Thesis presented to the Instituto Tecnológico de Aeronáutica, in partial fulfillment of the requirements for the degree of Doctor of Science in the Program of Engenharia Eletrônica e Computação, Field of Informática.

Aloizio Pereira da Silva

A NOVEL CONGESTION CONTROL FRAMEWORK FOR DELAY AND DISRUPTION TOLERANT NETWORKS

Thesis approved in its final version by signatories below:

Prof. Dr. Celso Massaki Hirata

Advisor

Prof^{a}. Dr^{a}. Katia Obraczka

Co-advisor

Prof. Dr. Luiz Carlos Sandoval Góes

Prorector of Graduate Studies and Research

Campo Montenegro
São José dos Campos, SP - Brazil
2015

Cataloging-in Publication Data
Documentation and Information Division

Silva, Aloizio Pereira da
A Novel Congestion Control Framework For Delay and Disruption Tolerant Networks / Aloizio Pereira da Silva.
São José dos Campos, 2015.
175f.

Thesis of Doctor of Science – Course of Engenharia Eletrônica e Computação. Area of Informática – Instituto Tecnológico de Aeronáutica, 2015. Advisor: Prof. Dr. Celso Massaki Hirata. Co-advisor: Prof[a]. Dr[a]. Katia Obraczka.

1. Wireless Networks. 2. Performance. 3. Congestion Control. I. Instituto Tecnológico de Aeronáutica. II. Title.

BIBLIOGRAPHIC REFERENCE

SILVA, Aloizio Pereira da. **A Novel Congestion Control Framework For Delay and Disruption Tolerant Networks**. 2015. 175f. Thesis of Doctor of Science – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSION OF RIGHTS

Aloizio Pereira da Silva
Av. Cidade Jardim, 679
12.233-066 – São José dos Campos–SP

A NOVEL CONGESTION CONTROL FRAMEWORK FOR DELAY AND DISRUPTION TOLERANT NETWORKS

Aloizio Pereira da Silva

Thesis Committee Composition:

Prof. Dr.	XXXXX		-	NASA
Prof. Dr.	Celso Massaki Hirata	Advisor	-	ITA
Prof^{a}. Dr^{a}.	Katia Obraczka	Co-advisor	-	UCSC
Prof. Dr.	ZZZZZ		-	SPACE-X
Prof. Dr.	SSSSSSSSS		-	GOOGLE
Prof. Dr.	RRRRRRRRR		-	HP
Prof. Dr.	EEEEEEEEEEEE		-	CISCO

ITA

“Man is still the most extraordinary computer of all. ”

— JOHN F. KENNEDY

Resumo

Redes tolerantes a atrasos e desconexões, também conhecidas como DTNs (*Delay and Disruption Tolerant Networks*) são redes capazes de lidar com atrasos longos e desconexões intermitentes, diferentemente das redes tradicionais, tais como Internet TCP/IP. Outra característica que diferencia DTNs das redes convencionais é que não existe nenhuma garantia de uma conectividade fim-a-fim entre fonte e destino. Essas características apresentam uma série de desafios técnicos no projeto de funções chaves da rede tais como mecanismos de roteamento e mecanismos de controle de congestionameto. Detectar e tratar o congestionamento em DTN é particularmente um desafio e extremamente importante no contexto de performance da rede. Atualmente o controle de congestionamento fim-a-fim em redes tradicionais é tratado pelo TCP (Transmission Control Protocol) que previne a rede de entrar em colapso mas não evita que o desempenho da rede degrade quando em presença de congestionamento. TCP/IP funciona bem quando não existe interrupção da comunicação fim-a-fim. Em particular, DTN endereça alguns ponto fracos do TCP, por exemplo implementando mecanismos de controle de congestionamento onde cada roteador pode tomar decisão considerando a informação local (e.x. aceitando dados de outro roteador). Essa informação local pode incluir disponibilidade de armazenamento e o risco de aceitar os dados baseado em experiências anteriores as quais podem ter tido efeitos adverso na rede resultando por exemplo em perda de dados. Mecanismos de controle de congestionamento DTN existentes tipicamente tentam usar alguma informação global da rede ou foram projetados para operar em um cenário especifico sujeito a uma estratégia especifica de encaminhamento de dados. Como resultado, estes mecanismos não tem boa performance quando eles são usados em diferentes cenários com diferentes protocolos de roteamento.

Neste trabalho, nós primeiramente revisamos os desafios em DTNs e exploramos o dominio dos mecanismos de controle de congestionamento nestas redes. Além disso, nós propomos uma taxonomia para classificar mecanismos de controle de congestionamento DTN discutindo pontos fracos e fortes no contexto de suas suposições e aplicabilidade. Nós também apresentamos uma analise quantitativa de alguns mecanismos de controle de congestionamento DTN para avaliar como eles se comportam no cenário de comunicação espacial considerando que eles foram projetados para operar em cenários terrestres. Nós avaliamos estes mecanismos extensivamente através de simulação, usando duas aplicações diferentes e três protocolos de roteamento e padrões de mobilidade. Os resultados mostraram que os mecanismos selecionados não

tiveram boa performance no cenário espacial. Portanto, do ponto de vista das caracteristicas DTN, para estudar novos mecanismos de controle de congestionamento e entender o impacto do congestionamento em DTN nós modelamos o problema de congestionamento DTN usando a teoria da percolação. Nós deixaremos mais claro ao longo deste trabalho que a formulação do problema de congestionamento DTN como um processo de percolação acontece naturalmente e o modelo de percolação resultante é simples e facil de derivar. Além disso, outra caracteristica importante do modelo de percolação proposto é o fato de que ao invés de requerer o uso de informação global da rede, ele depende exclusivamente de informação local, por exemplo, informação relacionada ao nó ou aos nós vizinhos dele. A principal vantagem do nosso modelo matematico é permitir validar experimentos reais e simulação. Consequentemente, o modelo proposto pode ser usado para predizer e controlar congestionamento em DTNs.

Dado que longe das redes tradicionais, DTN é um novo tipo de rede derivada de pesquisas realizadas no contexto de comunicação espacial e considerando a importância do controle de congestionamento que diretamente afeta a performance da rede. O desenvolvimento de redes DTN deve depender de um mecanismo de controle de congestionamento que garanta confiabilidade, estabilidade e a extensão da rede. Objetivando melhorar a confiabilidade da entrega de mensagens nestas redes de desafio, este trabalho propoõe DTN-learning, um framework adaptativo and autonomo sensível a congestionamento que mitiga o congestionamento usando *reinforcement learning*, uma estratégia de aprendizado de máquina. Esse framework permite que os nós da rede adaptem o comportamento deles durante o aprendizado (*online*). O framework é genérico e pode facilmente ser combinado com esquemas de controle de congestionamento DTN existentes para controle local. Resultados preliminares mostram que usando nossa abordagem adaptativa, os nós da rede exibem um comportamento mais preciso, aumentando a taxa de entrega e diminuindo a taxa de descartes, quando comparado a enfoques que não usam aprendizado. Isso mitiga o fenômeno de congestionamento observado em mecanismos de controle de congestioanmento não adaptativos e ajuda a rede a alcançar alta performance mais rápido.

Abstract

Delay and Disruption Tolerant Networks (DTNs) are networks that experience frequent and long-lived connectivity disruptions. Unlike traditional networks, such as TCP/IP Internet, DTNs are often subject to high latency caused by very long propagation delays (e.g. interplanetary communication) and/or intermittent connectivity. Another feature that sets DTNs apart from conventional networks is that there is no guarantee of end-to-end connectivity between source and destination. Such distinct features pose a number of technical challenges in designing core network functions such as routing and congestion control mechanisms. Dectecting and dealing with congestion in DTNs is an important and challenging problem. Currently end-to-end congestion control is handled by the TCP (Transmission Control Protocol) that prevents the network from collapsing, but network degradation does occur when the network becomes congested. TCP/IP works well when there is no disruption to end-to-end communication. DTN addresses weaknesses in TCP such as implementing congestion control mechanisms where each router can make decisions based on local information such as accepting a bundle of data from another router. This local information may include storage availability and the risk of accepting the data bundle based on previous experience which may have had adverse effects on the networking resulting in loss of data. Existing DTN congestion control mechanisms typically try to use some network global information or they are designed to operate in a particular scenario and they depend on forwarding strategy, for example, replication forwarding. As a result, existing DTN congestion control mechanisms do not have good performance when they are applied in different scenarios with different routing protocols.

In this thesis, we first review important challenges of DTNs and survey the existing congestion control mechanism of this domain. Furthermore, we provide a taxonomy of existing DTN congestion control mechanisms and discusses their strengths and weaknesses in the context of their assumptions and applicability in DTN applications. We also present a quantitative analysis of some DTN congestion control mechanisms to evaluate how they behave in deep space communication scenario since they were designed to operate at terrestrial DTN. We extensively evaluated these mechanisms using two different applications and three different routing protocols and mobility patterns. The evaluation results show that the selected mechanisms poorly perform in deep space scenario. Therefore, in view of DTN characteristics, to study new congestion controls and better undersand the impact of congestion in DTN we modeled DTN con-

gestion problem using percolation theory. As will become clear in the remainder of this work, we argue that formulating the DTN congestion problem as a percolation process happens quite naturally and the resulting percolation model is simple and easy to derive. Another important feature of the proposed percolation model is the fact that instead of requiring global information about the whole network, it relies exclusively on local information, i.e., information related to a node and its neighboring nodes. The principal advantage of our mathematical model is to provide a fast way of having an idea of the system's performance being modeled and allow us to validate either simulation or realistic experiments. Consequently the proposed model can be used to predict and control congestion in DTNs.

Being aware that far from the traditional network, DTN is a new kind of network derived by deep space communication and as congestion control is an important factor that directly affects network performance. The development of DTN must rely on the perfect congestion control mechanism to ensure reliability, stability and extensiveness of the network. In order to enhance the reliability of data delivery in such challenging network, this thesis proposes DTN-Learning, an adaptive and autonomous congestion aware framework that mitigates the congestion by using Reinforcement Learning. This allows the network nodes to adapt their behavior online in a real environment. It is general and can easily be combined with existing schemes for local control. Preliminary results show that using our adaptive approach, the network node exhibits a more accurate behavior, increasing the delivery ratio and decreasing drop ratio, as compared to approaches that do not use learning. This mitigates congestion phenomena observed in non-adaptive local congestion control mechanisms and helps the network to reach high performance faster.

List of Figures

List of Tables

List of Abbreviations and Acronyms

AI	artificial intelligence
ACM	association for computing machinery
AFNER	average forwarding number based on epidemic routing
AIMD	additive increase multiplicative decrease
AOBMP	adaptive optimal buffer management policies
ARPANET	advanced research and project agency network
BER	bit error rate
BORV	buffer occupancy rate value
BPA	bundle protocol agent
CAFE	context aware forwarding algorithm
CAS	congestion adjacent state
CBORV	current buffer occupancy rate value
CCC	credit based congestion control
CLA	convergence layer adapter
CN	congestion notification
CS	congestion state
CV	congestion value
DCCP	datagram congestion control protocol
DCCR	dynamic congestion control based routing
DCN	decrease congestion notification
DNS	domain name system
DTN	delay and disruption tolerant network
DTNRG	delay tolerant networking research group
ECN	explicit congestion notification
E-Drop	equal drop
EID	end point identifier
EWMA	exponentially weighted moving average
FBD	flood based drop
FR	following routing
GDB	global knowledge based drop

GEO	geosynchronous satellite
GT-SIM	georgia tech internetwork topology models
GUI	graphical user interface
HBD	history based drop
ID	identity
IEEE	institute of electrical and electronics engineers
IETF	Internet engineering task force
IPN	interplanetary network
IRTF	Internet research task force
IMRASFC	incentive multi-path routing with alternative storage
ION	interplanetary overlay network
IP	Internet protocol
JPL	jet propulsion laboratory
LEO	low Earth orbit
LEPR	evict least probable first
LTP-T	linklider transmission protocol - transport
MANET	mobile ad hoc network
MACRE	message admission control based on rate estimation
MDC	message drop control
MDP	markov decision processes
MOFO	evict most forwarded first
MOPR	evict most favorably forwarded first
NACK	negative acknowledge
OIS	Orbital International Station
ONE	opportunitic network
OPNET	optimized network engineering tools
POI	point of interest
ProPHET	probabilistic routing protocol
RED	random early detection
RFC	request for comments
RF	radio frequence
RL	reinforcement learning
RRCC	retiring replicants congestion control
RTT	round-trip time
RWP	random way point
RW	random walk
TD	temporal difference
SARM	simulated annealing and regional movement
SDNV	self-delimiting numeric value

SHLI	evict shortest life time first
SPMBM	shortest path map-based movement
SR	storage routing
TCP	transmission control protocol
T-Drop	threshold drop
TRMG	transport modeling research group
TTL	time to live
UDP	user datagram protocol
URI	universal resource identifier
UWB	ultra wide band
WoLF	win or learn fast
WSN	wireless sensor network

Contents

1 Introduction

Delay and Disruption Tolerant Networks (DTNs) were initially motivated by the idea of deploying an Interplanetary Internet (IPN) (JPL-NASA, 1998) for deep space communication. As a result, a framework for an IPN which aims to use an interplanetary backbone to connect internetworks in space was developed. Over time, a diverse set of other DTN applications for "extreme" environments on Earth have emerged including vehicular networks (GERLA; KLEINROCK, 2011), emergency response and military operations (STERBENZA *et al.*, 2010), surveillance (STERBENZA *et al.*, 2010), tracking and monitoring applications (LEE *et al.*, 2006), and bridging the digital divide (PELUSI *et al.*, 2006). In these applications, long delays are a consequence of the long distances and/or episodic connectivity which are characteristic of "extreme" environments.

The arbitrarily long delays and frequent connectivity disruptions that set DTNs apart from traditional networks imply that there is no guarantee that an end-to-end path between a given pair of nodes exists at a given point in time. Instead, nodes may connect and disconnect from the network over time due to a variety of factors such as mobility, wireless channel impairments, nodes being turned off or running out of power, etc. Consequently, in DTNs, the set of links connecting DTN nodes, also known as “contacts,” varies over time. This fundamental difference between DTNs and conventional networks results in a major paradigm shift in the design of core networking functions such as routing, forwarding, congestion and flow control.

The DTN architecture described in (FALL, 2003a) uses the so-called *store-carry-and-forward* paradigm, as opposed to the Internet’s *store-and-forward*, to deliver messages from source to destination. In *store-carry-and-forward*, nodes store incoming messages and forward them when transmission opportunities arise. Note that in traditional networks, nodes also store messages before forwarding them; however, the time scales at which data is stored locally while waiting to be forwarded are typically orders of magnitude smaller when compared to DTNs. Therefore, storage in *store-carry-and-forward* typically uses persistent storage which implies that DTN nodes need to be equipped accordingly.

According to the *store-carry-and-forward* paradigm, when a DTN node "encounters" another DTN node, it decides whether to forward messages it is carrying to the other node. Therefore, the concept of *links* in traditional networks (wired or wireless) is replaced with the no-

tion of *contacts*. In scenarios where these encounters are random, *store-carry-and-forward* is also referred to as *opportunistic forwarding*. On the other hand, when contacts are known a priori (e.g., in deep space communication applications), *store-carry-and-forward* is known as *scheduled forwarding*. Finally, there are scenarios where node encounters follow a probability distribution based on past history; in these cases, *store-carry-and-forward* is based on *probabilistic forwarding* (MATSUDA; TAKINE, 2008). Note that, since *contact times* are finite and may be arbitrarily short, a node may need to choose which messages to forward based on some priority; a node may also decide whether the new neighbor is a "good" candidate to carry its messages. A node's "fitness" as a relay for a particular message depends on several factors that can be dependent on the message's ultimate destination (e.g., how often that potential relay encounters the destination, etc.) and there are also factors that are destination-independent, for example, the relay's mobility patterns, its capabilities (e.g., storage, energy, etc.) (IBRAHIM, 2008) (SPYROPOULOS *et al.*, 2011).

The simplest DTN forwarding technique is called *epidemic forwarding* (MATSUDA; TAKINE, 2008) (VAHDAT; BECKER, 2000a) (HARRAS *et al.*, 2005) (LINDGREN; DORIA, 2008) (LINDGREN *et al.*, 2003b), which is to DTNs what *flooding* is to traditional networks. To address issues such as limited contact times and limited network and node resources, several variants of "pure" *epidemic forwarding* (HARRAS *et al.*, 2005) (SPYROPOULOS *et al.*, 2005b) (BURGESS *et al.*, 2006) have been proposed. For instance, before a node forwards its messages to another node upon contact, the two nodes perform an initial "handshake" in which they exchange a summary of the messages each one has. Then they only exchange messages that the other does not already carry. There are also a number of "controlled" epidemic variants that try to, implicitly or explicitly, limit the number of copies of the same message in the network.

The fact that in DTNs the existence of an end-to-end path between any pair of nodes at all times cannot be guaranteed raises fundamental challenges in end-to-end reliable data delivery. In DTNs, the Internet model of end-to-end reliability (as implemented by TCP) is not applicable. The DTN architecture proposed in (FALL, 2003a) replaces TCP's end-to-end reliability with *custody transfer*, which uses hop-by-hop acknowledgements. This confirm the correct receipt of messages between two directly connected nodes. Additionally, due to the inability to guarantee end-to-end connectivity at all times, functions based on the TCP/IP model such as congestion and flow control will not always work in DTNs. Instead, hop-by-hop control can be employed.

In this thesis, we present a novel congestion control framework for DTN. First we survey the state-of-the-art on DTN congestion control mechanisms. To this end, we propose a taxonomy to help (1) map the DTN congestion control design space and (2) compare existing DTN congestion control mechanisms. One important issue from our exploration of existing DTN congestion control techniques is that there is no "universal" congestion control mechanism that will be applicable to all DTN scenarios and applications. Furthermore, we concluded that DTN

congestion control should employ a hybrid of open-loop and closed-loop control so that nodes can also make congestion control decisions based on local information. That way they do not have to rely exclusively on the network to make congestion control decisions.

In this direction, we performed a quantitative study where we picked a representative subset of DTN congestion control mechanisms from the schemes we have classified using our taxonomy. More specifically, we conducted a study comparing the performance of these congestion control schemes in both IPN and terrestrial scenarios to exploring the feasibility of a universal congestion control mechanism. Our goal with this study was to understand the performance trade-offs raised by existing DTN congestion control mechanisms and how they behave in a wide range of DTN scenarios. We also explored the impact of different routing protocols and node mobility models and evaluated the different congestion control strategies using such performance metrics as average delivery ratio, average latency and overhead. As a result of this quantitative analysis we summarized some good design principles for congestion control schemes in DTN scenarios. Among these principles, we concluded that adopting a proactive or hybrid (proactive and reactive) approach may significantly improve performance. Thus the design of more intelligent, efficient, and routing-protocol-independent congestion control should be considered.

Motivated by the desire to have an in depth understanding of DTN congestion problem and based on our qualitative and quantitative results, we developed a simple, yet general mathematical framework to model congestion in DTNs based on percolation theory (BROADBENT; HAMMERSLEY, 1957). Our goal with the resulting modeling framework is to make possible to evaluate and validate existing and future DTN congestion control mechanisms taking into account the design principles previously pointed out. An important feature of the proposed percolation model is the fact that instead of requiring global knowledge about the whole network, it relies exclusively on local information, i.e., information related to a node and its neighbors. This has the effect of facilitating the evaluation of autonomous control. A key observation of formulating the DTN congestion problem as a percolation process is that it happens quite intuitively and the resulting percolation model is simple, general and easy to derive. As a result our model makes possible to evaluate DTN congestion control mechanisms in terms of how cost-effective they are in preventing/containing congestion. The intuition here is that existing congestion control schemes cannot perform well over a large range of DTN applications because they were designed for either a specific application or for a specific routing protocol. In addition, in regard to DTN features some of these mechanisms cannot be retrofitted to work in DTNs.

Henceforth, this thesis has conducted a comprehensive study that employs reinforcement learning which overcomes some of the limitations of existing DTN congestion control schemes. Specifically, we propose an approach where mobile nodes learn to do congestion control known its buffer occupancy ratio and a set of actions they can apply either when the congestion takes

place or the congestion is likely to happen. Figure 1.1 shows the main idea of our approach. Let's a DTN node with buffer overflow. This node can, for example, broadcast a beacon that inform it is not able to receive any message. This might avoid that others nodes to sending message. Consequently, reducing the number of dropped messages. To accomplish this, it would have to run intelligent programs that search the set of possible solutions (actions), and select the best, after analyzing the effect of various combinations. Simultaneously, it could use online machine learning programs that suggest which option(s) is better in the light of past experience. For truly intelligent behavior, the mobile nodes need to do congestion control simply avoiding buffer overflow and display the ability to adapt intelligently to their changing environments. Our technique can be easily combined with many existing schemes for local congestion control.

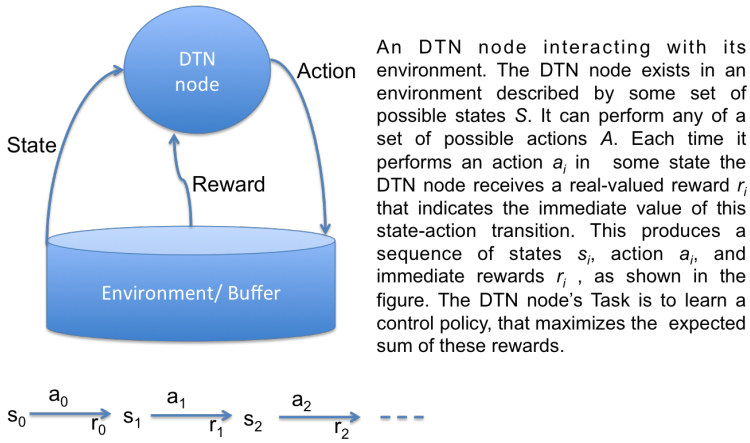


FIGURE 1.1 – DTN-Learning congestion aware framework

1.1 Motivation

Traditional DTN congestion control mechanisms fail either in the face of imperfect information or different applications. First, because the accurate and complete information that they need to make decisions is not available. Second, they are not adaptable to different DTN scenarios mainly because they were designed to a specific scenario with particular characteristics. On the other hand, machine learning often provides results with a high degree of accuracy once they have been trained adequately, even when the input data is incomplete. We believe that this approach is especially relevant to DTNs, where information is always delayed and often incomplete. To keep the network running at an acceptable efficiency, nodes in DTNs would need to assess the likelihood of congestion, by analyzing local information. In additon, machine learning techniques provide the ability to learn and to adapt to the dynamic environments. In other words, since the DTN environment is unknown and dynamic, the DTN node has to be able to adapt or learn its skills in order to react adequately to the environment. Congestion avoidance is a basic and fundamental task of autonomous DTN nodes. It is particularly critical in extreme

environments where an end-to-end communication is not always possible to reach a goal. Furthermore, the need to minimize human intervention in some extreme environments requires the ability to learn from both successes and failures, and evolve mechanisms to diagnose and fix problems in the network (CLARK *et al.*, 2003).

One good method to perform this capability of adaptive behavior is reinforcement learning because it does not require knowledge from the environment and enables continuous online learning. This learning system basically permits the DTN node to learn from its mistakes. The DTN node gets inputs from the environment, for instance, monitoring its buffer. This information represents a state used by the reasoning process to determine the right action to take. Then reasoning process receives a reinforcement signal measuring how the action performs with respect to the environment and what the consequences are. This feedback rewards the action when it matches with the objectives. The goal of the learning is to choose the action that maximizes this reward feedback.

The issues pointed out before strongly support the choice for machine learning (reinforcement learning) based approach to congestion control.

1.2 Problem Statement

The challenges of controlling congestion in DTNs are mainly due to two reasons: (1) episodic connectivity, i.e., end-to-end connectivity between nodes cannot be guaranteed at all times and (2) communication latencies can be arbitrarily long caused by high propagation delays and/or intermittent connectivity. Consequently, traditional congestion control does not apply to DTN environments. A notable example is TCP's congestion control mechanism which is not suitable to operate over a path characterized by extremely long propagation delays, particularly if the path contains intermittent links. Basically, TCP communication requires that the sender and the receiver negotiate an end-to-end connection that will regulate the flow of data based on the capacity of the receiver and the network. Establishment of a TCP connection typically takes at least one round-trip time (RTT) before any application data can flow. If transmission latency exceeds the duration of the communication opportunity, no data will be transmitted (BURLEIGH *et al.*, 2003a) (FARRELL *et al.*, 2006). Furthermore, there is a two-minute timeout implemented in most TCP stacks: if no data is sent or received for two minutes, the connection breaks. However, in some DTNs RTTs can be much longer than two minutes.

For example, in an interplanetary network supporting Earth-Mars communication (as illustrated in Figure 3.1), the RTT is around 8 minutes when both planets are closest to one another, with a worst-case RTT of approximately 40 minutes. In this scenario, the terrestrial satellite is connected to an Orbital International Station (OIS) in orbit around the Sun, which in turn connects to a Martian satellite and a space probe. Additionally, there is a Martian terminal con-

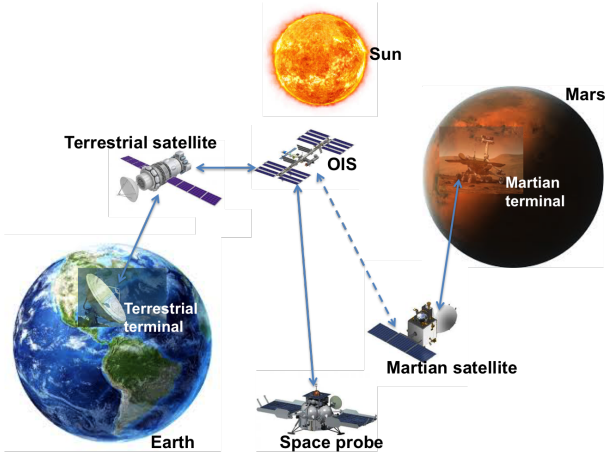


FIGURE 1.2 – Deep space communication scenario

nected to the Martian Satellite. The link between the OIS and the Martian Satellite is interrupted whenever the planet Mars is between the OIS and the orbiting satellite, as well as whenever the Sun is between Mars and the OIS. Therefore, traffic on the "link" between the Terrestrial and Martian satellites may need to be buffered at the OIS for long and varying periods of time. If the OIS becomes heavily congested, it will significantly hamper communication between the Terrestrial satellite and the space probe.

Alternatives such as UDP (User Datagram Protocol) or DCCP (Datagram Congestion Control Protocol) (KONLER *et al.*, 2006) are not generally appropriate because they offer limited reliability and, in the case of UDP, no congestion control. Consequently, efficient techniques are thus needed to effectively control congestion in DTNs so that network utilization is maximized.

In this thesis we investigate the congestion problem in delay tolerant networks and describe a novel congestion control strategy. More specifically we investigate the benefits and cost of congestion control techniques based on reactive and proactive control policy, open-loop approach and an implicit scheme of control.

Fall and Farrel (FALL, 2003b) emphasize the research problem as it follows: "Congestion management is an issue that has been raised since earliest days of the IPN and DTN designs, yet has received relatively little attention. Perhaps this situation is the result of insufficient use (there is no significant congestion on links with low durty cycle) or perhaps this problem is so formidable that solutions remain elusive. Without the conventional type of low-latency feedback available in ordinary networks, congestion management and control can be exceedingly difficult.” It is possible to observe that after several years of design, the behavior of DTN congestion remains misunderstood until the DTN architecture is more widely deployed and carries significant traffic loads. It is clear that congestion is a prominent problem that needs to be addressed.

1.3 Research Goals

Our research aims to develop an efficient DTN congestion control framework applicable to different types of DTN applications and scenarios. Our approach is based on using computational intelligence techniques to learn the conditions in which the network operates and adjust congestion control accordingly. More specifically, we have been using reinforcement learning which tries to achieve an optimal, or close-to-optimal solution when the system offers alternate choices. It has attracted considerable attention because it provides an effective approach for problems in which optimal solutions are analytically unavailable or difficult to obtain. Reinforcement learning is based on the idea that if an action is followed by a satisfactory state (e.g., performance improvement), then the tendency to produce that action is strengthened (reinforced). On the other hand, if the state becomes unsatisfactory, then that particular action is suitably penalized.

We have been using Q-Learning, a variant of reinforcement learning, which is characterized by having the learning agent first learn a model of the system on-line and then utilize this knowledge to find a satisfactory solution. One of our main design goals is to base congestion control decisions on information available locally at nodes (from the node itself and its neighboring nodes), since global knowledge is both difficult and expensive to acquire in DTN environments. However, if global information is available, it can also be considered. According to our proposed approach, a node stores a table of "Q-values" that estimates the quality of alternate control strategies. These values are updated each time conditions (such as buffer occupancy, drop ratio, local congestion) change and are used to select future congestion control alternatives. Examples of control actions include increase/decrease node message generation rate, discard buffered messages (e.g., based on their age), broadcast locally congestion notification messages and migrate messages to neighboring nodes.

To the best of our knowledge, this thesis is the first to propose a DTN congestion control framework that uses computational intelligence to adapt to different DTN applications and environments dynamically in order to deliver adequate performance. Our framework was able to operate well in scenarios ranging from "opportunistic" node encounters, i.e., where node encounters are random to scheduled encounters, i.e., where contact between nodes are known a priori, which is the case of some IPN scenarios.

1.4 Contributions

Henceforth, in light of the aforementioned observations and limitations of existing congestion control works, this thesis makes the following contributions:

- The definition of a taxonomy to map the DTN congestion control design space and use it to classify existing DTN congestion control mechanisms.
- An DTN congestion problem modeling based on percolation theory (GRIMMETT, 1999) (BROADBENT; HAMMERSLEY, 1957) in order to archieve the probability of delivering a message between a given source-destination pair in such way that the path contains only non-congested nodes (nodes that contains available buffer space for new arriving messages) and active contact opportunity (when a connection exists between two nodes) exist in the network.
- A congestion aware framework based on Machine Learning, in particular Reinforcement Learning that allows the DTN node to adapt their behavior in different environments, takes actions in it in order to mitigate congestion, and receive numeric rewards and punishments from some reward function based on the consequences of the actions it takes. By trial-and-error, the DTN node learns to take actions that maximize its rewards and consequently the network performance.

1.5 Publications

This thesis has resulted in the following papers:

- Aloizio P. Silva, Scott Burleigh, Celso M. Hirata, and K. Obraczka,"A Survey on Congestion Control for Delay and disruption Tolerant Networks", Journal of Elsevier Ad Hoc Networks. August 2014.
- Aloizio P. Silva, Scott Burleigh, Celso M. Hirata, and K. Obraczka,"Congestion Control in Disruption-Tolerant Network: A Comparative Study for Interplanetary Networking Applications", ACM CHANTS 2014, Maui, Hawaii, USA, September 2014.
- Aloizio P. Silva, Scott Burleigh, Celso M. Hirata, and K. Obraczka,"Congestion Control in Disruption-Tolerant Networks: A Comparative Study for Interplanetary and Terrestrial Networking Applications", Journal of Elsevier Ad Hoc Networks, March 2015. Under review.
- Aloizio P. Silva, Scott Burleigh, Celso M. Hirata, and K. Obraczka,"Congestion Control in Disruption-Tolerant Network: A Performance Study", MOBIQUITOUS 2015, July 2015. Under review.
- Aloizio P. Silva, Celso M. Hirata, Marcelo R. Hilario and K. Obraczka,"A Percolation-Based Approach to Model DTN Congestion Control", IEEE MASS 2015, October 2015. Under review.

1.6 Thesis Structure

The remainder of this thesis is structured as follows.

Chapter 2. This chapter presents an overview about Delay and Disruption Tolerant Networks. Specifically, it provides a background of DTN architecture and highlight DNT characteristics.

Chapter 3. This chapter reports a survey of the related literature surrounding the focus of this thesis, specifically: congestion control methods for traditional networks and delay tolerant congestion control schemes. In addition, it proposes a taxonomy to classify DTN congestion control mechanisms.

Chapter 4. This chapter presents the analysis of four DTN congestion control mechanisms in two scenarios: terrestrial and interplanetary networks. Specifically, it provides an extensive quantitative comparison of all mechanisms using the ONE simulator, highlighting their experimental setup and outline their deficiencies in terms of design and research methodology.

Chapter 5. This chapter presents a novel DTN congestion problem modeling using percolation theory. In particular, it presents an overview of percolation theory, a detailed description of our proposed model and highlights an experimental analysis using MATLAB and the ONE simulator.

Chapter 6. This chapter presents a novel congestion control mechanism for DTNs. Specifically, this chapter proposes DTN-Learning Congestion Aware Mechanism, an adaptive and autonomous congestion control mechanism based on machine learning. More specifically, reinforcement learning that has the ability to learn online how to behave.

Chapter 7. This chapter concludes the thesis, and provides a summary of research outcomes and future research directions.

2 Delay and Disruption Tolerant Networks

In the last few years, DTN have grown from shy research activities to a visionary research topic attracting both network designers and application developers. DTN is now a recognized area in networking research, due in part to practical experiences with mobile ad hoc network (MANETs) (BASAGNI *et al.*, 2005) that are required to operate in situations where continuous end-to-end connectivity may not be possible.

At its beginning, the concepts behind the DTN architecture were primarily targeted at tolerantig longs delays and predictably interrupted communications over long distances (i.e. in deep space). At this point in time, the work was an architecture for the IPN. By march 2003, when the first draft of the eventual RFC 4838 was published, the term DTN was coined with the intention to extend the IPN concept to other types of networks, specifically including terrestrial wireless networks.

In this chapter we present the necessary background of DTN. We reaffirm that there are many benefits in using DTN ideas, for example, in places that suffer from disaster or where normal communications are destroyed (GRAVER; TIPPER, 2005).

2.1 DTN Architecture

The DTN architecture, which was originally proposed in (BURLEIGH *et al.*, 2003a), aims at providing implementations for reliable message delivery in intermittently-connected networks. It introduces the *store-carry-and-forwarding* paradigm under which messages may remain stored for relatively long periods of time in persistent storage at routers while in transit from source to destination. The DTN architecture was designed to operate as an intermediate layer, called the *bundle layer*, between the application and the transport layers of the networks it interconnects (see Figure 2.1). It provides services such as in-network data storage and retransmission, interoperable naming, authenticated forwarding, and coarse-grained classes of service.

The DTN architecture also specifies the *Bundle Protocol* (BURLEIGH *et al.*, 2003a; CERF *et al.*, 2007a; SCOTH; BURLEIGH, 2007; FALL; FARRELL, 2008) which controls the exchange of

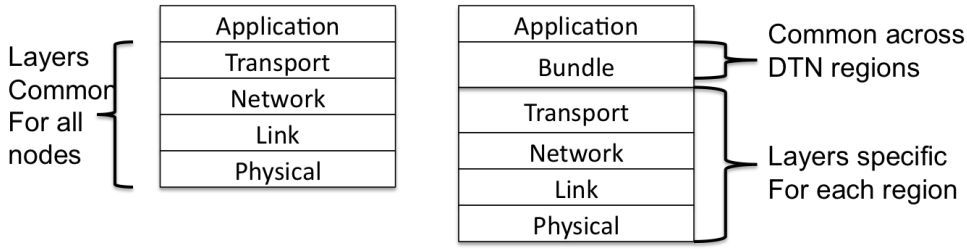


FIGURE 2.1 – TCP/IP stack and DTN stack.

bundles[1] , i.e., application-layer messages. It receives messages from the application layer, encapsulates them into bundles, whose format is depicted in Figure 2.2, and then forwards them to the next-hop DTN node. The final destinations of bundles are "endpoints" associated with nodes and identified by EIDs (End Point Identifiers).

