Universitat Autònoma de Barcelona

A controller-driven approach for Opportunistic Networking

MªCarmen de Toro Valdivia

Universitat Autònoma de Barcelona

Departament d'Enginyeria de la Informació i de les Comunicacions

PHD in Computer Science

A controller-driven approach for Opportunistic Networking

MªCarmen de Toro Valdivia

Supervisor Dr. Carlos Borrego Iglesias

Bellaterra, December 2022

This thesis was typeset with $\LaTeX\,2_\varepsilon$. It uses the *Clean Thesis* style developed by Ricardo Langner.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.

Front Cover Photograpy:

`https://www.rawpixel.com/image/4021287/photo-image-green-nature-dog`

I certify that I have read this thesis entitled “A controller-driven approach for Opportunistic Networking" and that, in my opinion, it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Bellaterra, December 2022

Dr. Carlos Borrego Iglesias
(Advisor)

Committee:

Enrique Hernández Orallo
Guillermo Navarro Arribas
Benedetta Picano

Subtitute:

Ramon Martí Escalé
Carles Garrigues Olivella

To Joan Borrell

Abstract

OPPORTUNISTIC NETWORKS (OppNets) leverage opportunistic contacts using a device-to-device communication scheme to flow data across an infrastructure-free network. Up to date, OppNets performance depends on applying the most suitable forwarding strategy based on the OppNet typology. On the other hand, Software Defined Networks (SDN) is a paradigm for wired networks that decouples the control and data planes. The control plane oversees the network to configure the data plane optimally. Our proposal uses SDN-like controllers to build a partial overview of the opportunistic network. The forwarding strategy uses this context information to achieve better network performance. As a use case of our proposal, in the context of an OppNet using a quota-based forwarding algorithm, we present a controller-driven architecture to tackle the congestion problem derived from multi-copy forwarding strategies. Particularly, the controller-driven architecture uses the context information of how congested the network is to dynamically determine the message replication limit used by the forwarding algorithm. Simulation based on real and synthetic mobility traces shows that using context information provided by the controller to configure the forwarding protocol increments the delivery ratio and keeps a good latency average and a low overhead compared with the baseline multi-copy-based forwarding protocols. These results strengthen the benefits of using supervised context information in the forwarding strategy in OppNets.

Resum

Les xarxes oportunistes (OppNets) aprofiten les oportunitats de contacte entre els nodes, el quals utilitzen l'habilitat de comunicar-se directament entre ells, per fer fluir les dades a través d'una xarxa mancada d'infraestructura. A dia d'avui, el rendiment de les xarxes oportunistes depèn d'aplicar l'estratègia d'encaminament que més s'escaigui al tipus de xarxa oportunista en qüestió. D'altra banda, les xarxes definides per software (SDN) és un paradigma aplicat a xarxes cablejades a on se separa el pla de control del pla de dades. El pla de control supervisa la xarxa per configurar, de la manera més òptima, el pla de dades. La nostra proposta utilitza controladors semblants als de SDN per fer-se una idea parcial de la situació de la xarxa oportunista. L'estratègia d'encaminament utilitza aquesta informació contextual per obtenir una millora del rendiment de la xarxa. Com a cas d'ús de la nostra proposta, en el context d'una OppNet utilitzant un algorisme d'encaminament basat en una quota màxima de replicació, presentem una arquitectura dirigida per controladors per abordar el problema de congestió derivat de la utilització d'estratègies d'encaminament basades en multi-còpia de missatges. En particular, l'arquitectura dirigida per controladors utilitza la informació contextual de com congestionada està la xarxa per, dinàmicament, determinar quin és el límit de replicació de l'estratègia d'encaminament. Simulacions basades en traces de mobilitat reals i sintètiques mostren que la utilització de la informació contextual, proveïda pel controlador, per configurar el protocol d'encaminament, incrementa la ràtio de missatges rebuts tot mantenint una bona latència mitjana dels missatges i un cost baix d'utilització de la xarxa en comparació amb els protocols d'encaminament marc basats en multi-còpia. Els resultats obtinguts corroboren com de beneficiosa és la utilització d'informació contextual supervisada en l'estratègia d'encaminament de xarxes oportunistes.

Resumen

Las redes oportunistas (OppNets) aprovechan las oportunidades de contactos entre los nodos, los cuales utilizan la habilidad de comunicarse directamente entre ellos, para hacer fluir los datos a través de una red sin infraestructura. A día de hoy, el rendimiento de las redes oportunistas depende de utilizar la estrategia de encaminamiento más conveniente al tipo de red oportunista en cuestión. Por otro lado, las redes definidas por software (SDN) es una paradigma aplicado a las redes cableadas donde se separa el plano de control del plano de datos. El plano de control supervisa la red para configurar de forma óptima el plano de datos. Nuestra propuesta utiliza controladores parecidos a los de SDN para hacerse una idea parcial de la situación de la red oportunista. La estrategia de encaminamiento utiliza esta información contextual para obtener una mejora del rendimiento de la red. Como caso de uso de nuestra propuesta, en el contexto de una OppNet que utilice un algoritmo de encaminamiento basado en una cuota máxima de replicación, presentamos una arquitectura dirigida por controladores para abordar el problema de la congestión derivado de la utilización de estrategias de encaminamiento basadas en multi-copia de mensajes. En particular, la arquitectura dirigida por controladores utiliza la información contextual de cuán congestionada está la red para, dinámicamente, determinar cuál es el límite de replicación de la estrategia de encaminamiento. Simulaciones basadas en trazas de movilidad reales y sintéticas muestran que la utilización de información contextual, proporcionada por el controlador, para configurar el protocolo de encaminamiento, incrementa la ratio de mensajes recibos manteniendo una buena latencia media de los mensajes y un coste bajo en la utilización de la red en comparación con los protocolos de encaminamiento marco basados en multi-copia. Los resultados obtenidos corroboran los beneficios de la utilización de información contextual supervisada en la estrategia de encaminamiento de las redes oportunistas.

Acknowledgements

LUCKILY MANY people have accompanied me during this long journey. All of them have contributed in very distinct ways to the fulfilment of this work, and I would like to praise them all for that. Firstly, I will be forever grateful to my parents, who worked very hard so I could complete my bachelor's studies, *millones de gracias mama, papa*.

I am very grateful to my supervisor for always believing in me and this work; offering me his wide knowledge, know-how, and good ideas; enlightening me to seek perfection; sharing with me all his bash scripts, R scripts, and knowledge of the simulator. Carlos, it has been so helpful! I especially like to thank him for his support in the last and hardest stages of this path, *moltíssimes gràcies* Carlos. I am also very thankful to my last tutor Sergi Robles for believing in me and for his brilliant ideas have been crucial to the success of this research, *moltíssimes gràcies* Sergi.

I would have never sailed this ship if Joan Borrell had not pushed me to it. *Moltíssimes gràcies* Joan for believing I was capable of reaching this horizon when I did not. We have sailed this ship together. Joan, you always will be in my mind and my heart.

I am immensely thankful to Ian Blanes for getting me out of the hole I was in and for actively supervising the successful completion of the research paper that supports this PhD. Ian, without your help, know-how, support and encouragement, I would not be writing this dissertation right now, *moltíssimes gràcies* Ian.

I also want to thank my friend Cristina Fernandez for always being there for me. You are such a bright and powerful woman. You inspire me and so many women and men, *moltíssimes gràcies Cristina*. I am very grateful to JosepM^{a} Basart for showing me a different way of living this process and life. I commend your generosity for sharing your knowledge and always being willing to listen to me, *moltíssimes gràcies* Josep M^{a}. Josep M^{a}, Cristina, it has been life-changing and an absolute privilege to share a mindfulness meditation session on Fridays, which has helped me in stormy times, *moltíssimes gràcies* Cristina, Josep M^{a}.

I want to thank the Deic department for procuring me a warm and inspirational place of work and cheering me up along this journey. I especially like to thank “el consell de savis" for those cheerful lunches. It has been an honour to share this time with you.

Last but not least, I want to show my gratitude to my husband and two kids, who have lived first-hand the upside-downs of this whole path, which has not been easy. Roger, Sara, I hope you keep the good of it, that effort and resilience overcome despair and help to keep going.

I do not know where life will drive me from now on. Wherever it will be, I will carry all of you with me as you all have changed me for good. *Gràcies*.

This research has been funded by a four-year PIF grant from Universitat Autònoma de Barcelona. I am very grateful to this institution for having supported this work economically.

MªCarmen de Toro Valdivia
Bellaterra, December 2022

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations list

$\mathfrak{a}$	Application layer message
φ	Message's flag alive
b	Buffer size in bytes
B	Buffer: a list to store messages
$\mathfrak{c}$	Controlled variable
c_{t+n}	Congestion state prediction for time $t+n$
d	Decay
δ	Directive: $\delta = (Id, \vartheta)$
δ_l	Directive encapsulating a replication limit
ϕ	Factor to calculate t_{t+n}
Id_p	Identifier of the network setting P
Id_l	Message's replication limit Id setting
k_1	Multiplicative decrease factor
k_2	Additive increase factor
g_i	A message
$\#g_d$	Number of delivered messages
$\#g_c$	Number of created messages
$\#g_r$	Number of relayed messages
l	Max. message copies in the network (replication limit)
$\overline{\lambda}$	Latency average
λ_i	Latency of message g_i
L_1	Highest message generation rate for the ISTH dist.
L_2	Lowest message generation rate for the ISTH dist.
m_{l_i}	Node n_i's local network measurement: $m_{l_i} = (v_{l_i}, 1, t_{c_i})$
m_i	Aggregated received network measurements: $m_i = (v_i, \eta_i, t_{c_i})$
m_{t+n}	Predicted network measure value at time $t+n$
M_i	Aggregated network measurements list for node n_i
$\breve{m}$	Aggregation of the received measurements during $\hat{t}$s
$\breve{M}$	List of $\breve{m}$ values
$\breve{z}$	Max size of $\breve{M}$
μ	Control system's manipulated variable
η	Number of aggregated measurements
n_i	A specific node
o_i	Buffer occupancy in bytes of node n_i
o_{t+n}	Buffer occupancy prediction for time $t+n$
o_{min}	Min. buffer occupancy rate
o_{max}	Max. buffer occupancy rate
θ	Overhead ratio
r	Reduction factor
$\mathfrak{r}$	Control system's reference input
ϱ	Number of times a message has been relayed
ρ	Linear regression function
σ	Delivery ratio
t	Current time
t_{t+n}	Time ahead when calculating the congestion prediction
t_{c_i}	Creation time for either a measurement or an aggregation
$\hat{t}$	Time period for aggregating received measurements
$\breve{T}$	List of the timestamps of the aggregations performed
ϑ	Network setting's value
v_{l_i}	Node n_i's local network measurement
v_i	Result value after aggregating the received measurements in M_i

Part I

Preliminaries

Introduction 1

THE INTERNET is growing fast with the ever-increasing Machine-to-Machine communication (M2M), the expansion of micro and proximity services, the use of cloud services, the continuous increase of connected devices, among other reasons. Based on Cisco's Networking Visual Index[1], by 2023 there will be 29.3 billion networked devices. The fastest-growing mobile device category is M2M with 14.7 billion, followed by smartphones and other smart devices. The International Telecommunication Union report for the years 2020 to 2030 [78] estimates 100 billion connected devices by the year 2030. The global M2M subscriptions will grow to 97 billion, and the smartphones and smart devices will increase to 17 billion.

Improving the legacy infrastructure to cope with the overcoming demand in terms of bandwidth, coverage, quality of service (QoS) and specific requirements for emerging applications is costly and, therefore, not the ultimate solution. Thus, offloading traffic from core networks is a concern. For that matter, Device-to-Device (D2D) communication performs direct transmissions between peers in range without the need of a base station. Precisely, 5G has implemented a D2D communication protocol to enable such direct communication [7].

A type of network based on D2D communication is Opportunistic Networks (OppNets). OppNets are characterized by the mobility of their nodes, which leads to an undefined network topology that hinders contemporaneous end-to-end connectivity. Therefore, in OppNets, communication is led by the contact opportunity between peers. This paradigm is very convenient in networks as Vehicular Ad hoc Networks [73], Mobile Wireless Sensor Networks [60], Pocket-Switched [35], People-Centric [19], and Mesh Networks [4], among others.

OppNets, due to the mobility of their nodes, are prone to frequent disconnections, segmentation and long delay paths. Hence, traditional routing schemes based on end-to-end connectivity are not applicable. Therefore, OppNets nodes act as routers which use the principle of store-carry-and-forward (SCF) [24] to forward data from source to destination on a hop-to-hop basis. The idea behind the SCF paradigm is that nodes may temporally store data in their buffers and carry it until there is a communication opportunity to forward the data to another node or its destination. This scheme adds a new temporal dimension to forwarding decisions. Moreover, in OppNets, routing efficiency directly affects the network performance

[1] `https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html`

and is highly coupled with the type of application running over the OppNet [17]. In this environment, routing orchestration is a challenge. Therefore, forwarding in OppNets has been widely studied, such as in [52, 17, 68, 53], and [44].

In this regard, Software Defined Networks (SDN) [42] is a network paradigm applied to connected networks that programmatically orchestrates the traffic routing and network configuration by using a program called controller. Essentially, these network paradigms decouple the control plane (routers and switches taking high-level routing decisions) from the data plane (packet-forwarding). This decoupling results in a control plane driven by the aforementioned controller. The controller proactively gathers network information by running network application services on the underlying network elements and uses this information to overview the whole network. With this overview, the controller fulfils the application requirements by configuring a table with data flow rules instead of having them hardwired into a physical device's firmware. These rules are pushed-down to the data plane. Although SDN protocols are based on TCP and, therefore, are not straightforwardly applicable to OppNets, we take on board the concept of having a controller providing a dynamic context-aware system and SDN-like agents receiving/sending information to the controller.

Hence, the goal of this proposal consists in using the SDN building blocks to build a context-aware system over an OppNet. This context-aware system leverages from context information to dynamically configure the OppNet's forwarding policy parameters aiming to achieve better performance. In the proposed context-aware system, some OppNet's nodes perform as SDN-like controllers (controllers) and the rest as SDN-like agents (nodes). The nodes gather context measurements, specifically device measurements and send them opportunistically upon a hop-to-hop basis to the controllers, which will use this information to tune the forwarding algorithm parameters on the go. The nodes extend the SCF paradigm to (1) gather device context measurements, (2) aggregate the gathered measurements, (3) forward them to the controllers and (4) apply the received policies generated by the controllers. In this manuscript, we name this paradigm GAFA. We refer to the OppNet extended with the control system and the GAFA functionalities as *controller-driven* OppNet.

1.1 Contributions

In developing this proposal, we make the following contributions:

- Design of the architecture of a novel context-aware system for OppNets inspired by the SDN paradigm where nodes operate based on the GAFA paradigm to feed the controllers with device-context information and apply the controllers' policies. The controllers would use those feedings to tune the forwarding algorithm parameters through configuration policies emitted to the nodes to obtain a better network performance.

- Use of the controller-driven OppNet architecture for tackling the congestion in an OppNet characterized by a high unpredictability of the nodes' mobility and a multi-copy replication forwarding strategy. Specifically, the OppNet's controllers orchestrate the value of the replication limit of the forwarding algorithm based on the buffer occupancy readings gathered by the nodes.

- Evaluation of the performance and benefits of the controller-driven OppNet for the use case of congestion control: To evaluate our proposal, we have simulated diverse network scenarios based on real and synthetic mobility traces using different message generation distributions. We have evaluated the controller-driven OppNet on the basis of the standard performance metrics for OppNets. Also, we have run the aforementioned simulations over an OppNet without the control layer (context-oblivious) using an epidemic and a quota-based forwarding protocol. We have compared and evaluated the performance of both configurations. We have proved that a controller-driven OppNet performs better than a context-oblivious one.

1.2 Thesis Structure

The thesis document is structured as follows:

Chapter 2
This chapter introduces the state of the art of Opportunistic Networks, focusing on data forwarding and congestion control. Also, we accost SDNs, as our proposal adopts the fundamental architectural building blocks of such networks.

Chapter 3
This chapter presents the controller-driven Opportunistic Network generic architecture. It illustrates how the SDN structural building blocks are adapted to the OppNets specifics to build a supervised context-aware information system that uses context information to tune the forwarding strategy to improve the network performance.

Chapter 4
This chapter describes how the controller-driven OppNet architecture is used to manage congestion. First, we expose an OppNet using a multi-copy forwarding algorithm scheme facing a congestion issue as a drawback of its forwarding strategy.

Second, we show how to tailor the controller-driven OppNet architecture presented in Section 3 to mitigate the congestion effects of this particular forwarding strategy.

Chapter 5

In this chapter, firstly, we present the simulation-based testbed environment for evaluating the aforementioned controller-driven OppNet congestion use case. Secondly, we describe which are the different scenarios the controller-driven OppNet will be simulated over. Next, we determine the performance metrics used to evaluate the controller-driven OppNet against other OppNet benchmarking configurations. After, we specify the different message generation distributions used to simulate network traffic for all the testbed chosen scenarios. Finally, we provide the testbed-specific configuration parameters needed for its replicability.

Chapter 6

In this chapter, we show and evaluate the performance of the controller-driven OppNet compared with other OppNet benchmarking configurations. Also, we analyse the impact of the control-layer configuration settings on the delivery ratio performance for the testbed scenarios. Finally, we derive the default value for the aforementioned control-driven OppNet configuration settings for maximising the messages delivery ratio.

Chapter 7

Finally, this chapter presents the conclusions drawn from this work. Also, we envision the natural future lines derived from this research.

State of the Art

2

THIS CHAPTER overviews the nature of OppNets. It reviews the current positioning of Opportunistic Networks by reviewing which original OppNet targeted applications are still a niche for the OppNets technology despite emerging technologies aiming to provide universal connectivity. It also inspects which is, up to date, the real-life deployment of this technology. Next, provided that our proposal uses context information in the forwarding strategy, we review routing in Opportunistic Networks, focusing on context-based routing. As our proposal's targeted OppNet uses a multi-copy forwarding strategy prone to congestion, we inspect the congestion management strategies applied to OppNets, spotlighting the ones using context information. Finally, as the highlight of our proposal is incorporating the main building blocks of the SDN architecture, we overview the principal architectural elements of SDNs and explore how this architecture has been exported beyond connected networks. Figure 2.1 depicts the relation of our proposal with the concepts presented in this chapter.

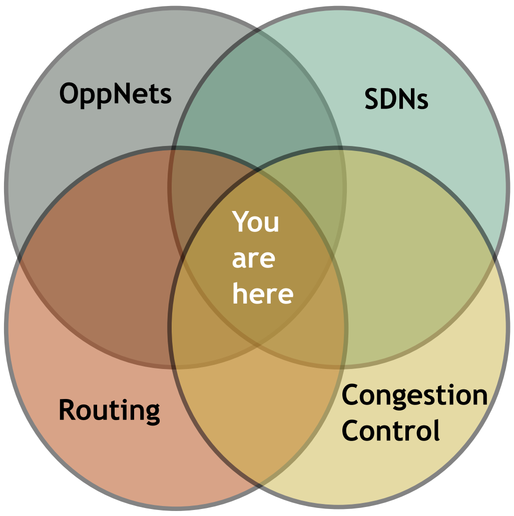


Fig. 2.1: Relationship between our proposal and the different presented concepts.

2.1 Opportunistic Networks overview

An OppNet is a structure-less multi-hop network built upon fixed and mobile nodes via wireless links. Due to the mobility of the nodes, OppNets are prone to disruptions, segmentation and long delay paths. Under these conditions, where there is no guarantee of an end-to-end path between the source and destination at a specific instant in time, the TCP/IP protocol suite is not effective. Hence, the communication in an OppNet is driven by the direct contact opportunity between peers in range, and data flow is achieved by exploiting the pair-wise contact opportunity provided by the nodes' mobility. OppNets have been conceived as a complement to connected networks to provide connectivity under specific conditions. Therefore, they are targeted to well-defined practical use case applications characterised by having a limited or even nonexistent infrastructure [24].

Trifunovic *et al.* in a recent review [77], characterize the current OppNets target applications. In their study, they review if the classical OppNets targeted applications are currently settled by new technologies providing a more reliable solution. Indeed, new technologies have emerged to provide a global scale Internet connectivity. They are classified into two main groups depending on the deployed infrastructure: (i) orbital, through low-earth orbit satellites or (ii) floating, through drones and high-altitude balloons. The two main representatives of the orbital infrastructure are Starlink by SpaceX [49] and OneWeb [32]. For the floating infrastructure, the main ones are project Loon by Google [39] and Internet.org by Facebook. Nevertheless, Trifunovic *et al.* state that these emerging technologies have intrinsic drawbacks which do not convert them in the ultimate solution for the traditional OppNet target applications, which leaves OppNets as the best solution so far.

According to [58, 52, 17] and more recently reviews, [77, 64], the beforementinoned traditional OppNets target applications are classified in five main use cases:

Challenged networks: These are networks where infrastructure is partially or fully unavailable. In this case, OppNets leverage their inherent instantaneous network support providing local communication and bridging to the available infrastructure. Challenged networks are classified in three groups:

- Emergency and extreme situations due to a natural or human-made disaster [6].
- Provision of Internet to rural or remote areas that could be inaccessible or do not seem cost-effective for companies to invest in infrastructure [38].
- Mines, gas and oil fields or construction areas where the signal suffers from a severe attenuation or where the geography hinders deploying an infrastructure [34].

Cellular traffic offload: During a crowded event, mobile operators can face bandwidth bottlenecks by the massive use of the same antenna, and thereby users could experience a low QoS. Mobile operators could rely on content cached in some access points and use opportunistic communication to propagate popular content within the attendants to decrease the load of their core network [83].

Censorship avoidance: Some governments and oppressive institutions censor the content disseminated through the Internet. Opportunistic communication favours the circumvention of this blockage as the information is disseminated out of the core network, and users are difficult to be tracked down [77].

