing and data transmissions. In our model, the encounters are precomputed, and it is possible to simulate the propagation of routing messages and their impact on the agent's configuration separately from actual data transmissions. This allows to precompute the internal routing state of agents in every moment of interest (i.e., graph node) and subsequently evaluate the behavior of many data generation scenarios.

III. Using the ON Network Model: the BOW Protocol

Our proposal's base is an ML algorithm that will manage routes and set forwarding decisions, trained from an oracle that provides optimal decisions. We will train this system on the traffic patterns of a single day and then verify how well it generalizes to other days.

A. The oracle

The oracle is tasked with computing the optimal decisions used as a reference when training the predictor. The optimal decisions are the edges that are part of the shortest paths from source to destination. The length of the path depends on the cost function used. We propose two simple possibilities:

- Oracle-Short. The cost of a path is the number of transmissions it uses. The cost of a transmission edge is 1. This cost function leads to simpler trajectories with fewer hops, with the intent of reducing the power consumption and radio interference.
- Oracle-Fast: The cost of a path is its latency; this is the time passed between the message emission and its arrival. The cost of a survival edge is the difference in the timestamps of the end nodes.

The graph is of considerable size and has a regular structure formed of sparsely interconnected chains. The path computing is done using *Uniform-cost Search* (UCS), a variation of Dijkstra where candidate nodes are added incrementally as the search horizon advances. The traversal stop condition is when a node corresponding to a collection agent is visited.

A complete run of the oracle over a day's graph takes under 10 minutes on an i3 class desktop PC, and the output is the subset of the transmission edges that participate in a shortest

delivery path. With Oracle-Fast, a typical selected set consists of around 3.000.000 edges when having a single collection point and 4.500.000 with four collection points. Under Oracle-Short, the numbers are 450.000 and 510.000, respectively.

Figure 2 shows a $1/500$ sample of the locations of bus encounters through 2014/10/01. The encounters in red are the ones selected for transmission by the Oracle-Fast, while the ones selected by Oracle-Short are in blue. The oracle finds a very high delivery rate of around 0.95 from all possible starting nodes. In the remainder of this work we will present results using Oracle-Short.

B. Transmission prediction

The oracle data is used to train a binary predictor, which the forwarding algorithm will use to classify new transmission opportunities as they occur. At the meeting moment, an agent has the following attributes from both meeting buses that can be used to classify the encounter:

- Timestamp, the current time in seconds from midnight.
- Longitude, latitude, and speed, obtained from the GPS unit.
- Average speeds over the last 10min, in the Latitude and Longitude directions (accumulated from the GPS unit).
- Bus unique identifier and assigned bus line.

The data from the meeting bus is obtained over the newly formed wireless link. Time, position, and speed features are captured from a GPS unit and thus do not depend on a per-bus configuration. These features are numerical. The bus id and bus line data are categorical (a value from a set of possible values).

In our application, the classifier is executed in real time by devices with limited computing resources, so the classification cost is of great importance. This suggests the use of Multi-Layer Perceptron, Logistic Regression, or Decision Tree classifiers.

C. BOW Protocol

The learned forwarding policy is executed on the agents to decide if a newly met agent is a good candidate for receiving the message. A redundancy mechanism is added to increase the robustness of the protocol, based on the BSW mechanism with an initial copies count that is diffused through the network. However, instead of doing it blindly on the first encounters as BSW does, messages are handled only over encounters deemed suitable by the classifier.

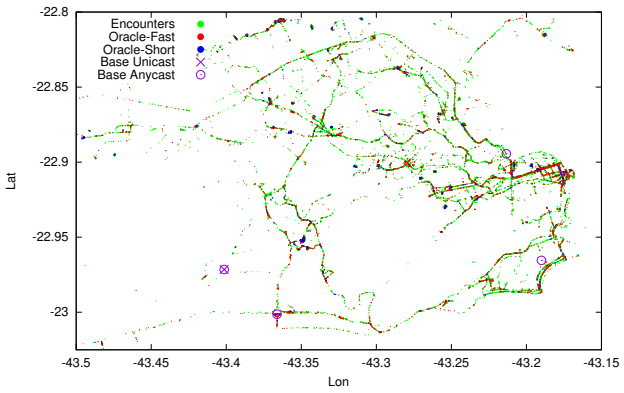


Fig. 2: Encounters and oracles' selections.

The classification is performed based on the participating nodes' features. In the following sections, two feature sets are studied:

Numerical features: Only position and speed features are used. These are numerical features obtained from a GPS unit. These features do not depend on per-agent configuration.

Full features: Besides the previous, Bus Id and Bus line features are used. Those are categorical features without an implicit order. When using these features, One-Hot encoding is applied.

The ML methods used are:

1) BOW-DT: The classifier is a Decision Tree of depth 15.

2) BOW-LR: The classifier is a Logistic Regression Classifier. Numerical features are normalized over the training set.

3) BOW-MLPC: The classifier is a Multi-Layer Perceptron using Relu as nonlinearity. When using only numerical features, two hidden layers are created, sized (40,20). With the full features, three hidden layers sized (3,2,2) are used. Numerical features are normalized over the training set.

All predictors were trained over a subsample of 100.000 edges taken from one day, evenly distributed between *true* and *false* examples. This dataset was split between training and development sets in an 80/20 ratio. The development set was used to select the best predictor parameters. Trained predictors were evaluated over the full trace of a different day.

In our scenario, every bus in the network emits a single message at 12:00.

Figure 3 show the simulation of the different protocols. BSW and BOW performance is shown with different numbers of initial copies of a message, while Prophet does single message forwarding.The BOW-Oracle curve corresponds to the BOW algorithm taking the optimal forwarding decisions from the *Short* offline solver applied to the tested day.

The worst performance is obtained with BSW. Logistic Regression provides slight improvement when using only numerical features (BOW-LR). In turn, Decision Trees and Multilayer Perceptrons do improve in this case (BOW-DT and BOW-MLPC). Finally, the best performance is obtained with Logistic Regression and Multilayer Perceptron using the full features set, i.e., including bus identity and bus line data (BOW-LR-FULL and BOW-MLPC-FULL). BOW with full features set reaches and surpasses Prophet performance using between 16 and 32 copies.

Figure 4 shows the proportion of delivered messages as time from the initial transmission passes. BOW and BSW are configured with 32 replicas. Prophet behaves better than BSW, but it shows a slow delivery, which indicates the propagation through a sub-optimal trajectory. When combined with an effective predictor, BOW displays better latency characteristics than Prophet and is a more robust protocol than BSW.

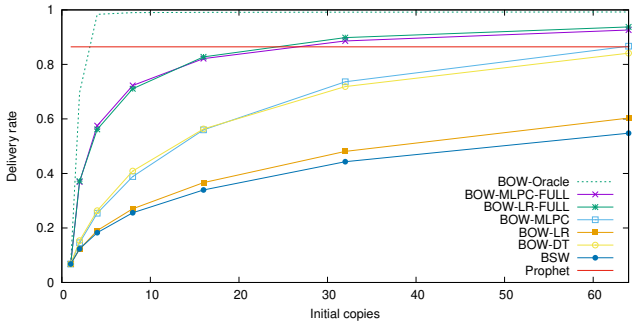


Fig. 3: Delivery rates.

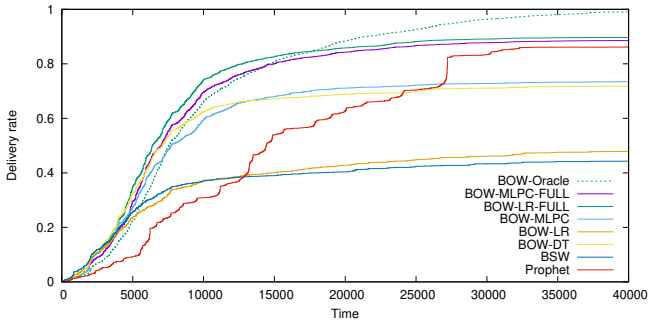


Fig. 4: Delivery-time behavior.

D. Traffic overhead comparison

Radio transmissions are energetically costly, and because the medium is shared, nodes communicating degrade the transmission conditions for other nodes in the vicinity.