As previously pointed out, since end-to-end paths cannot be guaranteed, a DTN route consists of a series of time-dependent *contacts*, i.e., communication links that are established whenever nodes come in range of one others. Contacts may be parameterized by their duration, capacity, latency, end points and direction. Also due to the inability to guarantee end-to-end routes, reliability is achieved hop-by-hop using *custody transfer*. A node taking custody of a message commits to deliver that message to its destination or another node that accepts the message's custody

The communication characteristics are relatively homogeneous in a DTN communication region. The wireless DTN technologies may be diverse, including not only radio frequence (RF) but also ultra wide band (UWB), free space optical, and acoustic technologies.

Each region has a unique region ID which is known among all regions of the DTN. DTN gateways have membership in two or more regions and are the only means of moving messages between regions.

Region ID uses the same name space syntax as the Internet's DNS (Domain Name System). Each name has a two part name, consisting of a region ID and an entity ID. Routing between regions is based only on region ID while routing within a region is based only on entity ID.

2.2 Bundle Layer

The unit of information exchange in a DTN is a bundle. The bundle layer receives messages from the application layer encapsulates them into bundles, whose format is depicted in Figure 2.2, and then forwards them to the next-hop DTN node. Nodes are identified by EIDs (End

[1]The term "bundle" was chosen to connote the self-sufficiency of the messages: application-layer messages are expected to contain sufficient metadata to enable processing by the recipient without negotiation, as if all relevant metadata query and response messages have been anticipated by the sender and "bundled" into a single application data unit.

point IDentifiers). A DTN node is an entity with a bundle layer. A node may be a host, router, or a gateway (or some combination) acting as a source, destination, or forwarder of bundles. A router forwards bundles within a single DTN region while a gateway forwards bundles between DTN regions. Figure 2.3 shows the differences between various kinds of DTN nodes.

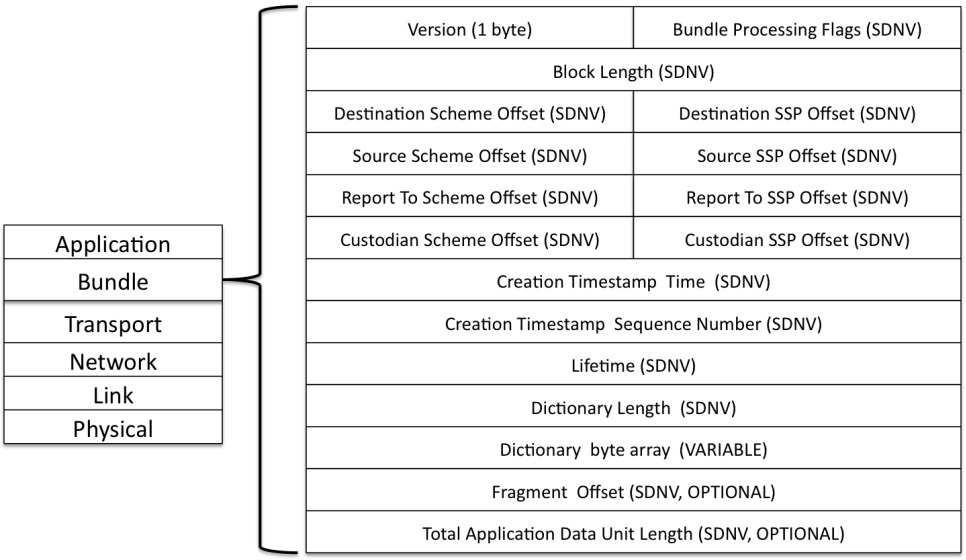


FIGURE 2.2 – DTN protocol stack and the structure of primary block of a bundle.

The bundle protocol is defined in RFC 5050 and implements a bundle layer in the DTN architecture defined in RFC 4838. The bundle protocol provides three priority classes that guide the routing and scheduling algorithms (bulk, normal and expedited). Delivery options, such as, custody transfer request, report when bundle delivered, deleted or forwarded, confidentiality, authetication and error detection request are supported by the architecture which gives the applications more flexibility. Interplanetary Overlay Network (ION) is an implementation of the DTN Bundle Protocol that is intended to be usable for interplanetary communications. The space flight specific constraints on this implementation are discussed in (BURLEIGH, 2007). DTN2 (DEMMER *et al.*, 2004)is another reference implementation of the Bundle Protocol by the Delay Tolerant Networking Research Group (DTNRG) featuring a flexible TCL console and XML inter- faces to external components.

In DTN, forwarding nodes can be authenticated. So, the sender information is authenticated by forwarding nodes, so that network resources can be saved by preventing the carriage of prohibited traffic at the earliest opportunity.

The unique characteristic of the bundle layer is the support for in-transit storage. Bundles received from a sender can be stored in a intermediate node for an excessive amount of time (minutes, hours, or even days). These store operations are performed by the network stack, at the bundle layer, transparently to the application. The in-transit storage is the means to: (i) overcome the delays and disruptions induced while a bundle moves hop-by-hop to its destination, (ii) avoid costly end-to-end retransmissions due to errors or timeout, and (iii) allow exchange of

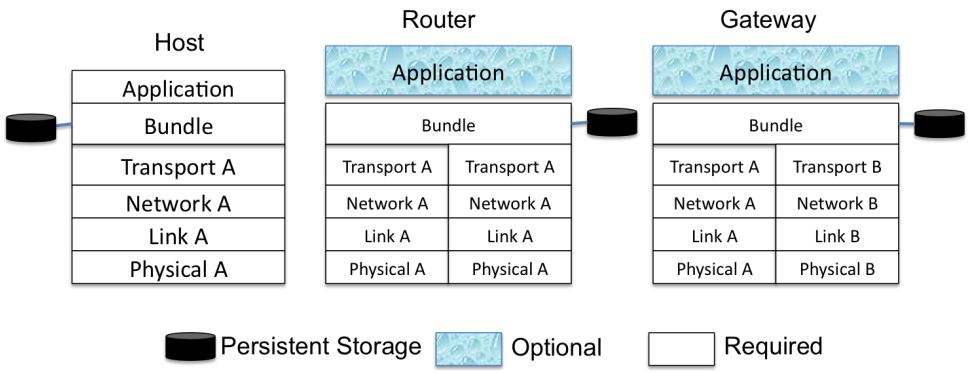


FIGURE 2.3 – Different DTN nodes structures

information between two nodes that share no end-to-end communication path at any given time moment. The bundle protocol defines a custody operation which allows an intermediate node to take on the responsability to handle bundle delivery to destination on behalf of a sender.

The bundle protocol tries to use as minimum bandwidth as possible while transmiting. This has been accomplished with the help of Self-Delimiting Numeric Values (SDNV) encoding technique. In this technique any positive numeric value is encoded into N octets, the Most Significant Bit (MSB) of the last octet is set to 0 while all the other octets have their MSBs as 1. The other 7 bits of every octet contain relevant information.

Figure 2.4 shows an implementation architecture for DTN where a central bundle forwarder, which can be the Bundle Protocol Agent (BPA) of a node to forwarder bundles (based on routing algorithm decisions) to the Convergence Layer Adapter (CLA), storage or local application. The arrows represent interfaces through which the bundle forwarder interacts with the applications, CLAs and management processes. Implementing these interfaces using inter-process communication rather than normal procedure calls has been quite beneficial for the development of the architecture. These interfaces carry bundles or directives that are represented as tiny green and yellow boxes respectively. The native Internet protocols provide different semantics that is not helpful to the DTN architecture. It is the task of a group of protocol CLAs to provide the necessary funcionalities required to carry the bundles on each of the required protocols (FALL, 2003b).

2.3 Naming, Addressing and Binding

Naming and addressing are some of the most fundamental aspects of a network architecture and one of the most tricky aspects to get right. Generally, naming has been thought of as something useful to people or organizations while addressing is more useful for network operations and routing. Names are generally expected to be variable length strings while addresses are expected to be fixed length identifiers. Some form of mapping or binding function is used

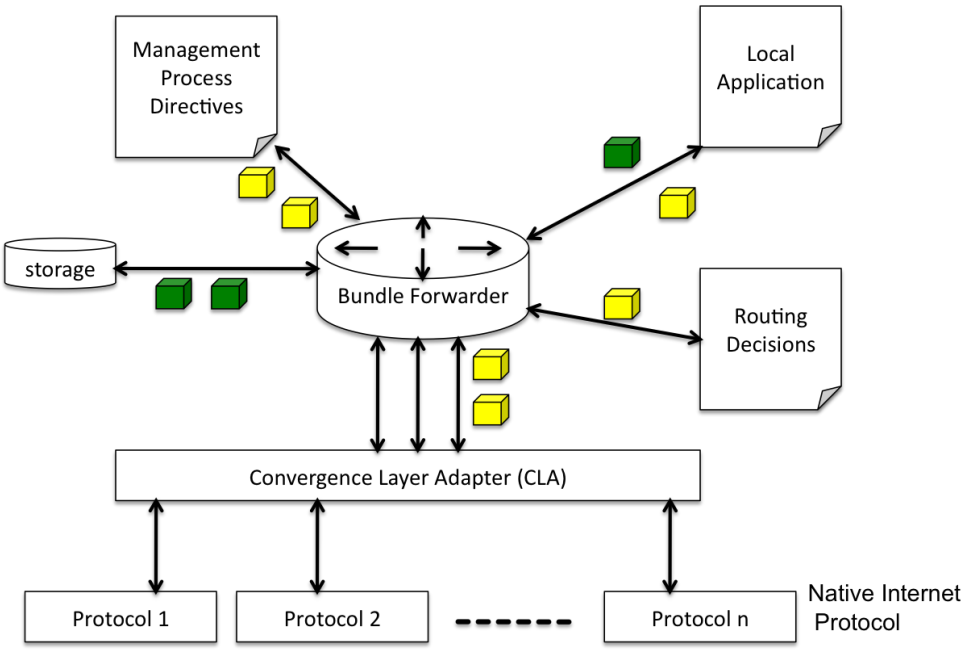


FIGURE 2.4 – An example of DTN implementation architecture: the architecture shows how bundles fowarder interactions. This is a modified version of the figure that appears in (MUKHERJEE, 2012).

to convert names into addresses. In the evolution of the DTN architecture, nodes have always had identifiers. These are used in the context of the bundle protocol (SCOTH; BURLEIGH, 2007), which provides the basic message delivery service for DTN. Originally, identifiers in the bundle protocol were constructed as a 3-tuple of the form (region, host, application) which was able to not only identify a host but also an application of interest on the host.

In recognizing that nodes may require multiple identifiers and even multiple types of identifiers, a naming structure was sought that is capable of encoding names or address from multiple different name spaces. Fortunately, work in the IETF (Internet Engineering Task Force) had already been accomplished in the area of generalized naming systems in the form of Universal Resource Identifiers (URIs) (LEE *et al.*, 2005). URIs as used in DTN are referred to as endpoint identifiers (EIDs). The addressing format for a DTN next-hop need not be of the same scheme as that of the source or destination in the bundle. This is in contrast to Internet routing entries, where next-hops are generally expressed using the same address format.

For a message containing DTN URIs comprising symbolic names, some binding step is performed by one or more nodes along the delivery path, such binding may be performed anywhere along the delivery path. In the Internet, this happens at multiple layers and at multiple locations, when DNS is invoked at a sending node, this is a form of early binding, which is used immediately in mapping a DNS name to a IP address. Subsequent mappings are performed on the IP address in delivering its containing packet toward its destination.

DTN supports direct forwarding based on symbolic names, so the early binding typical

of DNS in the Internet is not generally required. Instead, message are passed along toward their destinations based on fowarding entries present at DTN routing nodes that match against the name. This is known in the DTN literature as late binding. DTN supports late and early binding, depending on the scheme used.

2.4 Different Types of Contact

As mentioned previously in the *store-carry-and-forward* paradigm, when a DTN node "encounters" another DTN node, it decides whether to forward messages it is carrying to the other node. Therefore, the concept of *links* in traditional networks (wired or wireless) is replaced with the notion of *contacts*. To date, the following major types of contacts have been defined by (CERF *et al.*, 2007b).

- Scheduled Contacts: all information about the encounter are known. Specifically, there is an agreement to establish a contact at a particular time, for a particular duration. An example of a scheduled contact is a link with a low-earth orbiting satellite.
- Opportunistic Contacts: are encounters that present themselves unexpectedly. For example, an unscheduled aircraft flying overhead and beaconing, advertising its availability for communication, would present an opportunistic contact.
- Probabilistic Contacts: statistics about encounters are known. These contacts are based on no fixed schedule, but rather are predictions of likely contact times and durations based on a history of previously observed contacts or some other information.

Given a knowledge over different types of contacts, routes may be chosen based on this information. In the next section we give a briefly overview about DTN routing.

2.5 Routing

DTNs present many challenges that are not present in traditional networks. Many originate from the need to deal with disconnections, which directly impacts routing and forwarding. Sucessful data delivery in challenging environments can be exceptionally difficult or impossible. Intermittent connectivity is a typical characteristic of such environments, while even in case of predefined contacts there are imponderable factors such as physical phenomena. Moreover network and node resources are also limited, while topology information may be incomplete or obsolete. A category of routing algorithms attempts to compensate the lack of knowledge by spreading multiples copies of a message to the network, namely flooding algorithms. The basic

protocols in this category do not need any information about the network. A number of DTN routing algorithms can be found in (ZHANG, 2006) (JAIN *et al.*, 2004a) (JONES; WARD, 2006) for more details.

The DTN architecture claims applicability to a wide range of operating environments, and was intended to support pluralism between the naming formats, routing algorithms and network technologies. The routing problem can be coarsely divided into whether the routing graph is assumed to be connected or not, with DTN typically aiming at the latter. In addition, methods for routing bundles may involve creation and deletion of single or multiple copies of a bundle, various degree of knowledge about the topology and traffic pattern (e.g. past, current, and future contact, traffic load, and buffer occupancy), fragmentation, various levels of granularity in decision making, resource reservations, different routing for different class of service or custody bundles, and different options for the local of routing computation.

It is important to note that DTN concepts are also related to the MANET (Mobile Ad hoc Network) (BASAGNI *et al.*, 2005) literature, which has to date focused largely on routing in relatively dense mobile ad-hoc networks where end-to-end connectivity between any pair of nodes is possible. Combining DTN and MANET concepts together suggests that nodes may operate using some combination of simple fowarding and more delay tolerant store-carry-forward operation. Some such networks go a step further and couple the routing system to node control systems. This coupling supports the ability to beneficially place nodes to act as routers or data ferries to enable communication even in otherwise sparse and partitioned networks. The DTN architecture and bundle protocol do not restrict the type of routing that may be used in any particular operating environment.

DTN routing may eventually involve not only path or next-hop selection toward a destination using a single metric of goodness, but also possibly multiple routing solutions depending on the types of bundles being moved. In addition, DTN routing selects both next-hops at each forwarding node and the next protocol. Thus, a routing solution may involve not only a set of paths, but also a set of appropriate encapsulating protocols used to facilitate delivery using a heterogeneous set of transports.

2.6 Conclusion

This chapter has presented an overview about DTN architecture. We introduced the basic idea about DTN and its main features. In the end, DTN is a technology that provides network services for extreme environments that no end-to-end path exists through a network. More specifically, DTN based network systems are ideal for Interplaneatry Networks which allow for long distance communication capabilities where a continuous end-to-end link is not sustainable. Other places where DTN can be used include spacecraft, military/tactical, some forms

of disaster response, underwater, and some form of ad-hoc sensor networks. It may also include Internet connectivity in places where performance may suffer such as developing regions in the world. Development of DTN network in these environments must rely on the perfect congestion control mechanism to ensure reliability, stability, and extensiveness of the network. As some existing constraint condition of DTN network, so its congestion control problem is different from Internet. Therefore, in view of DTN features, to study new congestion control is important. At the same time, there are more researchers studying DTN in the last years and DTN congestion control schemes have been proposed. In the next chapter we survey the state-of-the-art on DTN congestion control mechanisms. To this end, we propose a taxonomy to help (1) map the DTN congestion control design space and (2) compare existing DTN congestion control mechanisms.

3 DTN Congestion Control

Delay and disruption tolerant networks (DTNs) may experience frequent and long-lived connectivity disruptions. Unlike traditional networks, such as the TCP/IP-based Internet, DTNs are often subject to high latency caused by very long propagation delays (e.g., interplanetary communication) and/or intermittent connectivity. Another feature that sets DTNs apart from conventional networks is that there is no guarantee of end-to-end connectivity between source and destination. Such distinct features pose a number of technical challenges in designing core network functions such as routing and congestion control.

In this chapter, we survey the state-of-the-art on DTN congestion control mechanisms. To this end, we propose a taxonomy to help (1) map the DTN congestion control design space and (2) compare existing DTN congestion control mechanisms. The survey presented in (SOELISTIJANTO; HOWARTH, 2013) considers reliability and congestion control proposals focusing on opportunistic networks. Note that opportunistic networks are a special case of DTNs where contacts between nodes are not known a priori. In our survey, we consider DTNs as more broadly defined: in addition to opportunistic networks, i.e., networks where contacts are random, we also explore networks in which contacts are scheduled well as networks in which contacts are probabilistic (based on some probability function derived from past contacts). The tutorial presented in (CAINI *et al.*, 2011) discusses the prospects of using DTN in future satellite networks, in particular LEO (Low Earth Orbit)/GEO (GEOsynchronous) satellite constellations. Studies such as (PSARAS *et al.*, 2009) and (VOYIATZIS, 2012) confirm that congestion control is a fundamental issue in DTNs and note that it has not received much attention from the DTN research community. Our work presented in this chapter goes a step further and provides a deeper analysis of existing DTN congestion control mechanisms.

3.1 Internet Congestion Control

The Internet is a worldwide computer network that transmits data by packet switching based on the TCP/IP suite. The Internet architecture defines two transport-layer protocols for data transmission end-to-end, i.e., between two communicating processes running on two different hosts connected to the Internet. The first one is UDP (User Datagram Protocol), which provides

an unreliable data delivery service. In other words, the different pieces of the same application message may arrive out of order, appear duplicated, or go missing without notice. UDP does not provide congestion– or flow control capabilities. The other Internet transport protocol is TCP (Transmission Control Protocol). TCP provides reliable, ordered delivery of a stream of bytes between two communicating processes.

In addition to reliable delivery, TCP also performs flow and congestion control. At this point, we should make clear the difference between flow and congestion control. Flow control is "all about the receiver", i.e., it tries to ensure that the sender does not outpace the receiver, sending data faster than the receiver can receive. On the other hand, congestion control is "all about the network" making sure that the sender does not send more data than the network can handle.

Therefore, congestion occurs when resource demands from users/applications exceed the network's *available capacity*. The need for congestion control on the Internet surfaced in 1986 when the Advanced Research and Projects Agency Network (ARPANET), the precursor to the Internet, suffered congestion collapse (JACOBSON, 1988). Congestion collapse generally occurs at choke points in the network, where the total incoming traffic to a node exceeds the outgoing bandwidth. As described in (WYDROWSKI, 2003), there are two fundamental approaches to the problem of controlling Internet congestion, namely: capacity provisioning and load control.

The **capacity provisioning** approach is based on ensuring that there is enough capacity in the network to meet the offered load. On the other hand, the **load control** approach ensures that the offered load does not exceed the capacity of the network. The latter approach inspired the development of the first Internet congestion control algorithm (SRIKANT, 2004). The basic idea behind the algorithm was to detect congestion in the network using *packet loss* as congestion indicator. Upon detecting a packet loss, the source reduces its transmission rate; otherwise, it increases it.

According to TRMG (Transport Modeling Research Group) (METRICS..., 2006) (FLOYD, 2008), performance metrics that can be used to evaluate Internet congestion control protocols include:

- **Convergence speed**: estimates time to reach the equilibrium, i.e., states how much time elapsed between the moment that the congestion was detected and the moment that congestion ceased to exist.
- **Smoothness**: is defined as the largest reduction in the sending rate in one RTT (Round-Trip Time). In addition, it reflects the magnitude of the oscillations through multiplicative reduction, which is the way TCP reduces its transmission rate.
- **Responsiveness**: is defined as the number of RTTs of sustained congestion required for the sender to halve the sending rate.

- **Fairness**: specify the fair allocation of resource between the flows in a shared bottleneck link.
- **Throughput**: characterizes the transmission rate of a link or flow typically in bits per second. Most congestion control mechanisms try to maximize throughput, subject to application demand and constraints imposed by the other metrics (network-based metric, flow-based metric and user-based metrics).
- **Delay**: can be defined as the queue delay over time or in terms of per-packet transfer times.
- **Packet loss rates**: measures the number of packets lost divided by total packets transmitted. Another related metric is the loss event rate, where a loss event consists of one or more lost packets in one round-trip time (RTT).

Some of the metrics discussed above could have different interpretations depending on whether they refer to the network, a flow, or a user. For instance, throughput can be measured as a network-based metric of aggregate link throughput, as a flow-based metric of per connection transfer times and as user-based utility metric.

These metrics were originally proposed for the Internet and in (MAMATAS *et al.*, 2007), they have been used to provide a categorized description of different congestion control strategies in packet networks using network-awareness level as a criterion. While some of them can still be used for DTNs, new metrics are needed. For instance, in DTNs, queueing delays are expected because of the high latencies and intermittent connectivity. Furthermore, paths are lossy, so losses do not necessarily indicate congestion as assumed in TCP. In Section 3.5, we discuss metrics employed by DTN congestion control protocols as well as metrics to measure their performance.

3.2 Congestion Control in DTN: Challenges

As previously pointed out, the challenges of controlling congestion in DTNs are mainly due to two reasons: (1) episodic connectivity, i.e., end-to-end connectivity between nodes cannot be guaranteed at all times, and (2) communication latencies can be arbitrarily long caused by high propagation delays and/or intermittent connectivity. Consequently, traditional congestion control does not apply to DTN environments. A notable example is TCP's congestion control mechanism which is not suitable to operate over a path characterized by extremely long propagation delays, particularly if the path contains intermittent links. Basically, TCP communication requires that the sender and the receiver negotiate an end-to-end connection that will regulate the flow of data based on the capacity of the receiver and the network. Establishment

of a TCP connection typically takes at least one round-trip time (RTT) before any application data can flow. If transmission latency exceeds the duration of the communication opportunity, no data will be transmitted (BURLEIGH *et al.*, 2003a) (FARRELL *et al.*, 2006). Furthermore, there is a two-minute timeout implemented in most TCP stacks: if no data is sent or received for two minutes, the connection breaks. However, in some DTNs RTTs can be much longer than two minutes.

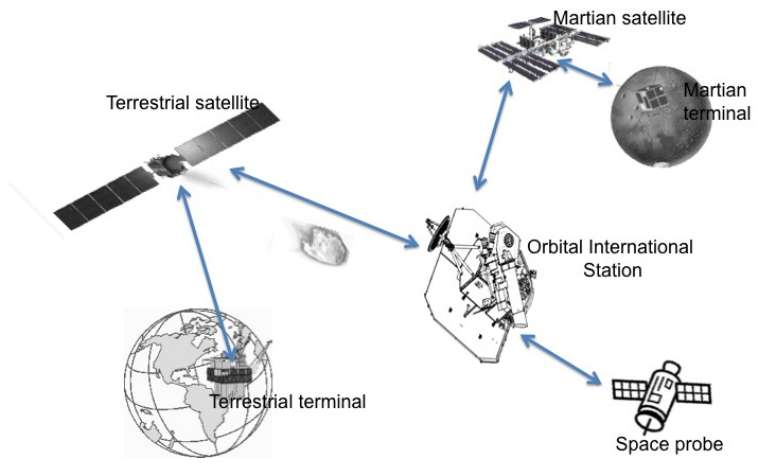


FIGURE 3.1 – DTN connection example: deep space communication scenario

For example, in an interplanetary network supporting Earth-Mars communication (as illustrated in Figure 3.1), the RTT is around 8 minutes when both planets are closest to one another, with a worst-case RTT of approximately 40 minutes. In this scenario, the terrestrial satellite is connected to an Orbital International Station (OIS) in orbit around the Sun, which in turn connects to a Martian satellite and a space probe. Additionally, there is a Martian terminal connected to the Martian Satellite. The link between the OIS and the Martian Satellite is interrupted whenever the planet Mars is between the OIS and the orbiting satellite, as well as whenever the Sun is between Mars and the OIS. Therefore, traffic on the "link" between the Terrestrial and Martian satellites may need to be buffered at the OIS for long and varying periods of time. If the OIS becomes heavily congested, it will significantly hamper communication between the Terrestrial satellite and the space probe.

Alternatives such as UDP or DCCP (Datagram Congestion Control Protocol (KONLER *et al.*, 2006)) are not generally appropriate because they offer limited reliability and, in the case of UDP, no congestion control. Consequently, efficient techniques are thus needed to effectively control congestion in DTNs so that network utilization is maximized.

3.3 DTN Congestion Control Taxonomy

One of the contributions of this survey is to propose a taxonomy to classify DTN congestion control mechanisms. The proposed taxonomy (see Figure 3.2) maps the DTN congestion

control design space and uses it as backdrop to put existing DTN congestion control techniques in perspective. We also discuss possible directions for future exploration of efficient congestion control for a range of DTN applications and scenarios. In this section, we present the criteria underpinning the proposed DTN congestion control taxonomy.

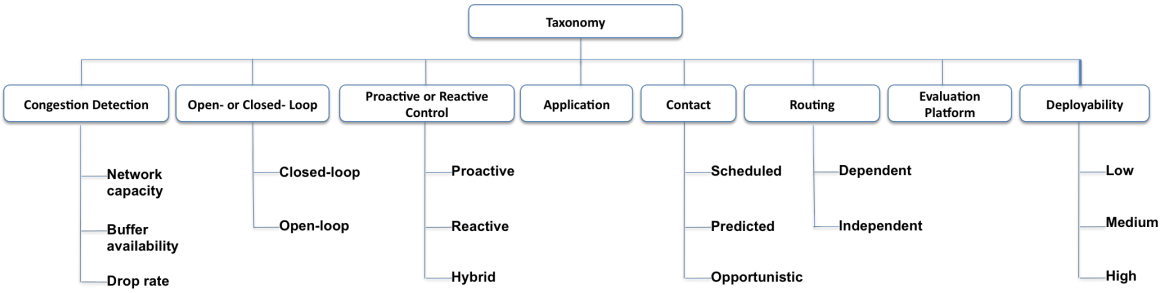


FIGURE 3.2 – Overview of the proposed taxonomy

3.3.1 Congestion Detection

How is congestion detected? In general, congestion occurs when resource demands exceed available capacity. Consequently, congestion detection can consider: *network capacity*, *buffer availability*, and *drop rate*.

- *Network capacity*: mechanisms that use network capacity to gauge congestion try to assess if the traffic arriving at the network is greater than the traffic that could be supported by network.
- *Buffer availability*: some DTN congestion detection strategies check whether the storage capacity at nodes is filling up as new messages are received.
- *Drop rate*: the fact that the drop rate goes beyond a certain threshold is used by certain mechanisms as a way to detect congestion.

3.3.2 Open- versus Closed-Loop Control

Another way to classify DTN congestion control mechanisms is based on whether they employ *open loop*– or *closed loop* control. In open-loop congestion control, end systems do not rely on feedback from the network; instead they try to "negotiate" their sending rate a priori which can be considered a congestion prevention approach. Closed-loop (or *reactive* control) mechanisms utilize feedback from the network to the end systems. Network feedback usually contains information about current conditions. End systems typically respond to congestion build-up by reducing the traffic load they generate (AHMAD *et al.*, 2009). Notable examples of open-loop congestion control mechanisms include the ones based on buffer management, e.g.,

dropping messages according to a variety of policies. Some of these techniques use classical drop policies (i.e. drop-random and drop-tail) which were inherited from traditional networks. In section 3.4.2 below, we list and discuss some of these classical policies as well as drop policies which have been proposed specifically for DTNs (i.e., E-Drop and Mean-drop).

3.3.3 Proactive versus Reactive Control

Congestion control mechanisms can also be classified as *proactive* or *reactive*. Proactive congestion control (also known as congestion avoidance) schemes take a preventive approach and try to prevent congestion from happening in the first place; in reactive congestion control, end systems typically wait for congestion to manifest itself (e.g., router queue build-up, packet loss, etc.) before any action is taken.

Because DTN systems may exhibit long delays, reactive congestion control may not be sufficient. Proactive congestion control or hybrid approaches combining proactive– and reactive control are interesting alternatives.

We should also point out that, at first glance, proactive– and open-loop control may seem to subsume one another. However, a closer look reveals that approaches based on open-loop control can be reactive. For instance, we can have a mechanism that is triggered solely based on the size of the router's queue, and therefore is open-loop. However, it can still be reactive if it starts dropping packets only when the queue is full. In a proactive open-loop approach, packets would start getting dropped sooner in an attempt to avoid letting congestion settle in.

3.3.4 Application

DTNs have a wide range of applications from deep space communication to mobile sensor networks on earth. Depending on the application, DTNs may exhibit very different characteristics to be able to address the requirements of the driving applications. Therefore, in our congestion control taxonomy, we consider the DTN application as an important classification criterion.

3.3.5 Contacts

When a communication opportunity appears between two DTN nodes we call this a *contact*. There are different kinds of contacts (HERBERTSSON, 22/12/2010) (KHABBAZ *et al.*, 2011) depending on whether they can be predicted. *Scheduled* contacts are predictable and known in advance. A typical example of a DTN with scheduled contacts is interplanetary networks, in which the mobility of nodes is completely predictable and known a priori. In *probabilistic*

DTNs, contacts are probabilistic following a particular distribution that is based on historical data. Finally, contacts in *opportunistic* DTNs are totally random and not known ahead of time.

3.3.6 Routing

In our taxonomy, we also consider whether there is any relationship between congestion control and routing. We then classify congestion control mechanisms as routing protocol- *independent* or *dependent*. Congestion control approaches that can work with any routing mechanism are said to be routing-protocol independent; on the other hand, congestion control mechanisms that are proposed taking into account specific DTN routing protocols are routing-protocol dependent.

An interesting aspect of protocol-dependent congestion control is that, often times, the same mechanism that is employed for routing also serves the congestion control mission. A simple example of such "serendipitous" congestion control is controlled epidemic, where the same mechanism used to limit the number of message copies in the network is also performing congestion control.

3.3.7 Evaluation Platform

Most experimental evaluations of proposed DTN congestion control mechanisms have employed network simulation platforms. A significant advantage of network simulators is the fact that they make it easy to subject protocols under evaluation to a wide range of network and traffic conditions. They also allow experiments to be reproduced easily. In our taxonomy, where applicable, we include information about the simulation platform used to evaluate a particular congestion control mechanism.

3.3.8 Deployability

In the context of DTN congestion control, we define *deployability* as the ability to deploy a protocol in realistic scenarios under real-world conditions. As previously pointed out, most existing DTN congestion control protocols have been implemented and tested using simulation platforms, which do not necessarily expose the mechanisms being evaluated to real-world conditions. In order to explore the deployability of existing congestion control mechanisms, in Section 3.4.8, we classify them using three different levels, namely: low-, medium, and high deployability. The criterion we use here to assess the deployability of DTN congestion control mechanisms is whether they relay on global knowledge of the network.

3.4 Classification of DTN Congestion Control Mechanisms

In this section, we provide an overview of existing DTN congestion control mechanisms in light of our DTN congestion control classification. We summarize the various mechanisms according to our taxonomy in Table 3.6.

3.4.1 Congestion Detection

Congestion detection techniques and the congestion indicators they use are critical to the congestion control effort. They determine how reliably a network is able to detect congestion and how quickly it is able to react to it.

Several DTN congestion control mechanisms use buffer occupancy rate, that is, the availability of buffer space, as an indicator of congestion (BISIO *et al.*, 2009)(BURLEIGH *et al.*, 2007) (CHAUHAN *et al.*, 2012) (CHENG-JUN *et al.*, 2013) (FARRELL, 2008) (GRUNDY; RADENKOVIC, 2010) (GRUNDY, 2012) (HUA *et al.*, 2010) (LAKKAKORPI *et al.*, 2011) (LEELA-AMORNSIN; ESAKI, 2010) (RADENKOVIC; GRUNDY, 2011) (SELIGMAN *et al.*, 2006) (SELIGMAN *et al.*, 2007) (WANG *et al.*, 2012a) (WANG *et al.*, 2012b) (YIN *et al.*, 2010) (ZHANG; LIU, 2008) (LO; LU, 2012) (AN; LUO, 2011).

As an example, the congestion control technique proposed in (SELIGMAN *et al.*, 2006) checks buffer availability of potential DTN custodians, i.e., nodes that will be assuming responsibility for carrying messages on behalf of other nodes. If the custodian's buffer is full or the new message will not fit, the DTN custodian is considered to be congested.

Network capacity has also been used as congestion indicator in DTNs (COE; RACHAVENDRA, 2010) (YUN *et al.*, 2010) (BURLEIGH, 2013). Token based congestion control (COE; RACHAVENDRA, 2010) tries to regulate the amount of traffic entering a network based on network capacity, where network capacity is measured by the amount of data a network can deliver to destinations in a given period of time.

Some efforts employ number of dropped packets à la TCP as a way to detect congestion (THOMPSON *et al.*, 2010) (THOMPSON; KRAVETS, 2010). However, due to frequent topology variations and high error rates, packets may be dropped for reasons other than congestion. For this reason, the congestion detection proposed in (THOMPSON *et al.*, 2010), partially follows TCP's strategy: the proposed congestion indicator is a function of the volume of message drops and replication.

3.4.2 Open- versus Closed-Loop Control

As illustrated in Table 3.6, over 80% of the DTN congestion control mechanisms considered here (RANGO *et al.*, 2008) (SELIGMAN *et al.*, 2007) (BURLEIGH *et al.*, 2007) (HUA *et al.*, 2010) (GRUNDY; RADENKOVIC, 2010) (GRUNDY, 2012) (COE; RACHAVENDRA, 2010) (SELIGMAN *et al.*, 2006) (YIN *et al.*, 2010) (CHENG-JUN *et al.*, 2013) (THOMPSON *et al.*, 2010) (LAKKAKORPI *et al.*, 2011) (RADENKOVIC; GRUNDY, 2011) (BISIO *et al.*, 2009) (THOMPSON; KRAVETS, 2010) (FARRELL, 2008) (LEELA-AMORNSIN; ESAKI, 2010) (LO; LU, 2012) (AN; LUO, 2011) (BURLEIGH, 2013) can be classified as closed loop. For example, the approach described in (HUA *et al.*, 2010) is based on the observation that congestion in DTN nodes builds up gradually. It then proposes three states for DTN nodes, namely: normal state, congestion adjacent state (CAS), and congestion state (CS). Congestion control is then performed depending on the state of the node. More specifically, the proposed congestion control mechanism adopts a closed loop approach by having nodes broadcasting congestion information to their neighbors once they enter into CAS or CS.