Vehicular networks (VANET): In VANETs, vehicles communicate while in range with other vehicles and roadside units and share efficient driver assistance and safety information [75, 81].

Mobile Wireless Sensor Networks (MWSN): MWSN are an extension of Wireless Sensor Networks(WSN) where nodes are mobile. In WSN, nodes are small low-power sensors with low battery life, so the network lifetime is short [3]. One example of application field for MWSN is wildlife monitoring. Wildlife monitoring refers to studying large populations roaming vast areas without human intrusion. These monitoring systems consist in equipping the species with a tracking device provided with a GPS, a small wireless computing device and special tags with sensing capabilities and having one or several mobile base stations to collect the data and send them to the processing center. Jino *et al.* in [63], survey the up-to-date research in WSN for different application areas, including wildlife monitoring.

Three more promising OppNet target applications are envisioned by Trifunovic *et al.* in [77]:

Opportunistic Mobile Sensing: It exploits the handheld/wearable devices' sensors to opportunistically collect data to study human behaviour, interactions, and relationships [29].

Opportunistic Mobile Computing: It considers the OppNets opportunistic contacts to, apart from disseminating data, share computational resources [20].

Device discovery in the IoT industry: Consists in exploiting opportunistic encounters to discover IoT devices [15].

Also, Kuppusamy *et al.* in [44], propose a classification of targeted applications for OppNets based on the size of the area and the number of nodes the OppNet can scale up to. They consider applications for: group monitoring; local information/special interest exchange between a random group of defined people, such as neighbours, team members like firefighters, rangers, etc.; the exchange of information in a campus; data exchange in a crowded event as a concert; a smart city; a smart country; Industry4.0, and disaster alert.

Indeed, a lot of the proposed OppNet solutions for well-defined applications have ended up with the design of a prototype. Notwithstanding, demonstrating the feasibility of OppNets, there are deployed solutions. Trifunovic et at. in [77] point out several companies that commercialize an OppNet solution for different sectors. Some of the mentioned companies are *Open Garden*, which offers *FireChat* [10] that provides a secure and private off-the-grid communication used during the Hong Kong protests. *Uepaa!*[1] is a safety application used in areas with no cellular coverage. *GoTenna* [62] provides a device that pairs with the smartphone for sharing text and location on a peer-to-peer basis. Guidec *et al.* in [28], stand out as an issue, the computation, communication, and storage eager energy consumption of handheld devices. They overcome the former handicap by using an external add-on hardware called Ligo. *Ligo* is a middleware linked via Bluetooth to a handheld device and establishes an OppNet with other Ligo devices. Touseaau *et al.* in [76], use the infrastructure above to develop a framework to build web applications accessed opportunistically. Furthermore, Lee *et al.* in [46], list up-to-date deployed prototypes for VANETs.

2.2 Routing in Opportunistic networks

In connected networks, communication is based on end-to-end connectivity. In this context, routing involves finding the best route to send data across multiple networks through routers. Forwarding consists in transferring packets from the router's incoming link to the outgoing link that fulfils the determined routing path. Conversely, OppNets are prone to long delay paths, disconnections and segmentation, hindering end-to-end connectivity. Hence, traditional routing based on contemporaneous end-to-end connectivity is not feasible. Therefore, in OppNets, data forwarding is driven hop-to-hop using the Store Carry and Forward (SCF) paradigm originally designed for DTNs [36]. This forwarding paradigm consists in the node *storing* the data and *carrying* it along the network according to its mobility until a contact opportunity occurs and then *forwarding* the data to the contacted node.

Data forwarding is principal in OppNets as the application deployment relies on the forwarding as a guarantee of their particular QoS requirements [36]. Moreover, these mobile wireless networks face the handicaps that nodes are usually equipped with short-range wireless devices, which brings short contact times between nodes in communication range, and the high battery consumption of the D2D communication mode [28]. These characteristics should also be considered to determine which data dissemination strategy is used to fulfil the QoS required by an application [17]. Therefore, much research has been conducted on developing new routing and

[1]https://safety.uepaa.ch/. Available for download from major digital distribution services.

forwarding strategies. Some surveys group the up-to-date forwarding proposals into a proposed forwarding taxonomy such as [17, 68, 53], and the most recent up to our knowledge, Sachdeva *et al.* in [64]. It is worth pointing out the importance of having a forwarding strategy taxonomy to classify a forwarding algorithm. It also seems reasonable to associate an OppNet that follows a specific typology with the most suitable forwarding strategy for that specific typology [25]. Although each of the surveys mentioned above presents different taxonomies and specificity when determining the categories, there are correspondences. It is sensible to leave this classification open and allow the researchers to use the most suitable taxonomy to classify their proposed algorithm.

Seeking forwarding efficiency, Jain *et al.* in [36] state that context information helps to make a more efficient forwarding. Also, Mota *et al.* in [52] suggest that context information such as the state of the neighbourhood, the history of contacts, and network attributes, along with prediction-based models, help to make a more efficient forwarding. They also point out the overhead that implies the acquisition and sharing of such context information, and thereby, they conclude that it is a matter of finding a trade-off between the effectiveness of the context information and its overhead cost. In this line, CC *et al.* in [68] classify the data forwarding algorithms into two main categories depending on the context information used to make routing decisions: social-based routing and pure opportunistic routing. Social-based routing is applied to OppNets driven by social behaviours such as social routines, mobility patterns, and interests, among others, where user location and application context information is helpful to make forwarding decisions. On the other hand, pure opportunistic routing encompasses both context-oblivious and context-aware forwarding techniques. Although it can leverage social context information, context-based pure opportunistic routing usually uses device context information related to the device's resources as energy and storage consumption from the node itself and its neighbours. CC *et al.* go deeper in the pure opportunistic routing classification tree branch and take into account (i) the forwarding algorithm's message replicas: single-copy or multi-copy and (ii) the message delivery type: unicast, multicast or anycast.

Under the pure opportunistic routing category, sound forwarding strategies have been proposed in the literature. Those proposals fit well under determined network conditions and application requirements. In this regard, flooding-based strategies, consisting in message replication, have proved to maximize the delivery ratio with a low latency when the OppNet is characterized by the unpredictable nodes' movement [17]. Under the aegis of multi-copy forwarding, the epidemic flooding approach proposed by Vahdat *et al.* in [80] is a context-oblivious strategy prone to suffer from the congestion derived from the replication overhead. Spyropoulos *et al.* in [72] address the congestion overhead by establishing a static configured replication quota. Context-aware strategies aim to reduce the effects of a naive replication by calculating the utility of a relay based on history information. CC *et al.* in [68] highlight the most relevant routing proposals under this category. A metric is usually employed to select the best relay candidate, such as how often a candidate is seen by the forwarder (inter-contact time) or the expected probability of a relay contacting the destination node in the future.

The performance of a forwarding algorithm is normally measured in terms of delivery ratio, end-to-end delay, and communication overhead [52, 22]. The *delivery ratio* measures the successfully delivered messages out of the created messages. The *end-to-end delay* is the average time it takes a message to be delivered. The *overhead* is the rate between the number of extra messages put in the network over the delivered messages.

Commonly, the evaluation of the forwarding algorithm performance is conducted by simulating the forwarding algorithm and the baseline algorithms in the taxonomy category the former belongs. Mota *et al.* in [52] and, more recently, Dede *et al.* in [22], and Kuppusamy *et al.* in [44], highlight the importance of following essential guidelines for a valuable evaluation of the forwarding proposal. Dede *et al.* propose as best-practices for the forwarding protocol evaluation: to select an appropriate mobility model according to the specific use case; to leave to testbeds the evaluation of the physical aspects of the OppNet as battery consumption, radio propagation, interference impact, etc.; to use a simple link-model for the simulation unless this is important for the algorithm itself, and to document the model and the default configuration used so the algorithm can be compared within a benchmark. Kuppusamy *et al.* add to this list the importance of selecting relevant baseline protocols to compare with, comparing the forwarding algorithm performance with optimal solutions, and emphasizes using a scenario with a realistic number of nodes.

2.3 Congestion control in Opportunistic Networks

Network congestion is defined as the situation when the network-wide suffers from delays and a decrease in the delivery ratio due to the overload of the network resources (buffer occupancy or bandwidth) [74]. In the Internet, TCP handles congestion through buffer management and flow-based feedback by establishing an end-to-end connection that negotiates the transmission rate depending on the receiver's and network's capacity [5]. Nevertheless, in OppNets, traditional mechanisms based on contemporaneous end-to-end connectivity to provide congestion feedback are unsuitable. Also, due to the node's mobility, congestion at a link level is very rare, thus, buffer overload is the main issue. Hence, OppNets perform the congestion control by optimising the buffer resource by monitoring the buffer occupancy and the drop rate.

OppNets use the SCF paradigm to forward data across the network. In this paradigm, if a node is affected by congestion, meaning that the buffer is overloaded, the node will need to reallocate, drop queued messages, or reject incoming ones. Either case is highly undesirable as due to the mobility of the nodes, the short contact time and limited bandwidth, losing messages caused by the node congestion could lead to a delivery failure, affecting the network delivery ratio and the messages latency. Therefore, congestion control is especially transcendent for OppNets.

Silva *et al.* in [66] present an interesting taxonomy of congestion control techniques for OppNets that helps to either design or identify which strategies a congestion control mechanism is using. This taxonomy is also helpful in comparing different proposals. They survey congestion control mechanisms and indicate which are the congestion strategies in the proposed taxonomy those congestion control mechanisms are applying. One of the proposed congestion control mechanisms in this taxonomy is the inherent OppNet's routing algorithm. In this regard, Soelistijanto *et al.* in [69] classify congestion control mechanisms depending on if the forwarding is based on a single-copy or multiple-copy. Both studies state that some congestion control mechanisms rely on the forwarding strategy.

Next, we go through the principal congestion control techniques for OppNets, focusing on the research relevant to our proposal. First, we will tackle the main progress in buffer management. Second, we will inspect congestion avoidance, and finally, we will go through resource allocation.

Buffer management is a strategy for congestion control applied on the relayed node [70]. Buffer management determines which messages must be dropped in case an incoming one needs to be fitted in. The basic buffer management policies are based on the local's node information such as message priority, lifetime, size, or delivery probability. Krifa *et al.* in [43] state that basic buffer management policies as drop-tail, drop-head, drop-oldest, drop-random, etc., are suboptimal. They propose an optimal policy associating a utility function to each queued message, producing a marginal value for a selected optimization metric (delay or delivery ratio). They use statistical learning about encounters to approximate the global knowledge of the network. This global knowledge is an input for the utility function. The messages with the lowest utility are the first ones to be discarded. Pan *et al.* in [57] propose a mechanism that integrates all the aspects of buffer management: (1) a queue policy by assigning a priority to every queued message; (2) a drop policy by assigning a utility measure to every queued message based on the message's remaining TTL, its size, and the number of times the message has been relayed; (3) a congestion policy based on limiting the number of copies of a message in the network; and (4) a mechanism to delete message's copies that have been already delivered.

Another congestion control strategy applied by the sending node is congestion avoidance [70]. Under this category, Goudar *et al.* in [27] state that basic congestion measures based on buffer management, such as message drops, are not accurate in detecting congestion. They state that under the inherent characteristics of a Mobile OppNet, congestion may occur before buffers are overwhelmed. They propose an analytical model where the forwarder node calculates the instantaneous forwarding probability of a relay out of the amount of the node's buffered messages to be relayed to the contacted node given a contact time and bandwidth. They experience that this probability decreases dramatically beyond certain buffer occupancy (buffer occupancy threshold). The node is considered to be congested when it reaches this occupancy threshold.

Also, Lakkakorpi *et al.* in [45] state that an effective congestion control system should not be based on the network conditions at the time the message was created. Instead, the congestion control system should consider the current network conditions before relaying the message. They propose a mechanism where each node, upon a contact, shares its buffer availability. Nodes use this information to determine if a relay node has enough buffer resources to custody the message. Thomson *et al.* in [74], measure the congestion as the ratio of drops over message replication per node. They use this information to adjust the replication limit of the messages. Goudar *et al.* in [26] propose a probabilistic model using an estimator to predict the average buffer occupancy of the nodes in the network. They use this information to discard relay nodes without enough storage to hold the messages to be relayed. Similarly, Batabyal *et al.* in [8] derive a steady-state probability distribution for buffer occupancy.

Another congestion control strategy is the resource allocation technique. This approach decouples the routing protocol from congestion control. Resource allocation involves moving stored messages from a congested node to nodes with available resources to prevent message loss. For that matter, Seligman *et al.* in [65] propose a distributed reallocation between different nodes for achieving a load-balanced storage. Using the principle of resource allocation, Son in [71] performs an early congestion detection by monitoring the increasing occupancy rate and predicting the congestion through an exponential weighted moving average (EWMA). In the case of predicting a buffer overflow, the system unloads the node's buffer by sending the unloaded message to another node. The drawback of the resource allocation technique is that the network link disruption may preclude contacting other nodes. Overall, as Silva *et al.* in [66] proclaim, in OppNets is better to take early measures to prevent congestion.

2.4 Software Defined Networking

The present-day Internet faces high throughput, speed and reliability requirements derived from cloud computing, extensive multimedia streaming, and pervasive IoT, among others. IP networks are based on predefined built policies. Configuring IP networks to be dynamically adaptive to changes is very difficult. Software-Defined Networking (SDN) [33, 42] is a new architecture that offers flexibility, scalability and adaptability for high-throughput internet applications at a very effective cost by using simplified hardware, software, and management. Signh *et al.* in [67] define SDNs as a programmable network that provides services on the fly. The fundamental principle of the SDN paradigm is that applications' or services' network requirements define the optimal network settings in terms of routing, bandwidth, lifetime, priorities and policies.

Current IP networks are vertically integrated as the control and data planes are bundled. The Control plane manages setting up routing flows, making routing decisions and applying network policies. Once the control plane has configured data flows, they are pushed down to the data plane. The Data plane handles data forwarding and data processing from the underlying routers and switches at a physical level. Conversely, SDN architecture entirely separates the data and the control planes in a horizontal integration way. It promotes the use of a software, referred to as "controller", which runs network services to gather network information from the data plane, out of which builds a perception of the global state of the network. Thereby, the controller sees the network as a single logical switch. The controller uses this global perception of the network to define high-level data flow rules, which pushes down to the data plane. It also can configure the data plane hardware's physical settings. It behaves as a single software component despite its possible distributed implementation. It programs the network, i.e. sets up the control plane dynamically, based on the network state and the application's network requirements. The control plane communicates with the data plane through a vendor-neutral well-defined API as OpenFlow [50].

OpenFlow is an open protocol that programs the data plane's flow tables (packet forwarding tables). The Data plane's switches/routers must be compliant with OpenFlow (or similar). OpenFlow switches no longer have the capacity to make packet-forwarding decisions, as such functionality has moved to the controller. The OpenFlow switch has a flow table containing rules which define actions. Those actions consist in forwarding or dropping packets based on the rule's defined policy. The controller communicates with the switch to update the switch's flow table through a secure channel.

Nevertheless, decoupling the control and data planes adds overhead to communications as control information is generated between the controller and the data plane (southbound communication). In return, programming the controller makes network management easier and more flexible than configuring the network at the interface level. With this decoupling between data and control plane, changes in the network's underlying infrastructure have a low impact on the applications.

SDN has been widely investigated for wired networks in the context of network providers, enterprise networks and data centres [31]. Several surveys explore the SDN architecture's feasibility and effectiveness, its protocols, services, applications, standards and design progress [42, 82, 56, 82, 33], and more recently [67, 84]. The demonstrated benefits of the SDN architecture have brought out the possibility of extending this architecture to wireless or hybrid networks. In this regard, Hacke *et al.* in [31] evaluate the adoption of the SDN paradigm to wireless, cellular, mesh, local and sensor networks. Kobo *et al.* in [41] conduct a study about Wireless Sensor Defined Networks (WSDN). They present the benefits of the SDN architecture over wireless networks and the latest WSDN architectures, protocols and implementations.

SDN are also present in Vehicular Ad hoc Networks (SDNVN). As examples, Chahal *et al.* in [16] present the insights of an SDNVN and the up-to-date deployed applications. Nkenyereye *et al.* in [54] present a classification of the existing SDNVN proposals based on their modelling and implementation to help decide which is the most suitable approach for a Vehicular Ad hoc Network service. Nobre *et al.* in [55] explore using fog computing in a software-defined vehicular network to improve the delay in disseminating emergency alerts.

SDN architecture has also been exported to IoT. As an example, Bera *et al.* in [9] explore different SDN-based technologies that can be adapted to IoT to provide a centralized controller to overview, monitor and coordinate IoT devices. Qin *et al.* in [59] develop an IoT SDN controller on top of the MINA (Multinetwork Information Architecture) middleware. This IoT SDN controller helps to give certain quality levels for differentiated tasks running over the IoT multinetwork. Rafique *et al.* in [61] goes through the state of the art on integrating an SDN controller to orchestrate an IoT multinetwork powered by edge computing.

5G enables device-to-device communication (D2D) to offload traffic from the core network to out-of-band technologies such as WiFi, Bluetooth, and ZigBee [79]. In this regard, as an example, Usman *et al.* in [79] propose an SDN architecture to deploy public safety applications, which have stringent requirements in terms of very low latency and high reliability. They use SDN controllers to orchestrate what they call mobile cloud devices, which use D2D communication. Those controllers coordinate with a central SDN controller in the core network. Other examples of how to use SDN and D2D communication are available at [48, 13, 30, 1].

Finally, to our knowledge, SDN and OppNets have converged by Li *et al.* in [47]. The authors apply the SDN paradigm in OppNets, introduced as Software Defined Opportunistic Networks (SDON), to implement a mobile crowdsensing system. In the proposed SDON, data are sent opportunistically, off-the-grid. However, SDN architecture has been conceived to be applied to a wired infrastructure to ensure reliable communication between the data plane and the controller. Therefore, the authors use a cellular network to send the control messages to guarantee southbound communication reliability. This hybrid approach has some privacy drawbacks. Using a cellular network lets the infrastructure provider know the node's position and, thus, the node's owner. With these premises, the presented research proposes an *SDN-like* architecture entirely opportunistic.

Finally, the proposed controller-driven OppNet architecture combines the concepts described in this chapter through a specific use case. This use case consists in a controller-driven OppNet using a multi-copy-based forwarding algorithm where the congestion control is driven by SDN-like controllers that manage a context-aware system based on the buffer occupancy of the nodes. More specifically, according to the congestion control mechanisms taxonomy proposed by Silva *et al.* in [66], the SDN-like controllers apply a threshold-based-hybrid-loop approach by combining the neighbourhood congestion information with a local perception of it to dynamically determine the replication limit used by the quota-based forwarding algorithm. Overall, the resulting congestion control mechanism applied by the controller-driven use case is proactive, as it works with a prediction of the congestion to avoid this congestion from happening beforehand, and it is also reactive as, if a message can not be allocated in the buffer, it locally applies basic drop policies.

Part II

Proposal

Control layer architecture 3

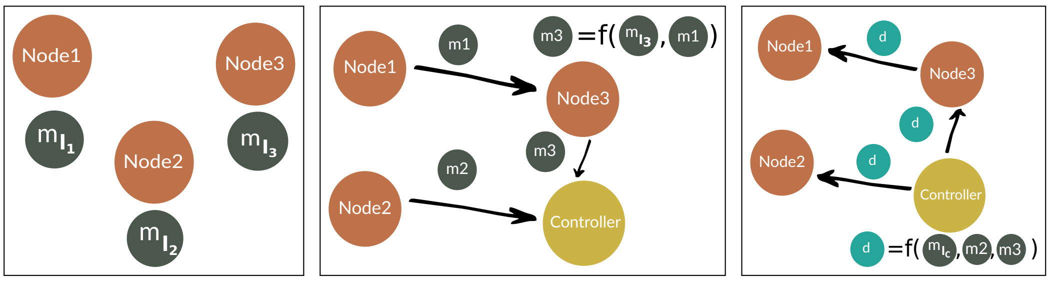
IN THIS chapter, we propose a context-aware architecture inspired by the building blocks of the SDN architecture: controllers and agents. First, we present an overview of the proposed architecture. After, we go through the insights of the proposed architecture. For that matter, initially, we describe the context-information entities the control layer works with. Then, we describe how the SDN-like agents (nodes) retrieve, process and disseminate this context information. Next, we describe the proposed SDN-like controller (controller) functionality regarding how it uses the information provided by the nodes to predict the future value of the context indicators. We describe how the controller uses this prediction to tailor the parameters of the forwarding strategy aiming to achieve a better network performance. Finally, we present how the controller disseminates the aforementioned forwarding algorithm setting value across the OppNet for the nodes to apply it.

3.1 Control layer overview

This proposal takes on board the controller concept from the SDN architecture. We have designed a control layer running on top of the convergence layer of the nodes as a context-aware system. Some of these nodes, the ones selected to be controllers, run the controller module of the control layer. Like the SDN controller, the proposed controllers keep an overview of the network by gathering network measurements from the data plane. On the other hand, in OppNets, the control and data planes are coupled in the node. Thereby, the controller-driven OppNet nodes perform the GAFA functionality introduced in Section 1 for gathering, aggregating and disseminating network measurements.

As mentioned before, a controller is a node running the controller module over the control layer. Any node could potentially be a controller. Which nodes act as controllers depend on the nature of the network. In a vehicular network (VANET), the controllers could be the roadside units; in an information-centric network, they could be well-connected nodes. For this particular research, it is considered a generic OppNet. Thus, the generic criteria of selecting the more central nodes, i.e. the ones with more contacts, has been applied.

Figure 3.1 shows the big picture of how the control layer implements the node's GAFA and the controller functionalities. Firstly, the nodes in the OppNet sense



(a) Taking network measurements. (b) Measurements aggregation and dissemination. (c) Directive creation and dissemination.