The overall traffic produced when the messages emitted are of size S_M is:

$$T_{tot} = E \times S_R + F \times S_F + T \times S_M \quad (1)$$

The number of encounters E is obtained by counting the transmission edges from the model graph. The values for the number of forwarding evaluations F and message transmissions T are obtained from the simulations described in Section II-C. Only Prophet produces routing data S_R, exchanging a predictability value as a floating-point number for every bus in the network. With S_F, to decide whether to forward Prophet transmits the delivery predictability; BOW transmits five numerical features as floating-point numbers (from the GPS unit), and the bus and line identifiers as 10 bytes long strings.

Figure 5 shows the delivery rate and total traffic produced in the network when generating messages of 100.000 bytes, uniformly distributed in time, in all the buses in the network.

It can be seen that Prophet has a high fixed cost due to routing messaging, independent from the data transmitted. The protocol does not scale well to very big networks. BSW has a high overhead due to retransmissions, which are not used effectively as the delivery rate remains low. Finally, BOW

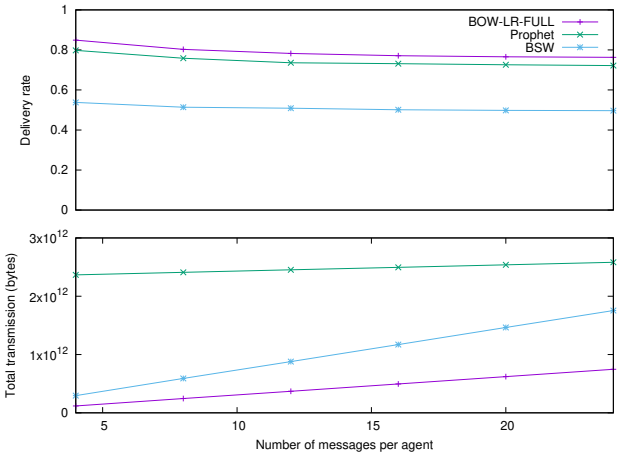


Fig. 5: Behavior under increasing message load.

displays good transmission characteristics with low network overhead.

IV. Conclusions and Future Work

In this work, we presented a model for ON networks that is simple to build and manipulate. To show its usefulness, we worked with it and developed BOW, an ML-based route management mechanism based on offline training, and shown it to be flexible and competitive with established protocols.

The BOW protocol shows good performance while using only offline training. As the training is made against a general oracle, the system can be trained to handle multiple use cases and different objectives. At the same time, we have shown the ability of the algorithm to capitalize on features beyond the ones generally used in routing algorithms. Having an offline oracle allows to easily include application-specific data, such as bus lines in our case, which are of great importance for efficient data routing. An application-agnostic protocol would have difficulties incorporating such information.

Finally, we believe that the presented graph model can be used as an efficient simulating platform.

References

[1] P. Holme and J. Saramäki, "Temporal networks," *Physics Reports*, vol. 519, no. 3, pp. 97–125, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0370157312000841

[2] M. R. Dale, *Spatio-temporal Graphs*. Cambridge University Press, 2017, p. 222–251.

[3] D. Dias and L. H. M. K. Costa, "CRAWDAD dataset coppe-ufrj/riobuses (v. 2018-03-19)," Downloaded from https://crawdad.org/coppe-ufrj/RioBuses/20180319, Mar. 2018.

[4] V. Kuppusamy, U. M. Thanthrige, A. Udugama, and A. Förster, "Evaluating forwarding protocols in opportunistic networks: Trends, advances, challenges and best practices," *Future Internet*, vol. 11, no. 5, 2019. [Online]. Available: https://www.mdpi.com/1999-5903/11/5/113

Appendix 5

rl4dtn: Q-Learning for Opportunistic Networks

Jorge Visca and Javier Baliosian. «rl4dtn: Q-Learning for Opportunistic Networks». In: *Future Internet* 14.12 (2022). ISSN: 1999-5903. DOI: 10.3390/fi14120348. URL: https://www.mdpi.com/1999-5903/14/12/348

Author's contribution The author provided the design and implementation of the rl4dtn algorithm. He also designed and implemented the test scenario and provided the analysis of the results.

future internet

MDPI

Article

rl4dtn: Q-Learning for Opportunistic Networks

Jorge Visca * and Javier Baliosian

Faculty of Engineering, Universidad de la República, Montevideo 11600, Uruguay
* Correspondence: jvisca@fing.edu.uy

Abstract: Opportunistic networks are highly stochastic networks supported by sporadic encounters between mobile devices. To route data efficiently, opportunistic-routing algorithms must capitalize on devices' movement and data transmission patterns. This work proposes a routing method based on reinforcement learning, specifically Q-learning. As usual in routing algorithms, the objective is to select the best candidate devices to put forward once an encounter occurs. However, there is also the possibility of not forwarding if we know that a better candidate might be encountered in the future. This decision is not usually considered in learning schemes because there is no obvious way to represent the temporal evolution of the network. We propose a novel, distributed, and online method that allows learning both the network's connectivity and its temporal evolution with the help of a temporal graph. This algorithm allows learning to skip forwarding opportunities to capitalize on future encounters. We show that explicitly representing the action for deferring forwarding increases the algorithm's performance. The algorithm's scalability is discussed and shown to perform well in a network of considerable size.

Keywords: opportunistic networks; DTN; Q-learning; reinforcement learning; routing

Citation: Visca, J.; Baliosian, J. rl4dtn: Q-Learning for Opportunistic Networks. *Future Internet* **2022**, *14*, 348. https://doi.org/10.3390/fi14120348

Academic Editor: Eirini Eleni Tsiropoulou

Received: 17 October 2022
Accepted: 20 November 2022
Published: 23 November 2022

1. Introduction

Opportunistic networks (ONs) are networks supported by mobile devices that meet sporadically. As data can only be transmitted during encounters, nodes must carry it with them for potentially prolonged periods until an opportunity to pass on the data arises. Unlike a traditional data network, ONs are formed by multiple network partitions that change continuously, and data delivery depends on multiple hops that do not necessarily form an end-to-end path at any given moment.

1.1. Applications for Opportunistic Networks

These kinds of networks (sometimes called delay- or disruption-tolerant networks—DTNs) have multiple applications. For example, sensor networks are composed of widely distributed low-cost devices that collect environmental data. This application is an important component of smart cities and Internet of Things (IoT) research [1]. The networking component is tasked with collecting all the produced data to be processed. An ON can transmit data without depending on external infrastructure, sometimes under very adverse conditions [2].

Opportunism can be used in conventional networks to reduce the load on the infrastructure and better use network devices' computational, storage and communication resources. Device2Device (D2D) functionality allows LTE network terminals, or smartphones, to download directly from neighboring terminals instead of doing it through the cellular network [3]. That can significantly reduce latency, power consumption, and infrastructure costs [4].

There are many applications wherein vehicles want to exchange information within a vehicle fleet: Cars could warn other vehicles of a dangerous event using a vehicular ad hoc network (VANET); A team of fire-fighting robots might coordinate tasks and provide

communication infrastructure while searching for survivors in a building on fire; a swarm of UAVs could communicate to keep formation; etc.

1.2. Network Properties

In general, ONs are intended to support communications over *Challenged Networks* [5], which are described by the following properties:

Disconnection: Nodes can be outside other nodes' range for long periods. This can happen for single nodes or groups.

Low duty cycle: Nodes can shut down network interfaces to save battery, only briefly coming online.

Limited resources: Nodes can have limited battery power supply and low computational resources, such as CPU and storage.

Low bandwith: Usually over wireless links, subject to high losses and latencies.

The characteristics mentioned above lead to the need for long queuing times, where messages must be held in buffers for minutes, hours, or days at a time, unlike the milliseconds usual in structured networks. Because of this, routing algorithms supporting ONs are called "carry-and-forward" algorithms.

ONs are usually highly stochastic systems. There are three main sources of randomness. In the first place, mobility. While there are ONs with deterministic behavior (e.g., orbiting spaceships [6]), most mobile devices do not exactly have repeating trajectories. Examples include service robots in a warehouse, cars in a city, or people in a building.

The second source of randomness is the generation of data to be transmitted. Data generation at the nodes is usually an external process outside the control of the network.

Finally, wireless communications are subject to interference from external devices and other environmental conditions.