The other mechanisms we examined, i.e., (CHAUHAN *et al.*, 2012) (ZHANG; LIU, 2008) (YUN *et al.*, 2010) (WANG *et al.*, 2012b) (WANG *et al.*, 2012a), fall into the open-loop control category. For example, in (YUN *et al.*, 2010) a congestion control strategy called the Average Forwarding Number based on Epidemic Routing (AFNER) is proposed. In AFNER, when a node needs to receive an incoming message and its buffer is full, the node randomly drops a message from those whose forwarding number is larger than the networks average forwarding number. The forwarding number of a message is defined as the number of copies that have been made of a message, and the average forwarding number is the mean forwarding number of all the messages currently in the network. According to the forwarding number queue, the strategy determines the packet forwarding sequence. Open-loop congestion control mechanisms are generally based on dropping messages, while closed-loop approaches use feedback information, i.e., messages from neighbors, to avoid having to drop messages. They typically only drop data as a last resort. Some existing DTN congestion control strategies use drop policies that have been proposed for "traditional" networks. A list of some traditional drop policies along with the DTN congestion control mechanisms that use them is showed in Table 3.1.

Table 3.2 show some drop policies (and the mechanisms that use them) which have been proposed specifically for DTNs.

While one could argue that open loop congestion control systems are a better match for DTNs, most existing mechanisms employ a closed loop approach.

TABLE 3.1 – Traditional drop policies

Drop Policy	Description
Drop-random (DAVIS *et al.*, 2001) (BISIO *et al.*, 2009) (GRUNDY; RADENKOVIC, 2010) (RADENKOVIC; GRUNDY, 2011)	a message from the queue is selected at random to be dropped.
Drop-head (DAVIS *et al.*, 2001) (SELIGMAN *et al.*, 2007)	the first message in the queue, i.e., the head of the queue, is dropped.
Drop-tail (DAVIS *et al.*, 2001) (SELIGMAN *et al.*, 2007) (BURLEIGH *et al.*, 2007) (SELIGMAN *et al.*, 2006) (COE; RACHAVENDRA, 2010) (BURLEIGH, 2013)	the most recently received message, i.e., the tail of the queue, is removed.
NHop-Drop (YUN *et al.*, 2010)	any message that has been forwarded over N hops is dropped.
Drop-least-Recently-received (DAVIS *et al.*, 2001) (YIN *et al.*, 2010)	the message that has been in the node buffer longest is removed.
Drop-oldest (DAVIS *et al.*, 2001)(LAKKAKORPI *et al.*, 2011) (SELIGMAN *et al.*, 2007)	the message that has been in the network longest is dropped.
Drop-youngest (DAVIS *et al.*, 2001)	drops the message with the longest remaining life time.

3.4.3 Proactive versus Reactive Control

Congestion control solutions can be classified as proactive (i.e., performs congestion avoidance), reactive (i.e., responds to congestion events), or hybrid (i.e., performs congestion avoidance and reacts to congestion build-up).

From Table 3.6, we observe that most existing DTN congestion control mechanisms (around 52%) adopt a hybrid policy using both proactive– and reactive control (BISIO *et al.*, 2009) (BURLEIGH *et al.*, 2007) (CHAUHAN *et al.*, 2012) (GRUNDY; RADENKOVIC, 2010) (GRUNDY, 2012) (HUA *et al.*, 2010) (LEELA-AMORNSIN; ESAKI, 2010) (RANGO *et al.*, 2008) (THOMPSON *et al.*, 2010) (YIN *et al.*, 2010) (YUN *et al.*, 2010) (LO; LU, 2012) (AN; LUO, 2011). An example of a hybrid approach that combines random early detection (RED) and explicit congestion notification (ECN) within the DTN architecture is described in (BISIO *et al.*, 2009).

We also note that approximately 31% of mechanisms studied adopt a purely reactive approach (CHENG-JUN *et al.*, 2013) (FARRELL, 2008) (SELIGMAN *et al.*, 2006) (SELIGMAN *et al.*, 2007) (THOMPSON; KRAVETS, 2010) (WANG *et al.*, 2012a) (WANG *et al.*, 2012b), while 17% of them are based on a purely proactive policy (COE; RACHAVENDRA, 2010) (LAKKAKORPI *et al.*, 2011) (RADENKOVIC; GRUNDY, 2011) (ZHANG; LIU, 2008) (BURLEIGH, 2013). The approach proposed in (LAKKAKORPI *et al.*, 2011) proactively advertises buffer occupancy information to adjacent nodes. Nodes can then avoid forwarding messages to nodes with high buffer occupancy. Following the mechanism described in (SELIGMAN *et al.*, 2006), when a DTN node becomes congested, it tries to migrate stored messages to alternate locations to avoid loss.

3.4.4 Application

The challenging peculiarities of interplanetary networking environments inspired the congestion control mechanisms proposed in (BISIO *et al.*, 2009) (FARRELL, 2008) (RANGO *et al.*, 2008) (BURLEIGH, 2013) which target deep space communication scenarios. In the case of (FARRELL, 2008), to evaluate the Linklider Transmission Protocol - Transport (LTP-T), a scenario

TABLE 3.2 – DTN drop policies

Drop Policy	Description
Drop-largest (RASHID; AYUB, 2010)	the message of the largest size is selected.
Evict Most Forwarded First - (MOFO) (INDGREN; PHANSE, 2006)	the message forwarded the maximum number of times is selected.
Evict Most Favorably Forwarded First - (MOPR) (INDGREN; PHANSE, 2006) (YUN *et al.*, 2010)	each message is related to a forwarding predictability FP. Whenever the message is forwarded the FP value is updated and the message that contains the maximum FP is dropped first.
Evict Shortest Life Time First - (SHLI) (INDGREN; PHANSE, 2006)(LAKKAKORPI *et al.*, 2011)	the message that contains the smallest TTL is dropped.
Evict Least Probable First - (LEPR) (INDGREN; PHANSE, 2006)	since the node is less likely to delivery a message for which it has a low P-value (low delivery predictability) and that it has been forwarded at least MF times, drop the message for which the node has the lowest P value.
Global Knowledge Based Drop - (GBD) (KRIFA *et al.*, 2008)	based on global knowledge about the state of each message in the network (number of replicas), drop the message with the smallest utility among the one just received and the buffered messages.
History Based Drop - (HBD) (KRIFA *et al.*, 2008)	a deployed variant of GBD that uses the new utilities based on estimates of m (the number of nodes, excluding the source, that have seen message since its creation until elapsed time) or n (the number of copies of message in the network after elapsed time). The message with the smallest utility is selected.
Flood Based Drop - (FBD) (KRIFA *et al.*, 2008)	accounts only for the global information collected using simple message flooding, that is, without considering message size.
Threshold Drop - (T-Drop) (AYUB; RASHID, 2010)	drops the message from congested buffer only if size of existing queued message(s) falls in Threshold range (T).
Equal Drop - (E-Drop) (RASHID *et al.*, 2011)	drops the stored message if its size is equal to or greater than that of the incoming message; otherwise does not drop.
Message Drop Control - (MDC) (AYUB *et al.*, 2011)	the largest size message will be dropped. This policy controls the message drop through the use of an upper bound.
Mean Drop (RASHID *et al.*, 2012)	drops messages that have size greater than or equal to the mean size of queued messages at the node.
Small-Copies Drop (KIM *et al.*, 2008)	drops messages with the largest expected number of copies first.
Adaptive Optimal Buffer Management Policies - (AOBMP) (LI *et al.*, 2009)	drops messages according to the utility function associated to the message.

that emulates a deep-space network consisting of nodes on Earth and around Mars was modeled. In this context, the proposed congestion control protocol is designed to withstand the noise and delays incurred by communication across astronomical distances.

Terrestrial applications are considered in (SELIGMAN *et al.*, 2007) (BURLEIGH *et al.*, 2007) (HUA *et al.*, 2010) (GRUNDY; RADENKOVIC, 2010) (GRUNDY, 2012) (COE; RACHAVENDRA, 2010) (LAKKAKORPI *et al.*, 2011) (RADENKOVIC; GRUNDY, 2011) (THOMPSON *et al.*, 2010) (LO; LU, 2012). In (THOMPSON *et al.*, 2010) one of the scenarios discussed models the behavior of mobile agents in disaster relief operations.

Dynamic scenarios with random generation of nodes following a statistic model are used in (CHAUHAN *et al.*, 2012) (CHENG-JUN *et al.*, 2013) (THOMPSON; KRAVETS, 2010) (LEELA-AMORNSIN; ESAKI, 2010) (WANG *et al.*, 2012a) (SELIGMAN *et al.*, 2006) (ZHANG; LIU, 2008) (YUN *et al.*, 2010) (WANG *et al.*, 2012b) (YIN *et al.*, 2010) (AN; LUO, 2011). Most of them are based on using mobile terrestrial communication applications to study the proposed congestion control mechanisms. In the case of (CHAUHAN *et al.*, 2012) the authors use a simple scenario where nodes and networks parameters are generated randomly and a mobility model is set.

3.4.5 Contacts

We observe from Table 3.6 that around 17% of DTN congestion control mechanisms (THOMPSON *et al.*, 2010) (WANG *et al.*, 2012b) (SELIGMAN *et al.*, 2007) (CHAUHAN *et al.*, 2012) assume in their scenarios both predicted and opportunistic contacts. Among the remaining congestion control mechanisms (BISIO *et al.*, 2009) (FARRELL, 2008) (RANGO *et al.*, 2008) (BURLEIGH, 2013), another 13% assume scheduled contacts. In particular, the hop-by-hop based mechanism proposed in (RANGO *et al.*, 2008) experiments with an interplanetary network scenario which employs scheduled contact between planets.

Approximately 39% of the mechanisms we studied, namely (CHENG-JUN *et al.*, 2013) (WANG *et al.*, 2012a) (LEELA-AMORNSIN; ESAKI, 2010) (LAKKAKORPI *et al.*, 2011) (YIN *et al.*, 2010) (YUN *et al.*, 2010) (ZHANG; LIU, 2008) (LO; LU, 2012) (AN; LUO, 2011) assume opportunistic contacts while around 30%, namely (BURLEIGH *et al.*, 2007) (GRUNDY; RADENKOVIC, 2010) (GRUNDY, 2012) (HUA *et al.*, 2010) (RADENKOVIC; GRUNDY, 2011) (COE; RACHAVENDRA, 2010) (SELIGMAN *et al.*, 2006) (THOMPSON; KRAVETS, 2010) use predicted contacts.

One example of opportunistic contact can be seen in (CHENG-JUN *et al.*, 2013) where a congestion control routing algorithm for security and defense based on social psychology and game theory is presented. In this case, DTN nodes are assumed to be randomly distributed and to perform random routing. Random routing results in randomness of each node's encounters.

A distributed congestion control algorithm that adaptively chooses the next-hop based on contact history and statistics is described in (GRUNDY; RADENKOVIC, 2010). It has a component called contact manager that executes a forwarding heuristic taking into account predicted contacts.

3.4.6 Routing

Most of the DTN congestion control mechanisms, e.g., (BISIO *et al.*, 2009) (FARRELL, 2008) (GRUNDY; RADENKOVIC, 2010) (GRUNDY, 2012) (HUA *et al.*, 2010) (RADENKOVIC; GRUNDY, 2011)(SELIGMAN *et al.*, 2007) (THOMPSON *et al.*, 2010) (THOMPSON; KRAVETS, 2010) (WANG *et al.*, 2012a) (WANG *et al.*, 2012b) (YIN *et al.*, 2010) (YUN *et al.*, 2010) (LO; LU, 2012) (AN; LUO, 2011), were proposed taking into account specific DTN routing protocols. We have classified them as routing protocol dependent. For example, the AFNER approach described in (YUN *et al.*, 2010) proposes a congestion control strategy for epidemic routing. On the other hand, there are mechanisms such as (BURLEIGH *et al.*, 2007) (CHAUHAN *et al.*, 2012) (CHENG-JUN *et al.*, 2013) (COE; RACHAVENDRA, 2010) (LAKKAKORPI *et al.*, 2011) (LEELA-AMORNSIN; ESAKI, 2010) (RANGO *et al.*, 2008) (SELIGMAN *et al.*, 2006)(SELIGMAN *et al.*, 2007) (THOMPSON *et al.*, 2010) (ZHANG; LIU, 2008) (BURLEIGH, 2013) that are independent of the routing protocol. This means that congestion control does not act- or rely on the underlying routing mechanism and

thus can be used with any routing infrastructure. A notable example is the Credit-Based Congestion Control mechanism (LEELA-AMORNSIN; ESAKI, 2010) which uses the age of a message as a heuristics to decide when the message will be discarded when congestion builds up in nodes.

We should point out that for DTNs that require interoperability through the Bundle Layer (SCOTH; BURLEIGH, 2007), congestion control mechanisms should be independent of the routing protocol. Intuitively, congestion control mechanisms that work independently from the underlying routing protocol are more general and applicable to a wide array of scenarios.

3.4.7 Evaluation Platform

Approximately 17% of the mechanisms covered in this survey were evaluated using custom simulators (BURLEIGH *et al.*, 2007) (GRUNDY; RADENKOVIC, 2010; GRUNDY, 2012) (COE; RACHAVENDRA, 2010) (ZHANG; LIU, 2008). For example, a discrete event-driven simulator was developed in (ZHANG; LIU, 2008) to test a congestion management strategy that uses the concept of revenue management and employs dynamic programming.

Another 17% of the techniques surveyed were evaluated using the *ns-2* network simulator (THE..., 2013), a simulation platform widely used in network research (RANGO *et al.*, 2008) (LAKKAKORPI *et al.*, 2011) (BISIO *et al.*, 2009) (YUN *et al.*, 2010).

Almost 44% of the approaches we researched (CHENG-JUN *et al.*, 2013) (WANG *et al.*, 2012a) (WANG *et al.*, 2012b) (THOMPSON *et al.*, 2010) (RADENKOVIC; GRUNDY, 2011) (THOMPSON; KRAVETS, 2010) (YIN *et al.*, 2010) (LEELA-AMORNSIN; ESAKI, 2010) (LO; LU, 2012) (AN; LUO, 2011) use the ONE simulator (KERäNEN *et al.*, 2009). ONE was designed specifically to evaluate DTN protocols and has become popular within the DTN research community. For instance, a local approach to detect and respond to congestion by adjusting the copy limit for new messages (THOMPSON; KRAVETS, 2010) has been implemented and evaluated using the ONE simulator.

The remaining approaches we surveyed employ other simulator platforms such as OPNET (HUA *et al.*, 2010), YACSIM (SELIGMAN *et al.*, 2007), GT-ITM (SELIGMAN *et al.*, 2007), Weka (CHAUHAN *et al.*, 2012), Netem (FARRELL, 2008) and ION (BURLEIGH, 2013) (LABORATORY, 2013).

We should also highlight the Interplanetary Overlay Network (ION) (LABORATORY, 2013) implementation of the DTN architecture. ION accomplishes congestion control by computing *congestion forecasts* based on published contact plans. These forecasts are presented to the mission teams so that they can take corrective action, revising contact plans before the forecast congestion occurs. The transmission rates in the contact plans are enforced automatically by built-in rate control mechanisms in the ION bundle protocol agent. In the case of rate control failure, causing reception rate to exceed what was asserted in the contact plan, the receiving

bundle protocol agent drops data according to a drop-tail policy to avoid congestion. The insertion of new bundles into the network can also lead to congestion. To avoid this, ION implements an admission control mechanism that may either function in a drop-tail manner or simply block the application until insertion of the new bundle no longer threatens to congest the node.

The general trend we observe is that most proposed DTN congestion control mechanisms have been evaluated experimentally using simulation platforms. Employing more realistic experimental environments including real world scenarios and testbeds should become a priority in DTN protocol research.

3.4.8 Deployability

DTN congestion control mechanisms that rely on global network knowledge are classified as "low" deployability. A notable example is the protocol proposed in (YUN *et al.*, 2010) which uses the network's average forwarding number, i.e., the average forwarding number considering all messages currently in the network, to decide which message(s) to drop in case nodes get congested (see Table 3.3). The low deployability of this approach is due to the fact that, in order to compute the average forwarding number, nodes need global knowledge of the network, which is hard to acquire, especially in DTNs.

TABLE 3.3 – DTN congestion control mechanisms with low deployability

Mechanism	Description
Average forwarding number based on epidemic routing (AFNER) (YUN *et al.*, 2010)	Nodes drop messages whose forwarding number is larger than the network's average forwarding number (a message's forwarding number is the number of copies of that message floating in the network).

Around 70% of the surveyed DTN congestion control mechanisms ((SELIGMAN *et al.*, 2007) (HUA *et al.*, 2010) (GRUNDY; RADENKOVIC, 2010) (GRUNDY, 2012) (COE; RACHAVENDRA, 2010) (SELIGMAN *et al.*, 2006) (YIN *et al.*, 2010) (CHENG-JUN *et al.*, 2013) (LAKKAKORPI *et al.*, 2011) (RADENKOVIC; GRUNDY, 2011) (BISIO *et al.*, 2009) (FARRELL, 2008) (RANGO *et al.*, 2008) (THOMPSON *et al.*, 2010) (THOMPSON; KRAVETS, 2010) (WANG *et al.*, 2012b) (WANG *et al.*, 2012a) (AN; LUO, 2011)) can be classified as "medium" deployability. For example, the approach presented in (THOMPSON; KRAVETS, 2010) tries to respond to congestion by adjusting the maximum number of message copies based on the current level of network congestion. Nodes estimate global congestion levels using the ratio between drops and duplicate deliveries obtained during node encounters. From Table 3.4, which shows a brief description of medium-deployability mechanisms, we observe that such mechanisms rely on local neighborhood information to perform congestion control.

The remainder of the mechanisms we investigated, namely (BURLEIGH *et al.*, 2007) (LEELAAMORNSIN; ESAKI, 2010) (CHAUHAN *et al.*, 2012) (ZHANG; LIU, 2008) (LO; LU, 2012) (BURLEIGH,

TABLE 3.4 – DTN congestion control mechanisms with medium deployability

Mechanism	Description
Hop-by-hop Local Flow Control (RANGO *et al.*, 2008)	Nodes use hop-by-hop flow control where the sender verifies if the link is active and if there is resource availability towards the next-hop receiver.
Storage Routing (SELIGMAN *et al.*, 2007)	Under congestion, nodes use a migration algorithm to transfer messages to other, less congested nodes.
Congestion Avoidance Based on Path Avoidance (HUA *et al.*, 2010)	This scheme manages node storage and defines three states: normal, congestion adjacent, and congested. Nodes broadcast their current state to their neighbors who avoid forwarding messages to congested nodes.
Context Aware Forwarding Algorithm (CAFÈ) (GRUNDY; RADENKOVIC, 2010) (GRUNDY, 2012)	Nodes adaptively choose a message's next hop based on contact history and statistics.
Token Based Congestion Control (COE; RACHAVENDRA, 2010)	In this scheme all nodes must have a token in order to inject messages into the network. Tokens are initially uniformly distributed and after some time move randomly throughout the network.
Push-Pull Custody Transfer (SELIGMAN *et al.*, 2006)	This approach uses buffer space availability information from neighbors to mitigate congestion. It includes a set of algorithms to determine which messages should be migrated to which neighbors and when.
Incentive Multi-Path Routing with Alternative Storage (IMRASFC) (YIN *et al.*, 2010)	This scheme uses an incentive mechanism to stimulate mal-behaving nodes to store and forward messages and also try to select alternative neighbors nodes with available storage.
Congestion Control Routing Algorithm for Security Defense based on Social Psychology and Game Theory (CRSG) (CHENG-JUN *et al.*, 2013)	This approach uses social psychology and game theory to balance network storage resource allocation. It does this by obtaining node buffer utilization during node encounters.
Node-Based Replication Management (RRCC) (THOMPSON *et al.*, 2010)	This scheme detects and responds to congestion by adjusting the message copy limit. It uses local measurements, e.g., the ratio of the number of dropped messages to the number of message replicas measured by a node.
Congestion Avoidance Based on Buffer Space Advertisement (LAKKAKORPI *et al.*, 2011)	Nodes advertise their buffer occupancy to adjacent nodes; neighboring nodes then use this information to decide their next hop when forwarding messages.
Congestion Aware Forwarding (RADENKOVIC; GRUNDY, 2011)	Congestion avoidance strategy which utilizes heuristics to infer shortest paths to destinations from social information (e.g., connectivity); it uses buffer occupancy and communication latency information to avoid areas of the network that are congested.
Combined Congestion Control for IPN (BISIO *et al.*, 2009)	This mechanism combines ECN (Explicit Congestion Notification) and RED (Random Early Detection) with storage-based routing strategies and makes use of neighbor buffer occupancy to mitigate congestion.
Threshold-Based Congestion Control (THOMPSON; KRAVETS, 2010)	This scheme tries to respond to congestion by adjusting the copy limit for messages based on the current observed network congestion level. The congestion level is estimated based on information collected during node encounters.
Lincklider Transmission Protocol Transport (LTP-T) (FARRELL, 2008)	LTP-T maintains a congestion timer for forwarded blocks. The idea is that an entire block has to be transmitted before the timer expires. If the congestion timer expires at an intermediate node, this node should signal the presence of congestion to upstream nodes using LTP-T's congestion notification.
Simulated Annealing and Regional Movement (SARM) (WANG *et al.*, 2012b)	SARM adopts a message deleting strategy based on regional characteristics of node movement and message delivery. As nodes move, the algorithm records the cumulative number of encounters with other nodes. When congestion happens, the cumulative number of the node as the transferred node and the destination node for all messages in these two nodes is calculated and then the message is selected to be deleted.
Following Routing (FR) (WANG *et al.*, 2012a)	FR assumes all nodes are mobile; if a node A tries to relay a message to node B but B is not able to receive the message (e.g., because its buffer is full), A tries to follow B's trajectory hoping to encounter a suitable next hop or the destination node.
Dynamic Congestion Control Based Routing (DCCR) (LO; LU, 2012)	DCCR is based on replication quotas. Each message is associated with an initial quota. During its time-to-live, a message can have its quota value updated according to a quota allocation function. This function reduces or increases the message's replication quota. Moreover, each node maintains buffer occupancy and contact probability with other nodes. In this case each node can compute the local measurement of congestion level information in order to update replication quotas.

2013) (see Table 3.5) can be included in the high-deployability category since they try to perform congestion control by using only local information, i.e., information about the node itself. For example, the scheme presented in (ZHANG; LIU, 2008) proposes a congestion management technique that decides whether to accept custody of a bundle if and only if the benefit of accepting custody is greater than or equal to the cost of the resources used to store the bundle.

3.5 DTN Congestion Control Design Guidelines

The majority of the congestion control techniques we investigated were developed for, and validated in, specific scenarios. Their applicability is therefore limited in various ways. For example, some control techniques depend upon anomalous conditions that are difficult to reproduce in an operating DTN environment but that induce temporary congestion in very special situations. Consequently, these mechanisms may not accurately manage routine congestion in a nominally operating network.

As previously pointed out, most existing congestion control protocols have been evaluated

TABLE 3.5 – DTN congestion control mechanisms with high deployability

Mechanism	Description
Autonomous Congestion Control (BURLEIGH *et al.*, 2007)	This mechanism adopts a financial model and compares the receipt and forwarding of messages to risk investment. When a new message arrives, the node decides whether to receive it or not according to a risk value of receiving and storing the message. The risk value is determined by local metrics, such as the node's own buffer space and the input data rate.
Distributive Congestion Control for Different Types of Traffic (CHAUHAN *et al.*, 2012)	Congestion control is accomplished by distributing traffic according to different priority levels. Messages with higher priority are ensured minimum bounded delay whereas those with lower priority are discarded at higher congestion levels.
Congestion Management Based on Revenue Management and Dynamic Programming (ZHANG; LIU, 2008)	Nodes accept custody of messages (or bundles) if and only if there is sufficient remaining resource capacitiy and the resulting benefit of acting as relays is greater than or equal to the cost associated to the use of local resources.
Credit-Based Congestion Control (LEELA-AMORNSIN; ESAKI, 2010)	According to this strategy, some credit is associated to each message; when a two nodes encounter each other, the amount of credit for each message is updated. When buffers become full, messages that have the least credit are dropped.
Message Admission Control Based on Rate Estimation (MACRE) (AN; LUO, 2011)	This congestion control scheme decides whether to admit a message according to the relationship between a node's input rate and output rate.
Interplanetary Overlay Network (ION) (LABORATORY, 2013) (BURLEIGH, 2013)	ION's congestion control is anticipatory and is performed based on the "contact plan" between the nodes. The maximum projected occupancy of a node is based on the computation of the congestion forecast for the node. Thus, congestion control is performed essentially manually.

experimentally using simulation platforms and some "traditional" performance metrics, i.e., metrics used to evaluate Internet congestion control as discussed in Section 3.1. DTN-specific performance metrics have been proposed including buffer occupancy and message replication. Additionally, DTN congestion control techniques have also employed "DTN-aware" indicators to help in the congestion control effort, e.g., social behavior of nodes, node mobility, message retention in storage, node encounter probability, node connectivity and betweenness, etc. Moving forward, employing more realistic experimental environments including real world scenarios and testbeds should become a priority in DTN protocol research.

Some DTN congestion control techniques were designed to operate over specific routing protocols. In a heterogeneous DTN environment where no single routing protocol is universally supported, routing-specific techniques' control over congestion is limited to the nodes where the corresponding routing protocols are in operation. The dependence of congestion control techniques on specific routing protocols can be mitigated by limiting the scope of congestion control information to the node at which that information was generated, i.e., merely using that information as the basis for local admission control. But a more powerful mitigation would be to enable interoperation among congestion control techniques, enabling protocol-neutral network

topology information and congestion control cues generated in one routing environment to be propagated to environments governed by other protocols. For example, network regions in which routes are computed from lists of scheduled contacts should be able to make use of congestion control and network topology information developed in regions where routing is opportunistic.

Generality aside, other aspects of the congestion control techniques developed to date seem problematic. Many of these mechanisms function in a reactive fashion and attempt to manage congestion in closed control loops; while the simplicity of closed-loop protocols is appealing, these approaches are innately non-delay-tolerant. Closed-loop mechanisms work very well for local admission control but cannot be relied upon for timely operation over network links that are subject to lengthy lapses in connectivity and/or long signal propagation times.

Accordingly, we suggest that DTN congestion control should employ a hybrid of open-loop and closed-loop control: closed-loop local admission control coupled with open-loop control over network links, functioning both proactively and reactively.

The natural question, then, is whether or not adherence to this principle would enable development of a single universal congestion control mechanism for DTNs. Can any single mechanism be driven by information produced by all possible DTN routing protocols, in all possible operational scenarios (ranging from terrestrial networks to deep space mission operations), maximizing network utilization while minimizing end-to-end delivery latency? These are questions we plan to explore in our future work.

We should also highlight the work being conducted by the IRTF's Delay-Tolerant Networking Research Group (DTNRG) (DTNRG, 2014), under which a DTN transport-layer overlay is being proposed. This could be an ideal building block upon which a universal, routing-neutral DTN congestion control framework could be built.

TABLE 3.6 – Flow and Congestion Control Mechanisms

Mechanism	Congestion Detection	Open- or Closed-Loop Control	Proactive or Reactive Control	Application	Contacts	Routing	Evaluation Platform	Deployability
Hop-by-hop local flow control (RANGO *et al.*, 2008)	Network capacity	Closed-loop	Hybrid	Interplanetary Communication	Scheduled	Independent	NS-2	medium
Storage routing for DTN congestion control (SELIGMAN *et al.*, 2007)	Buffer availability	Closed-loop	Reactive	Environmental Monitoring	Predicted and Scheduled	dependent	YACSIM and GT-ITM	medium
Autonomous congestion control (BURLEIGH *et al.*, 2007)	Buffer availability	Closed-loop	Hybrid	Data Dissemination	Predicted	Independent	–	high
Congestion avoidance based on path avoidance (HUA *et al.*, 2010)	Buffer availability	Closed-loop	Hybrid	Community-based Terrestrial Communication	Predicted	Dependent	OPNET	medium
Context aware forwarding algorithm (CAFÉ) (GRUNDY; RADENKOVIC, 2010) (GRUNDY, 2012)	Buffer availability	Closed-loop	Hybrid	Community-based Terrestrial Communication	Predicted	Dependent	own trace driven opportunistic simulator	medium
Token based congestion control (COE; RACHAVENDRA, 2010)	Network capacity	Closed-loop	Proactive	Environmental Monitoring	Predicted	Independent	own discrete event simulator	medium
Push-Pull custody transfer (SELIGMAN *et al.*, 2006)	Buffer availability	Closed-loop	Reactive	Mobile Terrestrial Communication	Predicted	Independent	DTNSim	medium
Incentive Multi-paths Routing with Alternative Storage (IMRASFC) (YIN *et al.*, 2010)	Buffer availability	Closed-loop	Hybrid	Mobile Terrestrial Communication	Opportunistic	Dependent	ONE	medium
Distributive congestion control for different types of traffic (CHAUHAN *et al.*, 2012)	Buffer availability	Open-loop	Hybrid	Mobile Terrestrial Communication	Predicted and Opportunistic	Independent	Weka undeployed	high
Congestion Control Routing Algorithm for Security Defense based on Social Psychology and Game Theory (CRSG) (CHENG-JUN *et al.*, 2013)	Buffer availability	Closed-loop	Reactive	Community-based Terrestrial Communication	Opportunistic	Independent	ONE	medium
Node-based replication management algorithm (RRCC) (THOMPSON *et al.*, 2010)	Drop rate	Closed-loop	Hybrid	Mobile Terrestrial Communication	Predicted and Opportunistic	dependent	ONE	medium
Buffer space advertisement to avoid congestion (LAKKAKORPI *et al.*, 2011)	Buffer availability	Closed-loop	Proactive	Mobile Terrestrial Communication	Opportunistic	Independent	NS-2 and ONE	medium
Congestion aware forwarding algorithm (RADENKOVIC; GRUNDY, 2011)	Buffer availability	Closed-loop	Proactive	Data Dissemination	Predicted	Dependent	ONE	medium
Combined congestion control for IPN (BISIO *et al.*, 2009)	Buffer availability	Closed-loop	Hybrid	Interplanetary Communication	Scheduled	Dependent	NS-2	medium
Threshold-based congestion control protocol (THOMPSON; KRAVETS, 2010)	Drop rate	Closed-loop	Reactive	Mobile Terrestrial Communication	Predicted	Dependent	ONE	medium
Lincklider transmission protocol transport (LTP-T) (FARRELL, 2008)	Buffer availability	Closed-loop	Reactive	Interplanetary Communication	Scheduled	Dependent	network emulation with Netem (HEMMINGER, 2005)	medium
Congestion management strategy based on revenue management and dynamic programming (ZHANG; LIU, 2008)	Buffer availability	Open-loop	Proactive	Mobile Terrestrial Communication	Opportunistic	Independent	own discrete event-driven simulator	high
Credit-based congestion control (LEELA-AMORNSIN; ESAKI, 2010)	Buffer availability	Closed-loop	Hybrid	Mobile Terrestrial Communication	Opportunistic	Independent	ONE	high
Average forwarding number based on epidemic routing (AFNER) (YUN *et al.*, 2010)	Network capacity	Open-loop	Hybrid	Mobile Terrestrial Communication	Opportunistic	Dependent	NS-2	low
Simulated annealing and regional movement (SARM) (WANG *et al.*, 2012b)	Buffer availability	Open-loop	Reactive	Data dissemination	Predicted and Opportunistic	Dependent	ONE	medium
Following routing (FR) (WANG *et al.*, 2012a)	Buffer availability	Open-loop	Reactive	Mobile Terrestre Communication	Opportunistic	Dependent	ONE	medium
Dynamic congestion control based routing (DCCR) (LO; LU, 2012)	Buffer availability	Closed-loop	Hybrid	Mobile Terrestre Communication	Opportunistic	Dependent	ONE	medium
Message Admission Control based on Rate Estimation (MACRE) (AN; LUO, 2011)	Buffer availability	Closed-loop	Hybrid	Data dissemination	Opportunistic	Dependent	ONE	high
Interplanetary Overlay Network (ION) (LABORATORY, 2013) (BURLEIGH, 2013)	Network capacity	Closed-loop	Proactive	Interplanetary Communication	Scheduled	Independent	ION framework	high

3.6 Conclusion

This chapter reviewed the state-of-the-art in DTN congestion control. We started by highlighting how DTN congestion control is different from "traditional" Internet congestion control. We then proposed a DTN congestion control taxonomy which we use to describe existing congestion control mechanisms and place them in context of one another. We anticipate that the proposed taxonomy will help to map the DTN congestion control design space and put in perspective the many existing DTN congestion control techniques. Furthermore, our exploration of the DTN design space will also help to identify important issues and questions that are yet to be addressed.

Our exploration of the DTN congestion control shows that a considerable number of protocols have been proposed, most of which have common goals, namely: increase successful message delivery while decreasing delivery delay and keep network utilization high without congesting it. To achieve these goals, several existing DTN congestion control mechanisms adopt a reactive approach by simply dropping messages stored at DTN nodes to make room for incoming messages. Both "traditional"– and new drop policies, i.e., specifically designed for DTN environments, have been employed. Alternatively, a number of DTN congestion control techniques, instead of discarding messages, try to transfer them to neighboring nodes, using sometimes "back pressure" to adjust message generation rate and admission control to restrict the number of flows entering the network.

We also observe that, in DTNs, cost-performance trade-offs become even more exacerbated. For example, reactive congestion control may have an even more considerable impact on performance because of the inherently high delays: by the time congestion is detected, a large number of messages may have been dropped. Dropping messages is detrimental to performance in general, but even more so in DTNs due to their episodic connectivity and limited resources. When messages are dropped, not only won't they be delivered to their final destination, but they will also have consumed precious network resources. Hybrid approaches, i.e., combining reactive and proactive congestion control strategies, are attractive in DTN environments. Hybrid approaches will try to avoid congestion from happening in the first place but can resort to reactive measures, such as dropping messages, if needed.

Another interesting observation is that several congestion control mechanisms have been proposed to resolve congestion that may result from replication-based message forwarding. As a result, they are interoperable with a variety of routing protocols as long as routing uses message replication as a basis for message forwarding. This is the case of networks that use routing based on epidemic forwarding or a variant thereof (e.g., Prophet [59], Spray-and-Wait [60]). However, these congestion control schemes cannot be classified as routing-independent since they only apply to routing based on message replication. The RRCC mechanism [44] is a notable example as it performs congestion control by limiting the number of message replicas.

Even though RRCC's authors consider it to be independent of the routing protocol, according to our classification, RRCC [44] is routing protocol dependent as it would not apply to forwarding-based routing (i.e., routing that does not replicate messages) such as [61], [62] and [63]. One interesting direction for future research is to design effective congestion control mechanisms that can interoperate with any routing scheme.

Traditional Internet congestion control mechanisms typically rely on closed-loop approaches that use end-to-end feedback. However, in DTNs, closed-loop strategies usually employ feedback on a hop-by-hop basis. We contend that DTN congestion control should employ a hybrid of open-loop and closed-loop control so that nodes can also make congestion control decisions based on local information. That way they do not have to rely exclusively on the network to make congestion control decisions.

One interesting observation is that DTN congestion control research relies heavily on simulation platforms to test and evaluate proposed protocols and algorithms. We argue that while simulation-based experimentation is a necessary step, it does not replace real-world experimentation. As such, employing more realistic experimental environments including real world scenarios and testbeds should become a priority in DTN protocol research.