Fig. 3.1: Node's GAFA functionality: (a) gathering measurements; (b) aggregating and forwarding them; (c) applying the directive created by the controller.

local context measurements (Figure 3.1a). Detailed information can be found in Section 3.2. The nodes disseminate an aggregation of context measurements upon a contact (Figure 3.1b). Section 3.4 develops the aggregation methodology and Section 3.3 shows how the dissemination of context information is performed. Finally, nodes acting as controllers come into action (Figure 3.1c). Section 3.5 presents the logic of the controllers. Section 3.6 develops how the controller processes the received context information to get a prediction of a network indicator in a future time. Out of this prediction, the controller determines an action to be done by the nodes consisting in the modification of a forwarding algorithm parameter. Section 3.7 details how the controller creates this action. The controller disseminates this action upon a contact with another node (see Section 3.8).

3.2 Control metadata

The control metadata used in the control layer is context measurements and control directives. When two nodes come into range, they disseminate control metadata before delivering or relaying buffered messages. The following two sections describe the beforementioned concepts.

3.2.1 Context indicator measurement

A *Context indicator measurement* is the local reading of a context indicator taken at a time by the nodes and the controllers (Figure 3.1a). The controller receives those measurements upon opportunistic contacts with nodes (Figure 3.1b) and builds an overview of the network based on this information (Figure 3.1c). In an OppNet, the contact time, bandwidth, and nodes' energy are limited resources, so we have opted to aggregate those measurements (see Section 3.4). Therefore, the context information we consider is this aggregation.

3.2.2 Control directive

A *control directive* is an action to be performed by the contacted node to modify a forwarding algorithm parameter. A directive is represented as the tuple: $\delta_s = (Id_s, \vartheta_s)$, where Id_s identifies a forwarding algorithm setting from the list of settings managed by the controller, and ϑ_s is the value for this setting. The controller generates the directive based on context information (Figure 3.1c) aiming to improve the forwarding performance. The controller disseminates this directive to the nodes. When a node receives the directive, applies it by modifying the node's forwarding setting Id_s, with the new value ϑ_s. As examples of forwarding settings, we could consider the message TTL, weights and thresholds intrinsic to the forwarding algorithm, among others.

3.3 Context measurements dissemination

In this section, we describe how the node processes and disseminates the local context indicator measurement and the context indicator measurements received from contacted nodes.

The node that receives a context measurement stores it in an indexed list. Considering $\{n_1, \cdots, n_z\}$ the set of nodes in the network at time t, the indexed list of received context measurements for the node n_i for $1 < i \leq z$ is represented as $M_i = [m_j \mid 1 \leq j \leq z]$ where $M_i(j) = m_j$.

Figure 3.2 shows how a node (*n1*) disseminates its context measurement when comes into range with another node (*n2*). The control metadata is shared bidirectionally, hence, *n2* will follow the same flow when it is its turn to do the dissemination.

In Figure 3.2, *step* s_1, when node n_1 contacts node n_2, n_1 takes a local context indicator reading, m_l. In *step* s_2, n_1 aggregates this context reading along with all the context measurements received by contacted nodes stored in the list M_1. The aggregation process is described in Section 3.4. In *step* s_3, the aggregated measurement, m_1, is shared with the contacted node n_2 which stores it in its context measurements list M_2 (*steps* s_4 and s_5). At this point, n_2 follows the same flow to calculate m_2 with the slight difference that it will not use the the entry $M_2[n_1]$. This entry contains the received measurement m_1 built out of the information provided by n_1. Using m_1 would interfere with the network perception n_2 is about to share with n_1.

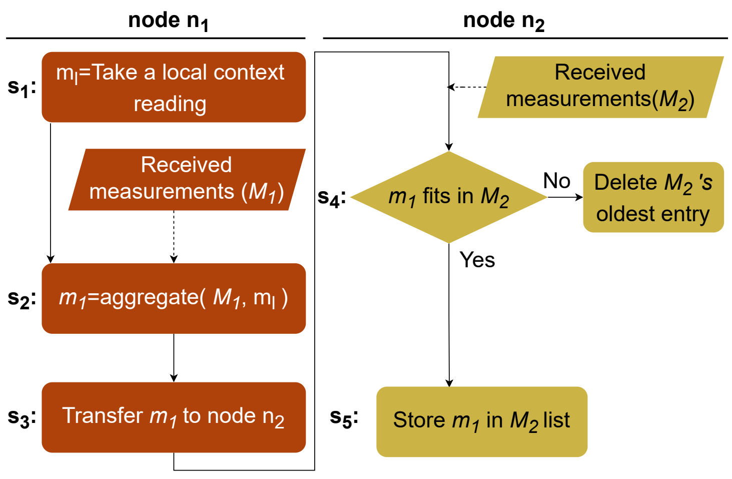


Fig. 3.2: Context measurements dissemination when two nodes are in range.

3.4 Context measurements aggregation

As we have mentioned in Section 3.2.1, due to the resources constrain in an OppNet, instead of sharing with the contacted node the context measurements list, the node aggregates this information along with its context indicator reading (v_l). Considering $\{n_1, \cdots, n_z\}$ the set of nodes in the network at time t, we represent the aggregated context measurement generated by node n_i, for $1 \leq i \leq z$, as the tuple $m_i = (v_i, \eta_i, t_{c_i})$, where v_i is the aggregated result; η_i is the number of measurements used for the calculation of v_i, and t_{c_i} is the time at what the calculation was made. The node n_i's local context measurement is represented as $m_{l_i} = (v_{l_i}, 1, t_{c_i})$, where $\eta_i = 1$ as v_{l_i} is a straight reading, not the result of an aggregation.

Algorithm 1 shows how the context measurements are aggregated. Specifically, the process that performs the aggregation is *getAggregatedContextMeasurement()* (*line 28*). First, the node n_i gets the local context reading, v_l (*line 29*). The node uses equation 3.3 as a two-factor weighted average to aggregate all the context measurements stored in M_i and its context reading v_l. Each one of the context measurements (m_j) stored in M_i, for $1 \leq j \leq z$, is weighted by two factors: (1) the number of aggregations used to create it (η_j), in the case of the local measurement would be 1, and (2) its decay (d_j) at the current time t. In the case of the local measurement, the decay would be 1 (no decay).

The following logistic function calculates the decay of a measurement at time t:

$$d(t) = \frac{1}{1 + r^2 t} \tag{3.1}$$

where r is the reduction factor, or decay degree, to apply to get a certain decay. By isolating this variable we obtain:

$$r(t) = \left(\frac{1-d}{td}\right)^{\frac{1}{2}}. \tag{3.2}$$

This resulting equation is used to get the necessary decay degree (r) to be used in (3.1) to get the desired decay at a particular time *t*. The decay is inversely proportional to time (t), and it goes from 0 to 1, where the decay of 1 means no decay. With a decay of 1, the congestion reading is not lowered when aggregated. In contrast, the older the reading is, the higher the decay (lower value), hence the more diminished the reading is when the controller aggregates it along with the other readings received during the aggregation interval.

The aforementioned two-factor weighted average used to aggregate the context measurements in M_i along with the perceived local context reading v_l (*lines 32-39*) in Algorithm 1 is:

$$v_i = \sum_{j=1}^{k} v_j(\alpha\frac{\eta_j}{\sum_{p=1}^{k}\eta_p} + (1-\alpha)\frac{d_j}{\sum_{p=1}^{k}d_p}) \tag{3.3}$$

where v_i is the value resulting from this aggregation; k is the size of M_i plus one (to include the local context reading); v_j is the $value$ of the measurement being aggregated ($m_j = M_i[j] = (v_j, \eta_j, t_{c_j})$); η_j is the number of aggregations used to generate m_j; d_j is the decay of m_j at the current time; and α is the weight factor, specified as a control setting (see Section 5.4.2), used to weigh the two factors of the weighted average. The function *getSumOfNrofAggrs()* at *line 17*, calculates the normalizing factor applied in (3.3) over the number of aggregations the context measurement being processed is formed by ($\sum_{p=1}^{k}\eta_p$). Similarly, the function *getSumOfDecays()* (*line 1*) calculates the normalization factor to be applied over the decay of a measurement ($\sum_{p=1}^{k}d_p$).

Finally, the node n_i creates the aggregated measurement $m_i = (v_i, \eta_i, t_{c_i})$ where v_i is the calculated aggregation value; η_i is the number of entries in M_i that have been aggregated plus the node's own reading (*lines 32,39*); and t_{c_i} is the current time. At this point, *step 3* in the flowchart in Figure 3.2, the calculated aggregated measurement m_i is transferred to the contacted node.

Algorithm 1 Context measurements aggregation algorithm.

```
▷ $M_i$: List of context measurements received from contacted nodes.
▷ *hostID*: Identifier of a host.
▷ *excludeHost*: Id of the host whose measurement in $M_i$ will not be used.
▷ *sumDecays*: Sum of the decays of all the measurements in $M_i$.
▷ $M_i\_keys$: Measurement list indexes.
▷ $m$: A received context measurement represented by the tuple $(v, \eta, t)$
▷ *threshold*: Decay threshold to determine that a measurement is expired
▷ *sumNrAggr*: Sum of the # aggregations a measurement is made of.
▷ $\#aggrEntries$: # of elements in $M_i$ that have been aggregated.
▷ *aContextReading*: A local context measurement.
▷ $v_i$: Aggregated measurement value.
function GETSUMOFDECAYS($M_i$,excludeHost)
    $sumDecays \leftarrow 0$
    for all $hostID \in M_i\_keys$ do
        if $hostID$ != $excludeHost$ then
            $m \leftarrow M_i[hostID]$
            $d \leftarrow decay(current\_time - m[t])$
            if $d < threshold$ then
                $M_i$.remove(hostID)
            else
                $sumDecays += d$
            end if
        end if
    end for
    $sumDecays ++$                                   ▷ Adding the decay of my own reading.
    return $sumDecays$
end function
function GETSUMOFNROFAGGRS($M_i$, excludeHost)
    $sumNrAggr \leftarrow 0$
    for all $hostID \in M_i\_keys$ do
        if $hostID$ != $excludeHost$ then
            $m \leftarrow M_i[hostID]$
            $sumNrAggr += m[\eta]$
        end if
    end for
    $sumNrAggr ++$                                   ▷ Considering its own reading.
    return $sumNrAggr$
end function
function GETAGGREGATEDCONTEXTMEASUREMENT($M_i$, excludeHost)
    $v_l \leftarrow aContextReading$
    $decays \leftarrow getSumOfDecays(M_i, excludeHost)$
    $aggrs \leftarrow getSumOfNrofAggrs(M_i, excludeHost)$
    $\#aggrEntries \leftarrow 1$                        ▷ Considering its own reading.
    $v_i \leftarrow v_l((\alpha \frac{1}{aggrs}) + ((1-\alpha)\frac{1}{decays}))$
    for all $hostID \in M_i\_keys$ do
        if $hostID$ != $excludeHost$ then
            $m \leftarrow M_i[hostID]$
            $d \leftarrow decay(current\_time - m[t])$
            $v_i += m[v]((\alpha \frac{m[\eta]}{aggrs}) + ((1-\alpha)\frac{d}{decays}))$
            $\#aggrEntries ++$
        end if
    end for
    return $m_i = (v_i, \#aggrEntries, t)$
end function
```

3.5 Controller architecture

As we have mentioned before, the controller is a software service that runs over a subset of network nodes. The way to decide which nodes are running as controllers can be diverse and mainly depends on the nature of the network.

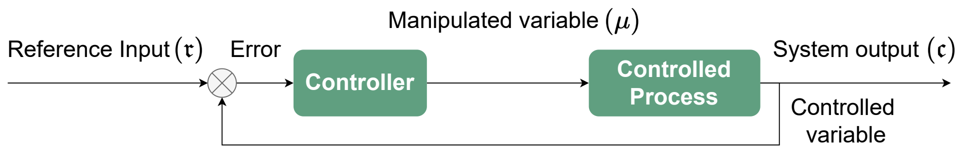

Manipulated variable (μ)
Reference Input $(\mathfrak{r})$
Error
Controller
Controlled Process
System output $(\mathfrak{c})$
Controlled variable

Fig. 3.3: Closed-loop control system.

The goal of a controller is to have an overview of its nearby part of the network, considering that the mobility nature of the nodes keeps changing the network's topology. This mobility brings on a network "segmentation" in terms of groups of nodes that eventually are connected between them. Ideally, some controllers would be required to cover all the possible network "segments".

The controller operates opportunistically, i.e. its actuation is triggered by contacting another node or controller. When that happens, the controller shares an aggregated context measurement and a directive with the contacted node.

The controller calculates an aggregated context measurement to be shared as a node does. To generate a directive, the control layer implements the closed-loop control system in Figure 3.3, also known as feedback control system [23]. A closed-loop control system is a control system that maintains a constant relation between the output of the system ($\mathfrak{c}$: controlled variable) and the desired value ($\mathfrak{r}$: the reference input) by subtracting one from the other as a measure of control.

In the proposed control system, the controller feeds from incoming context aggregated measurements (m_i) and determines the value of the manipulated variable μ out of those measurements and the system's reference input $\mathfrak{r}$. The resulting manipulated variable μ is encapsulated in an outgoing directive and shared with any node that might come into contact.

Besides, Goudar *et al.* in [26] state that considering the OppNet current conditions to trigger an actuation is not effective as it would be too late, given the variability of the network. Following their considerations, proactively, we decide the actions to be taken based on predicting the network situation. By this time ahead, presumably, the directive would have been propagated and applied throughout the network. Indeed, if the control system would directly inject the raw aggregated context measurements received from other nodes to the controller for generating a directive containing μ, when eventually this directive would reach the nodes, the network situation might have changed and, possibly, the directive would be no longer adequate for the current situation. The following section shows how we predict the value of a context indicator.

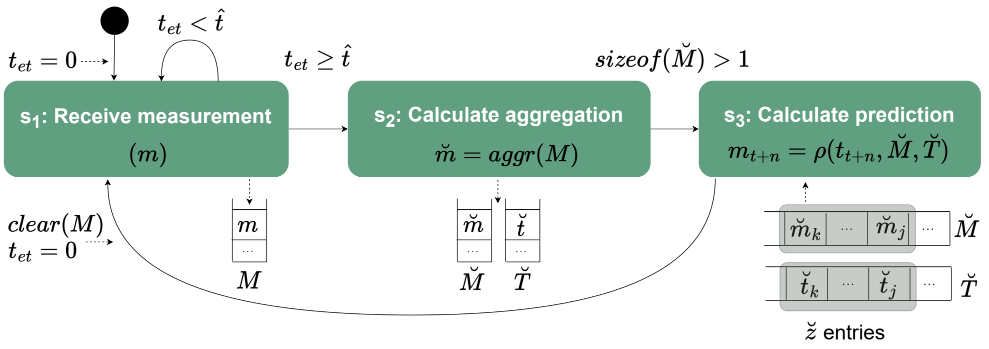


Fig. 3.4: State diagram for metrics aggregation.

3.6 Context indicator prediction

The controller anticipates the context indicator value using a two-step strategy. Firstly, instead of directly considering the received context measurements, the controller aggregates them for a configurable time ($\hat{t}$) using (3.3). This step results in two lists: $\breve{M}$ and $\breve{T}$. $\breve{M}$ contains the aggregation of the measurements received for periods of $\hat{t}$ seconds. This list is represented as $\breve{M} = [\breve{m}_j \mid 1 \leq j \leq \breve{z}]$, where $\breve{z}$ is the max size of $\breve{M}$. $\breve{T}$ contains the time when each entry in $\breve{M}$ was calculated, and is represented as $\breve{T} = [\breve{t}_j \mid 1 \leq j \leq \breve{z}]$. Secondly, once several aggregated samples of context measurements are available, these are used as an input of a linear regression (ρ) to calculate the prediction of this context indicator value for time t_{t+n}:

$$m_{t+n} = \rho(t_{t+n}, \breve{M}, \breve{T}) \tag{3.4}$$

where $\breve{M}$ and $\breve{T}$ are sliding lists of size $\breve{z}$; t_{t+n} is the time ahead for the context indicator prediction and it is calculated as $t = t + n$ where n is an offset; and m_{t+n} is the resulting predicted value of the context indicator value at time t_{t+n}.

Algorithm 2 and the state diagram in Figure 3.4 show how the controller calculates the prediction of a context indicator. The *addContextMeasurement()* procedure is executed by the controller when a contacted node shares its aggregated context measurement (m). This measurement is stored in the controller's received measurements list M_i (*line 2*) (*state* s_1 in the state diagram). If the window time for receiving measurements from contacted nodes has expired (*line 3*), the controller aggregates all the received aggregated context measurements in the list M_i and its local context measurement, by using the procedure *getAggregatedContextMeasurement()* (*line 5*) defined in Algorithm 1. This aggregation result ($\breve{m}$) is stored in the list $\breve{M}$ along with the current time (*lines 6-7* of Algorithm 2) (*state* s_2). The controller uses the entries in the above lists to calculate a prediction of the value of the context indicator (m_{t+m}) using a linear regression function ρ (*line 10*) (*state* s_3). The controller keeps the size of the aggregation calculations list $\breve{M}$ to the constant value $\breve{z}$ by using a

Algorithm 2 Controller's context indicator prediction.

```
▷ $M_i$: List of context measurements received from contacted nodes.
▷ $m$: Received context measurement (controlled variable).
▷ $t$: Current time.
▷ $aggrTimeout$: Timeout for aggregating incoming context measurements.
▷ $\hat{t}$: Time period for aggregating incoming context measurements.
▷ $\breve{m}$: Aggregation of the received context measurements during $\hat{t}$ s.
▷ $\breve{M}$: Sliding list of the aggregations so far.
▷ $\breve{T}$: Sliding list of the times when the aggregations were performed.
▷ $t_{t+n}$: Prediction time.
▷ $m_{t+n}$ Predicted value of the context indicator at $t_{t+n}$.
▷ $directive$: Manipulated variable ($\mu$) encapsulated in a directive
1: function ADDCONTEXTMEASUREMENT($m$)
2:     $M_i.add(m)$
3:     if $t >= aggrTimeout$ then
4:         //The window time for receiving context measurements has finished.
5:         $\breve{m} \leftarrow getAggregatedContextMeasurement(M_i, NULL)$
6:         $\breve{M}.add(\breve{m})$
7:         $\breve{T}.add(t)$
8:         if $\breve{M}.size() > 0$ then
9:             if $\breve{M}.size() > 1$ then
10:                $m_{t+n} = \rho(t_{t+n}, \breve{M}, \breve{T})$
11:                //Just keep $\check{z}$ values.
12:                $\breve{M}.removeEldestN()$                    ▷ Sliding the list.
13:                $\breve{T}.removeEldestN()$
14:            else
15:                $m_{t+n} = \breve{m}$
16:            end if
17:            $directive \leftarrow createDirective(m_{t+n})$
18:        end if
19:        $M_i.clear()$
20:        $aggrTimeout \leftarrow t + \hat{t}$
21:    end if
22: end function
```

FIFO discarding policy (*lines 12 - 13*). Notice that at least two inputs are needed to use the linear regression function ρ (*lines 8* and *9*). If this condition does not fulfill, the predicted measurement (m_{t+n}) gets as value the $\breve{m}$ calculated at *line 5* (*line 15*). Once m_{t+n} is calculated, M_i is emptied, ready to receive new context measurements from other contacts (*line 19*). A new aggregation window period ($aggrTimeout$) is configured, so all the measurements gathering and the prediction process starts over (*line 20*).

3.7 Directive generation

As mentioned in Section 3.5, the controller operates opportunistically upon a contact with a node/controller. So far, when a node comes into contact with a controller, the controller's function $addContextMeasurement(m)$ from the prediction algorithm (Algorithm 2) is executed. From this algorithm, at the end of the window time for collecting context measurements from contacted nodes, the controller generates a context indicator prediction (m_{t+n}) out of those measurements and its local one. The function *createDirective*(m_{t+n}) (*line 17*, Algorithm 2) is called to generate a

Algorithm 3 Directive creation algorithm.

```
    ▷ $m_{t+n}$: Predicted measurement.
    ▷ *opp*: Execution mode's flag.
    ▷ $\mu$: Manipulated variable's current value.
    ▷ $\mu$': Manipulated variable's new value.
    ▷ $\mathfrak{r}$ Controller system's reference input.
1: function CREATEDIRECTIVE($m_{t+n}$, $opp$ = FALSE)
2:     $\mu' = \mu$
3:     if $opp == false$ then
4:         $\mu' \leftarrow$ apply controller_adjustment($\mathfrak{r}, m_{t+n}$)
5:     end if
6:     return new Directive($\mu'$)
7: end function
```

directive encapsulating the calculated context indicator prediction (m_{t+n}). The former function, defined in Algorithm 3, calculates the manipulated variable's value (μ') out of the context indicator prediction and the reference value (*line 4*). Next, μ' is encapsulated in a directive: $\delta_s = (Id_s, \mu')$ (*line 6*).

As aforementioned, the reception of a context measurement after contacting a node triggers the generation of a directive. Nevertheless, if the window time for receiving context measurements is set to a high value, it would take a controller a long time to generate a directive. Hence, the nodes would not receive any directive to adjust their initial configured manipulated variable (μ) based on the current network condition, and they would have the perception that there is no controller nearby. Therefore, to prevent this situation, the controller is configured to generate a directive periodically, provided no directive has been generated opportunistically during this period. This directive encapsulates the last calculated manipulated variable (μ), and it acts as a beacon announcing the presence of a nearby controller. Algorithm 3 describes the above behaviour.