1.3. Performance Metrics

As a result, ONs operate under conditions hostile to data delivery, and there exists the possibility that a message will be lost. Thus, the main performance metrics of an ON are *delivery rate* and *latency distribution*. The delivery rate is the fraction of emitted messages that reach the destination. The latency is the travel time of the delivered messages. Notice that latency is only computed over messages that reach the destination. Higher delivery rates frequently depend on higher latencies, as harder-to-deliver messages must be held in the network waiting to be delivered. Because links are supported by shared and limited mediums, typically wireless, the *communication overhead* is also important. Excessive communications utilization can lead to the degradation of the network and thus impact the performance. The number of transmissions also impacts directly power consumption, a scarce resource in many mobile applications.

1.4. Opportunistic Algorithms

Due to the wide variety of use cases, many algorithms are developed [7]. A defining property of an ON algorithm is whether it generates extra message copies. *Forwarding* protocols route a single copy of the message; *Flooding* protocols produce multiple replicas to increase the delivery rate at the cost of increased network overhead.

Two big classes of algorithms are *stateless* and *stateful*. The former is based on efficiently disseminating the copies of a message, so it is also known as "dissemination-based". The later algorithms collect data on the ON behavior and attempt to use this knowledge to make more effective routing decisions. Many different techniques are used. Distance-vector style algorithms are popular, but more complex methods are also proposed.

We propose using machine learning (ML) techniques, specifically reinforcement learning (RL) and Q-learning, to solve the routing problem. As we will see below, multiple works use this approach. However, our proposal uses a particular way of modeling the network that captures more information from the network's behavior. In our previous work [8], this model was applied for ML routing in ON; however, in that work, the training

was performed offline using recorded traces. In this work, the ML process is performed online, and the model is built and maintained during the network's lifetime. We show that the application of this network model improves the network's performance compared to the conventional Q-learning application. It is also competitive with conventional protocols, especially regarding network overhead, as it does not depend on additional traffic for routing information.

2. Related Work

One of the simplest ON algorithms is *Epidemic Routing* [9], a stateless and flooding-based epidemic protocol wherein messages behave as contagious diseases, infecting new devices when they are met. This method achieves the highest possible delivery rate and the lowest latency possible at the cost of a massive load on the network, as every forwarding opportunity is taken, and every node ends up with a copy of every packet.

Various algorithms improve upon Epidemic Routing to reduce the network load. For example, *BSW* [10] is a version of *Spray and Wait* [11] which proposes the following method for controlling the propagation: first, messages have attached a "number of copies" attribute, a configuration parameter set at transmission time, which controls the total number of copies of the message in the network. Then, when an "infected" node meets a non-exposed one, half of the copies are handed over and half kept by the emitter. Once a copy count reaches 1, no more forwardings are performed until the message's final destination is encountered.

Unlike *BSW*, which does not collect information on the network, *PRoPHET* [12] attempts to learn the best forwarders for different destinations. *PRoPHET* is based on the exchange of distance tables between nodes in a process conceptually similar to *Distance Vector* protocols. The main difference is that the distance is represented by a "predictability" or delivery probability. These predictability vectors are exchanged between nodes upon meeting. The predictability of a node as a target is reinforced when meeting it and is subject to exponential decay as time passes. Furthermore, it can be transitively reinforced: when node A meets node B, then A's predictability values for nodes in B's table are reinforced. This transitive reinforcement is affected by the B's own predictability. The basic idea is that nodes encountered more frequently have higher predictability, as are nodes reachable through high-predictability nodes.

This process of learning better opportunistic routes can be attacked with other methods (for an overview of current algorithms, see [7]). We consider that machine learning techniques are a promising alternative for the following reasons:

- ML is very efficient at detecting and capitalizing on patterns, which are the basis of efficient ON routing.
- ML effectiveness frequently depends on the availability of training data, which an ON can provide, as every packet can be considered an instance of the routing problem.
- ONs are complex systems with different components –such as mobility, data generation, and wireless interference– that are difficult to model and characterize. One of the ML strengths is that it can be applied without a deep understanding of the system's dynamics.

In this work, we will discuss the application of reinforcement learning to ON routing.

3. Reinforcement and Q-Learning in Routing

Reinforcement learning (RL) is a machine learning (ML) technique based on the exploration of the solution space by learning agents. These agents interact with the environment or *state* through actions that produce a reward. These rewards serve as feedback to inform the agent about the appropriateness of the action in the given scenario, allowing it to learn a policy to produce optimal behavior.

Designing an RL system consists of defining the 4-tuple $< S, A, P, R >$, where S and A are the sets of states and actions; $P : S \times A \rightarrow S$ is a transition matrix, potentially stochastic. S, A, and P define a Markov decision process (MDP) described as $s_{i+1} \leftarrow \delta(s_i, a_i)$,

which can be either deterministic or not. $R : S \times A \times S \to \mathbb{R}$ is the reward function that assigns a state transition value. Rewards themselves can also be either deterministic or not.

Because the rewards are obtained from a single state transition, but the agent is expected to produce an effective sequence of actions, a mechanism to calculate a global reward V is also needed. Thus, the RL problem can be described as finding the optimum policy $\pi : S \to A$ to select the best action in a given state, as to maximize a cumulative reward $V^\pi : S \to \mathbb{R}$ where the S indicates the starting state for applying the policy π. The sequence of states evaluated by V^π must be produced by iteratively evaluating δ with the available actions in each moment. The reward itself is usually computed as the discounted cumulative reward from the produced sequence of states, starting from a state s_0, as:

$$V^\pi(s_0) \to \sum_{i=0}^{\infty} \gamma^i r_i \tag{1}$$

where $0 \leq \gamma < 1$ is a parameter that controls an exponential reduction in the weight given to instant rewards r_i further from the starting state.

This problem is challenging because to evaluate V^π, the state sequence produced by δ and the resulting rewards must be known, which is not always possible. In particular, the state transitions caused by actions or the resulting rewards might be non-deterministic.

To overcome the difficulty of optimizing a policy for an unknown sequence of states, actions, and rewards, Q-learning [13] proposes using a particular evaluation function:

$$Q(s,a) \leftarrow r(s,a) + \gamma \max_{a'} Q(\delta(s,a), a') \tag{2}$$

This function says that the evaluation of an action is the sum of the immediate reward and the best achievable evaluation from the action's target state. Notice that this is a recursive definition and that both $r(s,a)$ and $\delta(s,a)$ can be observed as they are only evaluated once in the present state s (they are sampled).

This equation iteratively approximates the Q function and obtains the associated policy in the process. For this, a learning agent stores the learned $Q(s,a)$ values in a table. Then, it repeatedly evaluates policies in *epochs*. After selecting an action a in state s, it updates the Q as follows:

$$\begin{aligned} Q(s,a) \leftarrow\ & (1-\alpha)Q(s,a) \\ & + \alpha(r(s,a) + \gamma \max_{a'} Q(\delta(s,a), a')) \end{aligned} \tag{3}$$

where α is a learning parameter controlling the convergence speed. As all actions for all states are sampled repeatedly, the Q estimation converges to the value of Equation (2) [13]. During the system's training, the optimal known policy is to select the highest valued action in any state. As usual in ML, a balance must be found between exploitation (using the highest valued actions) and exploration (evaluating alternate actions).

The fact that Q-learning correctly takes into account the future effect of actions, and provides a policy that does not depend on previous knowledge or domain model, made it very useful for attacking the problem of routing in dynamic networks [14,15].

The first and most influential application of Q-learning to routing is *Q-routing* [16]. As in most RL routing protocols, the learning agent is a network node, and the Q-learning is applied in a distributed manner where each node x maintains a Q_x table. Q-routing introduced several design decisions that were later applied by many other algorithms. First, Q-routing reduces a distance metric instead of maximizing a reward; $Q_x(d,y)$ is defined as the node x's estimation of the latency from node y to destination d. Furthermore, instead of exploring the solution space by sending data packets through sub-optimal paths, Q-learning introduced a separate channel for exchanging neighboring nodes' best Q values. When a node needs to recompute its own Q table, it will request $T_y = \min_{z \in N_y} Q_y(d,z)$ from all neighbors. That makes the algorithm somewhat similar to classical Bellman–Ford. Finally, node x updates its table as:

$$Q_x(d,y) \leftarrow (1-\alpha)Q_x(d,y) + \alpha(q+s+T_y) \tag{4}$$

where q and s are the local queue and transmission times. Notice that because the Q value represents a latency that is not subject to discounting for further away nodes, $\gamma = 1$ is used.

Q-learning-based routing has been adapted to various networks [17]. We will briefly mention some of the algorithms applied to ONs.