Finally, one important issue from our exploration of existing DTN congestion control techniques is that there is no "universal" congestion control mechanism that will be applicable to all DTN scenarios and applications. To complement our qualitative comparison of DTN congestion control mechanisms, we have been conducting a comparative performance study of different DTN congestion control techniques when applied to different application scenarios. In the next chapter, we present this comparative study for two main application domains, namely: inter-planetary and terrestrial networking applications.

4 Congestion Control in DTN: A Comparative Study for Interplanetary and Terrestrial Networking Applications

Controlling congestion is critical to ensure adequate network operation and performance. That is especially the case in networks operating in "challenged"- or "extreme" environments where episodic connectivity is part of the network's normal operation. Consequently, the "pure" end-to-end congestion control model employed by the Internet is not adequate. Our goal is to study congestion control mechanisms that have been proposed for these so-called disruption tolerant networks, or DTNs. In this chapter, we conduct a performance study comparing existing DTN congestion control mechanisms for two main application domains, namely: inter-planetary (IPN) and terrestrial networking applications. Our results confirm that congestion control helps increase message delivery ratio, even in highly congested network scenarios. Furthermore, the results show that existing DTN congestion control mechanisms do not perform well in IPN scenarios. Our study also suggests that good design principles for congestion control in DTN scenarios include: combining reactive- and proactive control, using local information instead of global knowledge, and employing mechanisms that are routing protocol independent. One important conclusion from our quantitative study is that there is currently no universal congestion control mechanism that fits all DTN scenarios and applications.

4.1 DTN Applications Background

In this section, we introduce the two main application domains where our comparative study takes place.

4.1.1 Interplanetary Internetworking

As described in (AKYILDIZ *et al.*, 2004; ARANITI *et al.*, 2010; BURLEIGH *et al.*, 2003b), an Interplanetary Internet includes the IPN Backbone, IPN External Networks, and Planetary Net-

works (PNs). The IPN Backbone makes possible the communication among the Earth, other planets, space probes, and spacecraft through satellites. The IPN External Network includes, for instance, spacecraft flying in deep space between planets, space probes, and orbiting space stations. A PN is composed of the PN's Satellite Network and the PN's Surface Network. The former includes links among surface nodes, orbiting satellites, and IPN Backbone Nodes, providing relay services between surface networks and the backbone as well as between two or more parts of the surface network. Surface networks provide communication between surface elements, such as rovers and sensor nodes.

The main challenges affecting IPNs can be summarized as follows (AKYILDIZ *et al.*, 2004) (CHEN; CHEN, 2010) (WANG *et al.*, 2009) (ZHANG; ZHOU, 2013):

- **Intermittent connectivity:** disconnections can happen due to planetary motion as well as the movement of celestial bodies, spacecrafts, rovers, etc.
- **Long and variable delays:** the deep space connection may have extremely high round-trip latencies caused by astronomical distances, e.g., the round-trip time (RTT) for radio communication from Mars to Earth can take between 3 minutes minimum to 30 minutes maximum. This happens because the distance between the Earth and Mars varies enormously depending on their relative positions in their orbits around the Sun.
- **High error rates:** the uncorrected bit error rate (BER) for deep space radio communication is high (around 10^{-1}) due to extreme environment conditions (i.e., cosmic radiation leads to signal corruption). Strong forward error correction coding is applied to reduce the observed BER to a rate that is on the order of 10^{-5} to 10^{-6}, still far higher than in communications over optical fiber in the Internet.
- **Asymmetric data rates:** the asymmetry in date rates on space links is typically of the order of 1:1000 or higher. Communications channels between spacecraft and the ground are frequently asymmetric in terms of both channel capacity and error characteristics. This asymmetry is a result of various engineering tradeoffs (such as power, mass, and volume), as well as the fact that for scientific missions, most of the data originates at the satellite and flows to the ground. The return link is generally used for commanding the spacecraft, not bulk data transfer (DURST *et al.*, 1996).

4.1.2 Terrestrial DTN

Popular examples of such terrestrial scenarios, which have already been the subject of extensive research, include: Wireless Sensor Networks (WSNs) deployed in extreme regions (e.g. volcanic or dessert, etc.) and Mobile Ad-Hoc Networks (MANETs) consisting of nodes

mounted on continuously moving objects (e.g. animals or vehicles, etc.). In this work a terrestrial DTN is considered as a kind of network that is subject neither to high delays due to astronomical distances (as in the interplanetary network) nor to scheduled contacts like those imposed by planetary movement (HOLTON; BRUSH, 2001). Similar to deep space networks the main challenges affecting terrestrial networks can be summarized as follows:

- **Intermittent connectivity**
- **Arbitrarily long and highly variable delays**
- **Highly variable error rates:** the BER varies according to the environment (e.g., deep-sea sensor networks or military/civilian submarine communication). For example, in a wireless sensor network the BER may be on the order of 10^{-1} to 10^{-3} but is frequently lower than in the deep space environments.
- **Data rates:** Data rates are frequently symmetric but may be asymmetric in some environments (e.g., underwater and space communication).

4.2 Selected DTN Congestion Control Mechanisms

For our comparative study, we picked a subset of DTN congestion control mechanisms from the schemes surveyed in (SILVA *et al.*, 2014) that we consider representative of the current DTN congestion control state-of-the-art. More specifically, each one of the selected mechanisms employs at least one of the following techniques: (1) manage the number of message copies in the network; (2) use buffer scheduling (3) use buffer drop policies, and (3) use "store-carry-and-forward" routing. The DTN congestion control schemes we consider in our performance study are: RRCC (Retiring Replicants Congestion Control) (THOMPSON *et al.*, 2010), AFNER (Average Forwarding Number based on Epidemic Routing) (YUN *et al.*, 2010), SR (Storage Routing) (SELIGMAN *et al.*, 2006) and CCC (Credit-based Congestion Control) (LEELA-AMORNSIN; ESAKI, 2010).

Below, we briefly describe each of these mechanisms as follows: for each scheme, we specify the control strategy adopted and the steps performed to avoid or mitigate congestion. Table 4.1 summarizes the four mechanisms based on the taxonomy presented in (SILVA *et al.*, 2014). The column *Routing* indicates whether there is any dependency relationship between congestion control and routing. Congestion control approaches that can work with any routing mechanism are said to be routing-protocol independent; those that cannot are said to be routing-protocol dependent.

TABLE 4.1 – Classification of selected congestion control mechanisms according to (SILVA *et al.*, 2014)

Mechanism	Congestion Detection	Proactive or Reactive Control	Routing	Evaluation Platform
RRCC (THOMPSON *et al.*, 2010)	Drop rate	H	D	ONE
AFNER (YUN *et al.*, 2010)	Network capacity	H	D	NS-2
SR (SELIGMAN *et al.*, 2006)	Buffer availability	R	D	DTNSim
CCC (LEELA-AMORNSIN; ESAKI, 2010)	Buffer availability	H	I	ONE

H: Hybrid R: Reactive D: Dependent I: Independent

4.2.1 Retiring replicants congestion control (RRCC)

The mechanism presented in (THOMPSON *et al.*, 2010) looks to understand the global behavior of congestion in intermittently connected networks. It employs local knowledge to approximate network behavior at a global level and adjusts replication rates at individual nodes accordingly to maximize delivery rates. Basically, RRCC allows each node to react independently limiting the number of messages that can be replicated during each encounter. The authors assert that RRCC congestion control is independent from the routing protocol, not interfering with forwarding decisions. To determine local replication limits, nodes must observe the current level of congestion (congestion value) in the network. In this case, the ratio of drops over message copies is used to track the level of congestion. Whenever the congestion value is updated, the node also adjusts the number of message copies ($limit$) following an additive increase, multiplicative decrease (AIMD) algorithm. The pseudo-code for the RRCC algorithm is shown in Algorithm 1. For our study, we have implemented RRCC taking into account the approximation of global metrics (drops and number of message copies) as suggested by the authors. In the worst case, this algorithm has a cost $O(n)$, where n is the number of exchanged messages during the encounter.

The congestion value is calculated by Equation 4.1, where β is a weight assiged to CV sample. Its value is set in 0.9. According to the algorithm, if the current congestion value $CV^{'}$ is higher than the previous value CV, there is growing congestion and the limit is reduced by a multiplicative factor $MD = 0.2$ whereas when congestion is decreasing the limit is increased by $A_i = 1$ (see this in Algorithm 1 lines 19 and 21). In RRCC, when a node wishes to drop a message, it adopts a random drop policy.

$$CV^{'} = \beta CV_{sample} + (1 - \beta)CV \tag{4.1}$$

4.2.2 Average forwarding number based on epidemic routing (AFNER)

The congestion control strategy called “average forwarding number based on epidemic routing” (AFNER) is described in (YUN *et al.*, 2010). The algorithm is exercised when a node’s stor-

Algorithm 1 Pseudo-code of RRCC Congestion Control Mechanism

```
procedure RRCC
    drops ← 0;
    reps ← 0;
    limit ← 0;
    CV ← Double.MaxValue;
    A_i ← 1;                                   ▷ the replication limit could be increased by a fixed amount
    MD ← 0.2;                                  ▷ mutiplicative factor to congestion value
    if event.equal(DROP) then
        drops ← drops + 1;
    else if event.equal(RECEIVE_MESSAGE) then
        reps ← reps + 1;
    else if event.equal(CONTACT) then          ▷ contact node b
        hopsCount ← message.getHopsCount − 1;  ▷ № of hops this message has passed
        d ← drops + b.drops;
        r ← reps + b.reps + hopsCount;
        reset(drops, reps);
        CV' ← alpha × (d/r) + (1 − alpha) × CV;
        if CV' ≤ CV then
            limit ← limit + A_i;
        else
            limit ← limit × MD;
        end if
        CV ← CV';
    end if
end procedure
```

age is full and the node needs to accept another incoming message. The node randomly drops one of the messages whose *forwarding number* is larger than the network's average forwarding number. The forwarding number of a message is defined as the number of copies of that message, and the average forwarding number is the mean forwarding number of all the messages currently in the network. AFNER assumes that messages are not fragmented and are transmitted in FIFO (First-In First-Out) order from one node to another during the contact period. AFNER is based on two principles:

- First, messages in a node's buffer are sorted in ascending order according to their forwarding number.
- Second, a message will be delivered at ultimate destination if and only if its forwarding number is greater than or equal to the average forwarding number.

We should point out that AFNER's congestion control depends on the average forwarding number which is not easily computable in a real DTN. The pseudo-code of the AFNER algorithm is shown in Algorithm 2, which has been implemented as part of this quantitative study. Note that AFNER relies on epidemic routing. AFNER's computational complexity is given by is $O(n\ log(n)) + m)$ in the worst case, where n is the node's buffer size and m the total number of messages in network.

Algorithm 2 Pseudo-code of AFNER Congestion Control Mechanism

```
1: procedure AFNER
2:     Fnum;                                   ▷ the number of message in the buffer of congested node.
3:     Ft;                                     ▷ the length of the existing messages in the buffer at congested node.
4:     if node a encounter node b then
5:         both nodes share their summary vector (SVa e SVb);
6:     end if
7:     if currentQueueSize ≥ messageSize then
8:         congestion does not occur, the message can be stored;
9:     else                                    ▷ congestion has occurred.
10:        Ft = Fnum × 1;                      ▷ congested node computes Ft.
11:    end if
12:    if (messageSize − currentQueueSize) < Ft then
13:        NumMsgDeleted ← 0;
14:        while NumMsgDeleted ≠ (messageSize − currentQueueSize) do
15:            delete message;                 ▷ according to descending order of the forwarding number.
16:            NumMsgDeleted ← NumMsgDeleted + 1;
17:        end while
18:    else if (messageSize − currentQueueSize) ≥ Ft then
19:        delete all messages whose forwarding number is bigger than Fave
20:    end if
21: end procedure
```

4.2.3 Storage routing (SR)

A strategy of migrating messages to alternative storage locations during congestion is presented in (SELIGMAN *et al.*, 2006). The strategy, named "storage routing" (SR), avoids congesting a node by causing it to forward excess messages out to available neighbors. When nodes that were previously at risk of congestion manage to reduce their buffer occupancy, any messages that were migrated are retrieved. SR operates as a local routing protocol diverting messages from their conventional routing path for later forwarding.

Specifically, SR is a set of algorithms invoked at the DTN router as congestion increases and decreases. SR is invoked when a message arrives at a node that is nearing congestion: SR determines a set of messages to migrate to a set of selected receiving nodes. Nodes selected as migration targets are called alternate custodians. When the risk of congestion subsides, SR invokes a retrieval selection algorithm to determine which nodes it will contact with custody requests for previously migrated messages. If the alternative custodian no longer contains the migrated message a NACK (Negative-acknowledge) message is returned, otherwise the migrated message is returned.

SR can be summarized as follows (see Algorithm 3 for SR's pseudo-code):

Message selection chooses messages to be "pushed" to neighbor nodes (instead of simply discarding them which is the case in traditional drop-based buffer management). Messages are deleted only when there is no available storage at any neighboring node.

Node selection produces a set of nodes to which messages can be migrated. The algorithm selects nodes based on an aggregate migration cost metric ($C_{c,v}(l)$) calculated by Equa-

tion 4.2.

$$C_{c,v}(l) = T_{c,v}(l)w_t + S_v(l)w_s \tag{4.2}$$

where migration cost from custodian node c to a neighborhood node v of a message with length l is the weighted summation of the storage cost $S_v(l)$ and transmission cost $T_{c,v}(l)$. The values of w_t and w_s are set according to the importance of transmission cost versus storage cost. Transmission cost is a function of the latency $L_{c,v}$ and bandwidth $B_{c,v}$ on the path node $c \rightarrow v$ and the message size l. Thus, $T_{c,v}(l)$ can be obtained by equation 4.3.

$$T_{c,v}(l) = log((L_{c,v} + (l/B_{c,v}))/(10^{-6}))/10 \tag{4.3}$$

The storage cost is a function of the available storage for migration to node v and is given by Equation 4.4, where A_v is the available storage and Max_v is the maximum node storage.

$$S_v(i) = \begin{cases} A_v \div Max_v & \text{if} \quad l \leq A_v \\ +\infty & \text{if} \quad l > A_v \end{cases} \tag{4.4}$$

Algorithm 3 Pseudo-code of SR Congestion Control Mechanism

```
1:  procedure SR
2:      bufferSize;                                   ▷ free buffer space.
3:      S ← 0;                                        ▷ counter of the amount of available storage by migrating messages.
4:      if m.size() > bufferSize then
5:          R ← m.size() − bufferSize;
6:          Mp = n.MessageSelection();
7:          x = n.NodeSelection();
8:          if x ≠ null then
9:              n.sendMessage(Mp, x);
10:         else
11:             n.deleteMessage(Mp);
12:         end if
13:         S = Mp.size() + S;
14:         if S ≥ R then
15:             enqueue m at n;
16:         else
17:             go to 7
18:         end if
19:     end if
20: end procedure
```

Taking into account the node selection strategy, SR's worst case cost is $O(n) + k$, where n is the buffer size and k is the number of alternate nodes.

4.2.4 Credit-based congestion control (CCC)

A heuristic-based congestion control mechanisms called credit-based congestion control that handles congestion reactively is presented in (LEELA-AMORNSIN; ESAKI, 2010). In order

to yield high delivery ratio with low number of replicas, CCC tries to delete messages when congestion builds up at a node. Messages are dropped when deemed obsolete by their *time-dependent credit*.

CCC is based on the concept that messages that are obsolete should be deleted first, as implemented by the time-dependent credit mechanism. In addition, CCC also considers that the messages that have been forwarded too many times should be dropped first. To this end, a refilling and refunding technique is applied when node pairs encounter and exchange their messages. Note that this technique might not work if node encounters are not frequent enough.

CCC works as follows: when a message has been generated, it is assigned the maximum amount of credit. As time passes, one credit is decreased at every time unit. When two nodes encounter each other, they exchange messages, and for each message exchanged, the sender deducts a *penalty* value from the copy of the message it keeps in its buffer. The receiver in turn adds a *reward* value to the credit of the copy it just received. Our implementation of CCC uses Equations 4.5 and 4.6 to compute functions for refilling and refunding, respectively.

$$C_{sender}(t+1) = max(C_{sender}(t) - penalty, \\ minimum - value) \tag{4.5}$$

$$C_{receiver}(t+1) = max(C_{sender}(t) + reward, \\ initial - value) \tag{4.6}$$

where C_{sender} is the message's credit at the sender, $C_{receiver}$ is the credit of replicated message at the receiver, before and after the encounter. Operations executed by this mechanism, such as refilling and refunding, do take time, but this time is constant. In the worst case, CCC's cost is $O(n)$, where n is the number of exchanged messages during the encounter.

4.3 Experimental Methodology

We conducted experiments using the ONE (Opportunistic NEtwork) Simulator platform (KERäNEN *et al.*, 2009), which is a discrete event simulator specific for DTN environments. It provides a Graphical User Interface (GUI), mobility models, event generation, message exchange, DTN routing, and application protocols, a basic notion of energy consumption, visualization, analysis, and an interface for importing and exporting mobility traces. The simulator reads a configuration file that specifies the network, i.e., the movement of the nodes and locations of the nodes in the network. Each simulation run generates a trace file containing all the data packets that are sent between the nodes during the course of the simulation. By analyzing this file it is possible to study the performance of the different congestion control schemes, which we have implemented in the ONE simulator.

We use 95% confidence intervals. Results when no congestion control is employed are used as the performance baseline. Furthermore, each configuration based on Tables 4.3 and 4.4 was run $5\ times$ (5 different seeds adding up to 15000 runs for IPN scenario and 45000 runs for terrestrial scenario) and the statistical values (average, standard deviation) were obtained. Note that some graphics do not show the standard deviation, because the value is either zero or approximately zero.

4.3.1 DTN Scenarios

In order to study the performance of the four different congestion control mechanisms described in Section 4.2, we simulate two main DTN application domains, namely data transfer in space and terrestrial communication. The parameters of the ONE simulator and their values are listed in Tables 4.3 and 4.4, respectively. A scenario without congestion control is considered as baseline for our comparative study. Every pair of nodes that are in communication range of one another can transfer data between them if they have data to send. Another interesting aspect of our study is to understand the dependence of the congestion control schemes on the underlying routing protocol (JAIN *et al.*, 2004b). To this end, we use three different routing protocols, namely Epidemic (VAHDAT; BECKER, 2000b), ProPHET (Probabilistic Routing Protocol) (LINDGREN *et al.*, 2003a), and Spray and Wait (SPYROPOULOS *et al.*, 2005a), each of which is briefly described below. It is important to note that like the congestion control mechanisms we study, these routing protocols were not designed to operate in deep space communication. They were designed for terrestrial networking.

- **Spray-and-Wait**: A node sprays copies of messages, such that, the number of message copies is limited to a configurable maximum. In our experiments we set the number of copies to 10.
- **ProPHET**: When a node A encounters a node B, it decides whether to pass to B a copy of a message destined to C based on B's encounter history with C.
- **Epidemic**: A node delivers copies of its messages to all encountered nodes.

4.3.1.1 Interplanetary Network (IPN) Scenario

Our IPN (Interplanetary Network) scenario features high latency (because of astronomical distances) and scheduled contacts, i.e., node encounters that are known a priori. There are five nodes, representing a Base Station on the surface of the Earth which sends data to two rovers on Mars through two satellites located near Mars (see Figure 4.1). Simulation parameters are set to correspond to realistic conditions. As such, links between the satellites and the rovers on

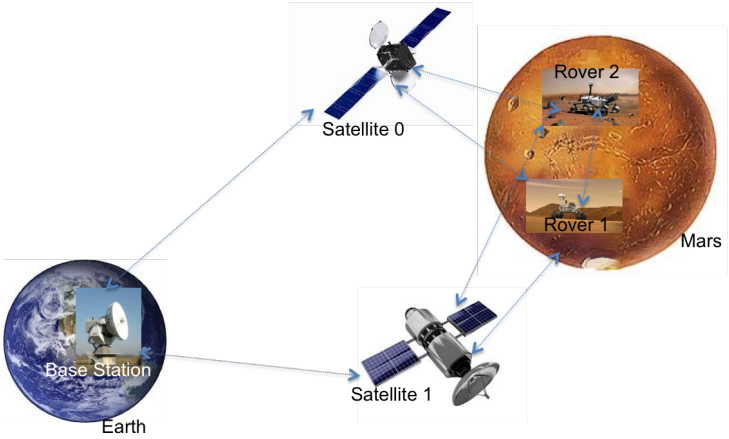


FIGURE 4.1 – Interplanetary network scenario.

the Martian surface are set to 1s propagation delay while the links between the base station on Earth and the satellites at Mars are set to 240s propagation delay.

TABLE 4.2 – Example of scheduled contact table.

Time (s)	Identifier	Initial Node	End Node	State
2000	CONN	Base Station	Satellite 0	up
3000	CONN	Base Station	Satellite 0	down
6000	CONN	Base Station	Satellite 1	up
7000	CONN	Base Station	Satellite 1	down
10000	CONN	Base Station	Satellite 0	up
11000	CONN	Base Station	Satellite 0	down
17000	CONN	Base Station	Satellite 1	up
18000	CONN	Base Station	Satellite 1	down
20000	CONN	Rover 2	Rover 1	up
21000	CONN	Rover 2	Rover 1	down
20000	CONN	Satellite 1	Rover 2	up
21000	CONN	Satellite 1	Rover 2	down
21100	CONN	Satellite 1	Rover 1	up
22100	CONN	Satellite 1	Rover 1	down
25000	CONN	Base Station	Satellite 1	up
26000	CONN	Base Station	Satellite 1	down
30000	CONN	Satellite 0	Rover 2	up
31000	CONN	Satellite 0	Rover 2	down
33000	CONN	Satellite 1	Rover 1	up
34000	CONN	Satellite 1	Rover 1	down
38000	CONN	Satellite 0	Rover 2	up
39000	CONN	Satellite 0	Rover 2	down

We mainly focus on testing how congestion is affected by the inter-contact time and the duration of contact. Thus a scheduled contact table was created (see Table 4.2) which records the time when a connection is created from one node to another node. Also, it records the time when a connection drops between two nodes. The Up and Down connection times are asserted a priori in the scheduled contact table. Note that the communication channel is asymmetric. Although Table 4.2 does not indicate when the connection from *Satellite 0* to *Base Station* is Up and Down, we assume that the state of the connection from *Satellite 0* to *Base Station* is always the same as the state of the connection from *Base Station* to *Satellite 0*. For example, according to the first line of Table 4.2, at time $2000s$, the connection between *Base Station* and *Satellite 0* is "up" but goes"down" at time $3000s$. Then, the same nodes are again in contact at time $10000s$ (line 5) for $1000s$ (line 6).

We use 5 different scheduled contact tables that differ in inter-contact time. We named the

five different contact schedules as Contact1, Contact2, Contact3, Contact4, and Contact5. We increase the inter-contact time by $1000s$ as we go from Contact$i$ to Contact$i+1$.

Parameters		
Name	Description	Value
Scenario.endTime	simulation time	43200 seconds
btInterface.transmitSpeed	bandwidth	2.5 Mbps
btInterface.transmitRange	transmitting range	150 m
Group.router	routing protocol	[EpidemicRouter, ProphetRouter, SprayAndWaitRouter (10 msg copies)]
Group.movementModel	mobility model	StationaryMovement
Group.bufferSize	node buffer size	[1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000] KB
Group.msgTTL	message time to live	43200 seconds
Group.nrofHosts	number of nodes in network	5
Movimentmodel.worldSize	area where simulation takes place	6km x 6km
Events1.size	message size	{50, 100} KB
Events1.interval	Creation interval in seconds, i.e. one new message every 1 to 100 seconds	[1-100, 1-200, 1-300, 1-400, 1-500] seconds

TABLE 4.3 – Simulation parameters and their values for the IPN scenario

Nodes generate messages according to the $Events1.interval$ parameter (see Table 4.3) where its value is $[x_i, x_j) = x_i \leq$ *next-event-time* $< x_j$. Uniformly distributed value between x_i (inclusive) and the x_j value (exclusive). We vary the message generation rate according to the values listed in Table 4.3 to show how this parameter affects the performance of the different congestion control schemes.

4.3.1.2 Terrestrial Scenario

The terrestrial scenarios we simulate include 50 nodes that move according to a pre-defined mobility regime. As they move, nodes encounter one another "opportunistically". During these "opportunistic contacts", nodes can exchange messages. Figure 4.2 shows the output of the ONE simulator's graphic interface illustrating an snapshot of a terrestrial DTN scenario. We assume that all nodes have the same transmission ranges. Table 4.4 lists the simulation parameter settings we used in our experiments.

TABLE 4.4 – Simulation parameters and their values for terrestrial scenario

Parameters		
Name	Description	Value
Scenario.endTime	simulation time	43200 seconds
btInterface.transmitSpeed	bandwidth	2.5 Mbps
btInterface.transmitRange	transmitting range	150 m
Group.router	routing protocol	[EpidemicRouter, ProphetRouter, SprayAndWaitRouter (10 msg copies)]
Group.movementModel	mobility model	[RandomWayPoint, RandomWalk, ShortestPathMapBasedMovement]
Group.bufferSize	node buffer size	1000 KB
Group.msgTTL	message time to live	43200 seconds
Group.nrofHosts	number of nodes in network	50
Group.speed	max and min speed that the nodes must move	{0.5, 1.5}m/s
Movimentmodel.worldSize	area where simulation takes place	1km x 1km (RandomWayPoint, RandomWalk) and 6km x 6km (ShortestPathMapBasedMovement)
Events1.size	message size	{50, 100} KB
Events1.interval	Creation interval in seconds, i.e. one new message every 1 to 100 seconds	[1-100, 1-200, 1-300, 1-400, 1-500] seconds

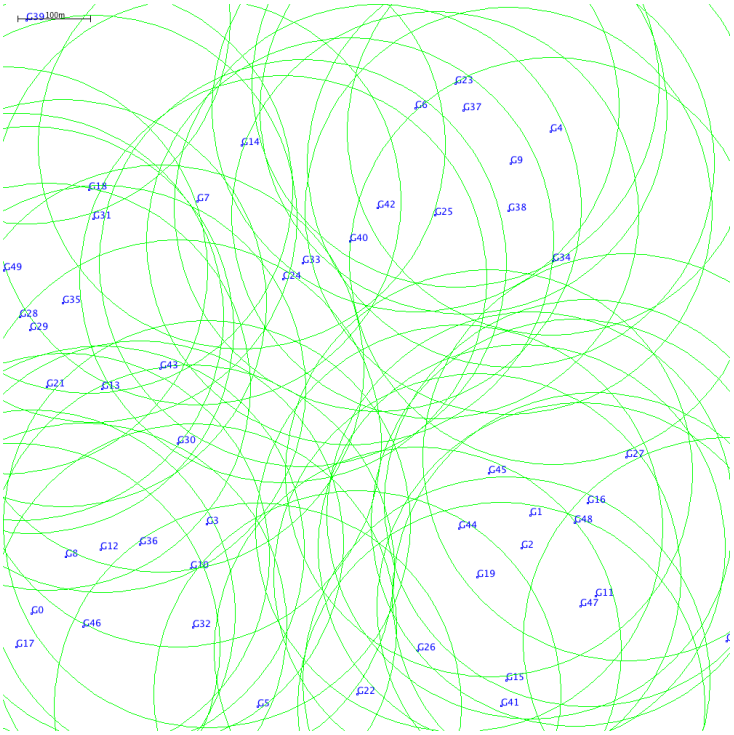


FIGURE 4.2 – Terrestrial network scenario.

Three mobility models were used in the scenario, namely Random Walk, Random Way Point, and Shortest Path Map-Based Movement. In Random Walk (RW) (CAMP *et al.*, 2002), a node moves from current location to a new location for a time interval t by randomly choosing a direction and speed from the predefined ranges. At the end of the time interval a new direction and speed are calculated. The Random Way Point (RWP) (CAMP *et al.*, 2002) model is a generalization of RW. In this model a mobile node stays at a given location for a certain period of time; then, the node moves to a new random destination at a random speed chosen from [0-MAXSPEED]. The Shortest Path Map-Based Movement (SPMBM) (KERANEN; OTT, 2007) model uses Dijkstra's shortest path algorithm to calculate the shortest path from the current location to a randomly selected destination. Destinations can be chosen in a random way or from a set of Points of Interests (POIs). When a node reaches its destination, it waits for the time interval t and then heads toward a new destination by using the shortest path. In SPMBM,

nodes do not move in a completely random way: they follow the shortest paths to reach their destinations.

In our simulations, for the RWP and SPMBM mobility models the minimum and maximum wait time after a node reaches the destination are 0 and 120 seconds, respectively.

4.3.2 Performance Metrics

We consider three main performance metrics, namely:

1. **Delivery ratio** is the ratio between the number of received messages at destination nodes to the number of created messages (see Equation 6.14).

$$\text{delivery ratio} = \frac{\text{number of received messages}}{\text{number of created messages}} \star 100\% \tag{4.7}$$

2. **End-to-End latency** is the average time interval to deliver messages to their destinations (see Equation 6.15 where t_i is defined as the time that message i took to reach the destination and t_c is the message creation time).

$$\text{end-to-end latency} = \frac{\sum_{i=1}^{\text{number of messages received}} (t_i - t_c)}{\text{number of messages received}} \tag{4.8}$$

3. **Overhead** is the difference between the number of delivered messages and the number of relayed messages, divided by the number of delivered messages (see Equation 6.16). The overhead reflects the number of messages relayed to deliver a single message. The lower the value of overhead, the more efficient the strategy is.

$$\text{overhead} = \frac{\text{number of relayed messages} - \text{number of received messages}}{\text{number of received messages}} \tag{4.9}$$

4.4 Simulation Results for Interplanetary Scenario

In this section, we present simulation results from ours IPN experiments. To evaluate the impact of congestion control, we try to generate enough load to congest the network. The goal of the evaluation is to show the performance of each congestion control mechanism relative to the baseline scenario in terms of delivery ratio and message delivery latency. We also consider the impact of the underlying routing protocol; to this end, except for AFNER which uses Epidemic routing, we run each congestion control strategy with different routing protocols, i.e., Epidemic, ProPHET, and Spray and Wait. Tables 4.2 and 4.3 list the simulation parameter settings.

Figure 4.3 shows message delivery ratio as a function of the message generation period. As

expected, for longer message generation periods, i.e., lower message generation frequencies, the delivery ratio increases. However, delivery ratios are quite low since newly generated messages cause older messages to be discarded before they have time to be delivered. At this level of congestion, even though message delivery ratio is quite low, congestion control is able to improve delivery compared when no congestion control is used.

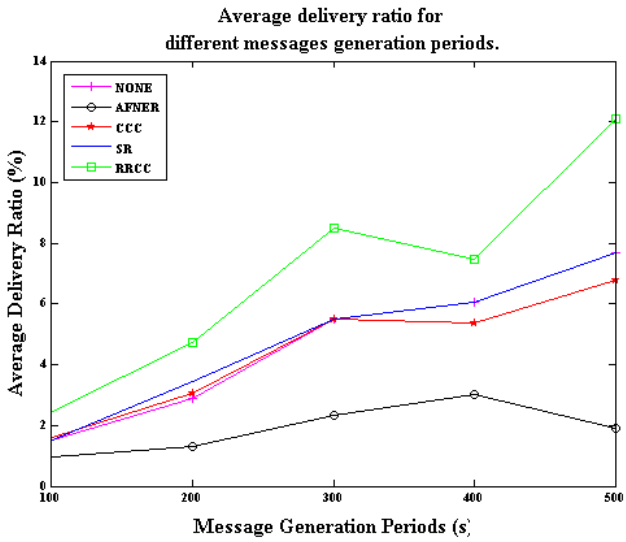


FIGURE 4.3 – Message delivery ratio for IPN Scenario as a function of the message generation period (Epidemic routing, buffer size of $4000k$, transmit speed of $2.5Mbp$s, contact duration $1000s$ and inter-contact time $1000s$).

The goal of the evaluation is to show the performance of each congestion control mechanism relative to the baseline scenario in terms of delivery ratio and message delivery latency. We also consider the impact of the underlying routing protocol; to this end, except for AFNER which uses Epidemic routing, we run each congestion control strategy with different routing protocols, i.e., Epidemic, ProPHET, and Spray and Wait. Tables 4.2 and 4.3 list the simulation parameter settings.

4.4.1 Different Inter-Contact Times

The results presented in this section allow us to analyze the impact on congestion control mechanisms when varying the inter-contact time. Table 4.5 shows an example of the configuration of inter-contact time for one pair of nodes, namely *Base Station* and *Satellite 1*, where column *Inter-Contact Time* depicts the inter-contact times which are managed by *up* connection and its time (columns *State* and *Time(s)*, respectively).

Figure 4.4 illustrates the concepts of inter-contact time and contact duration through an example involving the *Base Station* and *Satellite* nodes. In this example in *Day 1* at time $1000s$ the *Base Station* connects with *Satellite 1* (start of the contact). They remain in contact/ connected until $2000s$ (end of the contact). In this case, the contact duration was $1000s$ (see the

TABLE 4.5 – Inter-contact times for the encounters between *Base Station* and *Satellite 1*.

Tables	Inter-Contact Time	State	Time(s)	Initial Node	End Node
Table-1	1000	up	1000	Base Station	Satellite 1
		down	2000	Base Station	Satellite 1
Table-2	2000	up	2000	Base Station	Satellite 1
		down	3000	Base Station	Satellite 1
Table-3	3000	up	3000	Base Station	Satellite 1
		down	4000	Base Station	Satellite 1
Table-4	4000	up	4000	Base Station	Satellite 1
		down	5000	Base Station	Satellite 1
Table-5	5000	up	5000	Base Station	Satellite 1
		down	6000	Base Station	Satellite 1

solid horizontal line). In the *Day 5* they connect with each other at $5000s$ (start of the contact) and end the connection at $6000s$ (end of the contact). Note that the inter-contact time between the *Base Station* and *Satellite 1* increases for different days (see the dashed line).