Notice that the function *createDirective()* in Algorithm 3, receives the parameter *opp* which indicates whether the function is executed periodically, as described above or opportunistically after receiving a context measurement from a contacted node (see Algorithm 2). In the case of a periodic execution of the function *createDirective()*, the new value for the manipulated variable (μ') is directly the current one (μ) (*lines 2* and *6* in Algorithm 3). In the case of an opportunistic execution (*line 3*), the controller calculates the new value for the manipulated variable (*line 4*).

3.8 Directive dissemination

Although it is a controller that generates a directive, it is stored, carried and forwarded by any node in the network that receives it. Figure 3.5 shows this behaviour. When a node receives a directive (dir_{n_1}) (*step* s_4), if the directive is newer than

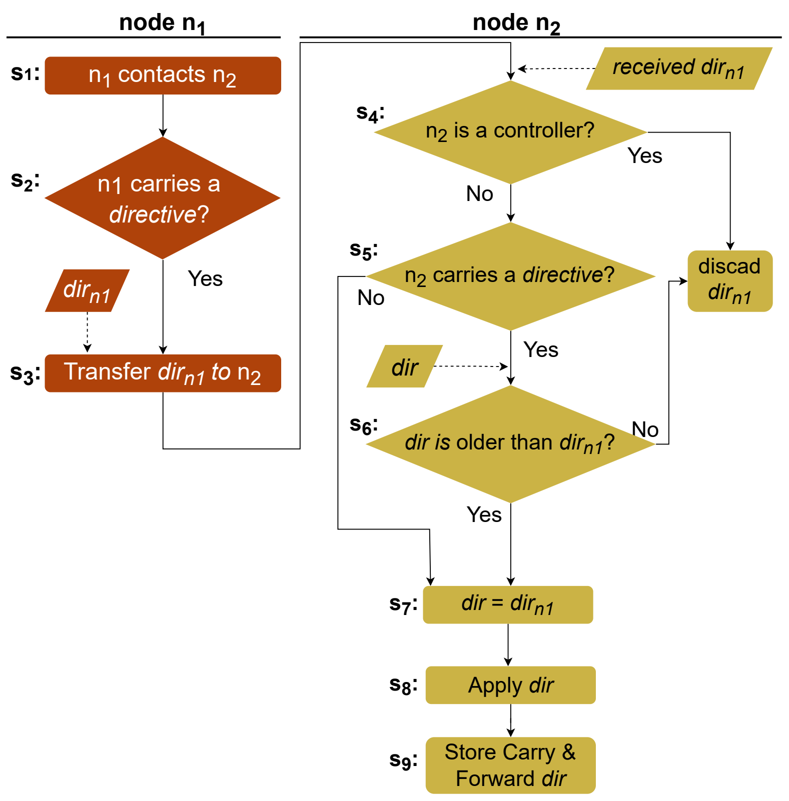


Fig. 3.5: Disseminating a directive when two nodes are in range.

the one the node might be carrying (*steps* $s_5 - s_6$), the node discards the old one (*step* s_7), executes the action encapsulated in the new directive (*step* s_8), and stores, carries and forwards the new directive (*step* s_9).

Discarding the older directive is a measure to deal with possible inconsistent directives since several controllers are allowed in the network. The fresher a directive is, the closer the node is to the controller and, therefore, the more appropriate the directive is. If a node receives several directives, if they are consistent, it means that the controllers are also nearby and are mainly sensing the same. Conversely, if the received directives are inconsistent, it glimpses that the controllers are sensing different parts of the network. In this case, the node likely belongs to the network "segment" controlled by the controller generating the fresher directive.

4 Use of the controller-driven OppNet architecture to manage congestion

THIS CHAPTER describes how the control layer, presented in Chapter 3, is applied to control the network congestion. As pointed out in Chapter 1, we focus on forwarding algorithms based on message replication, as they have demonstrated to be an efficient forwarding strategy for OppNets characterized by the unpredictability of the nodes' movement. Nevertheless, the message replication drawback is congestion [70]. Precisely, to mitigate the potential congestion caused by replication, this proposal considers approximating how congested the network is so that the forwarding algorithm adjusts the replication limit of the newly created messages and the ones being forwarded.

In the following sections, for the use case of congestion control, we specialize: (1) the control metadata required by the control layer for the specific case of congestion control (Section 4.1); (2) the adaptation of the controller architecture for this particular use case (Section 4.2); (3) the context indicator prediction (Section 4.3); (4) the directive generation (Section 4.4); and the directive execution (Section 4.5). Finally, we summarize the buffer management techniques used by the control layer as a congestion control strategy (Section 4.6).

4.1 Control Metadata

In this section, we present the control metadata used by the control layer (introduced in Section 3.2) for the specific use case of using the control layer for congestion control. First, Section 4.1.1 presents the context measurements to represent congestion. Secondly, Section 4.1.2 indicates the metadata used by the proposed congestion control mechanism.

4.1.1 Context indicator measurement for congestion

As a network congestion measure, we use the node's buffer occupancy rate (o):

$$o = \frac{\sum_{i=1}^{n} sizeof(g_i)}{b} \tag{4.1}$$

where n is the number of buffered data messages; *sizeof* is the function that returns a message size in bytes; g_i is a buffered message, and b is the buffer capacity in bytes.

Each node aggregates its local congestion measurement and the received ones following the guidelines in Section 3.4. Next, this aggregation is disseminated through opportunistic contacts as presented in Section 3.3. This way, the control layer is fed with the context information regarding congestion. Section 3.6 shows how the controller infers a prediction of the buffer occupancy out of the received congestion readings for the network "segment" in its reach.

4.1.2 Control layer data message

This section describes the data message representation used by the control layer for the use case of congestion control.

Before representing a data message, we will introduce the concept of message replication limit, which is a part of the message representation. In the scope of forwarding algorithms based on message replication, one of the forwarding algorithm parameters is the replication limit of the message. The replication limit determines the total number of copies of the message allowed to be in the network. As seen in Section 2.2, multiple replication strategies exist. For this particular use case, we consider a forwarding algorithm that uses a binary replication scheme consisting in relaying a copy of the message to the contacted node and reducing by half the replication limit of both the node's message and the relayed copy of it.

The control layer encapsulates the data messages generated by the application layer in the tuple $g = (\mathfrak{a}, l, \varrho, \varphi)$. From this tuple, $\mathfrak{a}$ is the data message generated at the application layer. l is the replication limit of the message (in the case of a binary replication scheme, the message can be relayed $\lceil l/2 \rceil$ times). ϱ is the number of times this particularly message copy has been relayed. This field is incremented each time the message is relayed to the next hop. φ is the *alive* flag. This flag is set to false to indicate that this message is marked to be deleted in case buffer space is required. This message is not deleted straightforwardly as it could happen that the next contact could be the message destination. Also, it could occur that applying

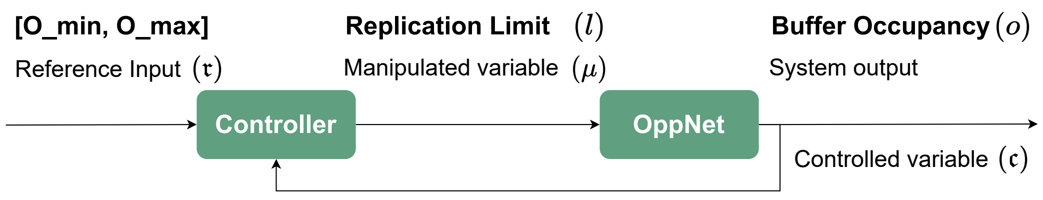

[O_min, O_max]
Reference Input $(\mathfrak{r})$
Replication Limit (l)
Manipulated variable (μ)
Buffer Occupancy (o)
System output
Controller
OppNet
Controlled variable $(\mathfrak{c})$

Fig. 4.1: Closed-loop control system for congestion control.

another directive would update this message's replication limit to a value equal to or higher than one, providing the message with more chances to be delivered. With this congestion measure, the messages with the field φ set to *false* are reactively removed in case of need, but also, in a proactively way, the message is given a chance to be carried along the network while there is no need for buffer space. This strategy requires the node not to consider the messages with the φ field set to *false* when calculating its buffer occupancy measurement.

4.2 Control layer tailored for congestion control

The control layer proposed in Section 3.5 has been adapted to use a congestion policy to efficiently limit the number of messages in the network to keep congestion at bay. This congestion policy dynamically determines the replication limit for newly created and buffered messages based on the network congestion perception of a controller.

Thereby, the generic closed-loop control system in Figure 3.3, for the use case of congestion control, is specialized in Figure 4.1 with the slight difference that the controller assumes calculating the difference between the reference input $(\mathfrak{r})$ and the controlled variable $(\mathfrak{c})$. As we can see in Figure 4.1, the controller feeds from the buffer occupancy (o) measurements received from contacted nodes. The controller, following Algorithm 2, uses the received buffer occupancy measurements (o) to calculate a buffer occupancy prediction (o_{t+n}) used to adjust the messages replication limit. Also, Figure 4.1 shows that the controller's reference input $(\mathfrak{r})$ is an optimal buffer occupancy congestion interval. This interval consists of a range of buffer occupancy rates, defined by a lower and upper bound, o_{min} and o_{max}, respectively. The buffer occupancy is considered optimal when the buffer is neither under-used nor close to its maximum capacity.

4.3 Network congestion prediction

This section shows how the context indicator prediction proposed in Section 3.6 is applied for forecasting congestion. For that matter, the controller uses the optimal congestion interval along with the calculated o_{t+n} to determine its prediction of the network's congestion. The network's congestion status falls into three states: *UNDER_USED*, *OPTIMAL*, and *CONGESTED*. The controller uses the following function to sentence which is its prediction of the network's congestion state (c_{t+n}):

$$c_{t+n}(o_{t+n}) = \begin{cases} \text{UNDER\_USED} & \text{if } o_{t+n} < o_{min} \\ \text{OPTIMAL} & \text{if } o_{min} \leq o_{t+n} < o_{max} \\ \text{CONGESTED} & \text{if } o_{t+n} \geq o_{max}. \end{cases} \tag{4.2}$$

Next section shows how the controller uses this congestion prediction to determine the message replication limit.

4.4 Directive generation for congestion management

Based on its congestion prediction (c_{t+n}), the controller calculates which should be the maximum number of copies of a message allowed to be in the network (l). For that matter, this proposal considers a proportional controller (P-controller) [37] using an additive increase and multiplicative decrease (AIMD) factor to adjust the manipulated variable (l). The considered AIMD factor uses k_1 as a multiplicative decrease factor and k_2 as an additive increase factor. Out of the predicted network congestion, the P-controller calculates the new replication limit l'.

$$l\text{'}(c_{t+n}) = \begin{cases} l & \text{if } c_{t+n} = \text{OPTIMAL} \\ l \cdot k_1 & \text{if } c_{t+n} = \text{CONGESTED} \\ l + k_2 & \text{if } c_{t+n} = \text{UNDER\_USED} \end{cases} \tag{4.3}$$

With this control function, if the predicted network's congestion state is *OPTIMAL*, it is not necessary neither increase nor decrease the replication limit of a message. When the predicted network's congestion state is *CONGESTED* or *UNDER_USED*, the controller decreases or increases, respectively, the message's replication limit.

Algorithm 4 Procedure to update the buffered messages with the l set by a directive.

```
      ▷ $\delta_l$, Received directive ($\delta_l = (Id_l, l')$)
      ▷ $l'$: The replication limit in the received directive ($\delta_l = (Id_l, l')$).
      ▷ $g$: A message ($g = (\mathfrak{a}, l, \varrho, \varphi)$).
      ▷ $B$: A buffer to store messages.
1: function APPLYDIRECTIVETOBUFFEREDMESSAGES($\delta_l$)
2:     for all $g \in B$ do
3:         $l'' = \frac{l'}{2^{\varrho}}$
4:         $g[l] \leftarrow l''$                    ▷ Setting the new rep. limit in the message's field $l$
5:         $g[\varphi] \leftarrow (l'' < 1)\ ?\ false : true$
6:     end for
7: end function
```

The controller encapsulates the calculated l' in a directive: $\delta_l = (Id_l, l')$, where Id_l is the identifier of the message replication limit setting and l' is the calculated replication limit. Following the flow-chart described in Figure 3.5, when the controller contacts a node, it forwards the former directive along with the calculation of its buffer occupancy.

4.5 Applying a directive

In Section 3.8, we have seen that when a node gets a directive ($\delta_l = (Id_l, l)$) from a contacted controller or a node carrying it, the node applies δ_l (Figure 3.5, *step s8*). Applying a directive $\delta_l = (Id_l, l)$ entails two actions: (1) using the encapsulated replication limit l' when creating new data messages, and (2) updating all the buffered messages according to the new l'. The first action consists in when an application creates a message $\mathfrak{a}$, the node encapsulates it in a message represented as $g = (\mathfrak{a}, l, \varrho, \varphi)$, specified in Section 4.1.2, where the field l is set to the new replication limit l' from δ_l. The second action consists in updating the message's field l of the buffered messages, considering that some are already copies.

Indeed, it would not be accurate to update the buffered messages with the new replication limit l' received through δ_l, as some of these messages are already copies. Therefore, the number of times this message has been relayed (message's ϱ field) is considered. This information is used to calculate the current replication limit of the buffered message, considering that: (1) the message was created with the replication limit set by the last received δ_l, and (2) $\lceil l/2 \rceil$ copies of the message are forwarded at each encounter:

$$l'' = \frac{l'}{2^{\varrho}} \tag{4.4}$$

where l'' is the remaining replication limit after relaying the message ϱ times.

Algorithm 4 shows how to modify the buffered messages' replication limit. For each one of the buffered messages (*line 2*), provided the replication limit it was assigned when created was l' (from the received directive $\delta_l = (Id_l, l')$) and taking into account the times the message has been relayed (ϱ), the remaining replication limit (l'') is calculated using (4.4) (*line 3*). If the calculated l'' were less than one, it would mean that by starting with the replication limit specified in δ_l, the message would not have any copies left to disseminate at this point, and, therefore, the message would not have reached the current node. If this is the case, the message's field φ is set to *false* (*line 5*), indicating that this message is marked to be deleted in case buffer resources are required.

4.6 Buffer Management

The control layer applies a congestion control mechanism based on buffer management. Besides the congestion policy limiting the number of message copies in the network, the control layer applies a hybrid drop policy.

The control layer foresees the network congestion (Section 4.3). Based on the predicted congestion, it limits the number of message copies (Section 4.4) and updates the copies left of the queued messages (Section 4.5). Proactively, if this update results in the message having no copies left, the message is marked to be deleted by setting the message's flag φ to false. Reactively, in case of buffer overflow, the messages marked to be deleted are dropped. The control layer applies basic drop policies if more buffer space is required. It first applies a *drop-oldest* policy based on the message's remaining TTL. Next, if necessary, it applies a *drop-head* policy removing the oldest ones.

Part III

Experimentation

5 Experimentation environment

THE CONTROLLER-DRIVEN OppNet, which uses a quota-based multi-copy forwarding with a dynamic message replication limit, is named Control configuration (Control). The Control configuration is compared with two *No-Control* multi-copy baseline forwarding algorithms: the Epidemic (EP) and a quota-based one (Static), both presented in Section 2.2. In Epidemic routing, nodes forward messages to every encountered node to achieve maximum network coverage. The Static routing sets a static upper bound of the number of message replicas in the network. It distributes half of these copies to each contact (provided the contact does not carry copies of the message yet) until the node has just one copy left, which will carry up to the destination. Both No-Control approaches are multi-copy baseline forwarding algorithms considered for benchmarking in Opportunistic Networking research [21].

This chapter describes the testbed setting up we will use to evaluate and compare the performance of the Control and No-Control configurations. The experimentation methodology follows the guidelines pointed out by Dede *et al.* in [22], and Kuppusamy *et al.* in [44]: (1) appropriate mobility models to favour different congestion degree situations have been designed; (2) our proposal is compared with benchmark multi-copy forwarding algorithms; (3) several performance metrics, listed in Section 5.1, are evaluated over the Control and No-Control configurations; moreover, the network performance is also evaluated for the Control configuration for different values of its configuration settings listed in Section 5.4.2; (4) the experimentation setup is detailed and well documented to be reproduced for benchmark purposes; and (5) the simulator provides the link model and the physical aspects are not considered.

5.1 Performance metrics

We use the standard metrics to measure the performance of the Control and No-Control configurations [52]:

- **Delivery Ratio (σ):** It measures the ratio of created messages that are delivered to the final destination:
$$\sigma = \frac{\#g_d}{\#g_c} \tag{5.1}$$
where $\#g_d$ is the number of delivered messages, and $\#g_c$ is the number of created messages.

- **Latency average ($\overline{\lambda}$):** It is the average time it takes for the created messages to get delivered to their final destination:
$$\overline{\lambda} = \frac{\sum_{i=1}^{w} \lambda_i}{w} \tag{5.2}$$
where *w* is the number of messages delivered to the destination, and λ_i is the elapsed time from the message creation to its delivery.

- **Overhead ratio (θ):** Measures the average of the message copies needed to deliver the message to its final destination:
$$\theta = \frac{\#g_r - \#g_d}{\#g_d} \tag{5.3}$$
where $\#g_r$ is the number of relayed copies, and $\#g_d$ is the number of delivered messages.

5.2 Scenarios

We use four scenarios which use different mobility patterns representing different network conditions. These scenarios are classified into two groups: 1) the ones based on real-world mobility traces, available at the Crawdad database [18], and 2) the synthetic ones generated by a Random Waypoint model (RWP) in a grid with reflective barriers where the nodes move at a configured speed for a configured distance. The nodes keep changing the direction each time they cover that distance. The scenarios based on real mobility traces are very convenient for evaluating this proposal under real network conditions. In contrast, the synthetic ones help us to recreate particular network conditions as emergencies, not covered yet with real mobility traces samples. The considered scenarios are:

- **Taxis:** Tracks 304 Yellow Cab taxis in the San Francisco Bay area for one week. The traces are available at [51].

- **Info5:** Tracks the movement activity of 41 students attending the Infocom conference in 2005 during three days [2].

- **Campus:** Is a synthetic map-based scenario that simulates the mobility activity of 80 students at Autonomous University of Barcelona campus, covering an area of 4.5 Km x 3.4 Km with defined points of interest (POI) corresponding to eight faculties and the railway station. The students walk throughout the Campus arriving/leaving their faculty and the railway station with a certain probability.

- **Emergency:** It is a synthetic RWP-based scenario with 100 nodes randomly walking in an area of one square kilometre. In this scenario, the pattern of message generation changes abruptly in the second half of the simulation, aiming to roughly simulate the network condition under the disrupting events of an emergency.

Each scenario has different nodes densities. The sparsity nature of a scenario entails fewer contact opportunities amongst the nodes, whereas dense ones are prone to more contacts between nodes. The Taxis is a sparse scenario, and the Campus is a sparse one with POI promoting a temporary high density of nodes, whereas Info5 and Emergency are both dense scenarios.

5.3 Message generation distribution

Depending on the scenario, for the data message generation, we will be using either a Constant Bit Rate (CBR) distribution or an Inverted Smoothed Top Hat distribution (ISTH) [11]. Next, we formalize the ISTH for the specific case of a 24-hour working day.

5.3.1 Inverted Smoothed Top Hat distribution (ISTH)

We aim to mimic the network traffic in a working day with the ISTH distribution (Figure 5.1). The Flat Region (FR) of the ISTH represents the working hours where the network traffic is the heaviest, i.e. messages will be generated at a higher rate. The Transient Regions (TR) are shaped in two ways: 1) to logarithmically increase the rate of message generation up to reach the peak rate of the FR, and 2)

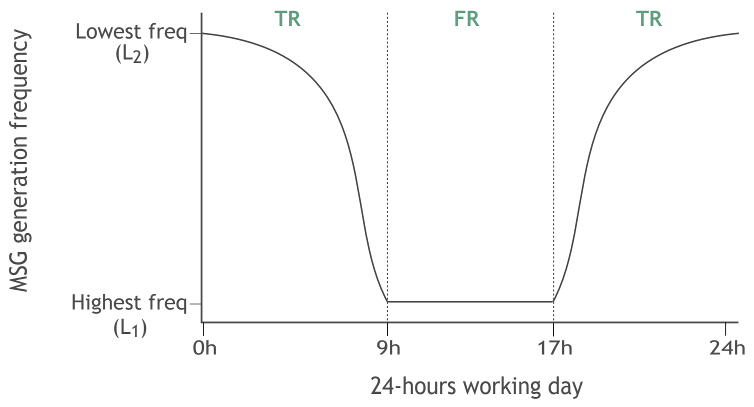


Fig. 5.1: Inverted Smoothed Top Hat function for a 24-hour working day.

to logarithmically decrease the rate of the message creation from the peak rate in the FR to a shallow rate when approaching the 24th hour of the day.

The ISTH function is built as a composition of a descendant and an ascendant logistics functions and a linear function. Both logistic functions determine the message generation frequency for a time:

- **Descendant logistic function** (exponential growth rate$(k) > 0$):

$$f(x) = \frac{L_2}{1 + ae^{k(x-x_0)}} \tag{5.4}$$

- **Ascendant logistic function** ($k < 0$):

$$g(x) = \frac{L_2}{1 + ae^{-k(x-x_0)}}. \tag{5.5}$$

These functions are bounded by two limits: L_2 and L_1. L_2 corresponds to the lowest message generation rate (highest value) used to generate very low traffic. L_1 corresponds to the highest message generation rate (lowest value) used to generate the highest network traffic. $x - x_0$ is the flexible horizontal translation. No horizontal translation is considered: $x_0 = 0$. k is the exponential growth rate.

From time 00:00 a.m. to 09:00 a.m., the descendant function defined in (5.4) needs to descend from L_2 down to L_1, i.e. $f(0) = L_2$ and $f(9) = L_1$. Both L_2 and L_1 limits are specified through the control configuration settings depending on the scenario. Hence, from (5.4) the only unknown variable (a) is isolated:

$$a = \frac{(L_2 - L_1)}{L_1 e^{k9}}\ .$$

The ascendant logistic function in (5.5) follows the same process to ascend from L_1, at the end of the working hours of the day (17:00h), up to L_2 at (00:00h): $g(17) = L_1$ and $g(0) = L_2$.