One of the most influential Q-learning ON routing is *Delay Tolerant Reinforcement-Based* (DTRB) [18]. DTRB is a flooding-based algorithm wherein messages are replicated, maximizing a reward computed by a dedicated distance-table gossiping algorithm.

In DTRB, nodes maintain two tables, a table with the distances to every known node in the network and a table of rewards for every known message destination node.

The distance stored in the first table is an estimation of the time needed to propagate a message to the target node. These temporal distance tables are maintained by exchanging time-stamped control messages. These messages flood information about the last time a given node was met in the network. Then, nodes use the time elapsed as a distance measure.

This learned distance value is then used to compute a reward used in a conventional Q-learning scheme (see Equation (5)). A node computes the one-hop reward R for a given neighbor as e^{-k}, where k is the time-distance measure stored in the distance table or 0 if the entry is older than a configuration parameter. Then, a Q *practicability of delivery* value is computed as an estimation of the future rewards after taking a particular action. The discount factor γ_x is dynamic, getting smaller the faster that the neighborhood of x changes. This means that the algorithm favors nodes with low mobility.

$$Q_c(d,x) \leftarrow (1-\alpha)Q_c(d,x) + \alpha(R + \gamma_x \times \max_{y \in N_x} Q_x(d,y)) \tag{5}$$

The computed rewards are distributed in the same control messages mentioned and are subject to further exponential aging. Finally, messages are replicated to nodes with higher computed Q-values than their own.

The basic idea of the algorithm is to distribute a routing metric through control messages and then use it as the immediate reward in a Q-learning scheme. The resulting Q-values are used as an estimation of the global cost of an opportunistic delivery path.

An alternative is to integrate the routing metric within the Q-tables. For example, CARL-DTN [19] is a flooding protocol that uses Q-learning to learn a Q-value associated with every destination. Q-values are updated when nodes meet or go out of range. The Q-value combines the hop count with other node characteristics such as TTL, node popularity, or remaining power. This combination is achieved using a fuzzy logic controller (FLC).

The update function for node c when meeting node x, for destination d is:

$$\begin{aligned} Q_c(d,x) \leftarrow{}& (1-\alpha)Q_c(d,x) \\ &+ \alpha(R(d,x) + \gamma \cdot FuzzTO \cdot \max_{y \in N_x} Q_x(d,y)) \end{aligned} \tag{6}$$

where the immediate return $R(d,x)$ is an indicator function which returns 1 if the destination d is directly reachable through x, and otherwise 0. *FuzzTO* is a fuzzy *transfer opportunity* metric that combines the social characteristics of a node and its estimated ability to complete message forwarding. When a node becomes disconnected, the associated Q-value begins to decay exponentially.

In these protocols, the Q-learning process is decoupled from the data delivery, respecting the classical separation between routing and forwarding. These depend on exchanging routing data, and in this regard, they inherit the *Q-routing* property of being a combination of Q-learning and a conventional routing algorithm. As a side effect, routes with associated messaging are maintained for all nodes in the network, whether there is actual traffic through them or not. Furthermore, they are only usable with destination-based routing.

Other algorithms take a more direct approach to Q-learning. In FQLRP [20], each message is considered an agent, and the set of possible states is the set of network nodes. Therefore, the set of actions available to an agent is the set of neighboring nodes at a given moment. Then, the reward is computed as a function of the current node and its neighboring nodes' attributes, such as available buffer space and remaining energy. The idea is that the rewards express the likelihood of a successful forwarding. A modification of the basic Q-learning scheme is that the candidate nodes are first filtered through a *Fuzzy Logic-Based Instant Decision Evaluation*.

All these protocols share in common that Q-learning is used to select a candidate node to replicate data to, but there is no explicit action to *not* copy. Q-values are computed for actions, and in the usual representation, the only actions considered are *forwardings*.

Furthermore, no explicit notion of time or sequencing is defined, using the simple exponential decay of Q-values to express the passing of time. This precludes the system from learning that it is better to avoid copying because there is probably a better opportunity in the future. It is important because many ONs have marked temporal patterns, such as the schedules of public transport systems, the workday and weekly mobility cycles of city inhabitants, or the migration patterns of wild animals. Integrating the notion of time and sequence in the learning system would allow capitalizing on those patterns. We will evaluate the gains of this representation in Section 6.

Finally, we want to mention a related but different concept to ON, the *Opportunistic Routing*. This term usually refers to the ability to take advantage of overheard messages in a broadcast medium. The *Opportunistic Routing* application is not restricted to ONs in the sense of networks whose devices encounter sporadically. In fact, most Opportunistic Routing research concentrates on ad hoc and mesh networks, which are connected networks, though with a potentially dynamic topology.

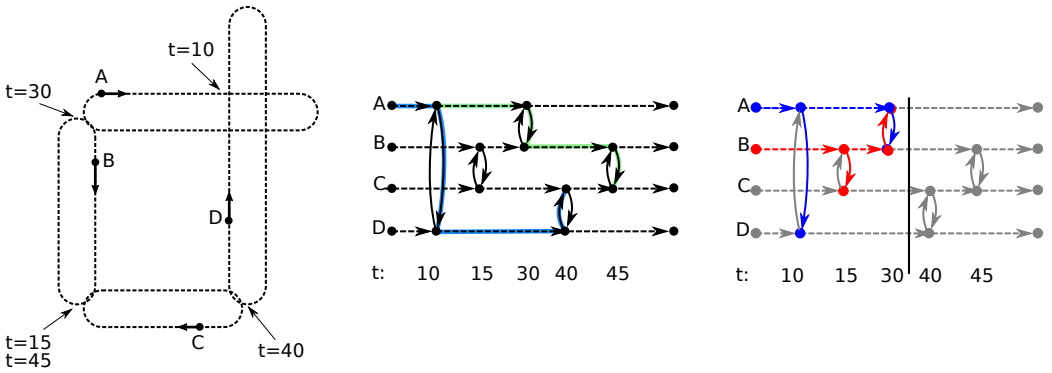
4. ON Model

We propose to apply Q-learning to a particular representation of the ON that we presented in [8]. In this model, a special *Opportunistic Network Model* (*ONM*) graph is built, which represents both the encounters between network devices and the temporal evolution of the network. Each network device is represented in this model by a set of graph nodes. Each node of this set captures the network device's state at the moment of an encounter with another device. In the resulting graph, there are two classes of edges: "copy" edges that link nodes belonging to different devices and represent an opportunity to copy a message from one device to another during an encounter; and "time" edges link all the nodes that model a device through time.

Figure 1a shows an example mobility scenario for four mobile devices, from A to D. The devices follow closed trajectories and meet at the points marked with arrows at the indicated times. In this example, devices B and C meet twice, at times 15 and 45.

The ONM graph that captures this scenario is shown in Figure 1b. Solid arrows represent encounters; dotted arrows represent time passing between encounters. The colored edges show two opportunistic paths between devices A and C.

Although this graph can be built from a complete trace of the mobility scenario, as described in [8] (useful for analyzing recorded scenarios or synthetic traces produced by simulators), in this work, the graph is built online, by the devices themselves, during the real-life scenario execution. In this case, each device builds a local view of the whole graph in a distributed manner, with only the edges it participates in. Figure 1c shows this process for devices A and B. Each device registers a local view of the graph as time passes and meetings occur. For example, in Figure 1c, the new nodes created at $t = 30$ have corresponding identical pairs created in both participating devices. These duplicates will be merged if all the local views are consolidated in a single representation.



(**a**) (**b**) (**c**)

Figure 1. Opportunistic network model construction. (**a**) Mobile devices' trajectories. The devices move in closed loops, meeting at the indicated times; (**b**) full opportunistic network model built offline using recorded traces (from [8]); (**c**) Online model construction. Learned nodes and edges up to $t = 35$ for devices A (blue) and B (red) are shown.

The online-build model captures the history of what happened and does not contain information on the future evolution of the network. To predict the future, in this work, we capitalize on the temporal patterns of many opportunistic networks, as seen in Section 6.

The pseudocode for the online graph construction is shown in Algorithm 1.

Algorithm 1: Distributed online ONM graph construction.