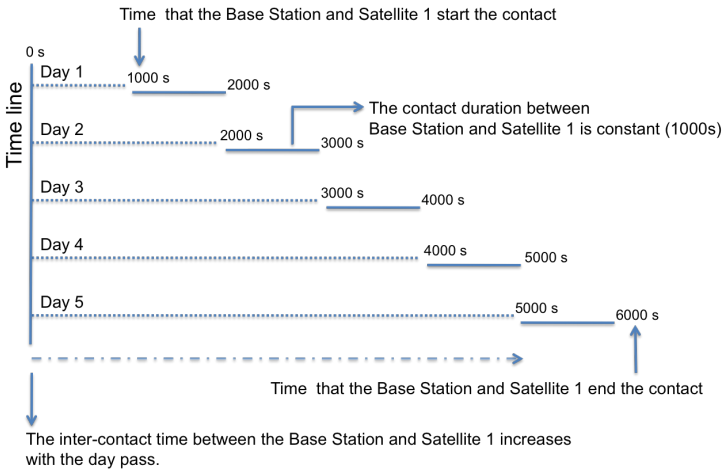
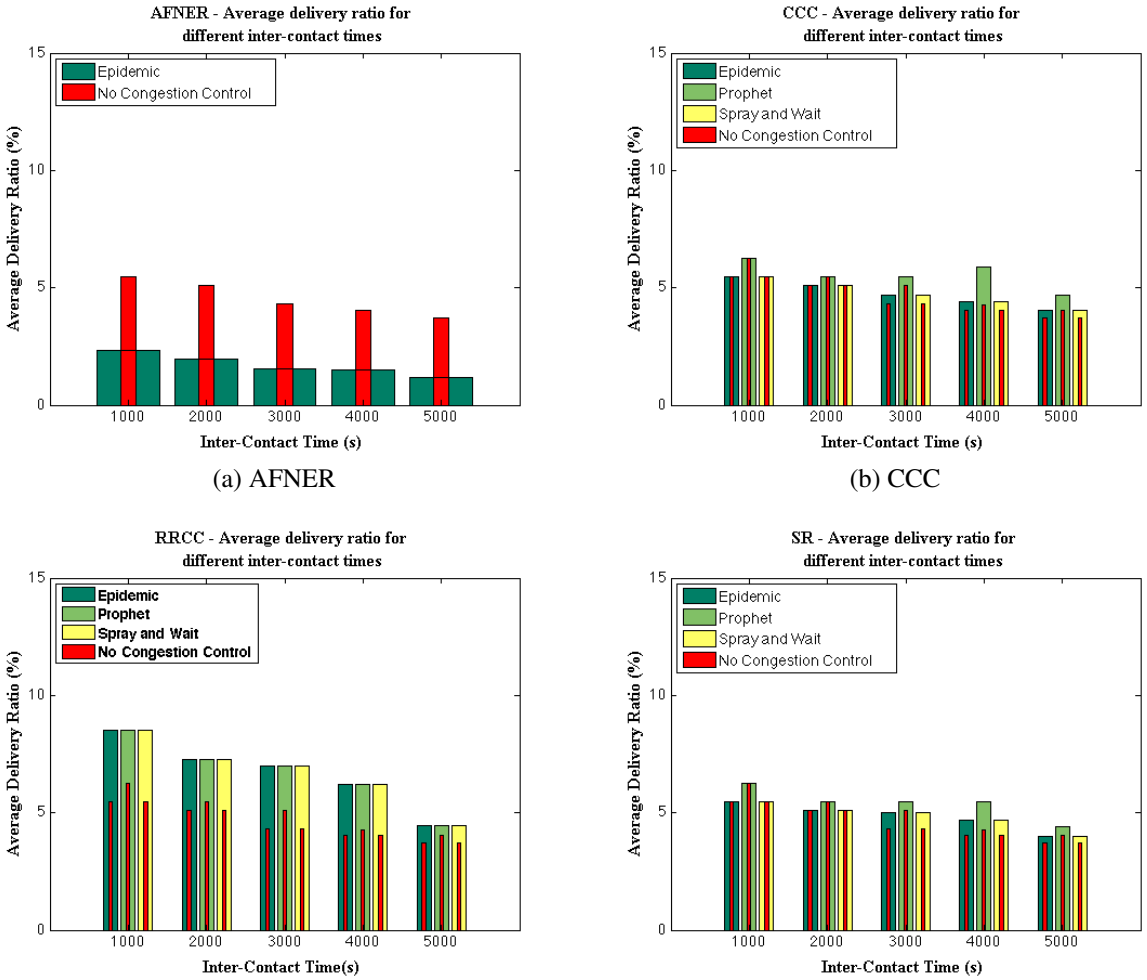


FIGURE 4.4 – Inter-contact time and contact duration for the encounters between *Base Station* and *Satellite 1*.

Figure 4.5 shows the average delivery ratio for different inter-contact times, where the slim bars refer to delivery ratio values when no congestion control is used. The delivery values confirm that for longer inter-contact times, the delivery ratio decreases. This happens because of the lower number of transmission opportunities, which results in longer data queues and consequently higher probability of data being dropped as node buffers fill up. As a result, the data delivery ratio decreases.

We observe that CCC's performance in terms of delivery ratio (Figure 4.5b) is quite similar to SR's (Figure 4.5d). Note that RRCC is routing protocol-independent since its performance is not affected by using different routing protocols (see Figure 4.5c); the difference in performance is very small. On the other hand, CCC's and SR's delivery ratios see a slight increase when ProPHET is used as the underlying routing protocol. Our hypothesis is that, since ProPHET bases its routing decisions on past contact history, it benefits from scenarios where contacts are repetitive, which is often the case in the scheduled behavior of IPN environments. In the case of Epidemic and Spray and Wait routing protocols the opportunistic contacts fit better. We argue



(a) AFNER (b) CCC

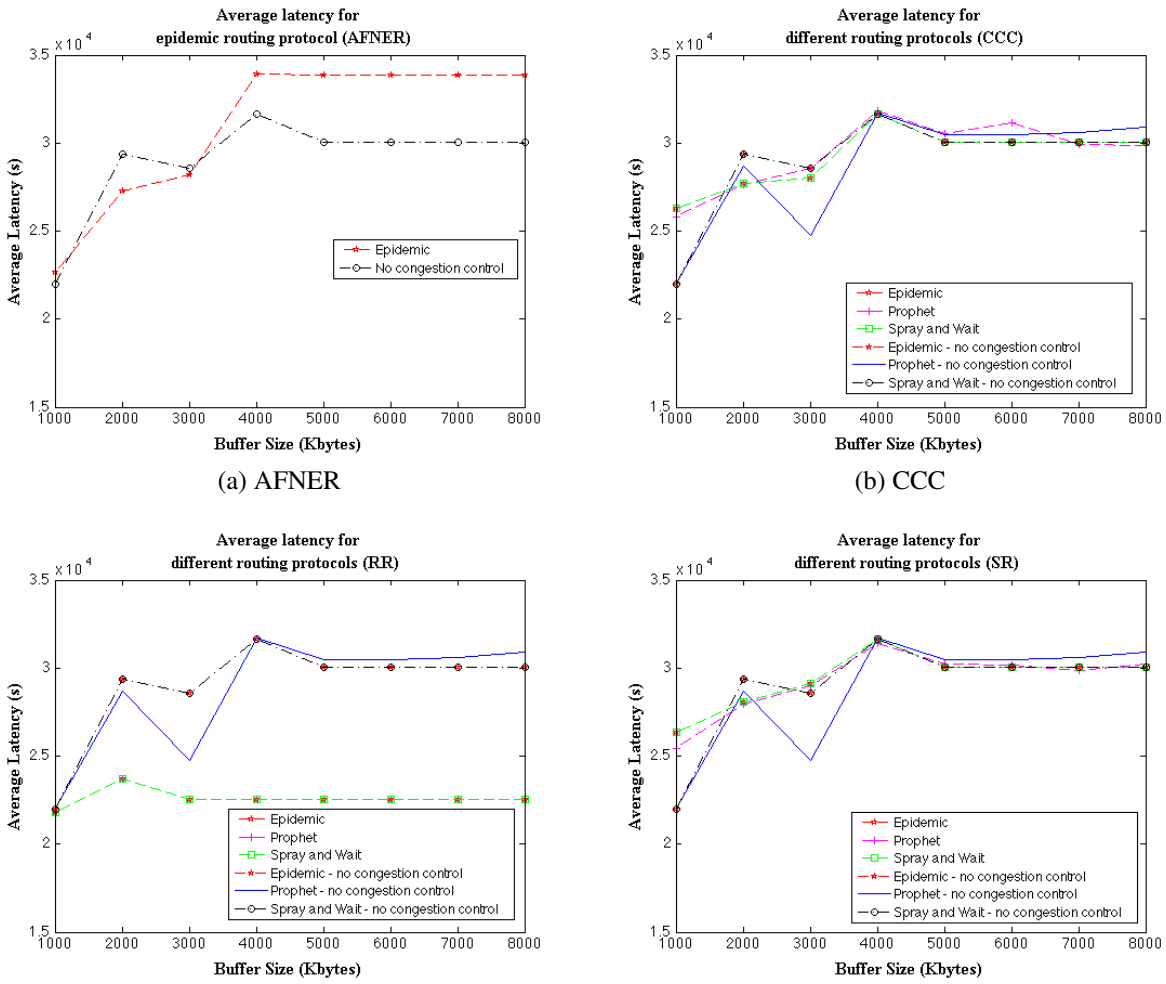
(c) RRCC (d) SR

FIGURE 4.5 – Average delivery ratio for different inter-contact times (IPN scenario, transmit speed of $2.5Mbps$, buffer size of $4000k$, contact duration $1000s$ and message generation period of $300s$).

that, due to the high congestion levels, the increase in delivery ratio is not substantial and is not consistent across all contact scenarios.

It is worth pointing out that, except for AFNER, as inter-contact times increase, the congestion control mechanisms yield greater delivery ratios when compared to the configuration that lacks congestion control altogether. AFNER uses the network's average forwarding number to mitigate congestion. Thus when the inter-contact times increase, more messages are waiting to be forwarded. As a result, buffers fill up and AFNER starts to discard messages based on the average forwarding number. This leads to lower delivery ratios. Note that for longer inter-contact times the nodes' buffer can fill up before the contact begins. This happens because the nodes are able to generate messages according to the message generation ratio. Essentially, delivery and forwarding happen only during a contact event. Therefore, the probability that a node's buffer is full rises with the increase of the inter-contact time. RRCC mechanism responds by randomly discarding messages. This reduces the delivery ratio. On the other hand, when the congestion

takes places during the nodes' contact intervals RRCC manages the message replication. It performs this management based on the previous number of drops and replication (which can be local information or an estimate generated from the node encounters). By limiting message replication, RRCC increases the delivery ratio when compared with the other mechanisms. CCC is totally based on message discard and its strategy of descreasing messages' credit results in older message discard. Consequently, obsolete messages and those that have been forwarded many times are dropped first. This approach helps to improve the delivery ratio when increasing inter-contact time. The improvement in delivery ratio achieved in case of SR is due to the message migration process, if there are neighbor nodes with space available. However, this leads to high delays and cannot work if no available node exists. In the latter case, SR starts to discard messages.



(a) AFNER (b) CCC

(c) RRCC (d) SR

FIGURE 4.6 – Average latency for different buffer sizes (IPN scenario, transmit speed of $2.5Mbps$, inter-contact time $1000s$, contact duration $1000s$ and message generation period of $300s$).

Figure 4.6 shows the average message delivery latency of selected congestion control mechanisms with different routing protocols. We can see that the latency presents fluctuations when

increasing the node buffer size.

One interesting observation is that the minimum latency value is approximately $2.3 \times 10^4 s$, that is, a hundred times greater than the propagation delay for the links between the base station on Earth and the satellites at Mars which were set up to $240s$. We argue that this happens due to the longer inter-contact times and the congestion effect. The IPN environment is subject to high latency and the contacts between nodes are normally scheduled or probabilistic. In this case, for the modeled scenario, the latency is not expected to decrease linearly with increasing buffer size, but rather to remain unchanged or decrease with some fluctuations. This can be explained by the fact that when buffer size is increased the message has a higher probability of being delivered, because the chance of the message being discarded due to buffer overflow is reduced. Thus, depending on the arrival time in the buffer and the time of the next scheduled contact, the message can remain either more or less time in the buffer before being forwarded. In addition, when the buffer is small, the forwarding mechanism drops the oldest messages which are then excluded from the latency computation. Therefore, the delivery delay fluctuates between $1000\ Kbytes$ and $5000\ Kbytes$. After $5000\ Kbytes$ the latency has a tendency to remain constant. Interestingly, there is a point at which buffer size variation does not give any better performance. Note that with increasing node storage, competition for local storage is supposed to decrease so that SR is not invoked as often. As a result the overhead introduced by the SR migration process (see Section 4.2) is not significant and so the delivery delay decreases. In the case of AFNER and CCC the message flow increases and therefore message delivery delay decreases. In addition, AFNER and CCC can get more accuracy in their congestion control parameters due to the increase in message forwarding. Observe that, in the case of RRCC, by limiting replications the congestion control increases the message delivery delay. But as we increase the buffer size, messages that otherwise would not be delivered are.

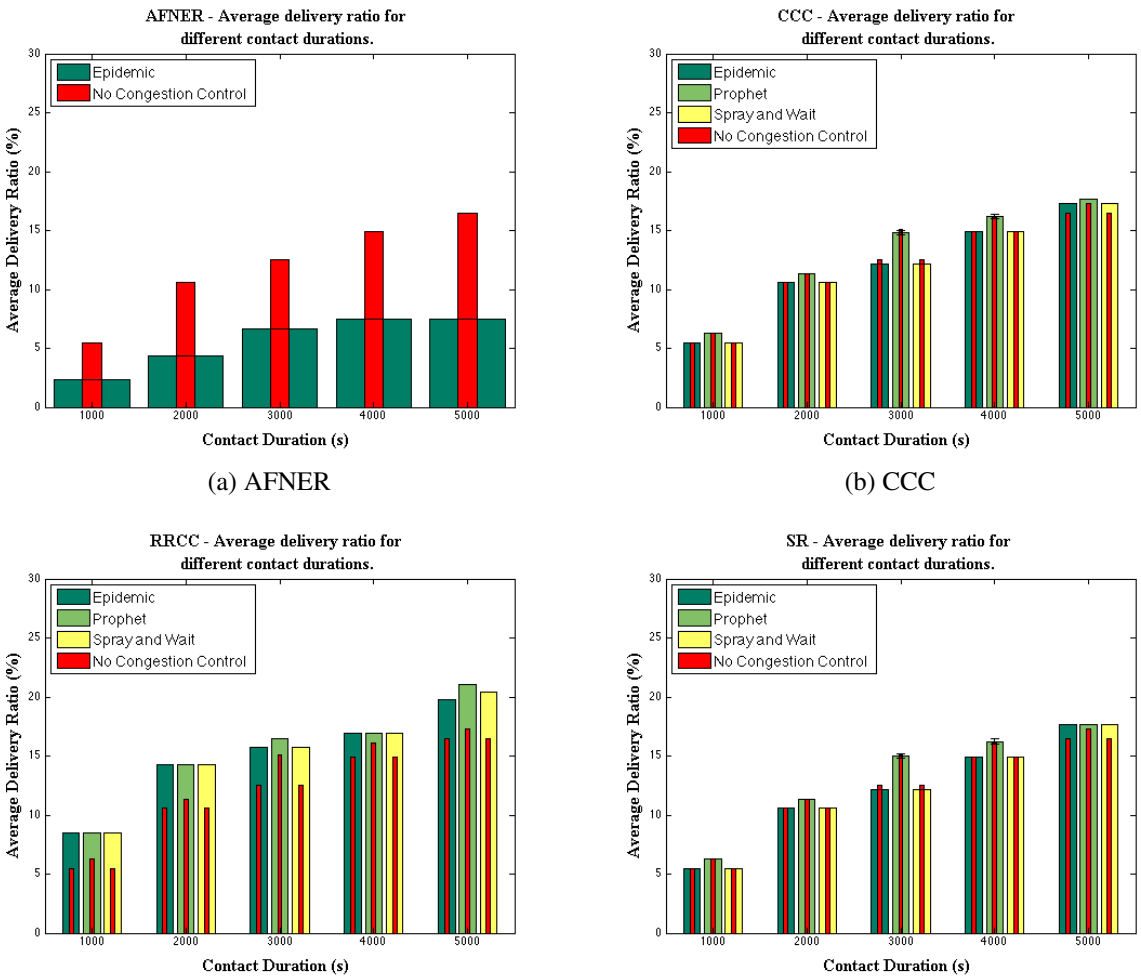
4.4.2 Different Contact Durations

In the next set of experiments, we use the same configuration parameters shown in Table 4.3. However, we introduce here a new parameter named contact duration. We fix the inter-contact time and vary the duration of the contacts. Table 4.6 shows an example of these configurations for one pair of nodes, namely *Base Station* and *Satellite 0*, where column *Duration* shows the contact duration which is managed by the *up* and *down* connection attributes and their times (columns *State* and *Time(s)*, respectively).

Figure 4.7 confirms that increasing the contact duration, thus making it possible to transfer more messages, increases the delivery ratio for all congestion control schemes. It is interesting to observe that RRCC yields higher delivery ratios when compared to AFNER, CC, and SR. We contend that RRCC's proactive AIMD algorithm contributes to RRCC's improved performance because it takes action before the congestion takes place. This reduces the number of message

TABLE 4.6 – Contact duration for the encounters between *Base Station* and *Satellite 0*.

Tables	Duration	State	Time(s)	Initial Node	End Node
Table-1	1000	up	2000	Base Station	Satellite 0
		down	3000	Base Station	Satellite 0
Table-2	2000	up	2000	Base Station	Satellite 0
		down	4000	Base Station	Satellite 0
Table-3	3000	up	2000	Base Station	Satellite 0
		down	5000	Base Station	Satellite 0
Table-4	4000	up	2000	Base Station	Satellite 0
		down	6000	Base Station	Satellite 0
Table-5	5000	up	2000	Base Station	Satellite 0
		down	7000	Base Station	Satellite 0



(a) AFNER (b) CCC

(c) RRCC (d) SR

FIGURE 4.7 – Average delivery ratio for different contact durations (IPN scenario, transmit speed of $2.5Mbps$, buffer size of $4000k$, inter-contact time $1000s$, and message generation period of $300s$).

copies per node. These results also show the routing protocol independence feature observed in previous experiments.

4.5 Simulation Results for Terrestrial Scenario

This section presents the experimental results obtained from the terrestrial scenario. Specifically, the experimental scenario was considered with the selected congestion control mechanisms and is different from the IPN scenario because the terrestrial scenario uses different mobility models and a large number of nodes.

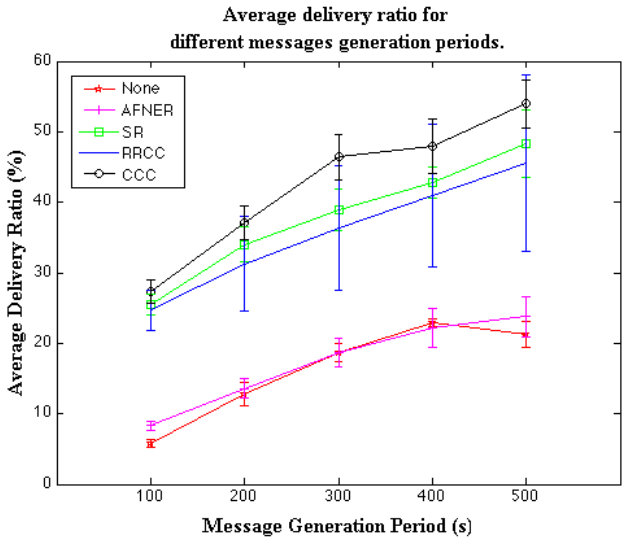


FIGURE 4.8 – Message delivery ratio for terrestrial scenario as a function of the message generation period (Epidemic routing, buffer size of 1000k, transmit speed of 2.5Mbps and RWP mobility model).

Figure 4.8 shows message delivery ratio as a function of the message generation period. As expected, for longer message generation periods, i.e., lower message generation frequencies, the delivery ratio increases. As expected, the results have similar trends as we have seen for IPN scenario. When comparing the results obtained from the IPN scenario (see Figure 4.3) with the terrestrial scenario, we clearly note that delivery ratios are superior for the terrestrial scenario since the selected mechanisms were designed for the terrestrial environment, as were the routing protocols. It is important to note here that the opportunism[1] of the routing protocols works better with a large number of nodes (we use 50 nodes in the terrestrial scenario and 5 nodes in the IPN scenario), enabling more opportunities for forwarding to arise. Therefore the average delivery ratio in the terrestrial scenario is greater than for the IPN scenario. Observe that the CCC mechanism exhibits a higher average delivery ratio than the other mechanisms (AFNER, RRCC, SR). We believe CCC's higher delivery ratio is due to its hybrid (reactive and proactive) control. The proactive mode is prospective or future-oriented, helping to prepare the network for upcoming congestion through the predictive use of context (the refilling and refunding tech-

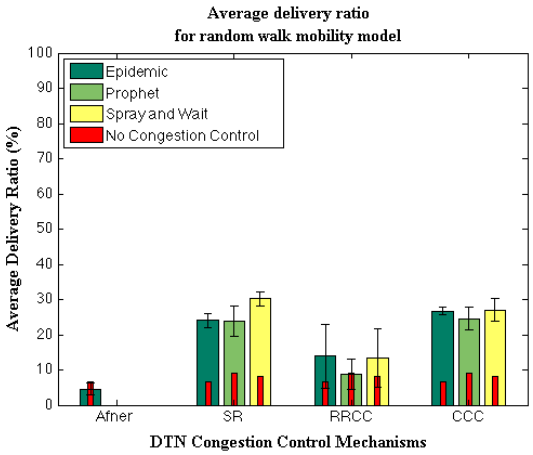
[1]The word *opportunism* refers here to the routing protocol action guided primarily by self-interested motives. For instance, the routing protocol assumes the network has a large number of nodes resulting in a large number of contact opportunities and thus better opportunities to forward each message to its destination.

nique of CCC, see Section 4.2). In contrast, reactive control is retrospective, responding to the presence of salient or imperative congestion by engaging control only if needed, via reactivation of previously stored information (messages' credit). We believe that the good performance of CCC is due to the effective use of its credit-based strategy, that is, the CCC mechanism adopts a message self-contained congestion control. Furthermore, its refilling and refunding technique benefits from the large number of contact opportunities that arises in the experimental scenario. Note that AFNER has the worst performance (the smallest average delivery ratio) even when compared to the baseline scenario (see Figure 4.8 *None* curve). This happens because AFNER requires information about all nodes in the network. Getting this information poses certain challenges. For instance, the network connectivity changes dynamically which causes the network to be intermittently partitioned. Ideally we would like to have global network information to do congestion control, but this is not only inefficient and costly but usually unfeasible in DTN environments.

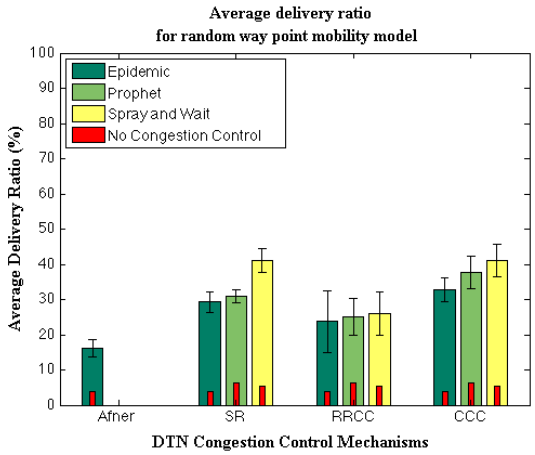
Node mobility is a critical factor influencing the performance of mobile networks. DTNs are no exception: the way nodes move determines their connectivity and thus impact their ability to relay messages.

Figure 4.9 shows the average delivery ratio for different mobility models and routing protocols, for each of the selected congestion control mechanism. As shown in Figure 4.9, we can see the influence of different mobility models on the average delivery ratio. In particular, when RW and RWP are considered, the delivery ratio is small. When we change the mobility model to SPMBM, the average delivery ratio for different protocols has a trend to increase. It is expected that the scenario with SPMBM (a denser network) has more contact opportunities. Thus, more messages can be forwarded, increasing the delivery ratio. However, the difference is minimal when we compare with RW and RWP. We argue that the delivery ratio becomes lower because the buffers in the intermediate nodes on the shortest path may already be at maximum capacity. As a result, messages are discarded and the delivery ratio decreases. Furthermore, observe that AFNER performs worse than other mechanisms when compared to the baseline scenario. AFNER uses the network's average forwarding number to control congestion. Therefore, for networks that are more sparse (as is the effect of the RW and RWP mobility models) there is a large probability of disconnection (network partition). For this reason, more messages are waiting to be forwarded. As a result, buffers fill up and AFNER starts to discard messages based on an inaccurate network average forwarding number, which leads to a low delivery ratio. Unsurprisingly, this behavior confirms that the use of network global information to mitigate congestion in DTN is not a good design principle.

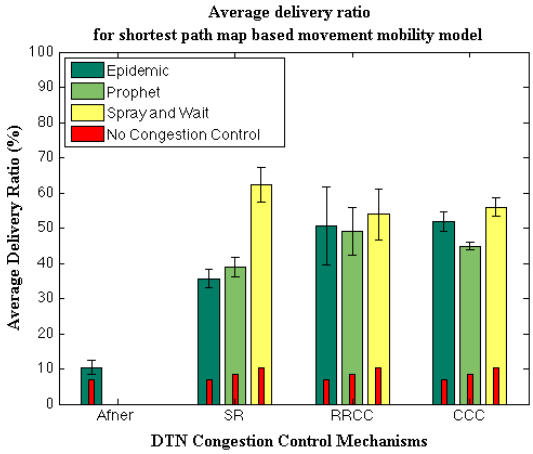
As depicted in Figure 4.9, for all mechanisms except AFNER (which was designed to operate over epidemic routing protocol), when different routing protocols are used the average delivery ratio values vary. Therefore, in the view of the taxonomy presented in (SILVA *et al.*, 2014), RRCC, CCC and SR can be classified as routing-protocol-dependent (since they regis-



(a) Random Walk



(b) Random Way Point



(c) Shortest Path Map Based Movement

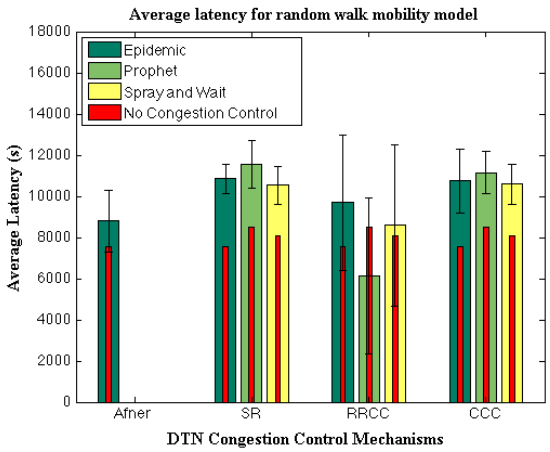
FIGURE 4.9 – Average delivery ratio per congestion control mechanism for different mobility models (terrestrial scenario, transmit speed of $2.5Mbps$, buffer size of $500k$ and message generation period of $300s$).

ter different delivery ratio values for different routing protocols). It is noticeable that the CCC mechanism performs better with RW and RWP mobility models. When SPMBM model is used, CCC and RRCC exhibit similar behavior. We argue that their control strategies benefit from a denser network where there are more contact opportunities. However, the RRCC mechanism in the IPN scenario exhibited a routing-protocol-independent behavior (see Figures 4.5c and 4.7c). In addition, we can immediately observe that the RRCC mechanism has good performance in the IPN scenario but it does not exhibit the same performance in the terrestrial scenario when compared with the other mechanisms. It is clear that the CCC mechanism performs better in the terrestrial scenario. From this we infer that the opportunism of the routing protocols which have been designed for terrestrial environments clearly affects the congestion control mechanism's performance. In particular, the opportunism of the routing protocols (epidemic, prophet, spray and wait) works better with a large number of nodes, enabling more opportunities for forwarding to arise. As evident inthis result, we can summarize that there is no congestion control

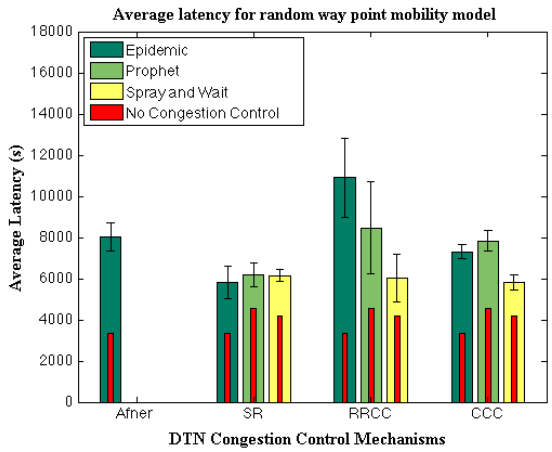
mechanism that is able to operate with similar performance in both scenarios.

In Figure 5.5b, we investigate the performance of the selected mechanisms when the RWP mobility model is used. Large values of average delivery ratio can be observed for all congestion control mechanisms when compared with the baseline scenario. Looking at Figures 5.5a and 5.5b, we can immediately see that the mobility model has significant impact on the congestion control mechanisms. When RW and RWP are used, the mechanisms register a large delivery ratio for RWP. This is because in the RWP model the mobile nodes are more likely to cluster in the geographical center of the simulation. Therefore , there are more contact opportunities and thus more messages are forwarded. This increases the delivery ratio. At the same time, in the RWP model, node mobility and frequent but shorter contacts lead to higher buffer fill ratio, and thus more queuing. Overall, the gain in delivery ratio may or may not come with reduced delay. Specifically, SR and CCC mechanisms have better performance when comparing with other mechanisms, and note that they do not present good performance in the IPN scenario. The speed distributions in the RWP lead to a situation where at the stationary state each node stops moving. In this case, the movement executed by the nodes is similar to the scheduled movement in the IPN scenario. We argue that this leads to the good performance of the RRCC mechanism in both scenarios in a particular case. Therefore, in view of the above comparison, it may be feasible to have different congestion control mechanisms for each scenario.

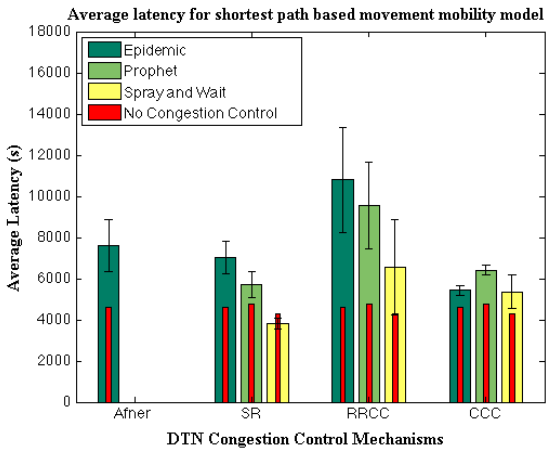
Despite the network's higher density resulting from the SPMBM mobility model, we observe large average message delivery latency values as shown in Figure 4.10c. This is for the reason that messages may be forced to take longer routes (leading to larger delays) because well-connected nodes, which would enable shorter paths, may already have full buffers and thus cannot be chosen as next hops. Observe that mobile nodes choose random directions when using RW (Figure 4.10a) or RWP (Figure 4.10b). In these trials, messages may travel over longer paths before arriving at their destinations, increasing the overall message average latency. The average latency decreases for SPMBM. The SPMBM model is an improved version of the RWP model, where nodes choose random destination decision points and move to those destinations following the map-based shortest path. Thus mobile nodes move less (due to the shortest path computation) when using SPMBM mobility model and, again, this model provides a denser network scenario in comparison to the RW and RWP, which are relatively sparse. As a result, in the SPMBM model the contacts are more frequent. Thus more messages are forwarded, reducing delivery delay and increasing delivery ratio. In all three mobility models, the combination of epidemic routing protocol and the AFNER mechanism results in relatively low average latency. AFNER's operation of dropping messages leads to reduced opportunity to forward older messages. The messages which have been discarded are those that have more copies in the network. As a result, newly arriving messages can be forwarded to their destination more rapidly, minimizing delivery latency. Again, though, AFNER uses the network's average forwarding number to decide which message to discard. This information can often be



(a) Random Walk



(b) Random Way Point



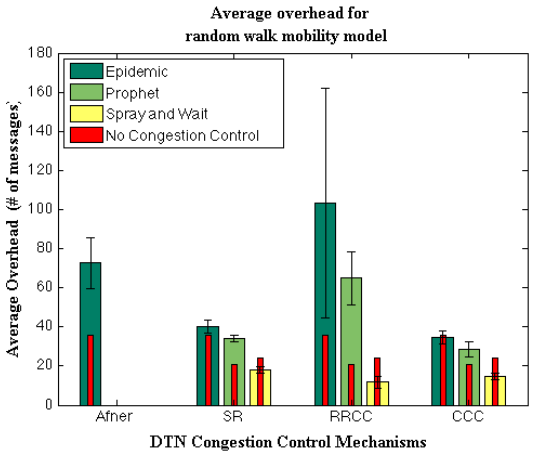
(c) Shortest Path Map Based Movement

FIGURE 4.10 – Average latency per congestion control mechanism for different mobility models (terrestrial scenario, transmit speed of $2.5Mbps$, buffer size of $500k$ and message generation period of $300s$).

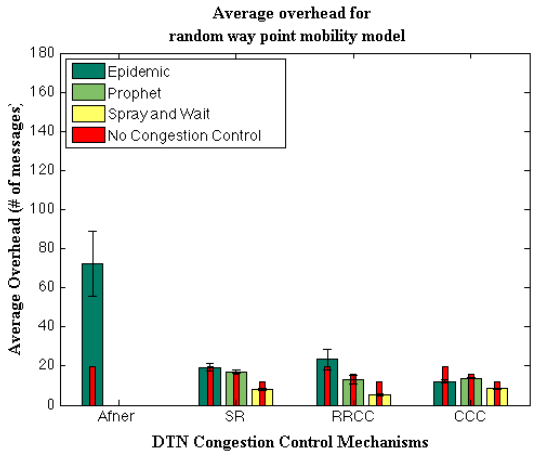
outdated because of intermittent connectivity, resulting in a low delivery ratio.

Figure 4.11 shows *overhead* incurred by each congestion control scheme under different routing regimes. It provides a measure of the average number of message transmissions required to deliver a message from the source to its destination. As such, the overhead graphic indicates the amount of network resources needed to deliver a message to its destination.

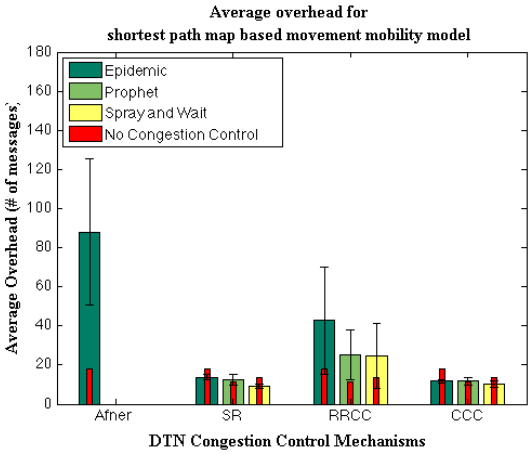
We expect that before congestion occurs, additional replicas will increase the probability that messages get delivered. However, as the network gets congested that relationship changes as more replicas mean more congestion and thus lower message delivery probability. As we have seen, Epidemic Routing results in the transmission of many more message copies within the network, compared to the Prophet and Spray and Wait protocols. This is because, under Epidemic Routing, each node replicates a message every time it encounters another node that does not have a copy of the message. This drastically increases the number of times a message



(a) Random Walk



(b) Random Way Point



(c) Shortest Path Map Based Movement

FIGURE 4.11 – Average overhead per congestion control mechanism for different mobility models (terrestrial scenario, transmit speed of $2.5Mbps$, buffer size of $500k$ and message generation period of 300s).

is relayed to intermediate nodes. We also observe that AFNER's overhead is higher than the overhead incurred by the other mechanisms. This behavior can be attributed to AFNER's use of Epidemic Routing as well as the fact that AFNER reactively discards messages based on the network's average forwarding number when congestion is detected. As a result, messages can be replicated and then until congestion has been mitigated. This increases the number of relayed messages in the network considerably while the number of delivered messages remains the same which increases the overhead. In other words, it increases the amount of network resources that are used to deliver one message to its destination. Note that for different mobility models the lower overhead values of the congestion control mechanisms yield higher delivery ratio values (see Figure 4.9).