Finally, to build the ISTH function it is necessary to combine the descendant and ascendant logistic functions with a flat linear function that covers the eight working hours of the day (from 09:00 a.m. to 17:00 p.m.). During this window time, messages are generated at L_1 rate:

$$h(x) = \begin{cases} 0 \leq x < 9 : & f(x) \\ 9 \leq x < 17 : & L1 \\ 17 \leq x < 24 : & g(x) \end{cases} . \tag{5.6}$$

5.3.2 Scenario's message generation distribution

Table 5.1 summarizes the message creation distribution for the different scenarios. The Taxis scenario uses an ISTH distribution that replicates two working days, where the messages are generated each 10 to 60 seconds uniformly distributed for each working day during eight peak hours. With all the scenario specifics, it is considered a low to medium congested scenario.

The Info5 scenario uses the aforesaid message generation distribution. Given the message generation distribution and the fact that the nodes congregate around the events of the congress, it is considered a medium to high congested scenario.

The Campus scenario uses an ISTH distribution resembling a workday where, during the peak hours, messages are created every 10 seconds. Hence, this scenario is considered a high congested one.

For the Emergency scenario, during the first eight hours, messages are generated at the high rate of each 10 seconds using a CBR distribution followed by eight hours of low rate message generation (every 80 seconds). Therefore, Emergency is a variable congested scenario.

Tab. 5.1: Message generation distribution per scenario.

Settings	Taxis	Info5	Campus	Emergency
Simulation time	69h	70h	34h	23h
Message distribution	ISTH	ISTH	ISTH	CBR
Message generation frequency	FR:[10 - 60]s; TR:1800s	FR:[10 - 60]s; TR:900s	FR:10s; TR:1800s	0-8h:10s; 8h-16h:80s
Congestion level	Low-Medium	Medium-High	High	Medium-High

5.4 Environment setup

For the experimentation, it has been used the Opportunistic Network Environment (ONE) simulator [40] designed specifically to simulate OppNets. Recent works show that it is the most used simulator for OppNets [44]. The ONE has proven easy to configure and provides an extensive set of mobility, traffic models, and propagation protocols [22]. The control layer has been developed on top of the simulator's network layer and is available through a public repository[1].

The simulations over the synthetic scenarios use a traces file with node contacts generated with the built-in RWP model of the simulator to preserve the same scenario over the different simulation rounds. Next, we will describe the configuration settings common to all the scenarios and the specific settings by scenario.

5.4.1 Common configuration settings

Table 5.2 lists the common simulation configuration settings. For all the scenarios, the nodes are configured with a Wi-Fi interface with a transmission speed of 100 Mbps and a transmission range of 60 meters as an approximation of the Wi-Fi 5 (802.11ac) standard.

In each simulation cycle, any random node in the network creates a message to a randomly selected node to approximate a real-world communication model. For the simulations, it is considered that when two nodes are in range, they have enough time to exchange the control protocol data, the messages to be delivered to the contacted node, and the messages to be relayed.

[1]https://github.com/MCarmen/the-one/tree/control

Tab. 5.2: Summary of the Common simulation settings for all the scenarios.

Setting	**Value**	**Setting**	**Value**
Network interface	Wi-Fi	Battery	none
Transmission speed	100Mbps	Type of nodes	pedestrians
Transmission range	60 meters	User behaviour	none
Interference	none	Application	Single destination
Power consumption	none		

Tab. 5.3: Summary of the Specific simulation settings per scenario.

Scenario setting	**Taxis**	**Info5**	**Campus**	**Emergency**
Simulation Time	69h	70h	34h	23h
Simulation Area	San Francisco Bay	hotel	4.5km x 3.4km	1km^2
Mobility model	Contact Traces	Contact Traces	Map-Based + POI	RWP
# Nodes	304	41	80	100
# Contacts	69412	22459	168442	38013
TTL (sec)	10000	10000	10000	4000
Buffer size	10M	10M	10M	10M
Message generation distribution	ISTH	ISTH	ISTH	CBR
Message generation frequency (s)	FR:[10 - 60]s; TR:1800s	FR:[10 - 60]s; TR:900s	FR:10s; TR:1800s	0-8h:10s; 8h-end:80s
Message size	[10 - 500]k	[10 - 500]k	[10 - 500]k	0-8h:500k; 8h-end:10k
Walk speed	N/A	N/A	0.5m/s	[0.5 - 1]m/s
Control settings	**Taxis**	**Info5**	**Campus**	**Emergency**
Nrof controllers	10	2	4	20
$[o_{min} - o_{max}]$	[0.6-0.7]%	[0.3 - 0.5]%	[0.6-0.7]%	[0.7 - 0.9]%
Additive increase (k_2)	1	1	1	1
Multiplicative decrease (k_1)	0.25	0.25	0.25	0.25
LR nrof inputs ($\check{z}$)	6	10	6	6
LR aggregation interval ($\hat{t}$)	60s	30s	300s	120s
Prediction time factor (ϕ)	2	5	2	2
Directive generation frequency	900s	900s	900s	900s
Reduction factor for a Decay (r)	0.103 (d of 5% at 1800s)	0.3 (d of 1% at 300s)	0.058 (d of 5% at 300s)	0.3 (d of 1% at 300s)
Decay Threshold	0.1	0.1	0.1	0.1
nrofAggregations weight (α)	0.2	0.2	0.2	0.2

5.4.2 Scenario and control configuration settings

The nodes' buffer size, the messages' TTL, size and generation frequency, which directly affect the network congestion, also vary for each scenario to create different congestion conditions. Table 5.3's first half lists the values for the above scenarios' settings. In this table, the intervals specifying the value for the settings: message generation frequency, message size, and walk speed, denote a uniform distribution between the two interval limits. Notice that for all the scenarios, the nodes' buffer size is set to 10M to favour congested situations, mainly when messages are generated at a high rate.

Also, the control layer can be customised through the settings listed in Table 5.3's second half. Following, we describe the customisable control settings:

- **Number of Controllers:** Indicates the number of nodes that will act as controllers in the network.

- **Optimal congestion interval** $(o_{min} - o_{max})$:] As presented in Section 4.2, this setting specifies the optimal range of the node's buffer occupancy. If the buffer's occupancy of a node fits in this range indicates that the buffer is neither *under-used* nor reaching its full capacity. This interval along with the buffer occupancy's prediction (o_{t+n}) are used in (4.3) to determine a prediction of the network's congestion state.

- **Additive increase (k_2), Multiplicative decrease (k_1) (AIMD):** Denote the factor to be added to (k_2) and the factor to multiply by (k_1) the current replication limit (l) to update l's value based on the network congestion status through (4.3).

- **Aggregation interval ($\hat{t}$):** Time while the controller gathers congestion measurements (see Section 3.6). After this time, the gathered measurements are aggregated, and the aggregated value is stored in the list $\breve{M}$ (see Algorithm 2). The $\breve{M}$ entries are used by the linear regression (LR) function in (3.4) to predict the network congestion.

- **LR nrof inputs ($\breve{z}$):** Max size of the congestion readings aggregation list $\breve{M}$. This list is used as an input for the congestion prediction function. $\breve{M}$ works as a sliding list of size $\breve{z}$ to consider a recent history of readings for the prediction.

- **Prediction time factor (ϕ):** It is the multiplicative factor applied over the aggregation interval setting ($\hat{t}$) to determine the time (t_{n+t}) for a congestion prediction (Section 3.6, Algorithm 2, *line 10*). ϕ is calculated by the equation:

$$t_{t+n} = t + \hat{t}\phi \tag{5.7}$$

  where t is the current time, and $\hat{t}$ is the time interval for aggregating congestion readings.

- **Directive generation frequency:** Determines the periodicity of the automatic directive generation (see Section 3.7) triggered in case the controller does not receive any congestion reading for this period of time.

- **Reduction factor for a specific decay (r):** It is the reduction factor to apply to get a certain decay. Used in (3.1) (Section 3.4).

- **Decay threshold:** When a controller receives a congestion measurement with a decay lower than *Decay threshold*, it is discarded, and hence, it is not used to estimate the congestion.

- **Number of Aggregations weight (α):** Weight factor applied over the number of aggregations a congestion measurement is built on. Used in (3.3) (Section 3.4).

Results 6

THIS SECTION shows and evaluates our proposal (Control) and the No-Control configurations, introduced in Chapter 5, for different scenarios, in terms of (1) the buffer occupancy, (2) the performance metrics listed in Section 5.1, and (3) the delivery ratio for different values of the controller settings listed in Section 5.4.2. Before delving into the comparison between Control and No-Control configurations, for the Control one, we analyse the replication limit (l) it tends to. The Control configuration will be compared with the Static one with the same l the Control tends to along this section (Static*).

For the No-Control Static routing policy, simulations have been run with different replication limits to show the tendency of the metric's value. We have narrowed the replication limit to the scenario's number of nodes. We consider that having as many copies of the message as nodes are in the network is an approximation of epidemically flooding the network.

Finally, we provide all the obtained results, the ONE configuration files for all the scenarios, the script files to run the simulations, and the data traces, for the reproducibility of these results[1].

6.1 Replication limit tendency for the Control configuration

Figure 6.1 depicts the tendency of the calculated Control replication limit along the simulation. We observe that the calculated l tendency is inverse to the filling up of the buffer for each scenario (Figure 6.2), i.e. the calculated l values are initially high, corresponding to the period where buffers are still not overwhelmed, but tend to decrease along the simulation depending on the congestion readings. Specifically, the l for Taxis tends to 8, Info5 to 2, Campus to 2, and Emergency to 4.

The controller's goal is keeping a high l aiming for a higher delivery ratio and lower latency while the buffers are not stressed, and lowering l to prevent this stress from happening. Following this behaviour, for the less congested scenario (Taxis), where the controller can keep a higher l value, we observe that the controller decreases the

[1]`https://deic.uab.cat/~mcdetoro/controller-driven_OppNet_results.zip`

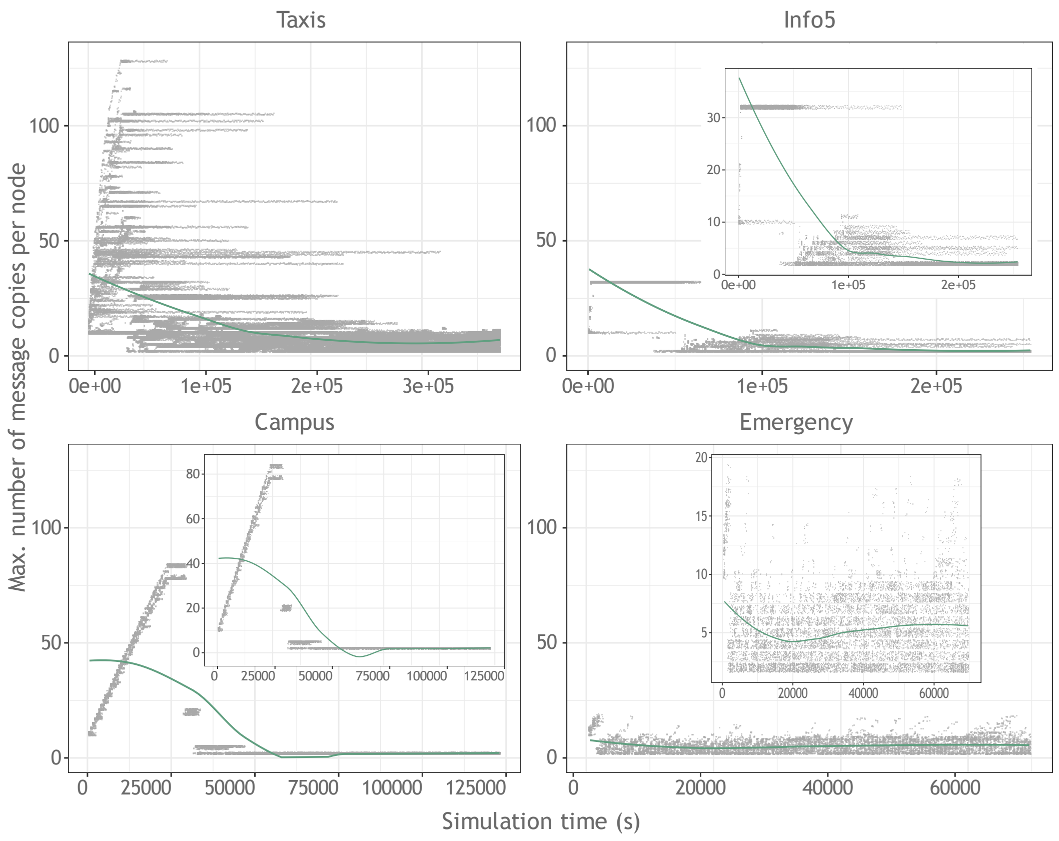


Fig. 6.1: For the Control configuration, progression of the replication limit used by each node at each time unit when a new message is created. In blue, we show the tendency of the replication limit over time (for Taxis is 8, for Info5 is 2, for Campus is 2, and for Emergency is 4). For the scenarios Info5, Campus and Emergency, a zoomed plot with a different scale has been embedded in the main plot.

l slowly. Conversely, for higher congested scenarios (Info5 and Campus), where high l values would rapidly overwhelm the buffers, we see that the controller decreases l much faster. Specifically, we observe that the controller reduces the l in the Info5 scenario faster than in the Campus one. Indeed, for the Info5 scenario for all the No-Control configurations, the buffer is overwhelmed similarly, whereas, for the Campus one, it depends on the No-Control replication limit configuration. More precisely, we can observe the controller's capacity to adjust the l in the Emergency scenario, where in the first half part of the simulation, the more congested one, the controller decreases the l and increases it in the second half of the simulation, the less congested one.

6.2 Buffer occupancy evaluation

Figure 6.2 shows that, for all the scenarios and for all the configurations except for the Control one, the buffer fills in a logarithmic way up to the buffer's total capacity. For the Emergency scenario, we can see an inflexion point that derives to a lower

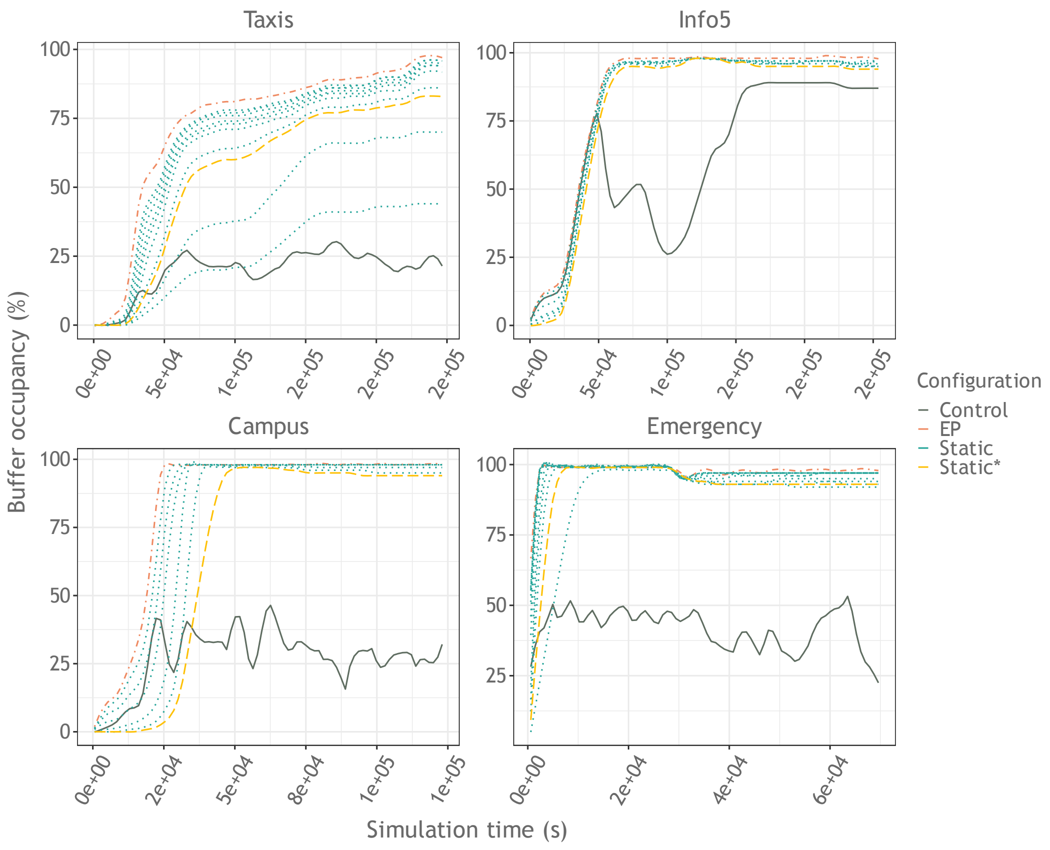


Fig. 6.2: Percentage of the buffer occupancy for the different policies, for each scenario, along the simulation time. In the legend, the configuration named Static* corresponds to the Static one with the replication limit the Control tends to.

buffer occupancy at the simulation time when the message generation distribution changes from high frequency to low frequency.

Without any replication limit (l), the EP policy fills up the buffer faster than the other policies. The Static policy's static l determines the speed at which the buffer fills up. For the Control configuration, as the control system regulates the replication limit based on congestion information readings from the nodes, the buffer occupancy fluctuates based on the effects of the new replication limit values.

Overall, the buffer occupancy is lower with the Control configuration. For all the scenarios, after a transient period, the buffer utilization by the Control policy tends toward 21% for the Taxis scenario, 87% for the Info5 scenario, 32% for the Campus scenario and 22% for the Emergency scenario. Nevertheless, this remarkable difference between the buffer utilization by the Control configuration and the No-Control ones is due to how we measure the buffer occupancy for the Control configuration. For Control, the buffer occupancy measure does not count the messages that are still buffered but have the flag φ to *false* so that if buffer space is required, those will be the first messages to be discarded.

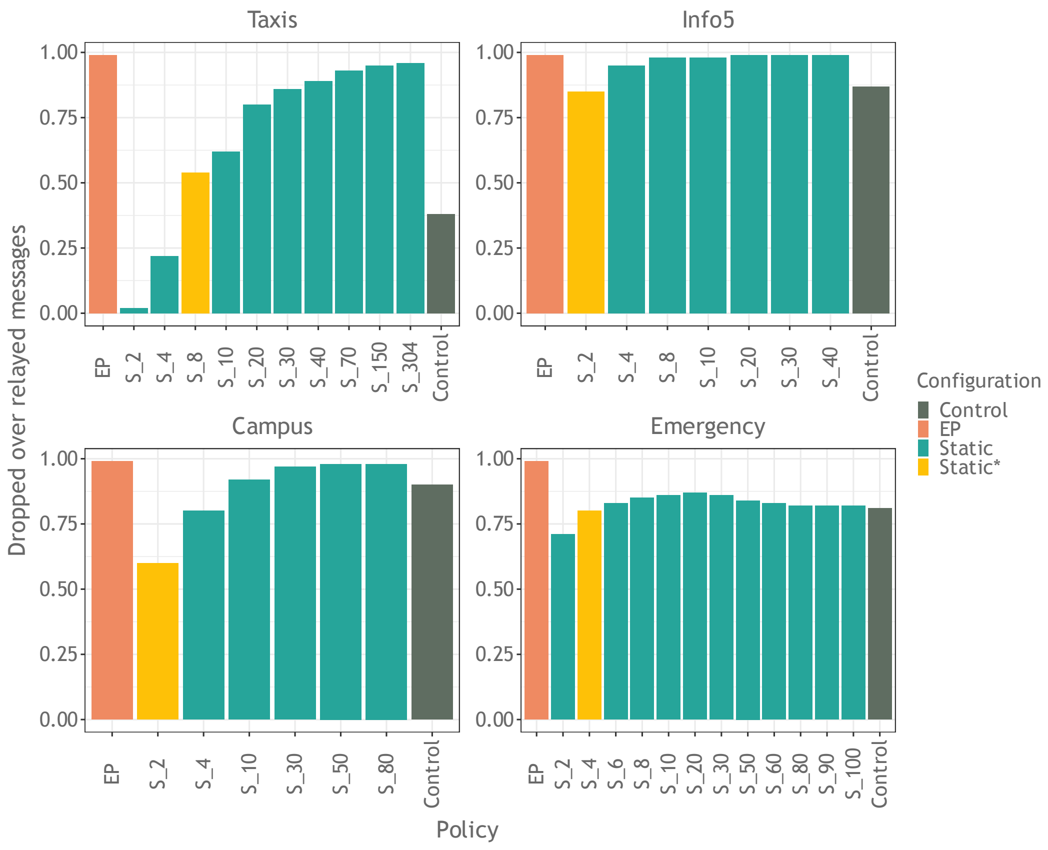


Fig. 6.3: Percentage of dropped over relayed messages for the different policies. The Static suffix denotes the replication limit. In the legend, the configuration named Static* corresponds to the Static one with the replication limit the Control tends to.

Figure 6.3 shows the percentage of dropped over relayed messages. As expected, for the No-Control configurations, the faster the buffer fills up (Figure 6.2), the more messages are dropped. Nevertheless, the Control configuration does not have a lesser drop rate despite its lesser buffer occupation. That is, precisely, because of the aforementioned detail of how Control measures the occupancy of the buffer so that, despite a message with the flag φ set to *false* will not be counted when calculating the buffer occupancy, when the buffer needs the space it will be dropped. Moreover, as expected, for the more congested scenarios (Info5, Campus and Emergency), the Control drop rate is slightly higher than the Static* configuration one. This difference is caused by the changes in the replication limit adjusted by the controller.