Data: Own device identity d
Output: Local view of the Model Graph
`/* Types for graph definition */`
1 Type Node: (Device, Time)
2 Type Edge: (Node, Node)
`/* Graph initialization */`
3 $N \leftarrow$ new Set of Node
4 $E_c \leftarrow$ new Set of copy Edges
5 $E_s \leftarrow$ new Set of time Edges
`/* Create initial node */`
6 $lastnode \leftarrow$ new $Node(d, 0)$
7 N.add($lastnode$)
`/* called when meeting other devices */`
8 **Function** `Encounter(`$neighbor$`):`
9 $newnode \leftarrow$ new $Node(d, time())$
10 N.add($newnode$)
11 $remote \leftarrow$ new $Node(neighbor, time())$
12 N.add($remote$)
13 E_s.add(new $Edge(lastnode, newnode)$)
14 E_c.add(new $Edge(newnode, remote)$)
15 $lastnode \leftarrow newnode$ `// update trailing node`

An important detail is that all the nodes associated with a single device have a single *time edge*, except for the last one, which has none. Furthermore, notice that *time edges* are added from the receiving node; this is once the edge has been "traversed".

A path over the ONM graph represents a possible propagation trace of a message. A trajectory starting at a given node represents a new message emitted by the corresponding device at the indicated time. A message traversing a *copy edge* represents that the message is copied between two devices. To traverse a *time edge*, a message must be stored in the device's buffer, surviving up to the destination node's time.

In an ON, the order of encounters is critical for delivering a message. In Figure 1, the path A–D–C depends on nodes A and D encountering before D and C. This dependency is naturally represented in the ONM.

In this model, routing messages in an ON is equivalent to routing them in a static graph. Edges can have different costs associated, allowing great flexibility in the definition of optimality, as discussed in Section 5.

The model allows representing multiple simultaneous transfers if the underlying network supports them, in the form of multiple *copy edges* leaving or arriving at a single node. For example, a broadcast transmission would be represented by multiple *copy edges* leaving a single node, each reaching a different device in range. Furthermore, additional information can be stored in the graph to support learning algorithms and performance modeling. For example, nodes may have associated the location and remaining battery storage, and *copy edges* can be assigned radio propagation characteristics.

As mentioned before, many ON networks are not entirely random but exhibit patterns. These patterns manifest themselves in the ONM. For example, the closeness between devices reflects in more frequent copy edges, while periodic trajectories manifest as a recurrence of copy edges at specific times. This allows using the ONM as a base to apply ML techniques.

5. Algorithm

As seen in Section 2, ON Q-learning-based algorithms usually do not maintain an explicit, graph-based model of the network. More specifically, the passage of time is usually captured by a simple parameter-decay mechanism, which depends on ad hoc configuration parameters that must be selected and tuned separately, usually through simulation.

Our strategy uses an ONM as the representation of the network. The ONM is a graph, though not a connectivity one. Nevertheless, a path connecting nodes in ONM still represents a data trajectory connecting a source with a destination. A shortest-path algorithm with adequately chosen edge weights can be used to find an optimal delivery trajectory.

There are two main classes of problems wherein RL is used to find the shortest path in a graph. The first class of problems is, naturally, network routing (see Figure 2b), where the graph is the connectivity graph of the network [21]. In this representation, edges are network links, and nodes are network routers. Each graph node is an agent that makes forwarding decisions that collectively solve the routing of messages from sources to destinations. As each node is an agent, each node trains only for the available actions; this is forwarding to neighboring nodes.

The second class of problems is the robot navigation problem (see Figure 2a). In this case, the graph represents a map; for example, nodes are rooms, and edges are doors connecting rooms. The learning agent is a robot that must navigate through the map to reach a destination [15]. The agent has a policy to be trained, which dictates the next action (edge) to follow as it moves through the graph, like in a maze. Each graph node is a state in this representation, with a single Q-table *attached* to the single agent. The Q-table is trained by repeatedly moving through the map.

Neither of these two representations is well suited as a base for applying RL for opportunistic routing using ONM. The main problem is that the RL states coincide with the nodes from the underlying graph in both representations. As a result, though they differ in whom they consider an agent and thus how many Q-tables there are and where they are stored, the content of the Q-table is similar in both cases: it assigns a Q-value to a target graph node.

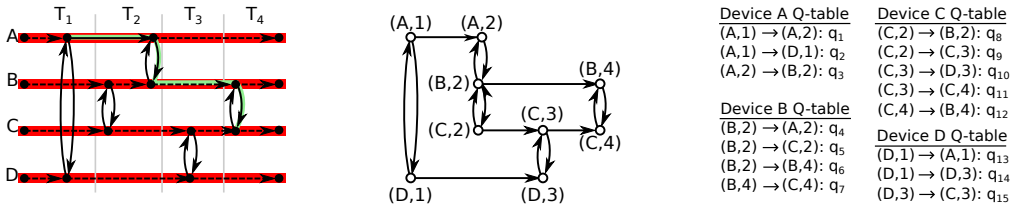
In the ONM graph representation, nodes and edges are not fixed beforehand. They are defined during the network's lifetime and describe a single instance of the mobility scenario. Furthermore, each node is visited only once during this instance.

(**a**) (**b**)

Figure 2. Models for routing over a graph using Q-learning. The learning agents are marked in red. Navigation problem. (**a**) The agent is the message, actions and states are graph nodes; (**b**) Routing Problem. Each node is an agent, the actions are next-hop nodes.

Thus, because the graph nodes are associated with a particular real-world instance of the network, and they are poor candidates for Q-learning actions and states, which must be usable through multiple learning episodes.

We propose *rl4dtn* an opportunistic routing algorithm based on applying Q-learning over an online-generated ONM graph (see Figure 3). The main characteristics are summed up in Table 1.



(**a**) (**b**) (**c**)

Figure 3. rl4dtn learning model. (**a**) The ONM model with the time slots and the learning agents marked in red; (**b**) Q-learning states ad actions. Each state is associated to a device and time slot; (**c**) The Q-tables for all devices.

Table 1. Reinforcement learning structure for rl4dtn algorithm.

Agent	Mobile device.
State	A (device, timeslot) pair, where the time is divided into set time slots.
Action	Connect two states, either on the same device (store data) or another (forward data).
Evaluation time	Encounter between devices (i.e., ONM node).
Q-Metric and task	Minimization of a distance metric.
Determinism	States and actions are non-deterministic.

5.1. States and Actions

In our approach, each network device is an agent and maintains the Q-table for all the associated system states and possible actions from its point of view.

Because the optimum action to choose depends on the time at which the agent is, we divide the scenario's duration into time slots (see Figure 3a) and then use each slot from every device as a potential state for the RL process. Thus, a state is defined by a pair $(networkdevice, timeslot)$, and actions are the possible state transitions (see Figure 3b). Notice that actions are not the edges from the ONM; ONM edges connect ONM nodes, which are defined by a device and an encounter time stamp. RL actions connect states

defined by a device and a discretized time slot. The resulting Q-tables are shown in Figure 3c.

Paralleling the ONM edges, there are two possible classes of actions. In one, the destination of the state transition is a state associated with the same origin device but at a later time slot; this represents storing data between different time slots. In the other scenario, the destination is a state with a different device but in the same time slot; this represents a data-forwarding action.

Notice that the state space can be seen as the time discretization of the ONM graph; multiple graph nodes from a device that occur in the same time slot map to a single state. For example, in Figure 3a, there are two encounters by node B in time slot T_2. Both encounters are represented by the same state $(B, 2)$ in Figure 3b.

Furthermore, an action can capture the effect of multiple ONM edges. For example, all encounters between a pair of devices that happened in the same time slot, each modeled by a separate ONM *copy edge*, are represented as the same action: forwarding between the involved devices in the given time slot.

Actions that connect subsequent states within the same device are called *survival actions*, the idea being that when the link exists, data have survived between time slots. Notice that a *survival action* is available on all states except for the last state associated with a device.

The Q-learning is performed each time a forwarding decision is made, this is on each ONM graph node. If there are multiple forwarding decision evaluations within a time slot, they all contribute to the computation of the time slot's Q-value.

The time slot size is a configuration parameter for rl4dtn. In one extreme, when there is a single time slot that spans the whole mobility scenario, the rl4dtn model behaves as a single, static connectivity graph. In this case, there is a single *forwarding action* computed for connecting each pair of nodes, and there are no *survival actions*. The Q-value for the actions is computed from all the forwardings made through the scenario, and no temporal distribution of encounters is captured. As the number of slots increases, the number of nodes per state is reduced since the time slots become shorter, and thus fewer nodes share a slot. This means there are fewer Q-learning evaluations to update the state's Q-value, while at the same time, the number of states increases. The result is a degradation of the system's learning convergence. Because we want to capture changing mobility patterns through time, a balance must be reached for the optimum time slot size. We discuss this effect in more detail in Section 6.