4.6 Discussion

In order to evaluate the selected mechanisms we implemented them as closely as possible to the original specifications published by the authors. However, the results achieved varying levels of success. AFNER was originally implemented in NS-2 using 50 nodes that move randomly at a speed of $20m/s$. In the results reported, AFNER registered equal or worse performance compared with the baseline scenario. For instance, when the buffer size is great or equal to 600 messages the delivery ratio is lower and the latency is higher than the values of baseline scenario. It performed worse than when no congestion control is used in both IPN and terrestrial scenario. In the case of RRCC the authors evaluated the mechanism's performance based on the behavior of delivery ratio when the buffer size and message generation period vary. In addition, 100 nodes were used. Unsurprisingly, the delivery ratio increased for large message generation periods. We observed the same behavior. However, the delivery ratio was higher in the terrestrial scenario and lower in the IPN scenario. The performance of the SR mechanism, in its original specification, was evaluated varying the buffer's capacity where 40 nodes and message generation rate of $2\ messages/s$ were used. This mechanism depends on available buffer space in the neighbors to work. Clearly, we can conclude that increasing the buffer's capacity improves the delivery ratio. But there is a point at which for any increase on the buffer's capacity the delivery ratio remains constant. CCC mechanism also has its performance evaluated varying the buffer's capacity and its impact on the delivery ratio. The authors use 100 nodes, node movement speed varying from $0\ km/h$ to $5\ km/h$, and a message generation period of 30s. Despite the variations from the original scenario where each mechanism has been evaluated, we were able to reproduce the control strategy of these mechanisms in our scenarios. Several experiments were performed to determine whether or not introducing variations in the inter-contact times, contact duration, mobile node movement model and routing protocol affect the overall performance in terms of delivery ratio. As expected, it was observed that the mechanisms behave differently in accord with variation in these parameters. Mainly the scenario characteristics seem to determine which mechanism will be more successful.

4.7 Conclusion

This chapter evaluated the performance of four DTN congestion control mechanisms which represent the state-of-the-art on DTN congestion control based on the survey we presented in (SILVA *et al.*, 2014). We examined performance in terms of delivery ratio, latency, and overhead. We also evaluated the congestion control schemes in terms of their routing protocol independence. As baseline, we use the same simulation scenarios without any congestion control. Although the selected congestion control mechanisms were not originally proposed for IPN scenarios, we argue that this kind of quantitative study provides useful insight to help guide the

design of effective and interoperable DTN congestion control mechanisms.

Our study shows that, since messages may be buffered for long periods of time before being acknowledged, buffer overflow becomes highly common; this results in excessive latencies and message losses which is aggravated by the fact that reactive congestion schemes such as AFNER and SR would be severely impacted by delayed control decisions. Our results show that adopting a proactive or hybrid approach as done by RRCC and CCC may significantly improve performance; this appears to be an interesting strategy for future DTN congestion control mechanisms.

One interesting conclusion we must point out is that, due to short contact duration and longer inter-contact times, message expiration itself provides a degree of congestion control in the IPN scenario. In IPN operations, messages may expire while propagating to their destination or remain a long time in buffers waiting to be forwarded, so early discarding of those message may reduce buffer occupancy.

Our results also indicate that DTN routing protocols have significant impact on congestion control mechanisms' performance (e.g. SR, RRCC and CCC in the terrestrial scenario and SR and CCC in the IPN scenario). Patterns of mobility show similar impact. The relation between routing and mobility appears clear here. Therefore, the design of more intelligent, efficient, and routing-protocol-independent congestion control mechanisms is of particular interest and should be considered.

In summary, our results indicate that congestion control helps to increase message delivery ratio, even in highly congested network scenarios. Furthermore it indicates that good design principles for congestion control in DTN scenarios include: using a combination of reactive and proactive control as well as using local information instead of relying on global knowledge. Accordingly, designing a congestion control mechanism that is routing protocol-independent helps with interoperability and applicability to a wide variety of DTN scenarios. A key challenge in DTN congestion control is to create a scheme that can provide good delivery performance and low delivery delay in a network of opportunistic or scheduled intermittent contacts. Apart from that, we investigated in the next chapter in details the DTN congestion problem. More specifically we formally model the DTN congestion problem based on percolation theory and we report extensive experiments using both MatLab and the ONE simulator. Percolation is an particular approach that studies network connectivity and an interesting observation about percolation, is that it can model various phenomena in terms of simple geometric considerations.

5 A Percolation-Based Approach to Model DTN Congestion Control

In this chapter, we propose a novel modeling framework to study congestion in delay- and disruption tolerant networks (DTNs). The proposed model is based on directed site-bond percolation where sites represent space-time position of DTN nodes, and bonds are contact opportunities, *i.e.* communication links that can be established whenever nodes come in range of each other. To the best of our knowledge, this is the first model of DTN congestion using percolation theory. The proposed modeling framework is simple yet general and can be used to evaluate different DTN congestion control mechanisms in a variety of scenarios and conditions. We validate our model by showing that its results match quite well results obtained from the ONE DTN simulation platform. We also show that our model can be used to understand how parameters like buffer management policy, buffer size, routing mechanism, and message time-to-live affect network congestion.

5.1 Introduction

Delay- and disruption-tolerant networks (DTNs) refer to networks that are characterized by intermittent connectivity, long delays, and often constrained bandwidth. They were originally motivated by space exploration and its need for deep space communication (JPL-NASA, 1998); however, over time, a diverse set of DTN applications for *extreme* environments have emerged including vehicular networks (GERLA; KLEINROCK, 2011), emergence response and military operations (STERBENZA *et al.*, 2010), surveillance (STERBENZA *et al.*, 2010), tracking and monitoring applications (LEE *et al.*, 2006), as well as bridging the digital divide (PELUSI *et al.*, 2006). In these scenarios, long delays arise as a consequence of either the fact that distances are long or that connections are episodic.

The DTN architecture proposed in (FALL *et al.*, 2003) addresses message delivery challenges posed by intermittently connected networks using the *store-carry-and-forward* paradigm where messages are forwarded only when a contact opportunity arises. As a result, messages might remain stored for long periods of time in persistent storage at intermediate nodes before reaching

their destinations. Because of the inability to guarantee end-to-end connectivity at all times, congestion may build up in the network. As a result, persistent storage at nodes may fill up and eventually overflow, causing data loss and consequently network performance degradation.

Understanding network congestion has motivated a number of research efforts that focus on modeling network behavior under congestion. Existing models employ a variety of techniques including renewal theory (PADHYE *et al.*, 1998) (MATHIS *et al.*, 1997), fixed point methods (CASETTI; MEO, 200), fluid models (MISRA *et al.*, 2000), financial models (BURLEIGH *et al.*, 2006), Markov chains (CELLO *et al.*, 2012), and control theory (CAVENDISH, 2004). In this work, we develop a simple, yet general mathematical framework to model congestion in DTNs based on percolation theory (BROADBENT; HAMMERSLEY, 1957). Our goal is to use the resulting modeling framework to evaluate and validate existing and future DTN congestion control mechanisms. An important feature of the proposed percolation model is the fact that, instead of requiring global knowledge about the whole network, it relies exclusively on local information, i.e., information related to a node and its neighbors. As will become clear in the remainder of the chapter, formulating the DTN congestion problem as a percolation process happens quite intuitively and the resulting percolation model is simple, general, and easy to derive.

To the best of our knowledge, our work is the first to model DTN congestion using percolation theory. The proposed model defines the probability of delivering a message between a given source-destination pair as the probability there exists a path between the source and destination containing only non-congested nodes (i.e., nodes that contain available buffer space for arriving messages) and active links (nodes in contact with one another). As a result, our model yields the following network congestion related metrics: average buffer occupancy, average buffer blocked time, average message delivery probability, and average delivery delay. Ultimately, our goal is to use the proposed model to evaluate DTN congestion control mechanisms in terms of how cost-effective they are in preventing/containing congestion.

5.2 DTN Congestion and Percolation

In this section, we discuss DTN congestion and introduce definitions, notations, and results of percolation theory which will be used in our model. More details including proofs of percolation results discussed here can be found in (GRIMMETT, 1999).

5.2.1 DTN Congestion

The challenges of controlling congestion in DTNs are mainly due to two reasons: (1) end-to-end connectivity between nodes cannot be guaranteed, and (2) arbitrarily long latency caused by high propagation delays and/or intermittent connectivity. Take for example the deep space

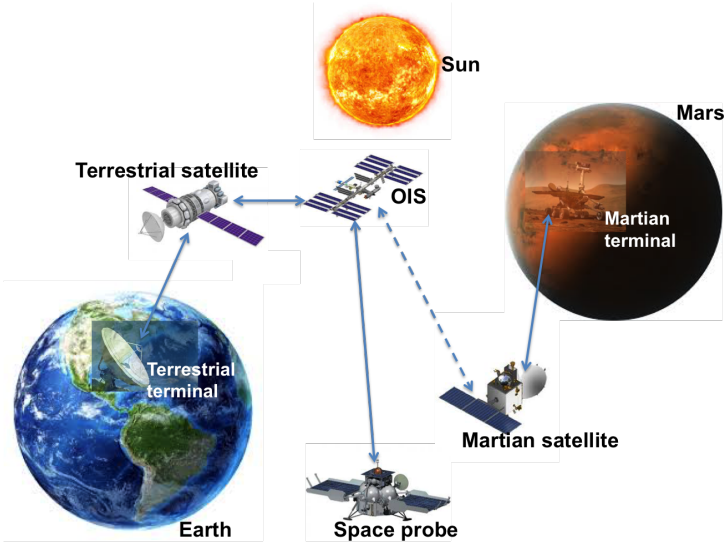


FIGURE 5.1 – Deep space communication scenario

communication scenario in Figure 5.1, in which a terrestrial terminal is connected to a terrestrial satellite. The terrestrial satellite is connected to an Orbital International Station (OIS) in orbit around the sun, which in turn connects to a Martian satellite and a space probe. Additionally, there is a Martian terminal connected to the Martian Satellite. The link between the OIS and the Martian Satellite is interrupted whenever the planet Mars is between the OIS and the orbiting satellite, as well as whenever the sun is between Mars and the OIS. Therefore, traffic on the "link" between the Terrestrial and Martian satellites may need to be buffered at the OIS for long- and varying periods of time. If the OIS becomes heavily congested, it will significantly hamper communication between the Terrestrial satellite and the space probe. Under these conditions, "traditional" end-to-end congestion control mechanisms, a la TCP, do not work well. As DTN nodes become congested, incoming messages may be discarded due to buffer overflow. This increases the drop rate and raises network overhead which leads to inefficient use of bandwidth and further worsens congestion conditions.

DTN congestion control has attracted considerable attention from the network research community. As a result, a number of congestion control techniques have been proposed (SILVA *et al.*, 2014), most of which have been evaluated empirically using network simulation platforms. Our goal, in this work, is to propose a simple, yet general framework that can model DTN congestion mathematically and that can be used to study the performance of DTN congestion control solutions, complementing and validating empirical studies. We explored a number of modeling approaches and found that percolation theory is well suited to describe congestion in DTNs. In the next section, we provide a brief overview of percolation theory and describe how we apply it to the DTN congestion problem.

5.2.2 Percolation Theory Overview

Percolation was first introduced in the mathematics literature motivated by the problem of how fluids flow (or percolate) in different materials (or media) (BROADBENT; HAMMERSLEY, 1957). It immediately caught the attention of mathematicians and physicists for its simplicity and wide applicability (GRIMMETT, 1999). To date, percolation has been employed in a variety of contexts ranging from complex networks, control of epidemic diseases, and wildfires.

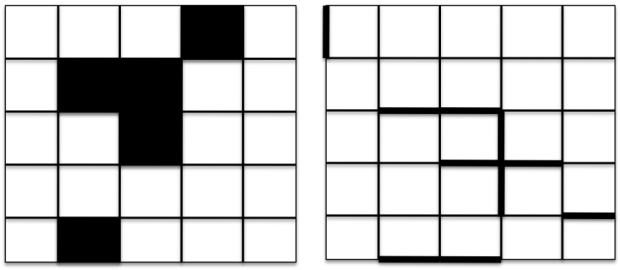
In percolation, the medium or network is modeled as a graph. A graph is a pair (V, E), where V is a countable set of points called vertices or sites, and E is a set of edges, *i.e.* unordered pairs of vertices $\langle v, w\rangle$, also called bonds. When $\langle v, w\rangle \in E$, we say that v and w are adjacent. A set of distinct vertices $\{v_1, v_2, v_3, \ldots, v_n\} \subset V$ is called $path$ when consecutive vertices are adjacent in the lattice. The graph distance between two vertices is defined to be the minimum amount of bonds necessary in order to establish a path that connects them. Consider each vertex $v \in V$ to be open with probability ρ and closed with probability $1 - \rho$, independently of every other vertex. Thus we can define P_ρ as being the probability distribution that describes the state of the graph as a whole. This distribution is defined in the sampling space $\Omega = \{0, 1\}^V$. Elements $\omega \in \Omega$ are represented as $\omega = (\omega(v) \colon v \in V)$. So, when $\omega(v) = 0$ we say that v is closed whereas, when $\omega(v) = 1$, we say that v is open.

Given a configuration $\omega \in \Omega$, we say that a path is open if all of its vertices are open, that is, $\omega(v_i) = 1$ for all $\{i = 1, 2, \ldots, n\}$. Two sites x and y in V are said to be connected $(x \leftrightarrow y)$ if there is a open path $\{v_1, v_2, \ldots, v_n\}$ where $x = v_1$ and $y = v_n$.

Given one configuration $\omega \in \Omega$ and a vertex x, we can consider the set of all vertices that are connected to x. This set is called the open cluster of x in the configuration ω and it is represented by $C_x(\omega)$, or simply by C_x. More formally, we have $C_x = \{y \in V; \omega \in (x \leftrightarrow y)\}$. When x is equal to the origin $0 \in \mathbb{Z}^d$ (where $\mathbb{Z}$ is an Euclidian integer lattice with dimension d) we drop the subscript and write $C = C_0$ to denote the open cluster containing the origin. Vertices in the lattice are called *sites* and edges *bonds*. In the so-called *bond percolation* models, bonds are declared "open" with probability ρ, or "closed" with probability $1 - \rho$. In the context of percolation's original application, open bonds correspond to channels through which fluid can flow. Open paths consist of a sequence of adjacent open bonds together with their end-vertices. Sites are declared "unblocked" with probability ρ or "blocked" with probability $1 - \rho$. An open path is a sequence of neighboring unblocked sites. Two sites belong to the same open connected component, or simply to the same cluster, if they are linked by an open path.

To illustrate bond- and site percolation, we consider the square lattice shown in Figure 5.2.

- In site percolation (see Figure 5.2a), we view the lattice as a rectangular array of squares. Each square is declared to be unblocked with probability ρ (black squares), and blocked with probability $1 - \rho$ (white squares). A *cluster* is defined as a maximal connected set



(a) Site Percolation (b) Bond Percolation

FIGURE 5.2 – Types of percolation

of neighboring unblocked squares.

- In bond percolation (see Figure 5.2b), the lattice is a graph consisting of horizontal and vertical edges. Each lattice edge is open (thicker) with probability ρ, and closed (thinner) with probability $1 - \rho$. A *cluster* is defined as maximal connected subgraph of the lattice consisting of open edges.

Bond- and site percolation are commonly used to study the connectivity properties of clusters in the lattice as a function of the parameter ρ. The main question is to determine wether the probability that there exists an infinite open path starting at the origin is non-zero, or equivalently, wether the origin belongs to an infinite cluster with non-zero probability.

In order to model congestion in DTNs, we use another formulation called directed site-bond percolation. In this formulation, both sites and edges have their states assigned randomly and independently. Furthermore edges will be oriented in order to represent the flow of information in the lattice. In the proposed percolation-based DTN congestion model: (1) *open bonds* represent contacts between DTN nodes, (2) *blocked sites* represent DTN nodes whose buffers are 100% occupied (i.e., congested nodes), and (3) *unblocked sites* represent DTN nodes that are not congested. Our model allows us to derive important network performance metrics such as average message delivery probability, average buffer occupancy ratio and blocked time, as well as average message delivery delay. These metrics are defined in Section 5.4.2. Our model also allows us to understand the impact of parameters such as buffer management policies, routing mechanisms, message time-to-live on network congestion control.

5.3 DTN Congestion Model

DTN congestion can be expressed as a percolation problem by examining whether there exists a path between data sources and destinations and whether the paths are open, i.e., non-congested. In other words, the probability of delivering data from a node x to a node y is equivalent to the probability of finding an open path between x and y. As such, our model tries

to find, in the graph represention of the network, *open paths*, i.e, sequences of adjacent *open edges* and *unblocked sites* over which data can be delivered from sources to destinations. To this end, in addition to finding end-to-end paths between DTN sources and destinations, the model also considers buffer occupancy at each node along the path.

While our percolation-based model is quite general and applicable to lattices of arbitrary dimension d, for simplicity, in this work we focus on the special case of $d = 1$ which still captures the main DTN congestion features that we want to study. The experimental validation of our model presented in Section 5.5.1 shows that results obtained from the model match quite well simulations run on the ONE DTN simulator (KERäNEN *et al.*, 2009). Figure 5.3 shows the DTN representation we use in our percolation model; the network is represented as a collection of nodes in a domain $\mathbb{S} \subset \mathbb{Z}^d$, where each node is a vertex in a $d = 1$ lattice, i.e., a line. We model the state evolution of the nodes by adding the temporal dimension, which is represented by parallel lines labeled $t_0, t_1, \cdots, t_n$ as shown in Figure 5.3. Our model is thus defined in $\mathbb{Z}^{d+1} = \mathbb{Z}^d \times \mathbb{Z}^1$. A site in $\mathbb{Z}^{d+1}, d = 1$ will be denoted by (x, t) where $x \in \mathbb{Z}^d$ represents the position of the site at time $t \in \mathbb{Z}$.

As previously highlighted, in order to establish whether a node x is blocked (congested) or unblocked (non-congested) at time t, our model calculates x's buffer occupancy given by: x's buffer occupancy at time $t - 1$, plus the number of new messages created at x, plus the number of incoming messages at x received from its neighbors, minus the number of outgoing messages from x. Below, we proceed to calculate these different buffer occupancy components. Table 5.1 summarizes the variables used in our model and their definitions.

We consider that, at each time t, a node x can only forward a message to its left and/or right neighbors ($x - 1$ and $x + 1$, respectively). This corresponds to the SE (southeast) and SW (southwest) oriented edges in Figure 5.3, respectively. We use vertical edges to represent the fact that a node can keep messages stored locally from time t to $t + 1$. These edges are represented as dashed lines in Figure 5.3.

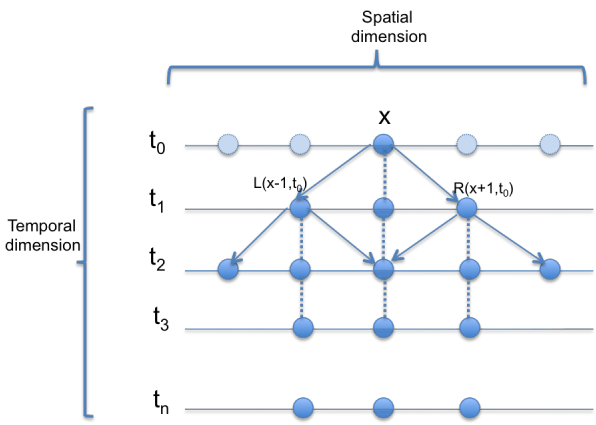


FIGURE 5.3 – Network model

Let $\mathcal{Q}$ be a positive integer denoting the buffer capacity at nodes, *i.e.* the maximum number of messages that can be stored at each node. Let $\mathcal{M}(x,t)$ denote the set of messages stored at node x at time t and $m(i,x,t) \mid i = \{1,2,\ldots,\mathcal{M}(x,t)\}$ refer to an individual message stored at x at time t.

TABLE 5.1 – DTN percolation model variables and definitions

Variable Name	Description
$\mathcal{N}(x)$	number of neighbors of x
$\mathcal{Q}$	maximum number of messages that can be stored in a node's buffer
$\mathcal{M}(x,t)$	set of messages stored at node x at time t.
$m(i,x,t)$	message i stored at x at time t
$f(i,x,t)$	specifies whether message $m(i,x,t)$ stored at x at time t was forwarded or not
$l(x_i,j,x,t)$	specifies whether message $m(j,x_i,t)$ forwarded by neighbor x_i already exists in x's buffer at time t
r_c	node's transmission radius
$O(x,t)$	x's buffer occupancy at time t
$B(x,t)$	number of messages generated by x at time t
E	set of all bonds
$\mathcal{S}(x,t)$	number of outgoing messages at x at time t
$\mathcal{D}(x,t)$	number of drop messages at x at time t
$\mathcal{C}_{msg}(x,t)$	number of messages consumed by applications running on x at time t
$F(x,t)$	number of messages forwarded by x at time t
e	bond between two sites/nodes
$\eta(e)$	specifies whether a bond between x and x_i is open or closed
β	specifies message forwarding mechanism: replication if $\beta = 0$ or "plain" forwarding if $\beta = 1$
$\omega(x,t)$	specifies whether x blocked (congested) or unblocked (uncongested) at time t
Γ	path that contains only unblocked sites and open bonds
$\kappa(x,t)$	number of messages received from right neighbor at time t
$\nu(x,t)$	number of messages received from left neighbor at time t
$L(x,t)$	number of messages forwarded to left neighbor at time t
$R(x,t)$	number of messages forwarded to right neighbor at time t

For a message $m(i,x,t)$:

$$f(i,x,t) = \begin{cases} 1 & \text{if } m(i,x,t) \text{ has already been forwarded,} \\ 0 & \text{if } m(i,x,t) \text{ has not been forwarded yet.} \end{cases} \tag{5.1}$$

For each site $(x,0)$, let us set an initial buffer occupancy $O(x,0)$. Here we assume that $\{O(x,0); x \in \mathbb{Z}\}$ is initially empty. We say that node x is blocked at time t if $O(x,t) = \mathcal{Q}$.

We define $B(x,t)$ as the number of messages created at x at time t; $B(x,t)$ is an independent random variable given by a Bernoulli distribution with parameter $\lambda \in (0,1)$ called the message generation rate. On average, a message is created at x every $1/\lambda$ unities of time.

Let $r_c > 0$ be an integer denoting the node's transmission radius. We add a bond for each pair of sites $e = ((x,t),(x_i,t+1))$ if $|x_i - x| \leq r_c$. Let us denote by E be the set of all bonds obtained by this procedure (Figure 5.3 presented an illustration for the case $d = 1$ and $r_c = 1$). We define $\{\eta(e) \mid e \in E\}$ a family of independent random variables with Bernoulli parameter ρ indexed by the bonds of the network. We say that the bond e is open if $\eta(e) = 1$ and closed otherwise. This allows us to model node encounters in DTNs, i.e., when a bond $e = ((x,t),(x_i,t+1))$ is open, nodes x and x_i are in contact and thus can communicate. This is also how we model the effect of node mobility which influences the appearances and

disappearances of bonds in the lattice.

Now, let $\mathcal{S}(x,t)$, which is given by Equation 5.2, be the total number of outgoing messages at node x at time t. $\mathcal{S}(x,t)$ includes the number of messages discarded ($\mathcal{D}(x,t)$), the number of messages that have been consumed by applications running on x ($\mathcal{C}_{msg}(x,t)$), as well as the number messages forwarded by x ($F(x,t)$). Note that to calculate $F(x,t)$, we use $\beta \in \{0,1\}$ which specifies whether the forwarding mechanism uses replication or not. More specifically, if $\beta = 0$, a copy of the message is forwarded and the original message is maintained in the buffer. Otherwise, if $\beta = 1$, the original message is itself forwarded and thus not maintained in the buffer. Indeed, this is how we model the underlying routing mechanism being used, i.e, whether it is based on replication or "plain" forwarding. $\mathcal{D}(x,t)$ represents the number of messages discarded either because the message time-to-live (TTL) has expired or because the buffer filled up. In the latter case, messages will be discarded according with a pre-defined drop policy (SILVA *et al.*, 2014). In the experimental evaluation of our model (see Section 5.5), we varied the drop policy and observed its impact on network congestion.

$$\mathcal{S}(x,t) = \mathcal{D}(x,t) + \mathcal{C}_{msg}(x,t) + \overbrace{\beta \sum_{i=1}^{O(x,t)} f(i,x,t)}^{\text{F(x,t)}} \tag{5.2}$$

Recall that, at time t, a node x can only forward messages to nodes $x-1$ (left neighbor) and $x+1$ (right neighbor). Thus, the number of messages forwarded by x, $F(x,t)$ can be written as $F(x,t) = L(x,t) + R(x,t)$, where $L(x,t)$ is the number of messages forwarded by x to its left neighbor $x-1$ at time t, and $R(x,t)$ is the number of messages forwarded by x to its right neighbor $x+1$ at time t.

According to Equation 5.3, $\kappa(x,t)$, the number of messages received by x from its right neighbor, depends on whether there is an edge between x and its right neighbor during $(t,t+1)$, which is determined by $\eta((x+1,t),(x,t+1))$. Clearly, if x and $x+1$ are not in contact with one another $\kappa(x,t) = 0$. If there is an edge between x and $x+1$, the number of messages received (and stored) by x is given by $L(x+1,t)$ minus the number of $x+1$'s messages which x already has in its buffer. These messages, if exchanged by the two nodes, will be discarded by x as duplicates. The number of duplicate messages is given by $\sum_{j=1}^{O(x+1,t)} l(x+1,j,x,t)$, where $l(x+1,j,x,t)$ is defined by Equation 5.5.

$$\kappa(x,t) = \eta((x+1,t),(x,t+1))[L(x+1,t) - \sum_{j=1}^{O(x+1,t)} l(x+1,j,x,t)]. \tag{5.3}$$

Similarly, $\nu(x,t)$, the number of messages received by x from its left neighbor is expressed

by Equation 5.4.

$$\nu(x,t) = \eta((x-1,t),(x,t+1))[R(x-1,t) - \sum_{j=1}^{O(x-1,t)} l(x-1,j,x,t)]. \tag{5.4}$$

$$l(x_i,j,x,t) = \begin{cases} 1 & \text{if } f(j,x_i,t) = 1 \text{ and } m(j,x_i,t) \in \mathcal{M}(x,t), \\ 0 & \text{otherwise.} \end{cases} \tag{5.5}$$

We can now write the expression for the buffer occupancy at x at time $t+1$ in Equation 5.6 as x's buffer occupancy at time t, plus the amount of new messages created at x at time t, minus $S(x,t)$, the number of messages leaving x at time t, plus the number of messages received from x's right neighbor at time t (if a contact existed between them), plus the number of messages received by x at t from x's left neighbor (if a contact existed between them).

$$O(x,t+1) = \{(O(x,t) + B(x,t) - \mathcal{S}(x,t) + \kappa(x,t) + \nu(x,t)) \wedge \mathcal{Q}\}, \tag{5.6}$$

where for any real numbers a and b, $a \wedge b \equiv \min\{a,b\}$.

We then say that x is blocked at time t when Equation 5.7 is true.

$$\{O(x,t) + B(x,t) - \mathcal{S}(x,t) + \kappa(x,t) + \nu(x,t)\} \geq \mathcal{Q}, \tag{5.7}$$

and define:

$$\omega(x,t) = \begin{cases} 0 & \text{if } x \text{ is blocked in time } t, \\ 1 & \text{if } x \text{ is not blocked in time } t. \end{cases} \tag{5.8}$$

Recall that our goal is to find open paths Γ, i.e., sequences

$$\Gamma = [(x^0,t^0),(x^1,t^1),(x^2,t^2),(x^3,t^3),\ldots,(x^n,t^n)] \tag{5.9}$$

such that

$$((x^i,t^i),(x^{i+1},t^{i+1})) \in E \quad \forall \quad i = 0,\ldots,n-1 \tag{5.10}$$

and, in addition,

- $\omega(x^i,t^i) = 1 \ \ \forall \ \ i = 0,\ldots,n$ and
- $\eta((x^i,t^i),(x^{i+1},t^{i+1})) = 1 \ \ \forall \ \ i = 0,\ldots,n-1$.

As shown in the results presented in Section 5.5, using our model we can compute network performance metrics such as buffer occupancy, buffer block time, message delivery probability, and delivery delay which are congestion indicators. Our model can also be used to evaluate the effectiveness of different congestion control mechanisms.

5.4 Experimental Methodology

The evaluation of our model was carried out in two ways. First, we validate the model by comparing its output against simulation results from the ONE DTN simulator (KERäNEN *et al.*, 2009). Then, we use our model to study how parameters like buffer management policy, buffer size, routing mechanism, and message time-to-live (defined in Section 5.4.2 below) affect network congestion and can be used to perform congestion control. We implemented our model in MatLab.

5.4.1 Experimental Setup

For experiments presented in Section 5.5, the parameters of our model and their values are summarized in Table 5.2 while the parameters of the ONE simulator (KERäNEN *et al.*, 2009) and their values are listed in Table 5.3.

TABLE 5.2 – Simulation parameters and their values for MatLab experiments

Parameters		
Name	Description	Value
Discrete clock time	simulation time	1000 seconds
TTL	messages time-to-live (TTL)	[100, 200, 300, 400, 500] seconds
Lambda (λ)	message generation rate	[1/5, 1/10, 1/20, 1/30]
BufferSize	buffer capacity in number of messages	[10, 20, 30 ,40, 50 , 60] messages
Rho (ρ)	connection success probability	0.5
BufferPolicy	message drop scheme if congestion happens	[DROP-INCOMING, DROP-HEAD, DROP-LAST, DROP-OLDEST]
Nodes	number of nodes	50
TimeApp	once the message arrived at the destination how long it will take for it to be consumed by the application	0 seconds
Beta (β)	forwarding strategy (replication or single-copy)	[0, 1]
MessageSize	message size	1 KB

TABLE 5.3 – Simulation parameters and their values for ONE simulations

Parameters		
Name	Description	Value
Scenario.endTime	simulation time	1000 seconds
btInterface.transmitSpeed	bandwidth	2.5 Mbps
btInterface.transmitRange	transmitting range	26 m
Group.router	routing protocol	[EpidemicRouter,FirstContactRouter]
Group.movementModel	mobility model	LinearFormation
Group.bufferSize	node buffer size	[10, 20, 30, 40, 50, 60] KB
Group.msgTTL	message time to live	[100, 200, 300, 400, 500] seconds
Group.nrofHosts	number of nodes in network	50
Movimentmodel.worldSize	area where simulation takes place	1km x 1km
Events1.size	message size	1KB
Events1.interval	i.e. one new message every 1 to 5 seconds	[1-5, 1-10, 1-20, 1-30] seconds

Some considerations about the implementation of our model are noteworthy. For example, when a message is delivered to its intended destination, it is consumed by the application

running at the destination within $TimeApp$ seconds. This is the period of time the message stays in the node's buffer after being received and before being removed from the buffer. In our experiments, unless stated otherwise, we use $TimeApp = 0$, which means that the message is promptly removed from the buffer as soon as it arrives. Each time two nodes are in contact with one another, they try to exchange all their stored messages if there is available space in their buffers. To this end, both nodes compute the buffer occupancy according to Equation 5.6. If the node's buffer is considered blocked (see Equation 5.7), the buffer management policy of choice is activated to discard the appropriate messages. Four drop policies have been considered in our experiments (see Table 5.4). Each message has a TTL (Time-To-Live) whose value is decremented by one unit at every clock tick. When a message's TTL expires, the message is automatically removed from the system.

TABLE 5.4 – Drop policies

Drop Policy	Description
DROP-INCOMING	the incoming message is dropped.
DROP-HEAD	the first message in the buffer, i.e., the head of the queue, is dropped.
DROP-LAST	the newly received message is first removed.
DROP-OLDEST	the message with the shortest remaining life time (closest to TTL expiration) is dropped.

In order to validate our model against the ONE simulator (KERäNEN *et al.*, 2009), we run ONE simulations using the line topology shown in Figure 5.4). We set the node transmission range such that a node can only communicate with its two adjacent neighbors in the line topology as depicted in Figure 5.4). To be able to simulate using the ONE the two forwarding strategies that we use in our model, namely replication and single-copy forwarding, we use two of the ONE's routing schemes. Replication in our model ($\beta = 0$) is simulated using *epidemic routing* (VAHDAT; BECKER, 2000b) in the ONE, which replicates a message and forwards it to all nodes that are within range (i.e., nodes that have open bonds with the current node). To match single-copy forwarding ($\beta = 1$), we use the ONE's implementation of the *first contact protocol* (JAIN *et al.*, 2004a), according to which only one copy of a message exists in the network at any point in time. This means that a message is relayed to the first encountered node it is a duplicate, and is then removed from the message buffer of the previous hop.

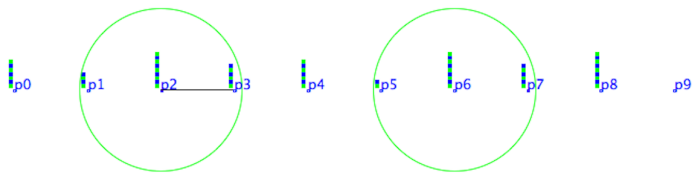


FIGURE 5.4 – Snapshot of the ONE simulator scenario for 10 DTN nodes.

In the ONE simulator, two nodes are in contact whenever they are within the communication range of one another. So, if the separation distance between the nodes is less than the r_c, all

links will be always active. In order to simulate the fact that node neighborhoods change, we simulate links between neighbors going up and down according to a Bernoulli distribution with parameter ρ. To this end, we modify the ONE simulator to include this feature and set $\rho = 0.5$ to match our model.

5.4.2 Evaluation Metrics

One of the goals of our model is to describe the congestion problem taking into account the main characteristics of a DTN, for example, limited resources, high delays, and intermittent connectivity. Additionally, we also want to be able to analyze DTN congestion under a variety of scenarios and conditions (e.g., deep space communication).

To this end, we define the following metrics that can be derived from the proposed model. These metrics not only provide us with insight into the performance of existing congestion control mechanisms but also help us to propose new ones that can be more cost-effective. We also use these metrics to cross-validate our model against simulation results from the ONE simulator.

1. **Average buffer occupancy** is given by the average ratio between the number of messages stored at a node and the size of the node's buffer; in our model it is equivalent to $O(x, t)$.

2. **Average buffer blocked time** is defined by the average time that each site remains blocked (or congested). It corresponds to $\omega(x^i, t^i)$ in our model.

3. **Average message delivery probability** is given by the probability of existence of a space-time path that connects the message source to its final destination. This metric allows us to infer network congestion levels and corresponds in our model to the probability that Γ exists.