More precisely, as expected, for the less congested scenario (Taxis), for all configurations, dropped messages rate correspond to the buffer occupancy, as the buffers are not stressed along the simulation. Also, the relation between the buffer occupancy and the dropped messages rate remains for the Info5 scenario, a medium-high congested one, where buffers are more stressed. Nevertheless, for the Campus scenario, a highly congested one, and for the Emergency scenario, which has a high congestion phase, for the Control configuration, the high buffer occupancy ends up dropping the

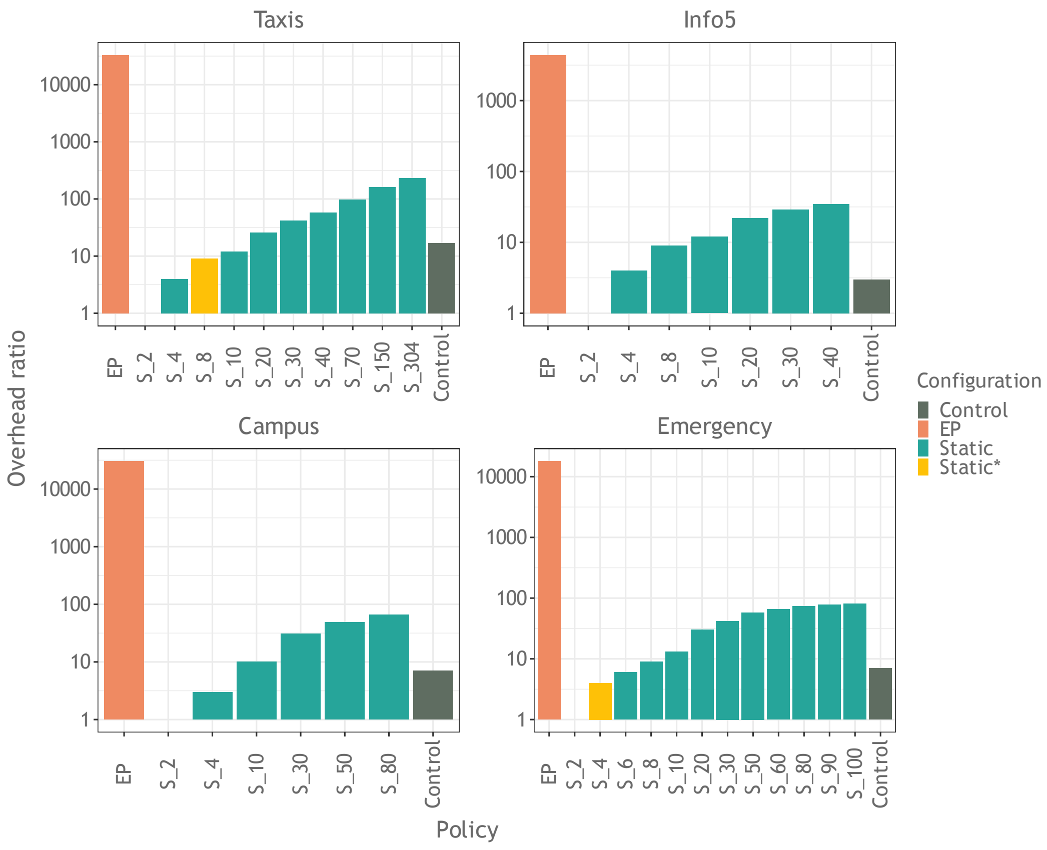


Fig. 6.4: The overhead percentage for the different forwarding policies in a logarithmic scale per scenario.

buffered messages with the flag φ set to *false*, and, therefore, the dropped messages rate is at par with the Static* configuration one, with slight differences caused by the changes on the replication limit adjusted by the controller.

6.3 Performance evaluation

This section presents and evaluates the results for the performance metrics listed in Section 5.1: overhead ratio, delivery ratio and latency average, for the Control and No-Control configurations for different scenarios.

6.3.1 Overhead ratio

Figure 6.4 shows that the overhead derived from the relay of the message copies depends on the l. EP's overhead surpasses Static and Control between two to three magnitude orders, whilst Statics's overhead increases for higher ls. The Control's overhead ratio is similar to the Static* one. The slight difference between the two configurations is due to the on-the-fly Control's recalculations of l.

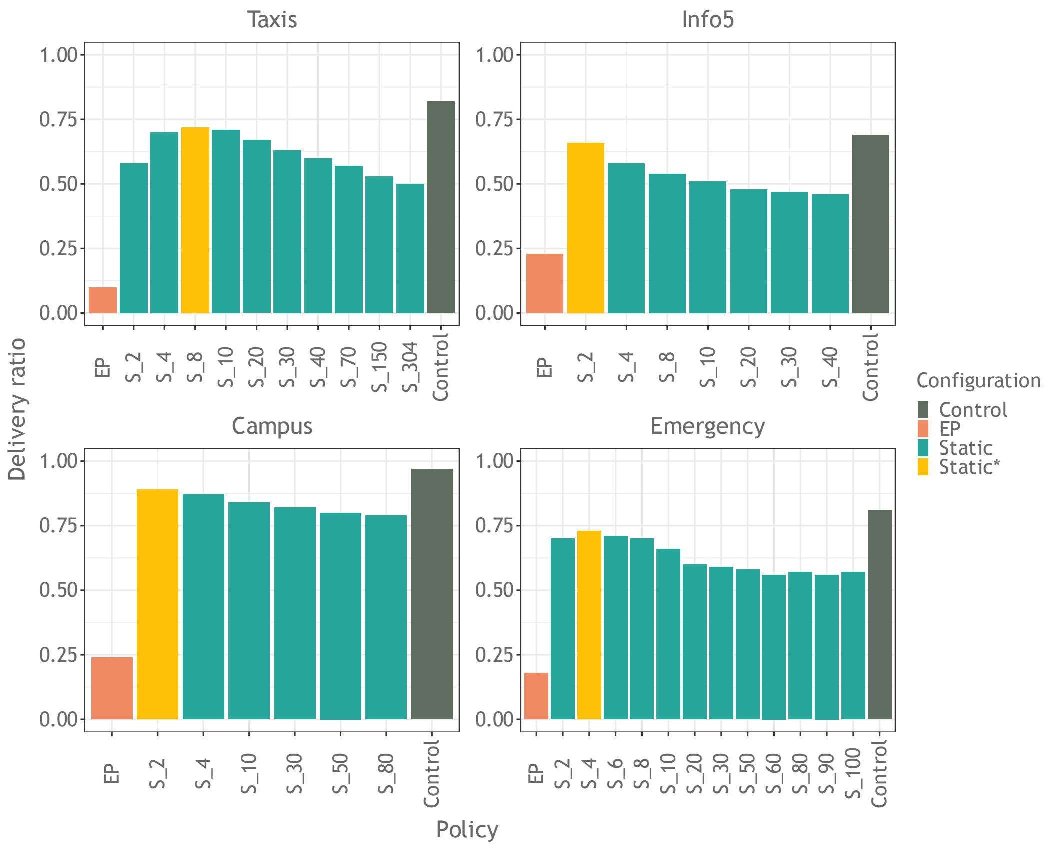


Fig. 6.5: Delivery ratio percentage for the different forwarding policies per scenario.

6.3.2 Delivery ratio

The dropped messages and the overhead directly affect the delivery ratio performance. As we can see in Figure 6.5, a high replication limit takes its toll on the delivery ratio performance.

With the highest replication limit, the EP policy floods the network, triggering a significant amount of drops and, therefore, getting the worst delivery ratio. The behaviour above also applies to the Static policy. The higher the l is, the poorer the delivery ratio we obtain.

Indeed, Info5 and Campus scenarios, the more congested ones, obtain the highest delivery ratio with a Static policy with a low l, 2 in both cases. As the l increases, the delivery ratio performance decreases. However, for the Emergency scenario, which combines a high message generation frequency with a low one, and for the Taxis scenario, with a low-medium congestion level, the delivery ratio is ascendant for the values of l up to the inflexion point of the Static's l with the best delivery ratio. This behaviour is coherent with the fact that high values of l strike the congested scenarios. In contrast, the low-medium congested scenarios admit higher values of l, favouring a higher delivery ratio.

Overall, the Control policy gets the best delivery ratio for all the scenarios. We obtain the best increase ratio in the scenarios with low-medium congestion levels. We specifically get a 14% and an 11% increment in the delivery ratio for Taxis and Emergency, respectively, over the Static configuration with the l performing the best. As we have previously seen, these scenarios admit higher l values, bringing on a higher delivery ratio. The ability of the Control policy to dynamically adapt the l provides the optimal l value depending on the current congestion situation. This flexibility makes the Control policy to outstand in low-medium congestion scenarios over the other configurations. On the other hand, for highly congested scenarios, where the best option is to keep a very low l close to direct delivery, the Control policy comes out also with a low l. It also benefits from the dynamism and slightly outperforms the Static policy with the l that performs the best, by 4% and 9% for Info5 and Campus scenarios, respectively.

6.3.3 Latency average

Figure 6.6 shows that, despite the crushing effects of the message flooding strategy over the buffer occupancy, delivery ratio and overhead, when it comes to the latency, message flooding benefits the arrival of the messages to their destination and, therefore, it obtains a good performance. Indeed, as pointed out by Krifa *et al.* in [43], a flooding-based replication benefits the latency of the messages at the expense of the delivery ratio in case of congestion. That is so because dropped messages will not get to the destination, undermining the delivery ratio. In contrast, a high message dissemination will favour that non dropped messages will have more chances to get to their destination upon an opportunistic contact resulting in a lower latency for those messages. With this premise, we can see that the configurations that fill up the buffer faster (Figure 6.2) as EP and the Static ones with the highest replication limit perform worst in terms of delivery ratio (Figure 6.5) but better in terms of latency (Figure 6.6).

As for the Control configuration, the premise above applies. More specifically, the Control configuration for the less congested scenario (Taxis) achieves a better performance than EP and Static* due to its low buffer occupancy, backed by a low dropped messages rate. Nevertheless, the configurations with a Static high replication limit leverage from high replication to have lower latency than the Control one. For the medium-to-high congested scenario Info5, where the buffer occupancy and dropped messages ratio are close to the No-Control configurations, a high replication benefits a lower latency. Also, for the most congested scenarios

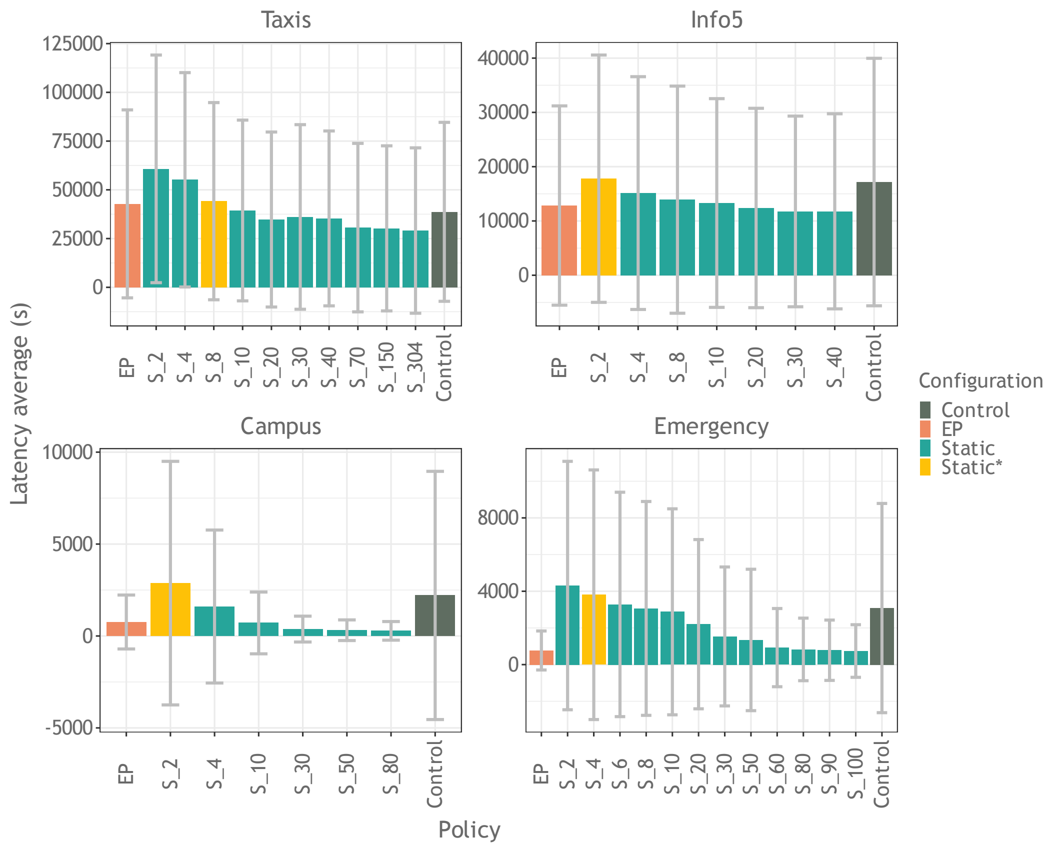


Fig. 6.6: Latency average for the different forwarding policies per scenario.

(Campus and Emergency), where the Control configuration ends up with a high dropped messages rate, the highest replication configurations obtain a better latency performance. Finally, Figure 6.6 includes the standard deviation of the latencies, showing up for all scenarios and configurations that there is a high variance between the messages' latencies.

6.4 Evaluation of the control settings impact on the delivery ratio

The control layer is configurable through the settings listed in Section 5.4.2. We have run simulations over the four selected representative scenarios to analyse the impact of the Control configuration settings on the delivery ratio over diverse scenarios and to find a general configuration that fits all of them. The following nine sections present our analysis.

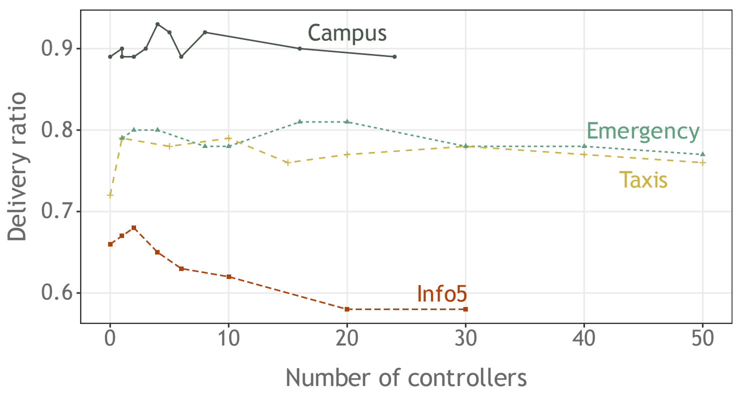


Fig. 6.7: Delivery ratio by number of controllers.

6.4.1 Number of controllers

Figure 6.7 shows the delivery ratio depending on the number of controllers used per scenario. In this plot, the max number of considered controllers is 50. Not all the scenarios have been simulated for all the considered number of controllers as, for example, the Info5 scenario has just 41 nodes.

Certainly, as pointed out in section 3.5, the number of controllers needed to orchestrate the OppNet depends on the nature of the network. Besides, the characteristics of the four simulated scenarios, including the number of nodes, are very different. Therefore, for each scenario, we did stop simulating as soon as the simulation results showed a clear descendant slope for an increasing number of controllers. Specifically, Figure 6.7 shows that for all the scenarios, a small number of controllers perform better in terms of delivery ratio. For the most connected scenarios, Campus and Info5, the best performance is achieved with just four and two controllers, respectively.

As the controller receives the congestion readings from the nearby nodes, it has an overview of the congestion of a part of the network, its nearest part. We can elaborate on this idea by considering that the network is kind of “segmented" by the number of controllers used. Each network “segment" consists of the number of nodes that can be reached by a controller directly or through short-time relays.

Having said that, in a highly connected network, using a high number of controllers results in overlapping the different network segments, as each controller can reach several of these segments. This overlapping effect results in the nodes receiving directives from different controllers. Of course, a directive emitted from a controller from a segment the node does not belong to has congestion information that is not entirely accurate for the node. This overlapping effect is why using many controllers decreases the network performance.

On the other hand, for the more sparse scenario (Taxis) and the scenario with an abrupt change in the communication conditions (Emergency), the implicit segmentation derived by the different controllers gets disjoint segments. Under these circumstances, having a higher number of controllers (10 for Taxis and 20 for Emergency) helps cover a broader network range, translating to better performance.

6.4.2 Optimal congestion interval $(o_{min} - o_{max})$

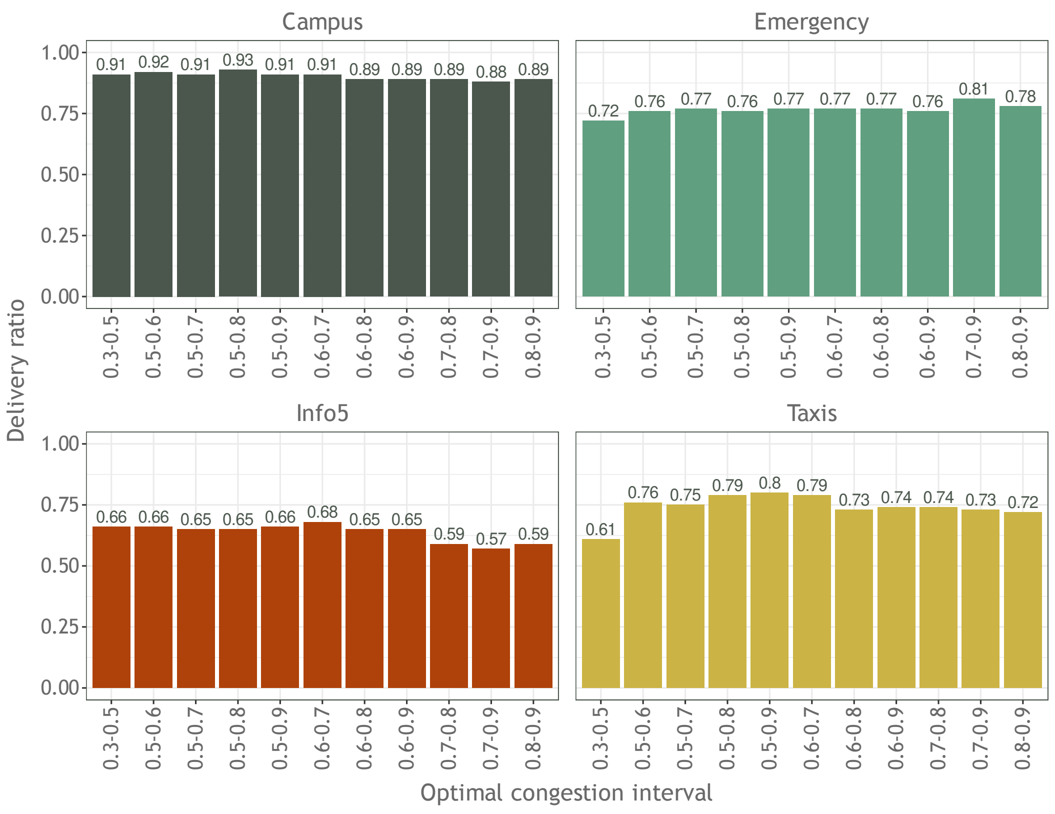


Fig. 6.8: Delivery ratio depending on the optimal congestion interval per scenario.

As presented in Section 4.4, the current replication limit is not modified when the congestion calculated by a controller falls in the optimal congestion interval.

Figure 6.8 shows that for the most congested scenario (Campus), there is a similar delivery ratio for all the optimal congestion intervals. The variance between the performance results for the different intervals is just 0.0002. Nevertheless, we get the best result for the interval [0.5-0.8].

These homogeneous results infer that in a congested scenario, if the congestion predictions fit in the configured optimal congestion interval setting for the current replication limit, the best strategy is to keep the current replication limit steady.

The medium-to-high congested scenario (Info5) shows a little more variance in the delivery ratio than the previous scenario (0.001). This variance can be appreciated mainly when the interval upper bound (o_{max}) is 0.8 and 0.9. Therefore, when the optimal congestion interval upper bound is set close to the maximum buffer occupation, the replication limit set by the controller is too high. Thus, a more conservative optimal congestion range gives better results which, in this case, is [0.6-0.7].

For the Taxis scenario, the most sparse one with fewer contact opportunities, it can be seen that the most conservative interval configuration, [0.3-0.5], performs by far the worst (24% less than the best interval). In contrast, the intervals with a high o_{max} have a good performance. This is because using a high level of message replication promotes a higher message delivery in a sparse scenario.

Finally, the Emergency scenario behaves similarly to the Taxis one. The performance of the most conservative interval is the worst (11% less than the best range). Nevertheless, the performance variance by the different intervals is 0.0004, whereas for the Taxis case is a bit higher: 0.002. For this unpredictable scenario, like for the Taxis one, the best strategy is to use an interval with a high o_{max} to keep a high replication limit and, therefore, to have more chances of message delivery.

Altogether, we have seen that in a congested scenario, the key is to keep a steady replication limit if it keeps the congestion within the configured optimal congestion interval. It is better to set a conservative optimal replication limit for a medium-to-high congested scenario to avoid high replication that could yield in a future congested scenario. On the contrary, for low-congested scenarios, setting an optimal congestion interval with a high upper bound leads to a higher replication limit favouring message replication, which increases the delivery ratio.

6.4.3 Additive Increase (k_2); Multiplicative Decrease (k_1)

The left plot in Figure 6.9 shows that for the additive increase factor used in (4.3), the value that gives the best performance in all the scenarios is, undoubtedly, 1. From this result, it can be stated that it is essential that the replication limit grows slowly to mitigate as much as possible the adverse effects of high replication. As for the multiplicative decrease factor (MD), for all the scenarios except for the Campus one, the best option is to reduce 75% the replication limit as a drastic measure to decrease the congestion caused by replication.