An agent can perform multiple actions simultaneously, representing the generation of replicas of the routed message, with each copy later being routed independently. Thus, if we configure the protocol to choose a single action, *rl4dtn* behaves as *forwarding* algorithm. If multiple actions are allowed, the algorithm becomes *flooding*-based.

5.2. Q-Value and Rewards

As is usual in routing, the RL problem is posed as distance minimization instead of reward maximization. Furthermore, as usual in these cases, we fix the discount factor $\gamma = 1$. The resulting Q-update function is shown in Equation (7).

$$Q_c(d, x) \leftarrow (1 - \alpha) Q_c(d, x) + \alpha (R + \min_{y \in N_x} Q_x(d, y)) \tag{7}$$

The definition of the immediate cost R itself is flexible. For example, Table 2 presents the immediate cost of edges under several distance metrics. The *hop count* metric minimizes the number of transmissions, while the *latency* attempts to deliver the data as fast as possible. The *power consumption* cost can be seen as a generalization of *hop count*, where the transmission cost is a function *Pwr* of the distance between the nodes, the radio environment, message size, etc.

Notice that the immediate cost R used to update a Q-value is computed on every encounter or ONM graph edge. As a result, the instances of evaluation of a state–action pair and the values of R are non-deterministic.

Table 2. Immediate edge costs R for rl4dtn for various distance definitions

	Time Edge	**Copy Edge**
Latency	$T_{dest} - T_{source}$	0
Hop count	0	1
Power consumption	0	$Pwr(source, dest)$

In RL algorithms, the definition of when a message is considered delivered is under the control of the reward function, which is an arbitrary function. It can accommodate concepts such as multicast (deliver to a set of nodes), anycast (deliver to any of a set of nodes), or generalizations such as subscription-based delivery (deliver to nodes that request messages with specific properties).

5.3. Opportunistic Route Exploration

An essential part of the behavior of ML systems is *exploration*; *rl4dtn* explores the space of solutions by two mechanisms. The first is the standard $\epsilon - greedy$ policy, where, at the action-selection time, there is a probability ϵ of selecting a random action instead of the known best (the one with minimum associated Q-value).

The second exploration mechanism is to generate multiple message copies, where each one is routed independently. This replication is managed using *Binary Spray and Focus* (*BSF*) [22]. This is similar to the *Binary Spray and Wait* technique mentioned in Section 2, differing only in its behavior when the copies count reaches one. In *BSW*, this single copy is held locally and delivered only to the destination device if it is met directly. In *BSF*, this copy can be handed over to another device and then remain hopping from device to device until meeting the destination. Notice that if the emitting device sets the *number of copies* parameter to one, the algorithm behaves as a pure forwarding algorithm.

As usual in Q-learning, the exploration parameters can be reduced during the system's lifetime, favoring exploration in the early stages and exploitation in later stages to improve performance and convergence.

The pseudocode for *rl4dtn* is shown in Algorithm 2. The entry point is on line 16, which is called on every encounter after updating the ONM model when a forwarding decision must be made (we assume we are handling a single message *msg*). For this purpose, a list of candidate destination states is built on lines 17-24. One of the candidates is the *survival* action (line 18). As mentioned before, this can be directly obtained from the *rl4dtn* model if already available from previous evaluations. If not, it can be a placeholder, as this action is known to exist. Besides the survival action, all copy actions are added to the list of candidates (line 19). If one of the neighboring agents is the target for the message, the message is delivered, and no further processing is made.

Once a list of candidates is built, line 25 selects a copy or *survival* target using the $\epsilon - greedy$ selection function (lines 3–7). After this, line 26 updates the Q value. That is done on lines 8–18, applying Equation (7). In our case, line 12 computes immediate hop count costs from Table 2.

If the selected action was a forwarding, *Binary spray and focus* is used on lines 27–33 to either forward half message copies if there are more than one (line 28) or migrate the single available copy to the new device (line 31).

As a baseline, we implemented a straightforward Q-learn algorithm not based on ONM. Unlike *rl4dtn*, only actions associated with forwarding are maintained (no *survival* actions), and forwarding is performed if the best candidate has a better Q-evaluation than its own. In Algorithm 2, this implies the removal of line 18, the condition on line 27, and

the substitution of the action selection function on line 7. The message-copies handling is identical to *rl4dtn*. This allows us to evaluate the impact of using ONM to learn explicit actions for keeping messages. Any improvement from Q-learn to *rl4dtn* can be assigned to the use of the ONM model. Please, notice that the computational complexity of both algorithms is roughly equal. Both are evaluated in the same situations (at forwarding decisions), the only difference being that *rl4dtn* has one more transition to consider: the explicit storing action.

Algorithm 2: rl4dtn pseudocode.

```
1  Type State: (BusID, Timeslot)                    // Type for Q-Learning State
2  Qtbl ← new MapOf(State, MapOf(BusID, Qvalue))  // Q-table datastructure
   /* Auxiliary functions                                                    */
3  Function PickAction(candidates):
4      if random() < epsilon then
5          return PickRandom( candidates )
6      else
7          return PickMinQ( candidates )
8  Function UpdateQ(target):
9      State currentS ← (myBusId, ts)
10     State targetS ← (target.id, target.ts)
11     minQtarget ← minValue( Qtbl[targetS] )
12     r ← EdgeCost ( target )
13     Q ← Qtbl[currentS][target]
14     Q ← Q + alpha(r + gamma * minQtarget − Q)
15     Qtbl[currentS][target] ← Q
   /* main function, called after updating ONM                               */
16 Function Encounter(N, msg):
17     candidates ← new Set Of Node
18     candidates.add( followup(N) )
19     for each n in neighbors(N) {
20         if isTarget( n, msg ) then
21             Qtbl[(myBusID, ts)][(n.id, ts)] ← 1.0
22             deliver( n.id, msg )
23             return
24         candidates.add( n )
25     target ← PickAction( candidates )
26     updateQ( target )
27     if target ≠ followup(N) then
28         if msg.copies > 1 then
29             forward( target, msg, msg.copies/2 )
30             msg.copies ← msg.copies − msg.copies/2
31         else
32             forward( target, msg, 1 )
33             drop( msg )
```

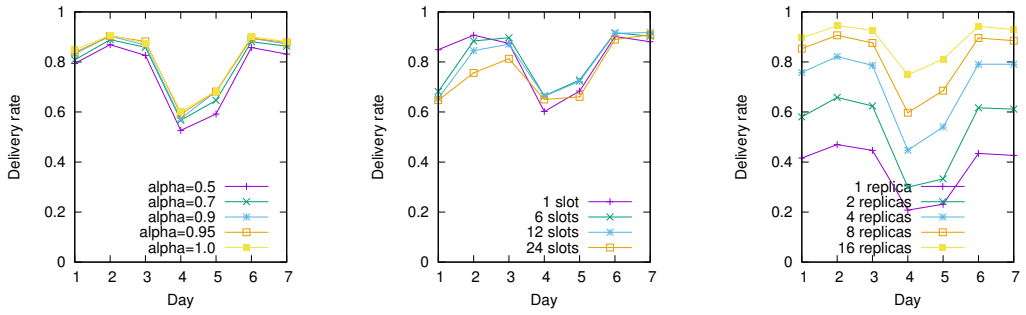
6. Simulation

We test the algorithms using the *RioBuses* dataset [23], which collects the GPS trajectories of buses in Rio de Janeiro, Brazil. It contains over 12,000 units, moving over 700 bus lines through November 2014. In this work, we down-sample the dataset to 1000 buses, which is still a large use-case for an ON [24]. We generate ten distinct down-sampled sets

to produce statistics. Nevertheless, the ONM representation allows simulating the entire dataset, even using desktop-grade hardware [8].

We run the algorithm in a scenario where all units emit two messages, one at 00:00 and another at 12:00, directed to a single fixed collection point located in one of the city bus terminals. Successful delivery is achieved when a message arrives within the same day. Messages underway are removed every night at midnight, but the learned protocol parameters are kept.

To calibrate the algorithm parameters, we simulate the first seven days of the trace. The results are shown in Figure 4. The delivery rate is computed at the end of the corresponding day. The reduction in delivery rate on days 4 and 5 corresponds to the weekend, with reduced bus services (1 October 2014 was a Wednesday).