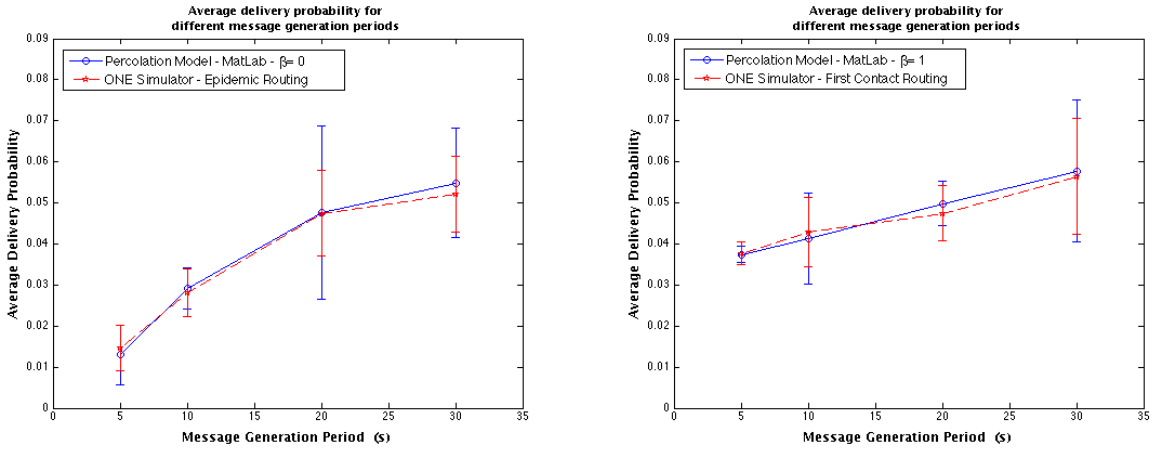
4. **Delivery delay** is defined as the total time elapsed between message generation and its delivery and is equivalent, in our model, to the time it takes for a message to be delivered over Γ, if Γ exists.

5.5 Results

5.5.1 Model Validation

We validate our model by comparing its results obtained from its Matlab implementation against simulation results from the ONE DTN simulator. To evaluate the impact of congestion on network performance, we try to generate enough load to congest the network. Figure 5.5 shows the average message delivery probability as a function of the message generation period

resulting from our model and from the ONE simulator. We observe that both curves essentially overlap with an average percentage difference [1] of 1.58% in the case of Figure 5.5a and 0.96% in Figure 5.5b. As expected, for longer message generation periods (i.e., lower message generation rate, or λ in our model), which results in less messages generated, the average delivery probability increases. Note that average delivery probabilities from this experiment are quite low since no congestion control scheme has been used. In this case, the drop policy is frequently activated, causing messages to be discarded before they are delivered. According to Figures 5.5a and 5.5b, we observe that the delivery probability when using $\beta = 0$ (epidemic routing) is slightly lower than when using $\beta = 1$ (first contact routing) because in the former case there are more message copies in the network resulting in buffer overflow and consequently increased drop ratios. As a result the average delivery probability is lower.



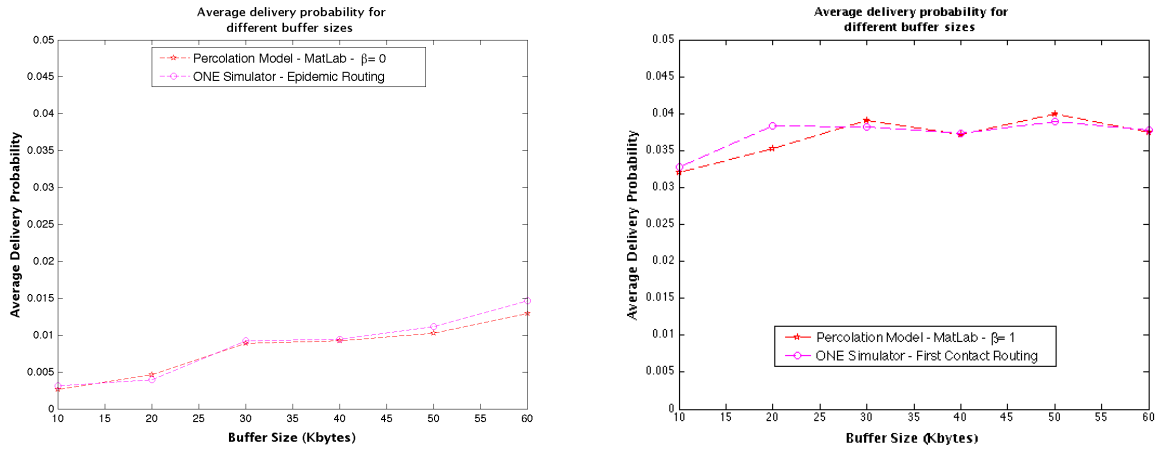
(a) (b)

FIGURE 5.5 – Average delivery probability for different message generation periods (TTL of $300s$, Drop-policy $DROP - OLDEST$, buffer size of $60\,kbytes$).

Figure 5.6 shows the delivery probability for different buffer sizes. Note that the percentage difference in Figure 5.6a is 16.32% and in Figure 5.6b is 1.15%. These percentage differences show that our model results is close to the ONE results. Without surprisingly when the buffer size becomes larger, the delivery ratio trends to increase. This result is mainly because there are more buffer space available in the network. Thus more messages are stored and the chance of message delivery increases.

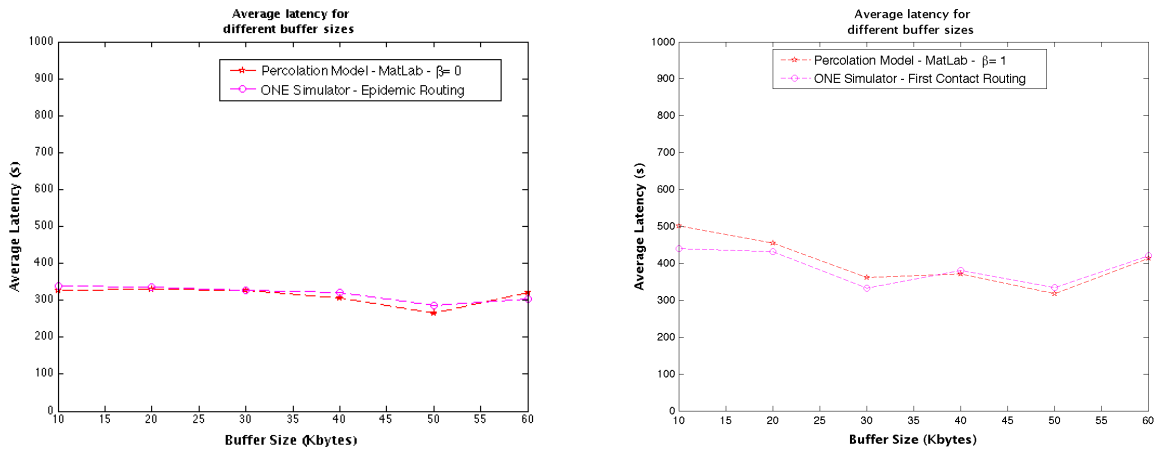
Figure 5.7 shows the message delivery latency for different buffer sizes. In both Figures 5.7a and 5.7b we can observe that the average latency slightly fluctuates and that exists a percentage difference of 56.16% in the case of $\beta = 0$ and 3.37% for $\beta = 1$. However, observe that the latency decreases when increasing buffer size until $50\,KB$.

[1] The percentage difference comparison calculates the percentage difference between two values in order to determine how close they are.



(a) (b)

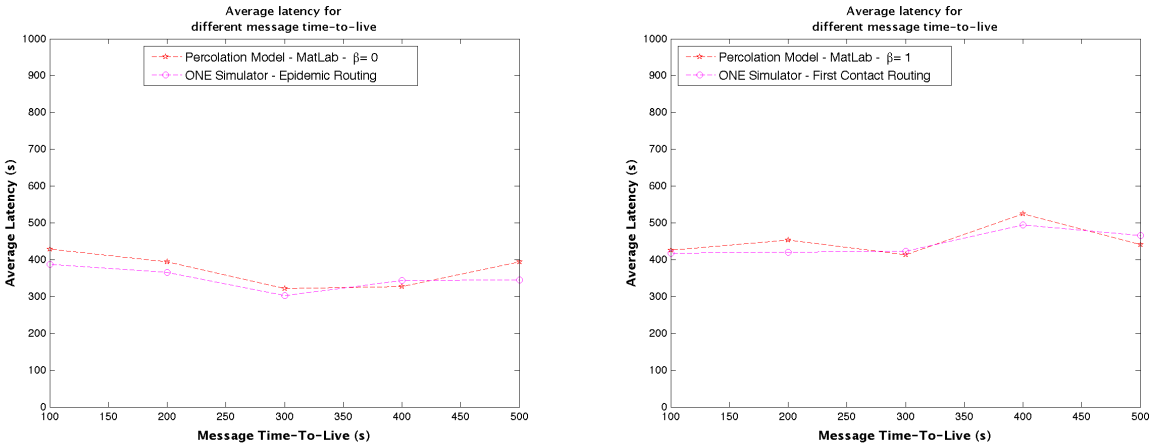
FIGURE 5.6 – Average delivery probability for different buffer sizes (TTL of $300s$, Drop-policy $DROP - OLDEST$, message generation period of $5s$, The standard deviation for the results from our model are 0.0038 for $\beta = 0$ and 0.0028 for $\beta = 1$. For the ONE experiments, we obtain a standard deviation of 0.0044 for $\beta = 0$ and 0.0022 for $\beta = 1$.).



(a) (b)

FIGURE 5.7 – Average latency for different buffer sizes (Drop-policy $DROP - OLDEST$, TTL of $300s$, message generation period of $5s$, The standard deviation for the results from our model are 25.02 for $\beta = 0$ and 66.97 for $\beta = 1$. For the ONE experiments, we obtain a standard deviation of 20.46 for $\beta = 0$ and 48.45 for $\beta = 1$.).

Figure 5.8 shows the delivery latency for different TTLs. Similarly to the average delivery probability results, the curves resulting from our model and the ONE simulator are quite close: the average percentage difference between the latencies obtained from our model and results from the ONE is 6.68% when $\beta = 0$ and 3.06% when $\beta = 1$ (Figures 5.8a and 5.8b respectively). We observe that when the message time-to-live increases, the latency has a tendency to increase. This is because higher message time-to-live results in a greater number of messages in the network, which in turn generates overload. Consequently, it increases the message delivery time since the messages stay longer into network.



(a) (b)

FIGURE 5.8 – Average latency for different message time-to-live (Drop-policy $DROP-OLDEST$, buffer size of $60\,kbytes$, message generation period of $5s$). The standard deviation for the results from our model are 46.49 for $\beta = 0$ and 43.70 for $\beta = 1$. For the ONE experiments, we obtain a standard deviation of 31.40 for $\beta = 0$ and 34.26 for $\beta = 1$.

5.5.2 Understanding Network Congestion

Here we use our model to understand network behavior under congestion and how certain policies and parameters can control congestion. We begin with the impact of different drop policies on delivery probability which is shown in Figure 5.9. What we observe is that the delivery probability for different policies does not vary considerably when $\beta = 0$. However, when $\beta = 1$, we note that the variability of the behavior among the different drop policies increases, favoring "drop-head" and "drop-last".

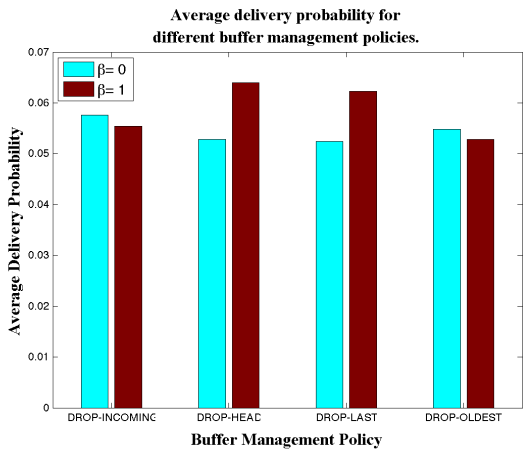


FIGURE 5.9 – Message delivery probability for different drop policies with $\beta = 0$ and $\beta = 1$ (TTL of $300s$, buffer size of $60\,kbytes$, message generation period of $30s$).

Figure 5.10 shows that, as expected, increasing the buffer size results in shorter buffer block times.

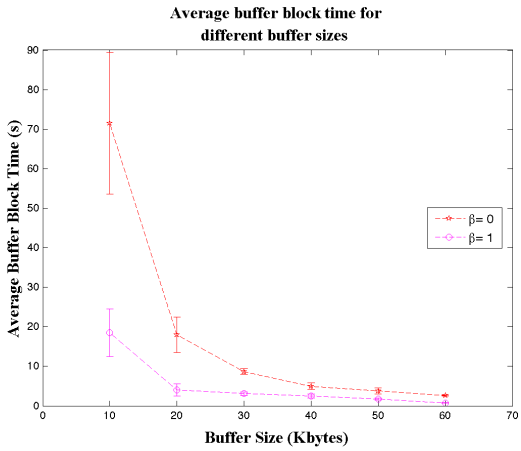


FIGURE 5.10 – Average buffer block time for different buffer sizes with $\beta = 0$ and $\beta = 1$ (TTL of $300s$, message generation period of $5s$).

From Figure 5.11, we observe that higher time-to-lives increase average buffer block time. This is mainly because messages can stay longer in the buffer until being forwarded or delivered. As a result the buffer becomes full faster.

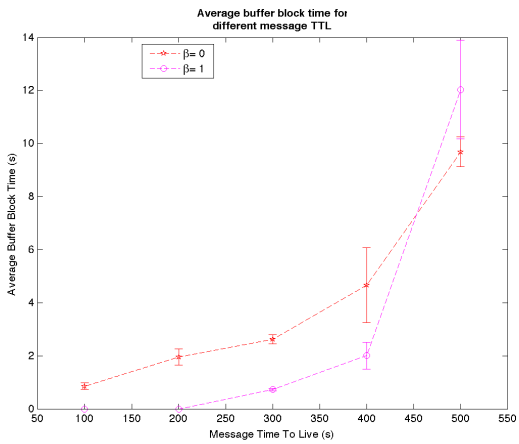


FIGURE 5.11 – Average buffer block time for different message time-to-live with $\beta = 0$ and $\beta = 1$ (buffer size of $60\,kbytes$, message generation period of $5s$).

Figure 5.12 shows that as the buffer size increases, so does the average buffer occupancy. Basically, for higher buffer sizes the node can store more messages, and consequently the buffer occupancy ratio has a tendency to increase.

As it can be seen in Figure 5.13, for higher time-to-live values, buffer occupancy tends to increase. The reason for this is that messages stay in the buffer longer. However, beyond a certain value of the TTL, buffer occupancy does not change significantly. This is probably due to the fact that messages have enough time to be delivered to their intended destination. For example, in Figure 5.13, for TTL values greater than $400s$, the average buffer occupancy does not vary significantly.

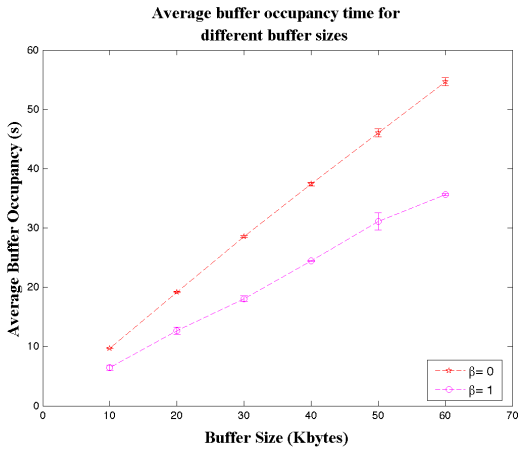


FIGURE 5.12 – Average buffer occupancy for different buffer sizes with $\beta = 0$ and $\beta = 1$ (TTL of $300s$, message generation period of $5s$).

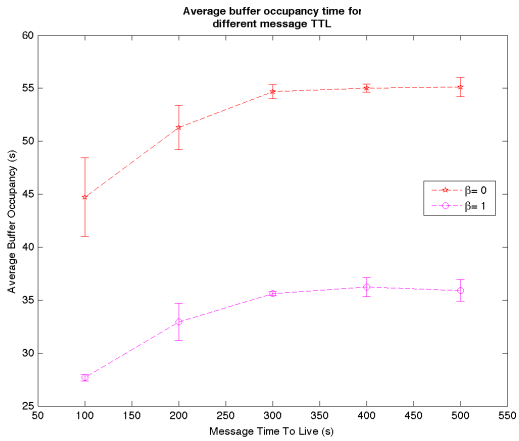


FIGURE 5.13 – Average buffer block time for different message time-to-live with $\beta = 0$ and $\beta = 1$ (buffer size of $60\,kbytes$, message generation period of $5s$).

5.6 Related Work

In this section we review prior research efforts focusing in the two areas related to our work, namely: percolation theory applied to mobile ad-hoc networks (MANETs) and network congestion modeling.

5.6.1 Applications of Percolation in MANETs

An evaluation of the probability of routing success in a dynamic network where links go up and down was presented (BASU *et al.*, 2012). The probability of source routing success in dynamic networks, where the link up-down dynamics is governed by a time-varying stochastic process, exhibits critical phase-transition (percolation) phenomena as a function of end-to-end message latency per unit path length. In addition, the performance of a stateless single-copy op-

portunistic forwarding algorithm on a 2D probabilistic grid is analyzed. Unlike a flooding-based approach, the opportunistic forwarding algorithm does not demonstrate a non-trivial percolation threshold in link-up probability as a flooding based approach does.

The existence and properties of percolating paths through dense random networks have been, in recent years, shown to be useful in uncovering a variety of new results and insights into wireless networking and communications. Some applications of the theory of directed percolation (where there is a general direction associated with the information flow) to wireless networking and present an analytical characterization of phase transitions in random dynamic path graphs have been studied (GUHA; BASU, 2012).

A model for dynamic networks, where the links or the strengths of the links change over time has been also focus of studies (PARSHANI *et al.*, 2010). The model uses directed percolation, where the direction corresponds to the time evolution of the network. Furthermore, the dynamic network undergoes a percolation phase transition at a critical concentration ρ_c that decreases with the rate r at which the network links change. Like our work, they argue that the evolution of the network over time generates many more possible configurations, enabling the percolation cluster to become much larger.

The problem of disseminating broadcast messages in wireless networks with time-varying links from a percolation based perspective is also investigated (KONG; YEH, 2009). A model of wireless networks based on random geometric graphs with dynamic links shows that the delay for disseminating broadcast information exhibits two behavioral regimes, corresponding to the phase transitions (subcritical and supercritical) of the underlying network connectivity. Thus, the information dissemination delay scales linearly with the Euclidean distance between the sender and the receiver when the resulting network is in the subcritical phase, and scales sub-linearly with the distance if the resulting network is in the supercritical phase.

Connectivity in ad hoc networks has been considered in some studies (ZHANG *et al.*, 2013) as an application of percolation theory in two dimensional square lattices. Given a message source and the bond probability to connect neighbor vertices on the lattice, percolation theory tries to determine the critical giant component with high probability. Following this description the connectivity from the source to any vertex on the square lattice following a certain direction. Using a recursive decomposition approach, they obtained the analytical expressions for directed connectivity. In our case, instead of directed connectivity, we model analytically the congestion problem.

The work (STAUFFER, 2012) about space-time percolation and detection by mobile nodes combines ideas from fractal percolation and multi-scale analysis to show that cells with a small node density do not percolate in space and time. The nodes are capable of detecting all points within distance r of their location. They study the problem of determining the first time at which a target particle, which is initially placed at the origin of $\mathbb{R}^d$, is detected by at least one node.

An extension of the random geometric graph model is applied to a mobile network setting by allowing nodes to move in space according to Brownian motion (PERES *et al.*, 2011). Three questions are studied: detection (the time until a given target point is detected), coverage (the time until all points inside a finite box are detected by the network), and percolation (the time until a given node is able to communicate with the giant component in the network).

A critical percolation threshold for aligned cylinders, which provides a lower bound for the required node degree for the performance of information in opportunitisc networking has attracted the attention of researchers (VIRTAMO *et al.*, 2012). The nodes are assumed stationary and the opportunistic content dissemination schemes, such as the "floating content", can be analyzed by using a three-dimensional continuum percolation model. From the information dissemination point of view, given a message and if it gets distributed everywhere it does not get extincted. This assertive is not true when we consider congestion problem, since any message can be discarded to make buffer space available.

A study of DTN and its capacity to store, carry, and forward messages so that messages eventually reach their final destination(s) was presented recently (HYYTIA; OTT, 2013). Percolation theory is used to characterize the mean density of nodes required to support communication in DTNs. They show that DTN communication is feasible when the mean node degree r is greater than $4\eta_c(y)$, where y is the ratio of the distance l to the transmission range d and $\eta_c(y)$ is the critical reduced number density of tilted cylinders.

A lower bound for node buffer size in intermittently connected wireless networks is described (XU; WANG, 2011). To investigate the fundamental requirements on node buffer size posed by the possibility of node inactivity, it is assumed that the capacity and node processing speed can be regarded as infinity when compared with the actual utilization of the network capacity. In addition, each node is assumed to have the same probability ρ of being inactive during each time slot and there exists a critical value $\rho_c(\lambda)$ for this probability from a percolation based perspective. When $\rho < \rho_c(\lambda)$ the network is in the supercritical case and there is an achievable lower bound for the occupied buffer size of each node, which is asymptotically independent of the size of the network. If $\rho > \rho_c(\lambda)$, the network is in the subcritical case, and there is a tight lower bound $\theta(\sqrt{n})$ for buffer occupation, where n is the number of nodes in the network.

An analytical framework based on bulk arrival and bulk service queues to model DTN node behavior is proposed in (CELLO *et al.*, 2012). The stationary discrete probability densities of the sizes of the arriving bulks is computed. In addition, a class of forwarding strategies based on epidemic rouing is investigated and an expression for the average buffer occupancy is derived.

5.6.2 Modeling Network Congestion

A variety of methods have been employed to model congestion in traditional networks, including queuing (GALANT, 1989) (ZEEPHONGSEKUL *et al.*, 2006), auto-regression (CHANDRASEKARAN,) (CHEN, 2007), and Markov chains (CHANDRASEKARAN,) (CHEN, 2007). Our work complements this prior work by considering congestion in DTNs.

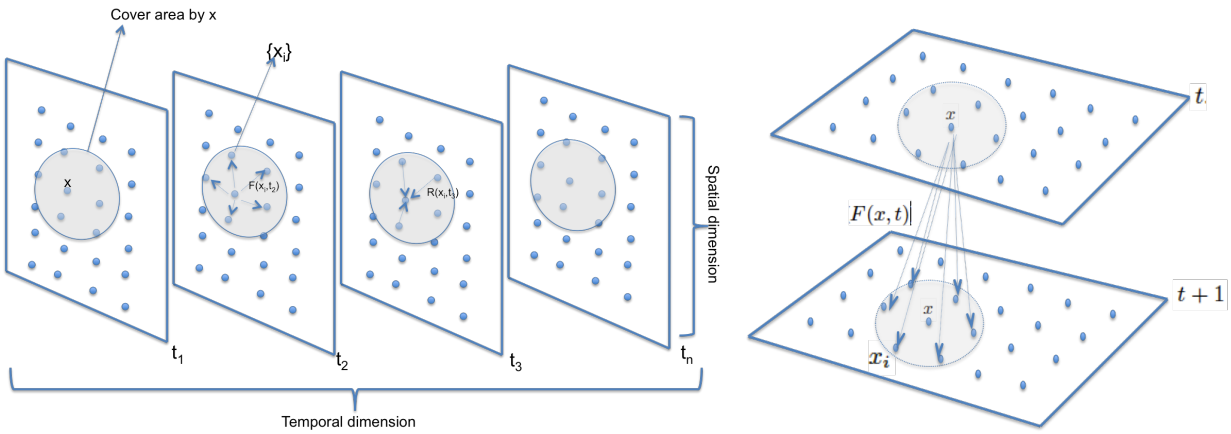
There are also some studies that treat the congestion problem using financial models. For instance, market theory suggests that differentiated service can arise from a pricing scheme based on the level of congestion. As the resource becomes overloaded, only those who are willing to pay higher usage prices remain on the network. An economic pricing model is used by (BURLEIGH *et al.*, 2006) to study the congestion problem and how to mitigate it in DTNs.

An analytical framework based on bulk arrival and bulk service queues to model DTN node behavior is proposed in (CELLO *et al.*, 2012). The authors compute the stationary discrete probability densities of the sizes of the arriving bulks. In addition, they investigate a class of forwarding strategies, based on epidemic routing, used by DTN nodes where an expression for the average buffer occupancy is derived.

5.7 General Model in $\mathbb{Z}^{d+1} \mid d \in \{1, 2, 3\}$

In this section we describe a general model that allows any number of neighbors and include node mobility. This model is part of our ongoing work.

Consider a network with a collection of nodes in a domain $\mathbb{S}$ of the plane (see Figure 5.14) where $\mathbb{S} \subset \mathbb{Z}^{d+1}$. Assume network structure as shown in Figure 5.14a where each segment is time-dependent . Refer to Table 5.5 for some variables definitions.



(a) Network time segment (b) Message forwarding $t \to t+1$

FIGURE 5.14 – Delay and disruption tolerant network snapshot for percolation mapping sampled at four time segments (3D).

TABLE 5.5 – Variable description - General model

Variable Name	Description
$\mu(x,t)$	the total number of received messages from all neighbors in time t
$R(x_i,x,t)$	the number of received messages by x from x_i in time t

According to Figure 5.14b a node x can forward a message at time t for its neighbors x_i where $i = \{1, 2, ...\mathcal{N}(x)\}$. In addition, we can say that the neighbors of x are all x_i's that are within the coverage area of x.

Suppose yet that all nodes can move each time unit one step (North, South, West or East) as shown in Figure 5.15.

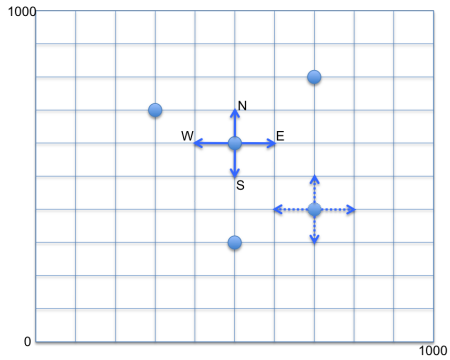


FIGURE 5.15 – Node motion on the general model

Let $\mu(x,t)$ be the number of messages received from neighbors according to Equation 5.11, where $R(x_i,x,t)$ is the number of message received from x_i at time t. We assume that a DTN node never stores a duplicated message, in this case the duplicated message is discarded.

$$\mu(x,t) = \sum_{i=1}^{\mathcal{N}(x)} \sum_{j=1}^{O(x_i,t)} \eta((x_i,t),(x,t+1))[R(x_i,x,t) - l(i,j,x,t)] \tag{5.11}$$

Where $R(x_i,x,t)$ is defined by an utility function as showed in Equation 5.12. As a forwarding node x, meets contacts on its way, it calculates the utility of each contact on encounter. The $\mathcal{U}(x_i,x,t)$ help the node x_i detects if it must forward any messages to node x and which messages must be forwarded to x. For instance, If there are multiple encunters, node x_i chooses the bestFit node that has the highest destination encounter probability compared to all of its neighbors, and send an adaptative number of message to it. Instead of sending all messages to all neighbors as happens in pure epidemic forwarding. Therefore, the $\mathcal{U}(x_i,x,t)$ is dependent on which forwarding algorithm is been used.

$$R(x_i,x,t) = \mathcal{U}(x_i,x,t) \tag{5.12}$$

Thus, using Equation 5.11 and 5.2 we can calculate the buffer occupation at time $t + 1$ by Equation 5.13.

$$O(x, t + 1) = \{(O(x, t) + B(x, t) + \mu(x, t) - \mathcal{S}(x, t)) \wedge \mathcal{Q}\} \tag{5.13}$$

Therefore, we say that a buffer is blocked at time t when Equation 5.14 has a true boolean value.

$$\{O(x, t) + B(x, t) + \mu(x, t) - \mathcal{S}(x, t)\} > \mathcal{Q} \tag{5.14}$$

5.8 Conclusion

In this chapter, we introduce the first DTN congestion model based on percolation theory, in particular, site-bond percolation. Using percolation theory to describe congestion in DTN scenarios allows us to propose a simple yet general model that derives the probability of delivering a message between a given source-destination pair. The resulting delivery probability is the probability that there exists a path that contains only uncongested nodes and active contact opportunities in the network. Ultimately, our goal is to use the proposed model to evaluate DTN congestion control mechanisms in terms of how cost-effective they are in preventing/containing congestion. Additionally, we use the theory of percolation to derive performance metrics that are key to understand DTN congestion and how to mitigate it.

The current model considers epidemic- and single-copy forwarding as data forwarding strategies. We plan to add to the model other forwarding strategies such as Prophet, Spray and Wait, Spray and Focus, etc. We will also extend our model to 2- and 3 dimensions as well as incorporate node mobility explicitly as described at Section 5.7. Henceforth, in light of the aforementioned observations and limitations of existing DTN congestion control mechanisms, in the next chapter we present a novel DTN congestion control framework called DTN-Learning which handles congestion in disconnected networks using Machine Learning easily deployable different DTN scenarios.

6 DTN-Learning: An Autonomous Congestion Control Mechanism

Dynamic ambients require systems that can learn and adapt, or otherwise interact intelligently with the environment in which they operate. The behavior of these systems must be achieved by means of intelligent algorithms, usually for tasks that involve some kind of learning. In this chapter, we describe DTN-Learning framework for congestion control in extreme environments (DTNs), in which a reinforcement learning (RL) module is embedded into each node of a challenge network. Our framework allows the nodes to adapt either their behavior or the way they perform congestion control "online[1]" while the environment is still being acknowleged.

6.1 Reinforcement Learning Overview

Reinforcement learning is a machine learning technique which has its origins on the cybernetics. It has been applied in statistics, psychology, neuroscience and computer science. In the last ten years, it has attracted attention of several scientific communities taking into account important applications, such as guiding the motion of characters in video games, control problems, simulating human crowds for planning and analysis, flight control systems for aircraft, sophisticated avionics systems, and coordinating teams of robots. Our work seeks to extend RL's concept which considers the problem faced by an agent that learns behavior through "trial-and-error" interactions with a dynamic environment (HARMON; HARMON, 1996) (SUTTON; BARTO, 1998) to DTN congestion problem.

Learning algorithms fall into three groups with respect to the sort of feedback that the learner has access to.

Supervised learning: for every input, the learner is provided with a target, in this case the environment tells the learner what its response should be. The learner compares its current

[1]In an on-line learning scenario, the program is given examples one by one, and it recalculates it hyphothesis of what it learns after each example. In contrast, in an off-line learning scenario the program receives all examples at once.

response to the target and adjusts its internal memory in such a way to produce a better response next time it receives the same input. Supervised learning can be seen as a simple categorization task. For example, if a program is learning to recognize the sounds of different birds and it is told each time what bird it is, it can compare its anwser with the correct one.

Unsupervised learning: the learner does not receive feedback from the environment. The learner's task is to represent the inputs in a more efficient way, as a cluster or using a reduced set of dimensions. There are no explicit target outputs or environmental evaluations associated with each input. Unsupervised learning can be seen as a perceptual system. For example, consider the case in which the inputs are the photoreceptor activities created by various images of an apple or an orange. In the space of all possible activities, these particular inputs form two clusters, with many fewer degrees of variation than, e.g. lower dimension. One natural task for unsupervised learning is to find and characterize these separate, low dimensional clusters.

Reinforcement learning: The input comes from an unpredicted environment and positive or negative feedback is given at the end of every small sequence of learning steps. The learner receives feedback about the appropriateness of its response. For example, consider a baby learning how to walk. It tries out various movements. Some work, so it moves forward and it is rewarded. Others fail, it stumbles or falls down and it is punished with some pain.

Note that both supervised and unsupervised learning are more appropriate for off-line learning scenario since the program receives all samples at once. Several researches have suggested that learning tasks are associated with the existence of a "teacher" in the environment that supplies performance information to the algorithm or controller (BARTO, 1991) (JORDAN; RUMELHART, 1990) (GULLAPALLI, 1992). With the exception of unsupervised learning tasks, the necessary information can be considered to be built into the learning system. In supervised learning tasks, the teacher provides target actions that specify how the controller should modify its actions so as to improve performance. In this case, the teacher instructs the controller about the actions to execute in order to improve performance. In contrast, the role of the teacher in reinforcement learning is more evaluative since it provides evaluations of consequences of the controller's actions, leaving it to the controller to determine how to modify its actions so as to obtain better evaluations in the future. Therefore in RL the teacher does not explicitly tell the controller what to do to improve performance, on the other hand, in supervised learning it does.

There are many situations where the correct answers that supervised learning requires are not known. For example, in a flight control system, the question would be the set of all sensor readings at a given time, and the answer would be how the flight control surfaces should move during the next milliseconds. A simple neural network cannot learn to fly the plane unless there

is a set of known answers, so if we do not know how to build a controller in the first place, simple supervised learning will not help.

For these reasons there has been much interest recently in RL which combines the fields of dynamic programming and supervised learning to yield powerful machine-learning systems (HARMON; HARMON, 1996). In RL the computer is given a goal to achieve. Then it learns how to achieve that goal by "trial-and-error" interactions with its environment.

6.1.1 Reinforcement Learning Model

In the standard RL model, autonomous agents can learn to act in an environment in order to achieve a desired goal (SUTTON, 1988). An agent performs actions that affect its state and environment, and receives a reward value indicating the quality of the performed action and state transition. This reward is used as feedback for the agent to make better future decisions.

Formally, the model consist of (see Figure 6.1):

- a discrete set of environment states S;
- a discrete set of agent actions A;
- a set of scalar reinforcement signals, typically $\{0, 1\}$ or the real numbers. In this case we have $R : S \times A \rightarrow r$.

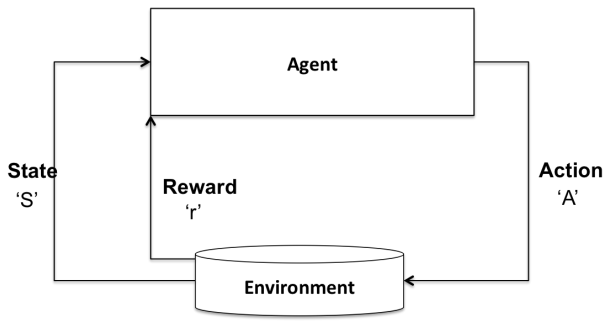


FIGURE 6.1 – Reinforcement learning model

The agent's goal is to find a policy π, mapping states to actions that maximizes the reward or some measure of reinforcement. If the RL system can observe perfectly the information in the environment that might influence the choice of action to perform, then the RL system chooses actions based on true states of the environment. This ideal case is the best possible basis for reinforcement learning and, in fact, is a necessary condition for much of the associated theory. In our case, the node chooses its action based on the buffer's state.

Algorithms for agent RL have been widely used for applications as robotics (MATARIC, 1997), game theory (BOWLING; VELOSO, 2002), scheduling (ZHANG; DIETTERICH, 1995), path

planning (SINGH *et al.*, 1994), and many others. A well known family of RL algorithms is based on the Temporal Difference (TD) prediction approach introduced in (SUTTON, 1988), which is a combination of Monte Carlo sampling methods and dynamic programming for learning in environments that are formulated as Markov Decision Processes (MDP). The most well known of these algorithms is Q-learning (WATKINS; DAYAN, 1992), in which agents learn an action value function that estimates the expected reward of choosing a specific action in a given state and following a fixed policy afterwards. We employ Q-learning in our framework, because it is able to perform online learning and cope with node mobility based on the contact history of neighbor nodes, without any prior assumptions of mobility patterns.