For all the scenarios except for the Taxis one, the delivery ratio keeps decreasing for higher MD values (implying less replication reduction). For the particular case

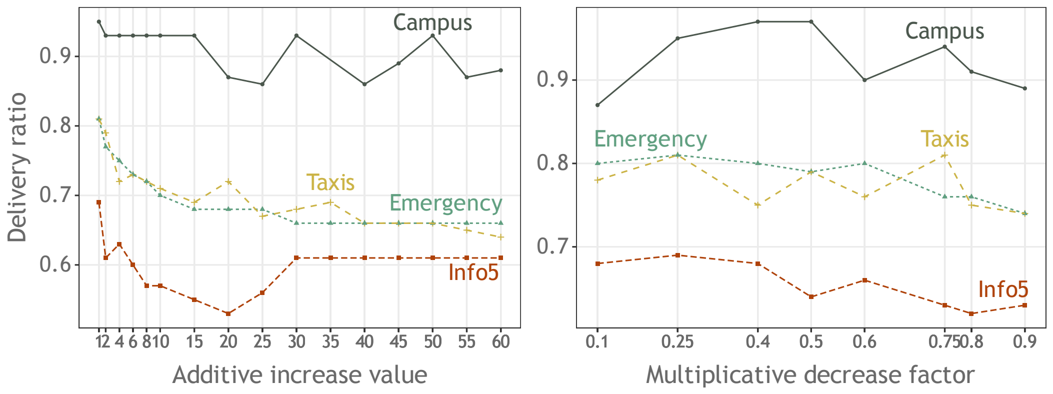


Fig. 6.9: Delivery ratio vs. different values for the AIMD control function.

of the Taxis scenario, its sparse condition results in more fluctuating delivery ratios depending on the MD factor, but none overcome the ceiling achieved with the 75% of reduction.

Going back to analysing the results, for the Campus scenario, which is highly connected and the most congested, we can see that we obtain the best result by applying a reduction factor of 50%, keeping a higher replication level. These results are consistent with those in the previous section, where it was stated that once the congestion status fitted in the congestion range thresholds, the best strategy was to keep the replication limit within the range. Precisely, reducing the replication limit to half is a good way to keep the congestion steady within the optimal congestion interval.

6.4.4 Aggregation interval $(\hat{t})$

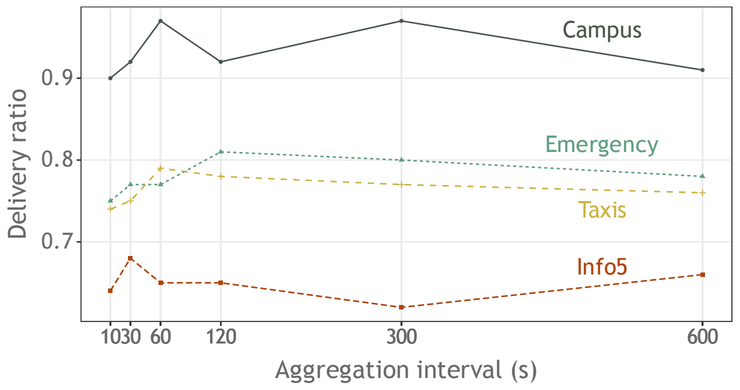


Fig. 6.10: Delivery ratio vs. the time interval while the controller is gathering congestion readings.

In Section 3.6 was presented how the controller foresaw the network congestion ahead through (3.4) based on a list of aggregated congestion readings $(\breve{M})$. The aggregation interval setting ($\hat{t}$) determines the time while gathering the readings to generate an $\breve{M}$'s entry. Indirectly, it determines the rate at which the controller emits a directive.

Figure 6.10 shows that gathering congestion readings during a short time interval ($\hat{t}$) results in obtaining the best performance for all the scenarios. Hence, we can say that when the controller generates directives at a higher rate obtains the best results. There are minor differences between the suitable intervals for each scenario.

The best performance for the Taxis and the Campus scenarios is at a 60 s interval, and for the Info5 scenario (the one with the fewest nodes) is 30 s. Thus, it can be said that the fewer nodes there are, the higher rate of directives emitted by the controller is required. For the Info5 scenario, even though in the plot it seems that from the interval of 600 s the function is increasing, simulations up to an interval of 10.800 s with samples every 1800 s have been run, and the delivery ratio remains constant to the value obtained at the 600 s interval.

Finally, the best interval time for the Emergency scenario is 120 s. This scenario drastically changes the message generation in the middle of the simulation. In that variable situation, it is understandable to have a wider time interval range for gathering congestion readings to soothe the effects of the changes.

6.4.5 Number of inputs ($\breve{z}$)

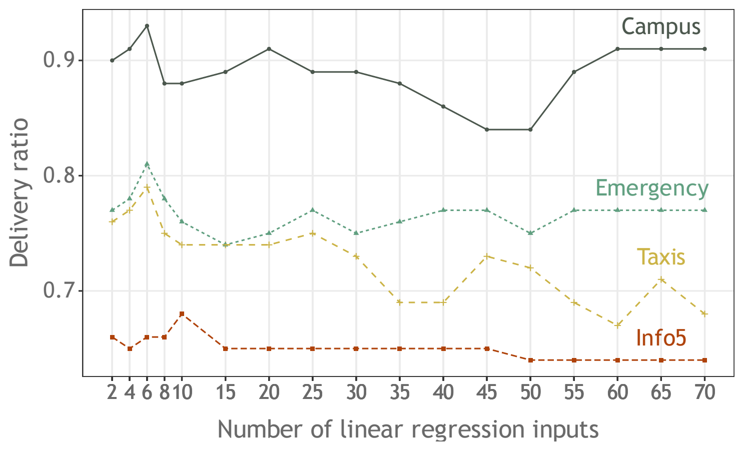


Fig. 6.11: Delivery ratio vs. the number of inputs the linear regression is fed with.

The number of inputs setting ($\breve{z}$) is the max size of $\breve{M}$ (max number of elements). As defined in Section 3.6, $\breve{M}$ behaves as an sliding list of size $\breve{z}$ to keep the congestion readings' recent history on-board. Figure 6.11 shows that, for all the scenarios, clearly, it is better to have a small $\check{z}$. That is fully understandable, as the fewer inputs we use, the fresher the information is.

6.4.6 Prediction time factor (ϕ)

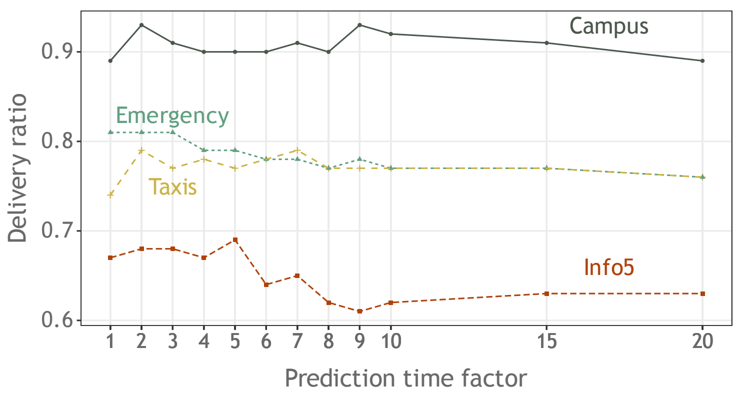


Fig. 6.12: Delivery ratio depending on the prediction time factor.

As shown in Figure 6.12, a minor factor, i.e. predicting the congestion-to-be in the short term, gives the best performance for all the scenarios. For all of them, the factor is 2 except for the Info5 one, which is 5. Hence, we can tell that we can predict the incoming future more accurately than a faraway one. This assertion implies that we need a small factor combined with a small aggregation interval ($\hat{t}$). This combination is feasible as, in the previous section, we have seen that a small $\hat{t}$ leads to better performance.

6.4.7 Reduction factor for a decay (r)

The decay Equation (3.1) is used to calculate the decay of a received congestion reading. Decay is used as a weight factor over the congestion reading in (3.3).

In Figure 6.13, we consider different decay at different times (300s, 600s, 1800s, 3600s). The decay goes from 1 (no decay) to (0.01) at each considered time. For all the scenarios, we can see that the delivery ratio overwhelmingly drops when the congestion reading is not “penalized" by a decay (decay weight close to 1) after an elapsed time. Hence, we conclude that considering a decay for the congestion readings is crucial.

From the transient of the results, for each scenario, it can be seen that the decay at t heavily affects the performance. Nevertheless, it can be observed that for high decay “penalties" (small decay weight values), better results are obtained at any time than with lesser decay (high values).

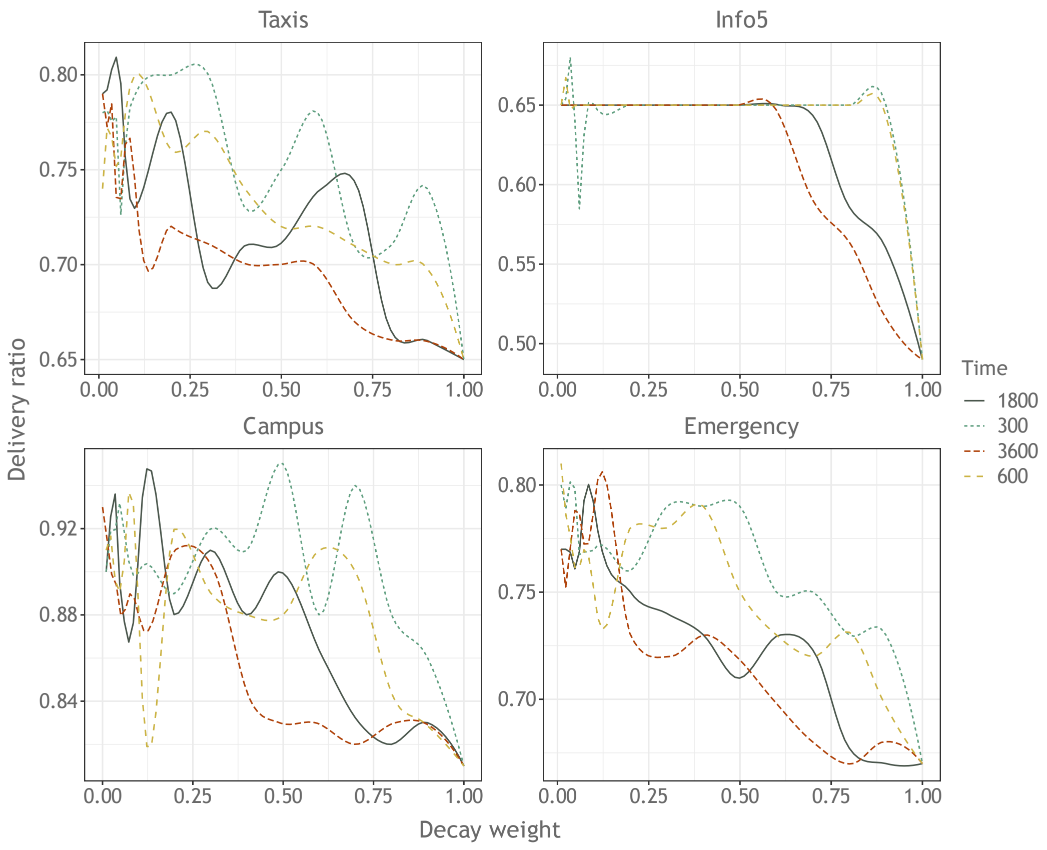


Fig. 6.13: Tendency of the delivery ratio when combining different decay percentages at different times per scenario.

In conclusion, the best strategy is to apply a high decay (small weight factor value), which implies a big diminish of the effect of the congestion measurement after a short elapsed time from its creation up to its reception by any node/controller.

6.4.8 Decay threshold

Section 3.4 shows how the received measurements are aggregated through (3.3). This equation double weights the congestion reading by the number of aggregations it is formed of and by its decay, which is calculated with (3.1). Consequently, a congestion measurement formed of a high number of aggregated congestion measurements would have a high impact despite its decay in the overall process of the congestion measurements aggregation. Hence, several of these congestion readings in the control's current aggregation process can lead to a long tail effect [12], where almost negligible old congestion readings would highly affect the whole aggregation result. In this case, we would have a congestion reading calculation based on, very likely, expired information. To avoid this undesirable situation, a decay threshold is specified so that the congestion readings with a decay lesser than decay threshold are not considered in the controller's congestion calculation.

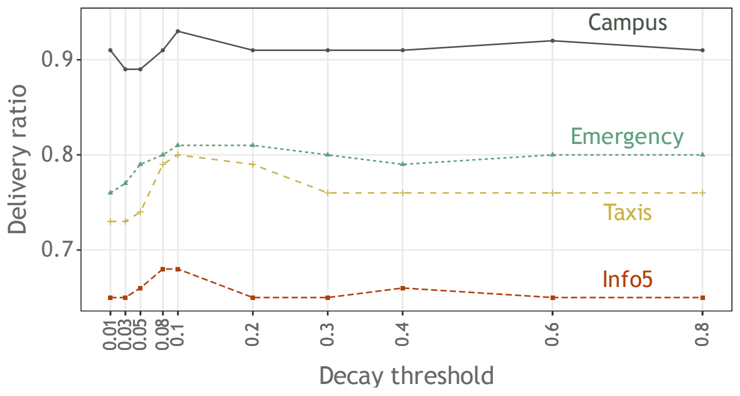


Fig. 6.14: Delivery ratio for different *decay thresholds*.

From Figure 6.14 it can be stated that, for all the scenarios, the best decay threshold is 10% of decay. As expected, either aggregating old readings (small decay threshold) or discarding new readings (high decay threshold) worsens the performance consistently for all the scenarios. Nevertheless, although small decay thresholds diminish the performance for the Emergency scenario, high decay thresholds don't significantly affect such performance. This behaviour is due to the high variance in the latency of this scenario, so considering old readings does not affect the performance much.

6.4.9 Number of aggregations weight (α)

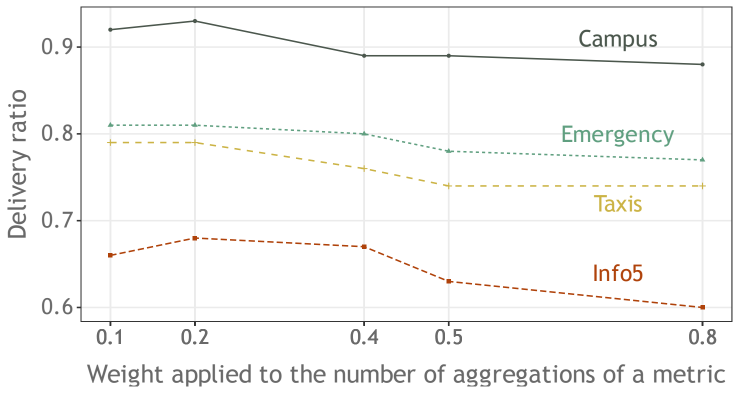


Fig. 6.15: Delivery ratio for different α weights.

The number of aggregation's weight setting (α) used in (3.3) is the weight applied to the *number of aggregations* an aggregated congestion measurement is formed of. As it is a two-factor weighted average, the weight related to the *decay* is the *alpha*'s complementary $(1 - \alpha)$.

As we can see in Figure 6.15, for all the scenarios, we obtain the best performance with an alpha of 0.2. This result concludes that the decay of a congestion measurement is more relevant than the number of aggregations this aggregated measurement is formed of.

6.4.10 Directive generation frequency

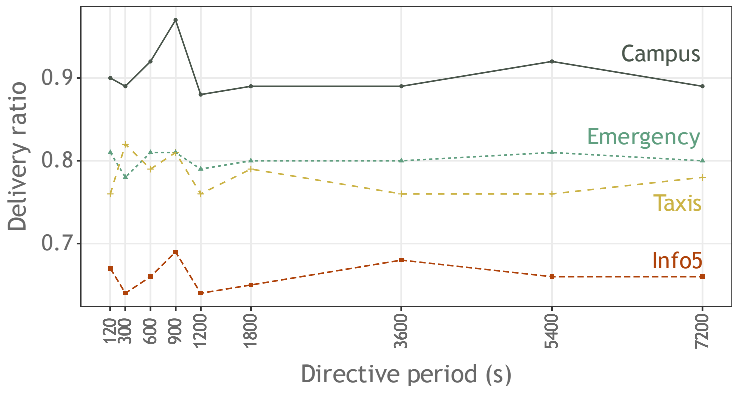


Fig. 6.16: Delivery ratio for different directive generation periods.

As we have seen in Section 3.5, the control system is opportunistic. Nevertheless, each *directive frequency* seconds, the controller re-sends the last generated directive for control beckoning purposes.

As we can see in Figure 6.16, for all the scenarios except for the Taxis one, we obtain the best performance by re-sending the last directive each 900s, provided no contact has happened before. Nevertheless, For the Taxis scenario, we get the optimal performance with a directive frequency of 300s. We can understand this slight difference as the Taxis scenario is the most sparse one, which implies fewer contacts between nodes. Hence, a more frequent directive beckoning gives the nodes more chances to receive a directive, and consequently, we get better performance.

Part IV

Conclusions and Future work

Conclusions 7

THE MOTIVATION for this work was bringing into OppNets the benefits of the SDN architecture by having an SDN-like controller managing a context-aware system fed with network information provided by the SDN-like agents and using this supervised context information to tune the forwarding strategy to achieve a better network performance. More precisely, we envisioned a controller-driven OppNet using a multi-copy forwarding algorithm where an SDN-like controller overviewed this OppNet (possibly a part of it) to determine on the fly the replication limit of the forwarding algorithm.

Next, in this chapter, we go through a summary of the achievements of this research. Also, we point out the immediate future lines derived from this study.

7.1 Achievements

A control layer has been devised. This control layer implements the node's functionalities of gathering, aggregating and forwarding network measurements, and applying the received directives (GAFA). The control layer also implements the controller functionalities. The control layer is executed by the nodes in the OppNet, becoming a controller-driven OppNet.

To state the soundness of the proposal, the control layer has been applied to manage the congestion derived from multi-copy-based forwarding algorithms. This controller-driven OppNet has been tested over four scenarios characterised by different mobility patterns and node densities against baseline forwarding strategies based on message replication. The scenarios have been simulated for the Control and No-Control (Epidemic and Static quota-based) configurations to evaluate the network's performance based on the indicators listed in Section 5.1.

For the scenarios with a message distribution following the ISTH function, the simulations show that for the Control configuration, the replication limit tends to an asymptote proximal to the replication limit of the Static policy performing the best in terms of delivery ratio (optimal). Therefore, it can be stated that under blind network knowledge, the Control configuration approaches the *optimal* replication limit.

Moreover, the Control configuration adapts to changes in the pattern of message generation distribution. It is precisely under these unpredictable conditions that the Control configuration stands out over the other configurations by leveraging its dynamic adaptability to the network conditions. This adaptability brings out a significantly lesser occupancy of the node's buffer and an important reduction of the overhead intrinsic to replication.

Besides, the Control configuration improves the delivery ratio for all the scenarios. Its goodness is accentuated for scenarios with medium-low congestion, as a wider replication limit range can be considered. In contrast, a high congested scenario is stuck to a low replication limit. Undoubtedly, latency benefits from a replication that does not overwhelm the nodes' cache system. The fact that the Control configuration keeps at bay the replication limit to avoid congestion to achieve a better delivery ratio affects the latency. Therefore, the application layer should determine whether maximise the delivery ratio or minimise the latency, so the control layer could apply forwarding strategies to optimise one or the other.

Furthermore, the control layer is highly configurable to provide the best performance depending on the Oppnet's nature. Nevertheless, generic values providing a good performance have been determined from simulations over the aforementioned scenarios. In this regard, simulations show that the controller must be fed fresh congestion readings from contacts. Therefore, applying a decay weight over the measurements used to predict the network conditions is decisive. In this regard, it is more effective a short-term than a long-term prediction.

Also, the simulation evinces that the more sparse the network is (fewer contacts between nodes), the more directives are needed. For the use case of congestion control, the replication limit needs to grow slowly, whereas, in a congestion state, a sharp reduction is required. The optimal congestion interval is highly coupled with the characteristics of the scenario.

Finally, simulations depict that, despite the disconnections, network partitioning and long delay paths prone to OppNets, a small number of controllers suffice.

Overall, this study asserts that (i) a context-aware system built upon the SDN pillar principles is a good approach for context-management in OppNets, and (ii) using this context-aware system to regulate the replication in an OppNet driven by a multi-copy forwarding strategy leads to better network performance.

7.2 Future work

Out of this research, we envision three possible lines of future work. The first proposal consists in nodes positing as controllers in an emerging role basis. The second one comprises sharing the context information each controller manages amongst the other controllers aiming to build a more global network context knowledge. Finally, we consider implementing the southbound SDN protocols over the nodes to achieve an SDN-compliant OppNet. The three proposals are expanded in the following sections.

7.2.1 Emerging controllers

Undoubtedly, as we have mentioned in Section 7.1, the most significant achievement of this research has been designing a context-aware system managed by SDN-like controllers. Also, in the introduction of Chapter 3, we point out that we have selected the most central nodes to perform as controllers. We have measured the centrality in terms of the number of different contacts per time unit a node has. Hence, the selection of controllers has been hand-picked. As the SDN-like controller is a crucial building block of this research, we devised different methodologies to determine the identity of the controllers. Selecting the central ones was the first approach. Following, we envision a dynamic way inspired by the Internet Group Management Protocol [14] named as emerging controller.

The idea behind the emerging controller is implementing a protocol where initially, there would be no controller. After a timeout without receiving control directives, a node would adopt the controller role. Likewise, when a controller would receive control directives from other nodes, it should decide whether to keep the role or withdraw it. We believe that this is the natural evolution of the proposed controller-driven OppNet.