(**a**) (**b**) (**c**)

Figure 4. Learning model parameters' impact on delivery rate. (**a**) Learning-rate parameter α; (**b**) Number of time slots; and (**c**) Number of message copies.

Figure 4a shows the impact of the α parameter on the learning process (see Equation (7)). The algorithm was configured with eight copies per message. The best learning is achieved with a very aggressive $\alpha = 1.0$.

As described in Section 5, the Q-parameters are learned per time slot to allow for learning different patterns throughout the day. On the other hand, more slots mean fewer encounters per slot, which slows down learning. The impact of the number of time slots is shown in Figure 4b. It can be seen as the single-slot scenario learns fastest, as it considerably outperforms the alternatives at the end of the first day. Nevertheless, 6 or 12 slots finally catch up and outperform after day 3. Having 24 slots degrades performance, possibly because the number of encounters per slot is too small to sustain effective reinforcement learning.

In *rl4dtn*, reinforcement learning is performed by observing the propagation of data messages through the network. Thus, having multiple replicas of a message improves the learning process. Additionally, it improves the delivery rate, as it is enough for a single message copy to arrive for a successful delivery. This effect can be seen in Figure 4c. On the other hand, producing multiple copies of messages increases the data transmitted in the network. This effect will be discussed in Section 6.

Figure 5 compares *rl4dtn* delivery performance with *BSW*, *PRoPHET* and the reference Q-learning protocol. The average over ten simulations with different buses samples is shown, with the corresponding interquartile range. The number of time slots used for *rl4dtn* and Q-learning is 6, the number of message copies is 8, and the ϵ parameter driving the Q-learning exploration is set at 1.0 at the beginning and then reduces linearly to 0.5 on the last day.

BSW underperforms considerably. *PRoPHET* suffers the most from the reduction in mobility through the weekend.

It can be seen that *rl4dtn* offers a better delivery rate. Furthermore, the performance difference between Q-learn and *ld4dtn* shows the impact of using the ONM as a base for the learning process.

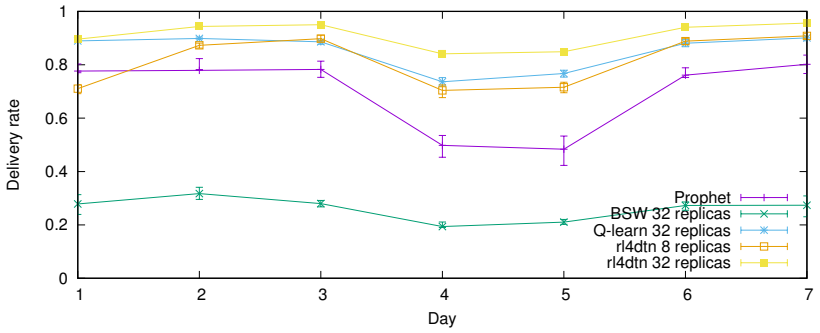


Figure 5. Delivery rate comparison.

Figure 6 displays a 1:50 sample of the Q-values as they evolve through the system's learning. This shows the convergence behavior of the Q-learning process. Notice that the Q-values estimate the cost of an opportunistic path; in our scenario, this cost is the hop count. It can be seen that the system starts with Q values up to 6, and from day three, it settles on many paths of one bus hop, considerably less but roughly similar paths of length 2 and 3, and very few of length 4. The gap with almost no deliveries in the early morning can be seen.

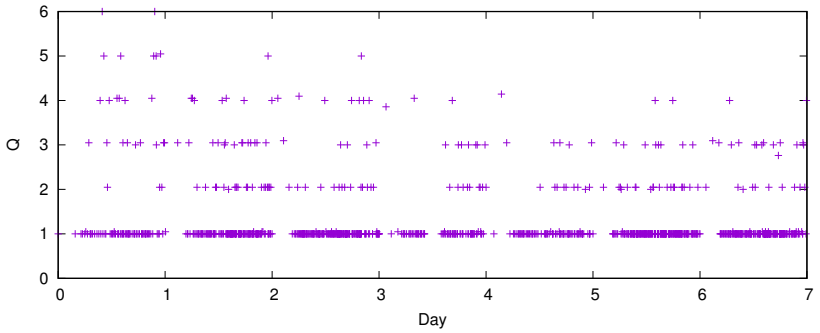


Figure 6. Q-values evolution through learning.

Figure 7 shows the latency histogram, which is the time between message generation and message arrival, for the messages emitted at 12:00 and successfully delivered. *rl4dtn* has a marked peak at around 2 h latency, while the other algorithms have a more indistinct behavior. Notice that two hours is a reasonable average trip duration, using bus lines, from anywhere in the Rio de Janeiro region to a given collection point. This suggests that rl4dtn finds more direct paths than the alternatives, which depend more on longer, more random journeys.

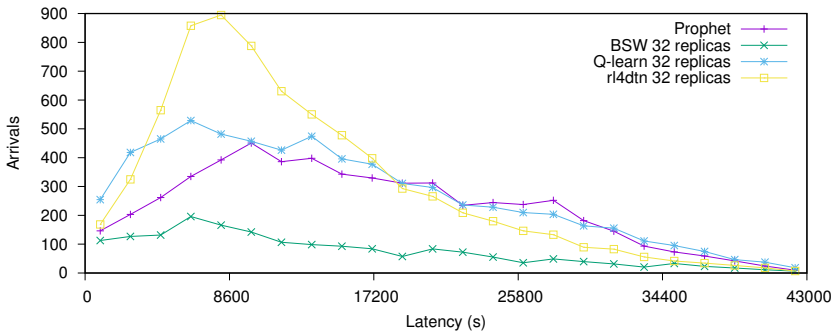


Figure 7. Message latency histogram.

Traffic Efficiency

The traffic produced by an ON algorithm is an important metric, as it directly impacts power consumption. This traffic is produced by agents exchanging routing data and when forwarding messages.

Thus, the total traffic in the network is:

$$T_{tot} = E \times S_R + T \times S_M \quad (8)$$

where the parameters are shown in Table 3, assuming 1 KB messages and 1000 buses.

Table 3. Data exchange parameters.

		BSW	**PRoPHET**	**Q-Learn,rl4dtn**
E	Number of encounters	From ONM graph (copy edges)		
T	Number of message forwardings	From simulation		
S_M	Size of message	1 KB + 4 B	1 KB	1 KB + 8 B + 4 B
S_R	Routing exchange size	0	1.000*8B	0

BSW does not attempt to learn from the network, and no routing information is exchanged. The only addition to the messages is the integer number of the number of copies.

PRoPHET exchanges a predictability vector during each encounter. This vector contains a floating-point predictability value for each device in the network. *rl4dtn* does not exchange routing data; in its place, a Q-value used to update a Q-table is retrieved from the destination when forwarding. Additionally, the messages have a copies-count integer.

The total traffic produced in the network is shown in Figure 8. *BSW* has a very high transmission cost, despite not producing routing data. This is related to the fact that every transmission opportunity is taken, and the maximum allowable number of replications is produced. Nevertheless, the delivery rate is poor because the replication is made very aggressively following the message generation, not achieving adequate dissemination through the network. It can be seen that *PRoPHET* has a high traffic consumption, the vast majority of it being routing information and not data transmissions. This signaling grows with the number of encounters (vector exchanges) and devices in the network (vector size). Furthermore, it must be noted that a considerable part of the routing information exchange might not be of interest to the actual data flows in the network. Finally, *rl4dtn* achieves better delivery performance with a lower data overhead.

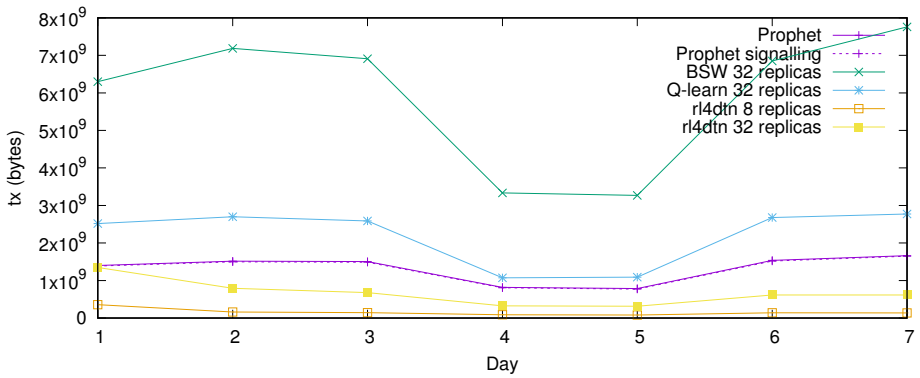


Figure 8. Total data transmitted in the network.

7. Conclusions and Future Work

We presented *rl4dtn*, an opportunistic networks routing algorithm based on Q-learning, that takes advantage of a particular opportunistic network model. This model captures the

evolution of the network through time using a temporal graph. The network devices build and use this model in a distributed fashion during the network's lifetime.

This work shows that this model allows our Q-learning-based method to outperform classic Q-learning approaches. By comparing rl4dtn with a conventional Q-learning algorithm, we show that the rl4dtn's advantage comes from the fact that the learning mechanism feeds from temporal data. These temporal data allow learning from the temporal patterns in the distribution of encounters arising from the devices' movement patterns. Temporal patterns are a defining characteristic of ONs, differentiating them from conventional networks.

At the same time, the resulting algorithm obtains a higher delivery rate with a considerably lower network overhead than representative ON routing algorithms; this is very important for networks with large numbers of devices where the routing data transmissions can represent a significant load.

Amongst the issues that need more attention, we point out that although the dataset used in this work is typical, more tests are needed as ONs cover a broad spectrum of mobility scenarios.

Another area of improvement is the management of time slots. In this work, the time-slot size is fixed and experimentally determined. A scheme can be devised where the system adjusts the slots dynamically and automatically without an explicit configuration. For example, the system could adjust the time-slot size to maintain a certain number of encounters per slot, enough to sustain effective Q-learning. As a result, for example, nighttime slots could be longer and peak-hour slots shorter.

In summary, the proposed routing algorithm builds on top of a previously presented network model, which allows it to outperform classic Q-learning approaches. It achieves higher delivery rates with lower network overhead than representative ON routing algorithms. There are several lines for improvement that must be explored.

Author Contributions: Investigation, J.V. and J.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not Applicable, the study does not report any data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Amah, T.; Kamat, M.; Abu Bakar, K.; Moreira, W.; Oliveira Junior, A.; Batista, M. Preparing opportunistic networks for smart cities: Collecting sensed data with minimal knowledge. *J. Parallel Distrib. Comput.* **2020**, *135*, 21–55. [CrossRef]
2. Coutinho, R.W.L.; Boukerche, A. Opportunistic Routing in Underwater Sensor Networks: Potentials, Challenges and Guidelines. In Proceedings of the 2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS), Ottawa, ON, Canada, 5–7 June 2017; pp. 1–2. [CrossRef]
3. Ansari, R.I.; Chrysostomou, C.; Hassan, S.A.; Guizani, M.; Mumtaz, S.; Rodriguez, J.; Rodrigues, J.J.P.C. 5G D2D Networks: Techniques, Challenges, and Future Prospects. *IEEE Syst. J.* **2018**, *12*, 3970–3984. [CrossRef]
4. Visca, J.; Fuentes, R.; Cavalli, A.R.; Baliosian, J. Opportunistic media sharing for mobile networks. In Proceedings of the NOMS 2016—2016 IEEE/IFIP Network Operations and Management Symposium, Istanbul, Turkey, 25–29 April 2016; pp. 799–803. [CrossRef]
5. Fall, K. A delay-tolerant network architecture for challenged internets. In *Sigcomm '03: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*; ACM: New York, NY, USA, 2003; pp. 27–34. [CrossRef]
6. Schlesinger, A.; Willman, B.; Pitts, R.; Davidson, S.; Pohlchuck, W. Delay/Disruption Tolerant Networking for the International Space Station (ISS). In Proceedings of the 2017 IEEE Aerospace Conference, Big Sky, MT, USA, 4–11 March 2017; pp. 1–14. [CrossRef]
7. Sachdeva, R.; Dev, A. Review of opportunistic network: Assessing past, present, and future. *Int. J. Commun. Syst.* **2021**, *34*, e4860. [CrossRef]
8. Visca, J.; Baliosian, J. A Model for Route Learning in Opportunistic Networks. In Proceedings of the NOMS 2022—2022 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 25–29 April 2022; pp. 1–4. [CrossRef]

9. Vahdat, A.; Becker, D. Epidemic Routing for Partially-Connected Ad Hoc Networks. *Tech. Rep.* **2000**. Available online: http://issg.cs.duke.edu/epidemic/epidemic.pdf (accessed on 16 October 2022).
10. Maitreyi, P.; Sreenivas Rao, M. Design of Binary Spray and wait protocol for intermittently connected mobile networks. In Proceedings of the 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 9–11 January 2017; pp. 1–3. [CrossRef]
11. Spyropoulos, T.; Psounis, K.; Raghavendra, C.S. Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks. In Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-Tolerant Networking, Philadelphia, PA, USA, 26 August 2005; Association for Computing Machinery: New York, NY, USA, 2005; WDTN '05, pp. 252–259. [CrossRef]
12. Lindgren, A.; Doria, A.; Davies, E.B.; Grasic, S. Probabilistic Routing Protocol for Intermittently Connected Networks. RFC 6693, 2012. Available online: https://datatracker.ietf.org/doc/rfc6693/ (accessed on 16 October 2022).
13. Watkins, C.J.C.H.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. Available online: https://link.springer.com/article/10.1007/BF00992698 (accessed on 16 October 2022). [CrossRef]
14. Chettibi, S.; Chikhi, S. A Survey of Reinforcement Learning Based Routing Protocols for Mobile Ad-Hoc Networks. In *Recent Trends in Wireless and Mobile Networks*; Özcan, A., Zizka, J., Nagamalai, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 1–13.
15. Khriji, L.; Touati, F.; Benhmed, K.; Al-Yahmedi, A. Mobile Robot Navigation Based on Q-Learning Technique. *Int. J. Adv. Robot. Syst.* **2011**, *8*, 4. [CrossRef]
16. Boyan, J.; Littman, M. Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach. In *Advances in Neural Information Processing Systems*; Cowan, J., Tesauro, G., Alspector, J., Eds.; Morgan-Kaufmann: Burlington, NJ, USA, 1993; Volume 6. Available online: https://proceedings.neurips.cc/paper/1993/file/4ea06fbc83cdd0a06020c35d50e1e89a-Paper.pdf (accessed on 16 October 2022).
17. Mammeri, Z. Reinforcement Learning Based Routing in Networks: Review and Classification of Approaches. *IEEE Access* **2019**, *7*, 55916–55950. [CrossRef]
18. Rolla, V.G.; Curado, M. A reinforcement learning-based routing for delay tolerant networks. *Eng. Appl. Artif. Intell.* **2013**, *26*, 2243–2250. [CrossRef]
19. Yesuf, F.Y.; Prathap, M. CARL-DTN: Context Adaptive Reinforcement Learning based Routing Algorithm in Delay Tolerant Network. *arXiv* **2021**, arXiv:2105.00544.
20. Dhurandher, S.K.; Singh, J.; Obaidat, M.S.; Woungang, I.; Srivastava, S.; Rodrigues, J.J.P.C. Reinforcement Learning-Based Routing Protocol for Opportunistic Networks. In Proceedings of the ICC 2020—2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6. [CrossRef]
21. Yu, H.; Bertsekas, D.P. Q-learning and policy iteration algorithms for stochastic shortest path problems. *Ann. Oper. Res.* **2013**, *208*, 95–132. [CrossRef]
22. Spyropoulos, T.; Psounis, K.; Raghavendra, C. Spray and Focus: Efficient Mobility-Assisted Routing for Heterogeneous and Correlated Mobility. In Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW'07), White Plains, NY, USA, 19–23 March 2007; pp. 79–85. [CrossRef]
23. Dias, D.; Costa, L.H.M.K. CRAWDAD dataset coppe-ufrj/RioBuses (v. 2018-03-19). 2018. Available online: https://crawdad.org/coppe-ufrj/RioBuses/20180319 (accessed on 16 October 2022).
24. Kuppusamy, V.; Thanthrige, U.M.; Udugama, A.; Förster, A. Evaluating Forwarding Protocols in Opportunistic Networks: Trends, Advances, Challenges and Best Practices. *Future Internet* **2019**, *11*, 113. [CrossRef]