7.2.2 Approximation to a global network overview

As we have mentioned in Section 3.5, under the ever-changing topology of an OppNet, the controller has an overview of the network “segment" within its reach. Nevertheless, opportunistic contacts also favour receiving control information from other controllers in the network. The controllers could use the control information received from other controllers to build a more extensive network overview. It would be interesting to evaluate if working with more global context information would be more beneficial than using nearby information.

Moreover, upon the control information received from other controllers, they could build a dynamic graph of the controllers' influence range, indicating which nodes are under the range of a particular controller. This information could be disseminated along the OppNet so the nodes could use it to decide whether a node is a good relay candidate based on the congestion perception of the controller influencing the relay candidate node.

7.2.3 Implementation of a connectionless SDN southbound protocol

Finally, as we have mentioned in Section 1, up to our knowledge, SDN protocols are based on TCP and, therefore, do not apply to OppNets. In SDN, the Control and the data planes communicate through a southbound protocol such as OpenFlow or P4Runtime, among others. We propose checking the feasibility of developing a connectionless southbound protocol to communicate the OppNet nodes (data plane) and the controllers (control plane) to achieve a Software Defined Opportunistic Network (SDON) fully compliant with the SDN southbound communication specifications.

Part V

Bibliography

Bibliography

[1] Mehran Abolhasan, Mahrokh Abdollahi, Wei Ni, et al. "A routing framework for offloading traffic from cellular networks to SDN-based multi-hop device-to-device networks". In: *IEEE Transactions on Network and Service Management* 15.4 (2018), pp. 1516–1531 (cit. on p. 17).

[2] Dimitrios-Georgios Akestoridis. *CRAWDAD dataset uoi/haggle (v. 2016-08-28): derived from cambridge/haggle (v. 2009-05-29)*. Downloaded from `http://crawdad.org/uoi/haggle/20160828/one`. Aug. 2016 (cit. on p. 43).

[3] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. "Wireless sensor networks: a survey". In: *Computer Networks* 38.4 (2002), pp. 393–422 (cit. on p. 9).

[4] Ian F Akyildiz, Xudong Wang, and Weilin Wang. "Wireless mesh networks: a survey". In: *Computer networks* 47.4 (2005), pp. 445–487 (cit. on p. 3).

[5] Mark Allman, Vern Paxson, and Ethan Blanton. *TCP congestion control*. Tech. rep. 2009 (cit. on p. 12).

[6] Flor Álvarez, Lars Almon, Patrick Lieser, et al. "Conducting a large-scale field test of a smartphone-based communication network for emergency response". In: *Proceedings of the 13th Workshop on Challenged Networks*. 2018, pp. 3–10 (cit. on p. 8).

[7] Rafay Iqbal Ansari, Chrysostomos Chrysostomou, Syed Ali Hassan, et al. "5G D2D networks: Techniques, challenges, and future prospects". In: *IEEE Systems Journal* 12.4 (2017), pp. 3970–3984 (cit. on p. 3).

[8] Suvadip Batabyal, Parama Bhaumik, Samiran Chattopadhyay, and Sudip Misra. "Steady-state analysis of buffer occupancy for different forwarding strategies in mobile opportunistic network". In: *IEEE Transactions on Vehicular Technology* 68.7 (2019), pp. 6951–6963 (cit. on p. 14).

[9] Samaresh Bera, Sudip Misra, and Athanasios V Vasilakos. "Software-defined networking for internet of things: A survey". In: *IEEE Internet of Things Journal* 4.6 (2017), pp. 1994–2008 (cit. on p. 16).

[10] Archie Bland. "FireChat–the messaging app that's powering the Hong Kong protests". In: *The Guardian* 29 (2014) (cit. on p. 10).

[11] John P Boyd. "Asymptotic Fourier coefficients for a C∞ bell (smoothed-"top-hat") & the Fourier extension problem". In: *Journal of Scientific Computing* 29.1 (2006), pp. 1–24 (cit. on p. 43).

[12] George W. Brown and John W. Tukey. “Some Distributions of Sample Means”. In: *The Annals of Mathematical Statistics* 17.1 (1946), pp. 1–12 (cit. on p. 65).

[13] Yegui Cai, F Richard Yu, Chengchao Liang, Bo Sun, and Qiao Yan. “Software-defined device-to-device (D2D) communications in virtual wireless networks with imperfect network state information (NSI)”. In: *IEEE Transactions on Vehicular Technology* 65.9 (2016) (cit. on p. 17).

[14] B Cain, S Deering, I Kouvelas, B Fenner, and A Thyagarajan. *RFC3376: internet group management protocol, version 3*. 2002 (cit. on p. 73).

[15] Pablo Calcina Ccori, Laisa Caroline Costa De Biase, Marcelo Knorich Zuffo, and Flávio Soares Corrêa da Silva. “Device discovery strategies for the IoT”. In: *2016 IEEE International Symposium on Consumer Electronics (ISCE)*. IEEE. 2016, pp. 97–98 (cit. on p. 9).

[16] Manisha Chahal, Sandeep Harit, Krishn K Mishra, Arun Kumar Sangaiah, and Zhigao Zheng. “A survey on software-defined networking in vehicular ad hoc networks: Challenges, applications and use cases”. In: *Sustainable cities and society* 35 (2017), pp. 830–840 (cit. on p. 16).

[17] Nessrine Chakchouk. “A survey on opportunistic routing in wireless communication networks”. In: *IEEE Communications Surveys & Tutorials* 17.4 (2015), pp. 2214–2241 (cit. on pp. 4, 8, 10, 11).

[18] *Community Resource for Archiving Wireless Data At Dartmouth*. url= https://crawdad.org/. Accessed on 04.11.2021 (cit. on p. 42).

[19] Marco Conti, Chiara Boldrini, Salil S Kanhere, et al. “From MANET to people-centric networking: Milestones and open research challenges”. In: *Computer Communications* 71 (2015), pp. 1–21 (cit. on p. 3).

[20] Marco Conti and Mohan Kumar. “Opportunities in opportunistic computing”. In: *Computer* 43.01 (2010), pp. 42–50 (cit. on p. 9).

[21] Renu Dalal, Manju Khari, John Petearson Anzola, and Vicente García-Díaz. “Proliferation of Opportunistic Routing: A Systematic Review”. In: *IEEE Access* (2021) (cit. on p. 41).

[22] Jens Dede, Anna Förster, Enrique Hernández-Orallo, et al. “Simulating opportunistic networks: Survey and future directions”. In: *IEEE Communications Surveys & Tutorials* 20.2 (2017), pp. 1547–1573 (cit. on pp. 12, 41, 46).

[23] Richard C Dorf and Robert H Bishop. *Modern control systems*. Pearson, 2011 (cit. on p. 27).

[24] Kevin Fall. “A delay-tolerant network architecture for challenged internets”. In: *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. 2003, pp. 27–34 (cit. on pp. 3, 8).

[25] Diego Freire, Sergi Robles, and Carlos Borrego. “Towards a Methodology for the Development of Routing Algorithms in Opportunistic Networks”. In: *arXiv preprint arXiv:2009.01532* (2020) (cit. on p. 11).

[26] Gourish Goudar and Suvadip Batabyal. “Estimating Buffer Occupancy sans Message Exchange in Mobile Opportunistic Networks”. In: *IEEE Networking Letters* (2022) (cit. on pp. 14, 27).

[27] Gourish Goudar and Suvadip Batabyal. "Point of congestion in large buffer mobile opportunistic networks". In: *IEEE Communications Letters* 24.7 (2020), pp. 1586–1590 (cit. on p. 14).

[28] Frédéric Guidec, Yves Mahéo, Pascale Launay, Lionel Touseau, and Camille Noûs. "Bringing Opportunistic Networking to Smartphones: a Pragmatic Approach". In: *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE. 2021, pp. 574–579 (cit. on p. 10).

[29] Bin Guo, Zhu Wang, Zhiwen Yu, et al. "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm". In: *ACM computing surveys (CSUR)* 48.1 (2015), pp. 1–31 (cit. on p. 9).

[30] C Yu Hans, Giorgio Quer, and Ramesh R Rao. "Wireless SDN mobile ad hoc network: From theory to practice". In: *2017 IEEE International Conference on Communications (ICC)*. IEEE. 2017, pp. 1–7 (cit. on p. 17).

[31] Israat Tanzeena Haque and Nael Abu-Ghazaleh. "Wireless software defined networking: A survey and taxonomy". In: *IEEE Communications Surveys & Tutorials* 18.4 (2016), pp. 2713–2737 (cit. on p. 16).

[32] Yvon Henri. "The OneWeb satellite system". In: *Handbook of Small Satellites: Technology, Design, Manufacture, Applications, Economics and Regulation* (2020), pp. 1–10 (cit. on p. 8).

[33] Fei Hu, Qi Hao, and Ke Bao. "A survey on software-defined network and openflow: From concept to implementation". In: *IEEE Communications Surveys & Tutorials* 16.4 (2014), pp. 2181–2206 (cit. on pp. 15, 16).

[34] Qingsong Hu, Juan Ding, and Shiyin Li. "A novel cognitive opportunistic communication framework for coal mines". In: *Mathematical Problems in Engineering* 2019 (2019) (cit. on p. 8).

[35] Pan Hui, Augustin Chaintreau, James Scott, et al. "Pocket switched networks and human mobility in conference environments". In: *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. 2005, pp. 244–251 (cit. on p. 3).

[36] Sushant Jain, Kevin Fall, and Rabin Patra. "Routing in a delay tolerant network". In: *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*. 2004, pp. 145–158 (cit. on pp. 10, 11).

[37] P.K. Janert. *Feedback Control for Computer Systems: Introducing Control Theory to Enterprise Programmers*. O'Reilly Media, 2013 (cit. on p. 36).

[38] Manuel Jesús-Azabal, Juan Luis Herrera, Sergio Laso, and Jaime Galán-Jiménez. "OPPNets and rural areas: an opportunistic solution for remote communications". In: *Wireless Communications and Mobile Computing* 2021 (2021) (cit. on p. 8).

[39] Soujanya Katikala. "Google project loon". In: *InSight: Rivier Academic Journal* 10.2 (2014), pp. 1–6 (cit. on p. 8).

[40] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. "The ONE simulator for DTN protocol evaluation". In: *Proceedings of the 2nd international conference on simulation tools and techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). 2009, p. 55 (cit. on p. 46).

[41] Hlabishi I Kobo, Adnan M Abu-Mahfouz, and Gerhard P Hancke. “A survey on software-defined wireless sensor networks: Challenges and design requirements”. In: *IEEE access* 5 (2017), pp. 1872–1899 (cit. on p. 16).

[42] Diego Kreutz, Fernando Ramos, Paulo Verissimo, et al. “Software-defined networking: A comprehensive survey”. In: *arXiv preprint arXiv:1406.0440* (2014) (cit. on pp. 4, 15, 16).

[43] Amir Krifa, Chadi Barakat, and Thrasyvoulos Spyropoulos. “Optimal buffer management policies for delay tolerant networks”. In: *2008 5th annual IEEE communications society conference on sensor, mesh and ad hoc communications and networks*. IEEE. 2008, pp. 260–268 (cit. on pp. 13, 57).

[44] Vishnupriya Kuppusamy, Udaya Miriya Thanthrige, Asanga Udugama, and Anna Förster. “Evaluating Forwarding Protocols in Opportunistic Networks: Trends, Advances, Challenges and Best Practices”. In: *Future Internet* 11.5 (2019) (cit. on pp. 4, 9, 12, 41, 46).

[45] Jani Lakkakorpi, Mikko Pitkänen, and Jörg Ott. “Using buffer space advertisements to avoid congestion in mobile opportunistic DTNs”. In: *International Conference on Wired/Wireless Internet Communications*. Springer. 2011, pp. 386–397 (cit. on p. 14).

[46] Michael Lee and Travis Atkison. “Vanet applications: Past, present, and future”. In: *Vehicular Communications* 28 (2021), p. 100310 (cit. on p. 10).

[47] He Li, Kaoru Ota, Mianxiong Dong, and Minyi Guo. “Mobile Crowdsensing in Software Defined Opportunistic Networks”. In: *IEEE Communications Magazine* 55.6 (2017), pp. 140–145 (cit. on p. 17).

[48] Jiajia Liu, Shangwei Zhang, Nei Kato, Hirotaka Ujikawa, and Kenichi Suzuki. “Device-to-device communications for enhancing quality of experience in software defined multi-tier LTE-A networks”. In: *IEEE Network* 29.4 (2015), pp. 46–52 (cit. on p. 17).

[49] Jonathan C McDowell. “The low earth orbit satellite population and impacts of the SpaceX Starlink constellation”. In: *The Astrophysical Journal Letters* 892.2 (2020), p. L36 (cit. on p. 8).

[50] Nick McKeown, Tom Anderson, Hari Balakrishnan, et al. “OpenFlow: enabling innovation in campus networks”. In: *ACM SIGCOMM Computer Communication Review* 38.2 (2008), pp. 69–74 (cit. on p. 15).

[51] Matthias Grossglauser Michal Piorkowski Natasa Sarafijanovic-Djukic. *Dataset of mobility traces of taxi cabs in San Francisco, USA (v. 2009-02-24)*. Feb. 2009 (cit. on p. 43).

[52] Vinicius FS Mota, Felipe D Cunha, Daniel F Macedo, José MS Nogueira, and Antonio AF Loureiro. “Protocols, mobility models and tools in opportunistic networks: A survey”. In: *Computer Communications* 48 (2014), pp. 5–19 (cit. on pp. 4, 8, 11, 12, 41).

[53] Anand Nayyar, Ranbir Singh Batth, Dac Binh Ha, and G Sussendran. “Opportunistic networks: present scenario-a mirror review”. In: *International Journal of Communication Networks and Information Security* 10.1 (2018), pp. 223–241 (cit. on pp. 4, 11).

[54] Lionel Nkenyereye, Lewis Nkenyereye, SM Islam, et al. “Software-defined network-based vehicular networks: A position paper on their modeling and implementation”. In: *Sensors* 19.17 (2019), p. 3788 (cit. on p. 16).

[55] Jéferson Campos Nobre, Allan M. de Souza, Denis Rosário, et al. “Vehicular software-defined networking and fog computing: Integration and design principles”. In: *Ad Hoc Networks* 82 (2019), pp. 172–181 (cit. on p. 16).

[56] Bruno Astuto A Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. “A survey of software-defined networking: Past, present, and future of programmable networks”. In: *IEEE Communications surveys & tutorials* 16.3 (2014), pp. 1617–1634 (cit. on p. 16).

[57] Daru Pan, Zhaohua Ruan, Nian Zhou, Xiong Liu, and Zhaohui Song. “A comprehensive-integrated buffer management strategy for opportunistic networks”. In: *EURASIP Journal on Wireless Communications and Networking* 2013.1 (2013), pp. 1–10 (cit. on p. 13).

[58] Luciana Pelusi, Andrea Passarella, and Marco Conti. “Opportunistic networking: data forwarding in disconnected mobile ad hoc networks”. In: *IEEE Communications Magazine* 44.11 (2006), pp. 134–141 (cit. on p. 8).

[59] Zhijing Qin, Grit Denker, Carlo Giannelli, Paolo Bellavista, and Nalini Venkatasubramanian. “A Software Defined Networking architecture for the Internet-of-Things”. In: *2014 IEEE Network Operations and Management Symposium (NOMS)*. 2014, pp. 1–9 (cit. on p. 16).

[60] Asmaa Rady, EL-Sayed M El-Rabaie, Mona Shokair, and Nariman Abdel-Salam. “Comprehensive survey of routing protocols for Mobile Wireless Sensor Networks”. In: *International Journal of Communication Systems* 34.15 (2021), e4942 (cit. on p. 3).

[61] Wajid Rafique, Lianyong Qi, Ibrar Yaqoob, et al. “Complementing IoT Services Through Software Defined Networking and Edge Computing: A Comprehensive Survey”. In: *IEEE Communications Surveys Tutorials* 22.3 (2020), pp. 1761–1804 (cit. on p. 16).

[62] Ram Ramanathan, Christophe Servaes, Warren Ramanathan, Ayush Dusia, and Adarshpal S Sethi. “Long-range short-burst mobile mesh networking: Architecture and evaluation”. In: *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE. 2019, pp. 1–2 (cit. on p. 10).

[63] SR Jino Ramson and D Jackuline Moni. “Applications of wireless sensor networks—A survey”. In: *2017 international conference on innovations in electrical, electronics, instrumentation and media technology (ICEEIMT)*. IEEE. 2017, pp. 325–329 (cit. on p. 9).

[64] Rahul Sachdeva and Amita Dev. “Review of opportunistic network: assessing past, present, and future”. In: *International Journal of Communication Systems* 34.11 (2021), e4860 (cit. on pp. 8, 11).

[65] Matthew Seligman, Kevin Fall, and Padma Mundur. “Alternative custodians for congestion control in delay tolerant networks”. In: *Proceedings of the 2006 SIGCOMM workshop on Challenged networks*. 2006, pp. 229–236 (cit. on p. 14).

[66] Aloizio P Silva, Scott Burleigh, Celso M Hirata, and Katia Obraczka. “A survey on congestion control for delay and disruption tolerant networks”. In: *Ad Hoc Networks* 25 (2015), pp. 480–494 (cit. on pp. 13, 14, 17).

[67] Sanjeev Singh and Rakesh Kumar Jha. “A survey on software defined networking: Architecture for next generation network”. In: *Journal of Network and Systems Management* 25.2 (2017), pp. 321–374 (cit. on pp. 15, 16).

[68] CC Sobin, Vaskar Raychoudhury, Gustavo Marfia, and Ankita Singla. “A survey of routing and data dissemination in delay tolerant networks”. In: *Journal of Network and Computer Applications* 67 (2016), pp. 128–146 (cit. on pp. 4, 11, 12).

[69] Bambang Soelistijanto and Michael P Howarth. “Transfer reliability and congestion control strategies in opportunistic networks: A survey”. In: *IEEE communications surveys & tutorials* 16.1 (2013), pp. 538–555 (cit. on p. 13).

[70] Bambang Soelistijanto and Michael P. Howarth. “Transfer Reliability and Congestion Control Strategies in Opportunistic Networks: A Survey”. In: *IEEE Communications Surveys Tutorials* 16.1 (2014), pp. 538–555 (cit. on pp. 13, 14, 33).

[71] Sanhua Song. “An effective congestion control scheme based on early offload for space delay/disruption tolerant network”. In: *International Journal of Security and Networks* 16.1 (2021), pp. 28–36 (cit. on p. 14).

[72] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S Raghavendra. “Spray and wait: an efficient routing scheme for intermittently connected mobile networks”. In: *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. ACM. 2005, pp. 252–259 (cit. on p. 11).

[73] Saif Al-Sultan, Moath M Al-Doori, Ali H Al-Bayatti, and Hussien Zedan. “A comprehensive survey on vehicular ad hoc network”. In: *Journal of network and computer applications* 37 (2014), pp. 380–392 (cit. on p. 3).

[74] Nathanael Thompson, Samuel C Nelson, Mehedi Bakht, Tarek Abdelzaher, and Robin Kravets. “Retiring replicants: congestion control for intermittently-connected networks”. In: *2010 Proceedings IEEE INFOCOM*. IEEE. 2010, pp. 1–9 (cit. on pp. 12, 14).

[75] Ozan Tonguz, Nawapom Wisitpongphan, Fan Bai, Priyantha Mudalige, and Varsha Sadekar. “Broadcasting in VANET”. In: *2007 mobile networking for vehicular environments*. IEEE. 2007, pp. 7–12 (cit. on p. 9).

[76] Lionel Touseau, Yves Mahéo, and Camille Noûs. “A Smartphone-Targeted Opportunistic Computing Environment for Decentralized Web Applications”. In: *2021 IEEE 46th Conference on Local Computer Networks (LCN)*. IEEE. 2021, pp. 363–366 (cit. on p. 10).

[77] Sacha Trifunovic, Sylvia T. Kouyoumdjieva, Bernhard Distl, et al. “A Decade of Research in Opportunistic Networks: Challenges, Relevance, and Future Directions”. In: *IEEE Communications Magazine* 55.1 (2017), pp. 168–173 (cit. on pp. 8–10).

[78] I Union. “IMT traffic estimates for the years 2020 to 2030”. In: *Report ITU* 2370 (2015). Available at `https://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-M.2370-2015-PDF-E.pdf` (cit. on p. 3).

[79] Muhammad Usman, Anteneh A Gebremariam, Usman Raza, and Fabrizio Granelli. “A software-defined device-to-device communication architecture for public safety applications in 5G networks”. In: *IEEE Access* 3 (2015), pp. 1649–1654 (cit. on p. 16).

[80] Amin Vahdat, David Becker, et al. *Epidemic routing for partially connected ad hoc networks*. Tech. rep. Duke University, 2000 (cit. on p. 11).

[81] Anna Maria Vegni, Carlos Borrego Iglesias, and Valeria Loscri. “MOVES: A MemOry-based VEhicular Social forwarding technique”. In: *Computer Networks* 197 (2021), p. 108324 (cit. on p. 9).

[82] Wenfeng Xia, Yonggang Wen, Chuan Heng Foh, Dusit Niyato, and Haiyong Xie. “A survey on software-defined networking”. In: *IEEE Communications Surveys & Tutorials* 17.1 (2014), pp. 27–51 (cit. on p. 16).

[83] Dianlei Xu, Yong Li, Xinlei Chen, et al. “A survey of opportunistic offloading”. In: *IEEE Communications Surveys & Tutorials* 20.3 (2018), pp. 2198–2236 (cit. on p. 8).

[84] Yuan Zhang, Lin Cui, Wei Wang, and Yuxiang Zhang. “A survey on software defined networking with multiple controllers”. In: *Journal of Network and Computer Applications* 103 (2018), pp. 101–118 (cit. on p. 16).

MªCarmen de Toro Valdivia
Bellaterra, December 2022