Different approaches have also been proposed to incorporate learning when multiple agents are present in a simulation scenario (BUŞONIU *et al.*, 2008) (UTHER; VELOSO, 1997). These multi-agent learning approaches originate both from extensions of RL algorithms and applications of game theory.

6.1.1.1 Reinforcement Learning and Control

In this section we highlight some points about RL and adaptive control. Mainly, we are interested here in some issues of learning to control complex systems.

Although problems in controlling complex process are usually considered the exclusive purview of control theory, many control problems of current importance suggest methods combining control theory and artificial intelligence (AI). The complexity of these control problems arise from non-linear, stochastic, or non-stationary process behavior, that makes them less appropriate to traditional approaches in control system design. Under such circumstances, researchers have proposed many heuristic techniques based on AI for intelligent systems that can operate in dynamic environments. "Learning control involves modifying the controller's behavior to improve its performance as measured by some predefined index of performance (IP)" (GULLAPALLI, 1992). If control actions that improve performance as measured by the IP are known, methods for learning from examples (supervised and unsupervised learning), can be used to train the controller. On the other hand, when control actions are not known a priori, appropriate control behavior has to be inferred from observations of the IP.

There are two methods that have been used for deciding how to modify the controller behavior accord to IP. Indirect methods refer to the construction of a model of IP to be used to supply training set to train the controller. Direct methods also called model-free methods try to disturb the environment and to observe the impact on the IP. Indirect methods are widely used in AI. On the other hand, prominent examples of direct methods are based on learning systems that infer appropriate actions depending on the feedback. Because of the similarity of this method with trial-and-error learning where the behavior of an animal (e.g., dogs) is modified by the ensuing reinforcement the psychologists have called it "reinforcement learning" (MENDEL; MCLAREN,

1994).

Although direct methods are often perceived as being weaker than indirect methods, a major point made in this dissertation is that they can yield faster and more reliable learning in control problems arising at dynamic environment. Clearly, as the complexity of the controlled process increases, it becomes more difficult for the designer to determine the control model, making adaptive methods more useful. For example, congestion control in traditional networks is governed by end-to-end paradigm opposite to DTN where end-to-end connectivity cannot be guaranteed. Therefore a global control in DTN is very complex and unfeasible. In this case a local control is more approriate since a controller can learn to recognize previously experienced situations. Then it select the appropriate control behavior it had stored for a long time in its memory. This implies that it is able to learn online leading to good overall performance.

As indicated by the above discussion, our research addresses issues in applying reinforcement learning to DTN congestion control problem. By implementing "shaping," a technique used in experimental psychology for training animals (SUTTON; BARTO, 1998) (SKINNER, 1951), we demonstrate how learning to solve DTN congestion problem in different DTN scenarios. In the next section we briefly describe the learning algorithm Q-learning that we have implemented on the ONE Simulator (KERäNEN *et al.*, 2009). It is the most popular RL technique and it is a very powerful algorithm, because theorems regarding its convergence to the optimal policy have set it on sound theoretical ground. It also seems to work well in practice (MAHADEVAN; CONNELL, 1992) (STRAUSS; SAHIN, 2008).

6.2 Q-learning Algorithm

Watkins (WATKINS; DAYAN, 1992) introduced the method of RL called Q-learning. The goal of the learning algorithm is to discover a policy that maximizes cumulative discounted reward. An agent tries to learn the optimal policy from its past interaction with the environment. Each agent has a history that is a sequence of *state-action-reward* (see Equation 6.1). This means that the agent was in state s_0 and did action a_0, which resulted in it receiving reward r_1 and so on.

$$\langle s_0, a_0, r_1 \rangle, \langle s_1, a_1, r_2 \rangle, \ldots \tag{6.1}$$

Let the history be a sequence of experiences, where an experience is given by a tuple $\langle s, a, r, s' \rangle$. In this case, the agent was in state s, it did action a, it received reward r and it went into state s'. These experiences are the data from which the agent can learn what to do.

In Q-learning, policies and the value functions are represented by a two-dimensional lookup table indexed by state-action pairs. Formally, for each state s and action a let $Q^*(s, a)$ be the expected value of doing a in state s and then following the optimal policy.

The Q-learning algorithm works by maintaining an estimate of the Q^* function and adjusting *Q-values* based on actions taken and reward received. This is done using Sutton's prediction difference (SUTTON, 1988) which the difference between the immediate reward received plus the discounted value of the next state and the *Q-values* of the current state-action pair (Equation 6.2).

$$r + \gamma V^*(s^{'}) - Q^*(s, a) \tag{6.2}$$

Equation 6.2 shows that r is the immediate reward, $s^{'}$ is the next state resulting from taking action a in state s and $V^*(s^{'}) = max_{a'}[Q^*(s^{'}, a^{'})]$. Then the values of Q^* are adjusted according to Equation 6.3

$$Q^*(s, a) = (1 - \eta)Q^*(s, a) + \eta(r + \gamma V^*(s^{'})) \tag{6.3}$$

where $\eta \in (0, 1]$ is a learning rate parameter. Note that the current estimate of the Q^* function implicitly defines a greedy policy by $\pi(s^{'}) = argmax_{a'}[Q^*(s^{'}, a^{'})]$. That is, the greedy policy is to select actions with the largest estimated *Q-values*.

Above we consider Q-learning in deterministic environments[2] (RUSSEL; NORVIG, 2003). In our case a nondeterministic case is expected. In the context of DTN congestion, the reward function can produce different rewards each time the transition $\langle s, a\rangle$ is repeated. Consequently, the training rule will repeatedly alter the values of $Q^*(s, a)$. Thus the training rule does not converge. To overcome this problem the training rule takes a decaying weighted average of the current Q^* values and the revised estimate. Let Q^*_n denotes the estimate on the nth iteration of the algorithm. The Equation 6.14 is sufficient to assure convergence of Q^* to *Q-values*.

$$Q^*_n(s, a) = (1 - \eta_n)Q^*_{n-1}(s, a) + \eta_n(r + \gamma V^*_{n-1}(s^{'})) \tag{6.4}$$

$$where$$

$$\eta_n = \frac{1}{1 + visits_n(s, a)} \tag{6.5}$$

$$and$$

$$V^*_{n-1}(s^{'}) = max_{a'}[Q^*_{n-1}(s^{'}, a^{'})] \tag{6.6}$$

Note that the $visits_n(s, a)$ is the total number of times the state-action pair has been visited up to and including the nth iteration. The main idea is that Q^* is made more gradually than in the deterministic case. Observe if we set η_n to 1 in Equation 6.4 we would have the deterministic

[2]Q-learning algorithm, assuming deterministic rewards and actions. The discount factor γ may be any constant such that $0 \leq \gamma < 1$.

case. Furthermore, the value of η_n in Equation 6.5 decreases as n increases, so that updates become smaller as training progresses. When reducing η_n during training, it is possible to reach the correct convergence of Q^* function. The standard Q-learning algorithm has several stages as shown in Algorithm 4. One particular benefit of using the Q-learning algorithm is that it does not require specific domain knowledge of the problem it is attempting to solve (ANDALORA, 2007).

Algorithm 4 Q-learning algorithm

```
procedure Q-LEARNING
    For each s and a, initialize the table entry Q(s, a) to zero
top:
    Observe the current state s
    Select action a through an action selection method and execute it
    Receive reward r
    Observe the new state s' and update the table entry Q(s, a) using values of r and Q(s', a)
    s ← s'
    goto top.
end procedure
```

It is important to note that the Q-learning method does not specify what actions the agent should take at each state as it updates its estimates. This means that Q-learning allows arbitrary experimentation while at the same time preserving the current best estimate of states' values. This is possible because Q-learning constructs a value function on the state-action space instead of the state space. The value function on the state-action is constructed indirectly by Q-learning. In addition, since this function is updated according to the ostensibly optimal choice of action at the following state, it does not matter what action is being followed at that state. Because of this, the estimated returns in Q-learning are not contaminated by experimental actions (WATKINS, 1989). Therefore Q-learning is not experimentation sensitive.

There are two methods for selecting action a from the possible actions in every state (SUTTON; BARTO, 1998).

- Exploration: allows selection of a different action from the one that it currently thinks is the best.
- Exploitation: allows selection of the action a that maximizes $Q^*(s, a)$.

It is clear that action selection is more exploration at the beginning of learning and is more explotation towards the end of learning. The next section describes two action selection methods we have been using in Q-learning to mitigate the DTN congestion.

6.2.1 Action Selection Methods

There have been a number of suggested ways to trade off exploration and exploitation. The greedy strategy is widely recognized in the RL and several exploration techniques have been suggested to overcome it (THRUN, 1992). One problem with a greedy strategy is that it treats all of the actions, apart from the best action, equivalently. If there are two good actions and more actions that look less promising, it may be more sensible to select among the good actions, putting more effort toward determining which of these promising actions is best, rather than putting in effort to explore the actions that look bad.

Another common strategy is to have the algorithm that selects actions with some amount of randomness during the initial learning period (WATKINS, 1989). But this approach has two drawbacks in the context of congestion control. First, the network is continuosly changing, thus the initial period of exploration never ends. Second random traffic has an extremely negative effect on congestion. Messages sent by nodes that do not know about the congestion level of their neighbors tend to add queue delays, slowing down all the messages passing through those queues, which adds further to queue delays, etc. Because the nodes make their policy decisions based on local information and sometimes based on neighbor's information, this increased congestion actually changes the problem the learners are trying to solve.

In real DTN scenarios, we cannot visit each $\langle s, a\rangle$ an infinite number of times and then exploit our knowledge, we can only approximate. It is possible to do a large finite amount of random exploration, then exploit. This is one of the simple methods we use on the Q-learning state spaces in this work.

On the larger states space Q-learning problems, random exploration takes far too long to focus on the best actions, so we use a method that interpolates exploration and exploitation. The idea is to start with high exploration and decrease it to nothing as time goes on, so that after a while we are only exploring $\langle s, a\rangle$'s that have worked out at least moderately well before.

The first action selection method we used is a fairly standard one in the field and originally comes from (WATKINS, 1989) and (SUTTON, 1990). The node tries out actions probabilistically based on their *Q-values* using Boltzmann distribution. Given a state s, it tries out action a with probability given by Equation 6.7.

$$\rho_s(a) = \frac{e^{\frac{Q^*(s,a)}{T}}}{\sum_{a' \in A} e^{\frac{Q^*(s,a')}{T}}} \tag{6.7}$$

Observe that $e^{\frac{Q^*(s,a)}{T}} > 0$ whether the *Q-values* is positive or negative. The "temperature" T controls the amount of exploration (the probability of executing actions other than the one with the highest *Q-values*). If T is high, or if *Q-values* are all the same time, this will pick a random

action. If T is low and *Q-values* are different, it will tend to pick the action with the highest *Q-values*.

In the beggining, Q^* is assumed to be inaccurate, so T is high (exploration) and actions all have an equal chance of being selected. T decreases as time goes on and it becomes more and more likely to pick among the actions with the higher *Q-values*, until finally, as we assume Q^* is converging to *Q-values*, T approaches zero (exploitation) and we tend to only pick the action with highest *Q-values*.

The second action selection method, that we called WoLF (Win or Learn Fast) was introduced in (BOWLING; VELOSO, 2001) and (GODOY *et al.*, 2013). The basic idea is to use two learning rates "to maximize the probability of choosing a profitable action and slowly decrease the probability of an action that is less beneficial to the agent". In (GODOY *et al.*, 2013), the authors use this strategy of action selection to improve the local navigation of agents in crowd simulations. In other words, they try to alleviate congestion phenomena observed in non-adaptive local navigation methods. The rationale behind the use of this strategy is that when a node is congested and it selects actions that return negative rewards, it must try to reduce the congestion level and make a slow transiton to a state that we call decrease congestion. This means that the node should keep a reasonable probability on actions that have proven to be profitable in the recent past. Hence, we want the node to gradually reduce its congestion level before make a direct transition to a non-congested state. In this sense, we belive that if the node is highly congested, the gradual decrease on probabilities will eventually force the node to pass by an intermediate state when it is very close to the non-congested state. Thus, the node can keep the buffer's utility.

WoLF defines two learning rates: γ_{min} and γ_{max}. They are used when the node is decreasing and increasing its rewards, respectively. Initially, because the node has not explored the environment and does not know the consequences of its actions, every action a has the same probability of being selected (Equation 6.8). Note that A denotes the set of all available actions for the node at state s.

$$\forall i \in A\colon \rho_{a_i} = \frac{1}{|A|} \tag{6.8}$$

Consequently, if the node took an action a_i at timestep $t-1$ that increases its reward when comparing to the previous reward at timestep t, at time t it updates the probability of selecting a_i in the future. The node performs this by Equation 6.9, where $\rho_{a_i}^{t-1} + \gamma_{max} < 1$. In the Equation 6.10 the probability of choosing another action j is randomly distributed.

$$\rho^t_{a_i} = \rho^{t-1}_{a_i} + \gamma_{max} \tag{6.9}$$

$$\rho^t_{a_j} = \frac{1-\rho^t_{a_i}}{|A-1|}, \forall j \in A, j \neq i \tag{6.10}$$

On the other hand, if the node's reward was decreased, $\rho^t_{a_i}$ is updated as shown in Equation 6.11, where $\rho^{t-1}_{a_i} - \gamma_{min} > 0$. In Equation 6.12 instead, the probability of choosing another action j is incremented at the same rate, with respect to its individual previous probabilty.

$$\rho^t_{a_i} = \rho^{t-1}_{a_i} - \gamma_{min} \tag{6.11}$$

$$\rho^t_{a_j} = \rho^{t-1+}_{a_j} \frac{\gamma_{min}}{|A-1|} \tag{6.12}$$

In fact, we can note that when a node at state s applies an action a and receives a higher reward, the probability of chosing the action a in the future is very high. However, the node still keeps a small probability for choosing other actions, which allows it to continuously explore the dynamic environment preventing the node from getting stuck at local optimal. On the other hand, if after applying an action a the node receives a smaller reward, the probability of selected action a in the future is reduced by γ_{min}. Thus gradually more weight is given to other more potential actions.

Note that with both action selection methods previously discussed, the node learns to take more conservative actions in the presence of congestion, and more operative ones in the absence of it. Nodes using our approach will be able to adapt to different DTN scenarios. In the next section we describe how we mapped Q-learning in the DTN congestion problem context and introduce a Q-learning based congestion-aware framework.

6.3 DTN-learning: A Congestion-Aware Framework

In this section, we present DTN-learning a novel congestion-aware framework for DTN environments based on reinforcement learning. The task of DTN-learning is to find an optimal congestion control strategy for networks subject to high delays and intermittent connectivity. In DTN-learning, Q-learning is used to first learn a representation of the node's buffer state in terms of *Q-values* and then use these values to make congestion control decisions. Each node x in the network represents locally its own view of the buffer state through its $Qtable$ (table that stores values of $Q(s, a)$). Given this representation of the state, the action a at node x is to mitigate congestion independently of routing protocol, actuating proactively and reactively to minimize the number of messages discarded and maximize the delivery ratio.

6.3.1 Node's State Machine

Our approach entails that a node makes congestion control decisions based on its local information (that sometimes includes neighbor information) - the input rate, output rate and available buffer space at the node, which along with their derivatives are used to predict the level of congestion in a DTN. The possibility of congestion is indicated by a congestion detection daemon where a node can be in one of the states shown in Figure 6.2: Congested, Prospective-congested (PCongested), Decrease-congested (DCongested) and Non-congested (NCongested). Note that each node is able to predict if its buffer occupancy rate is increasing (EWMA-POSITIVE) or decreasing (EWMA-NEGATIVE). It does this using EWMA (Exponentially Weighted Moving Average) (see Appendix A).

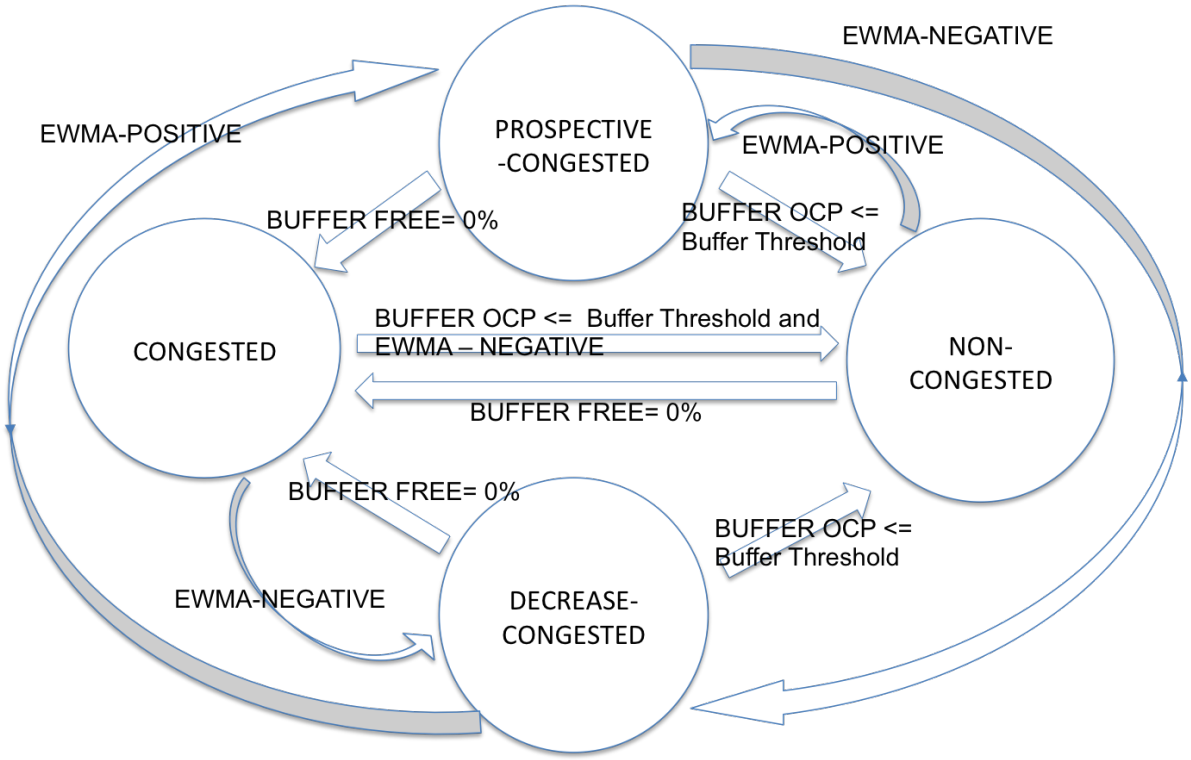


FIGURE 6.2 – Node states

In this context, the congestion level indicated by the daemon will depend on the available buffer space. In particular the node can make transition as shown in Figure 6.2 and as described above. Note that in Figure 6.2, *BUFFER OCP* is the current buffer occupancy rate and *BUFFER FREE* is the percentage of available buffer space.

1. Congested: the node makes a transition to *Congested* state if there is no available space on node's buffer (the node's buffer occupancy rate is 100%).

2. PCongested: If the node's buffer occupancy rate is growing (*EWMA-POSITIVE*) it moves to *PCongested* state. In that instance we are less conservative to implement a more proac-

tive congestion control approach. This state is important to indicate how imminent the threat of buffer exhaustion is.

3. DCongested: If the node's buffer occupancy rate is decreasing (*EWMA-NEGATIVE*) it moves to *DCongested* state. This state indicates that the nodes are already available to receive messages. Our approach uses this state to provide network utility.

4. NCongested: In this state the node is in a terminal state (the goal was reached by the controller). Our approach is conservative here in the sense that a buffer occupancy threshold (*Buffer Threshold*) exists that determines the transition for this state and at the same time the node's buffer occupancy rate must have a tendency to decrease (*EWMA-NEGATIVE*). This indicates the least and most likely of impending congestion the node is up to.

6.3.2 Actions

This section presents the action space that can be within one of the node's states. The set of actions available may differ from state to state. The action selection methods used by our approach were dicussed at Section 6.2.1. Our framework uses techniques from RL allowing the nodes to adapt their behavior online while still recognizing the environment. It is general and can be easily combined with existing schemes for DTN congestion control. We define a set of actions that represent different ways of treating DTN congestion and we have associate these actions with the node's buffer states.

Given the state space discussed in Section 6.3.1, we consider the following actions shown at Table 6.1 which are the actions that the node can perform.

Next we give a brief overview of each action.

- Increase message generation period: each node is assumed to generate local messages according to a message generation period. This period is multiplied by 2. This means that the node reduces its message generation frequency.

- Broadcast CN - Congestion Notification: the node generates a notification message that indicates how imminent the threat of buffer overflow is. The CN is broadcasted to all neighbors. Each neighbor that is able to listen to this CN, stores it in the memory. The next time they encounter each other before sending any message they check if any CN exists. If a CN exists and the information of buffer fill up time carried by the CN is zero (the node's state is Congested) no message is sent. Otherwise if a CN exists and the information of buffer fill up time carried by the CN is different of zero (the node's state is PCongested). If the node realises that the buffer fill up time is less its current time, it does not send any message. The Algorithm 5 describes how the broadcast CN happens. Note that Line 8 shows how we calculate the buffer fill up time.

Actions	**States**			
	Congested	**Prospective-Congested**	**Decrease-Congested**	**Non-Congested**
Increase message generation period	×	×		
Broadcast CN - Congestion Notification	×	×		
Discard expired message	×	×	×	×
Discard old message	×	×	×	×
Discard random message	×	×		
Discard message that will expire before next contact arises	×	×		
Discard oldest messages until space available	×			
Migrate messages	×			
Broadcast DCN - Decrease Congestion Notification			×	
Decrease message generation period			×	
Receive messages		×	×	×
Forward messages	×	×	×	×

×: the action can be taken when the node is in the state.

TABLE 6.1 – State action space table

- Discard expired message: the node discards all messages in the buffer that their TTL has already expired.
- Discard old message: the node discards the oldest message from its buffer. The message that has the smallest TTL.
- Discard random message: the node randomly chooses a message to discard.
- Discard message that will expire before next contact arises: each time a node has a contact it stores the information about this contact and calculates an estimative of when the next contact will happen using EWMA. Then it will verify if any message exists in its buffer that will expire (TTL) before the next predicted contact arises. Figure 6.3 shows how our approach works to predict the next contact. Note that each node keeps a contact map that contains the information about the next predicted contact for different and past encounters. $EWMA_t$ represents a prediction of the future contact at time t and Y_t a current contact at time t.

Algorithm 5 Broadcast CN

```
1:  procedure BROADCAST CN
2:      id ← message id
3:      time_for_buffer_fill ← time for available buffer space to fill up
4:      id_node ← node id
5:      if node state = Congested then
6:          time_for_buffer_fill ← 0.
7:      else if node state = PCongested then
8:          time_for_buffer_fill ← available_buffer_space / (input_rate - output_rate).
9:      end if
10:     create a CN(id, id_node, time_for_buffer_fill).
11:     broadcast CN to neighbors.
12:     if node is receiving a CN then
13:         check if exist an old CN with the same id_node if so remove it from CN list.
14:         store the new CN at CN list.
15:         check if exist an old DCN with the same id_node if so remove it from DCN list.
16:     end if
17:     if node is sending a message then
18:         if exist an CN in the CN list for the destination node then
19:             if time_for_buffer_fill = 0. then
20:                 Don't send any message.
21:             else if time_for_buffer_fill != 0. then
22:                 if the time_for_buffer_fill has already been reached then
23:                     Don't send any message.
24:                 end if
25:             end if
26:         end if
27:     end if
28: end procedure
```

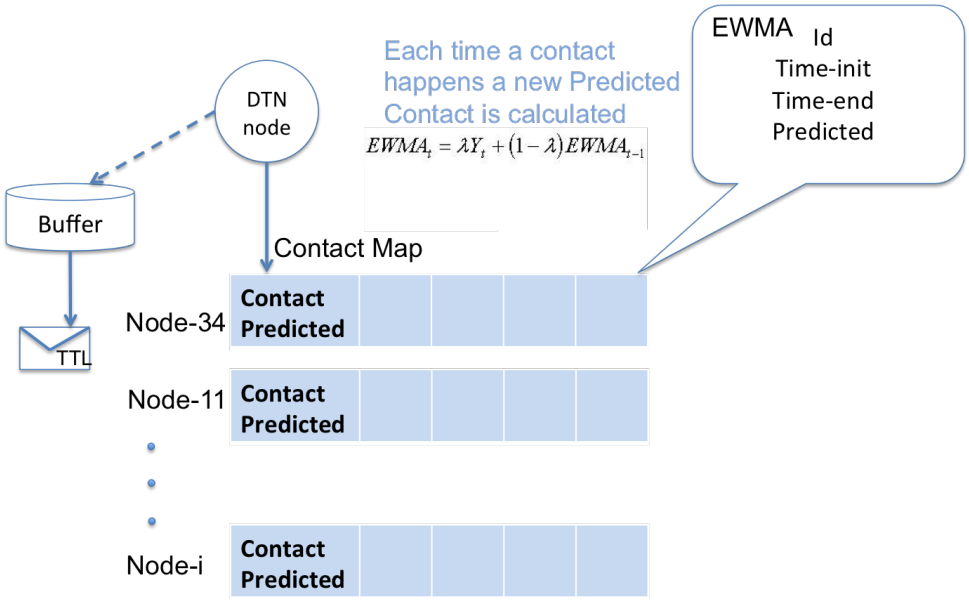


FIGURE 6.3 – Contact prediction using EWMA

- Discard oldest messages until space available: the node discards as many messages from

its buffer until enough available space exists to store the arriving message.

- Migrate messages: the node randomly selects messages and sends them out to neighbors with available storage.
- Broadcast DCN - Decrease Congestion Notification: the node broadcasts a DCN for all neighbors to indicate that it is available to receive messages. The Algorithm 6 shows how the broadcast DCN process happens.

Algorithm 6 Broadcast DCN

```
1:  procedure BROADCAST DCN
2:      id ← message id
3:      dcn_flag ← is the buffer occupancy rate decreasing (EWMA)?
4:      id_node ← node id
5:      if (node state = DCongested ) then
6:          dcn_flag ← true.
7:      end if
8:      create a DCN(id, id_node, dcn_flag).
9:      broadcast DCN to neighbors.
10:     if node is receiving a DCN then
11:         if exist an old DCN with the same id_node then
12:             remove the old DCN from DCN list and store the one arriving.
13:         end if
14:         if exist an old CN with the same id_node then
15:             remove the old CN from CN list.
16:         end if
17:     end if
18: end procedure
```

- Decrease message generation period: each node is assumed to generate local messages accord to a message generation period. This period is divided by 2. This means that the node increases its message generation frequency.
- Receive messages: the node has enough available space to store new messages.
- Forward messages: the node is in contact with other node and there is a high probability to forward messages.

We should point out that, in our framework, we are not only interested in nodes that reach a convergence in their local control, but also make them adaptable to different DTN environments, even if this sometimes implies they will choose suboptimal actions. As previously discussed, the nodes use feedback received from the environment to make decisions about actions that influence their behavior (e.g., change from Congested state to DCongested state). This implicitly leads to an increasing of delivery probability and consequently reduces the message drop ratio.

6.3.3 Feedback Function

A feedback function defines the goal in a reinforcement learning problem. it maps each perceived state (or state-action pair) of the environment to a single number, a reward, indicating the intrinsic desirability of that state. A reinforcement learning node's sole objective is to maximize the total reward it receives in the long run. The feedback function defines what the good and bad events are for the node.

The Q-learning process needs a feedback function that evaluates the results of an action in a given state in order to reward the correct behavior by increasing the *Q-values* of a good action or by decreasing the *Q-values* of a bad action. Typically, the feedback function is not defined in terms of what action we took but rather what state we arrived at (e.g., DCongested and NCongested states), that is, r is a function of the transition s to $s^{'}$. The general idea is that we cannot reward the right action a because we do not know what it is. That is why the node is learning. If we knew the correct a we could use supervised learning.

In our approach, rewards are associated with states. The node is not just rewarded for arriving at state s but also rewarded continually for remaining in state s. Thus the goal from the node's point of view, is that it keeps looking at not being in the Congested or PCongested state. Because being in one of these states causes punishment (negative reward). Since the node is congested the delivery ratio decreases and the drop ratio increases degrading the network's performance.

The rules dictated by the feedback function imposes certain behavior on the node when the congestion takes place. Therefore, the function itself and its coefficients have to be carefully chosen in order to emerge with wanted behaviors and eliminate unwanted behaviors. As mentioned above, our approach associates the reward with the states. To define this, Table 6.2 depicts the reward values for each state. The reward value is defined by a scale from -1 to 1 (which could be different for different situations and environments). Where -1 indicates that the node has made a transition to the worse state (Congested) and 1 indicates a transition to the best state (NCongested). In order to improve and to keep network's utility we set up a smaller positive reward value to the transition to DCongested state of 0.5 which is still a good state and a smaller negative reward value to PCongested of -0.5 since the node is not completely congested.

Intuitively, we consider the fact that a node is of the form shown in Equation 6.13. Where $r > r^{'}$, then it is irrelevant what r and $r^{'}$ actually are, the node will still learn the same policy. So setting $r = 1$ and $r^{'} = -1$ here does not reduce our options. If we replace 1 above for any number greater than zero, the node still learns the same pattern of behavior. It still sends the

States s	**States** s'			
	Congested	**Non-Congested**	**Prospective-Congested**	**Decrease-Congested**
Congested	-1	1	-	0.5
Non-Congested	-1	1	-0.5	0.5
Prospective-Congested	-1	1	-0.5	-
Decrease-Congested	-1	1	-0.5	0.5

- : the transition does not exist.

TABLE 6.2 – Feedback function

same preferred action a to the controller.

$$Node_i\ reward:\ if\ (good\ action)\, r\ else\ r' \tag{6.13}$$

The values given to the feedback function are experimental. The negative reward for Congested has to be strong to penalize the *Q-values* no matter what. And NCongested and DCongested have to be rewarded so the node will explore its environment and by exploring the environment, the node reinforces various values of state-action pairs.

6.3.4 Evaluation

The primary goal of our evaluation is to show that DTN-learning achieves a high message delivery ratio and good latency, while maintaining low overhead and good goodput. To demonstrate this, we first present the metrics used in our evaluation, followed by a brief description of two different scenarios used in our experiments. Then we show the performance of our reinforcement learning scheme. In addition, to determine how DTN-learning reacts to changes in internal parameters, we evaluate DTN-learning against itself using different parameter settings. Furthermore, to demonstrate the effectiveness of DTN-learning on IPN scenarios, we design some use cases for IPN scenarios where RL is combined with supervised learning. Finally, we present a comprehensive evaluation of DTN-learning in comparison to four other DTN congestion control mechanisms studied in Chapter 4. To perform our evaluation, we use the Opportunistic Network Environment simulator (ONE) (KERäNEN *et al.*, 2009), which is a simulation environment designed specifically for DTNs.

Each simulation lasts for twelve simulated hours unless otherwise specified. Each data point is the average of at least 5 runs, with 95% confidence intervals displayed. Results when no congestion control is employed are used as the performance baseline.

6.3.4.1 Metrics

The evaluation of many current DTN congestion control mechanisms is performed by traditional performance metrics including average message delivery ratio and end-to-end latency, while resource usage can be captured by goodput.

We consider four main performance metrics, namely:

1. **Delivery ratio** is the ratio between the number of received messages at destination nodes to the number of created messages (see Equation 6.14).

$$\text{delivery ratio} = \frac{\text{number of received messages}}{\text{number of created messages}} \times 100\% \tag{6.14}$$

2. **End-to-End latency** is the average time interval to deliver messages to their destinations (see Equation 6.15 where t_i is defined as the time that message i took to reach the destination and t_c is the message creation time).

$$\text{end-to-end latency} = \frac{\sum_{i=1}^{\text{number of messages received}} (t_i - t_c)}{\text{number of messages received}} \tag{6.15}$$

3. **Overhead** is the difference between the number of delivered messages and the number of relayed messages, divided by the number of delivered messages (see Equation 6.16). The overhead reflects the number of messages relayed to deliver a single message. The lower the value of overhead, the more efficient the strategy is.

$$\text{overhead} = \frac{\text{number of relayed messages} - \text{number of received messages}}{\text{number of received messages}} \tag{6.16}$$

4. **Goodput** is defined as the number of messages received divided by the total number of messages transferred (including those tranfers that did not result in a delivery) (see Equation 6.17).

$$\text{goodput} = \frac{\text{number of messages received}}{\text{number of relayed messages}} \times 100\% \tag{6.17}$$

6.3.4.2 Scenarios

For our experiments, we use two different scenarios. The first one is a terrestrial DTN scenario. In this scenario, 50 nodes are placed in an remote area and they must forward messages in an overload network. The second was the IPN scenario, 5 nodes are placed in a deep space environment and the nodes must communicate among them and some of them must immediately face congestion. We give more details about each scenarios in the next sections.

Parameters		
Name	Description	Value
Scenario.endTime	simulation time	43200 seconds
btInterface.transmitSpeed	bandwidth	2.5 Mbps
btInterface.transmitRange	transmitting range	150 m
Group.router	routing protocol	[EpidemicRouter, ProphetRouter, SprayAndWaitRouter (10 msg copies)]
Group.movementModel	mobility model	[RandomWayPoint, RandomWalk, ShortestPathMapBasedMovement]
Group.bufferSize	node buffer size	4000 KB
Group.bufferThreshold	percentage value that indicates if the node is Non-Congested	[50, 60, 70, 80, 90]%
Group.alphaEWMA	the weight assigned to the current observation at EWMA function	[0.05, 0.20, 0.40, 0.60, 0.80]
Group.gammaGlearning	the discounted estimated future value at Q-learning function	[0.20, 0.40, 0.60, 0.80, 1.0]
Group.actionSelectionMethod	action selection methods	[boltzmann, wolf]
Group.gammaMin	minimum learning rate for WOLF action selection method	0.1
Group.gammaMax	maximum learning rate for WOLF action selection method	0.9 -$\rho_{a_i}^{t-1}$
Group.msgTTL	message time to live	30000 seconds
Group.nrofHosts	number of nodes in network	50
Group.speed	max and min speed that the nodes must move	{0.5, 1.5}m/s
Movimentmodel.worldSize	area where simulation takes place	1km x 1km (RandomWayPoint, RandomWalk) and 6km x 6km (ShortestPathMapBasedMovement)
Events1.size	message size	{1, 100} KB
Events1.interval	Creation interval in seconds, i.e. one new message every 1 to 100 seconds	[1-100, 1-200, 1-300, 1-400, 1-500] seconds

TABLE 6.3 – Simulation parameters and their values for terrestrial scenario

6.3.4.2.1 Terrestrial DTN Scenario

The terrestrial scenarios we simulate include 50 nodes that move according to a pre-defined mobility regime. As they move, nodes encounter one another "opportunistically." During these "opportunistic contacts," nodes can exchange messages. Figure 6.4 shows the output of the ONE simulator's graphic interface illustrating a snapshot of a terrestrial DTN scenario. We assume that all nodes have the same transmission ranges. Table 6.3 lists the simulation parameter settings we used in our experiments.

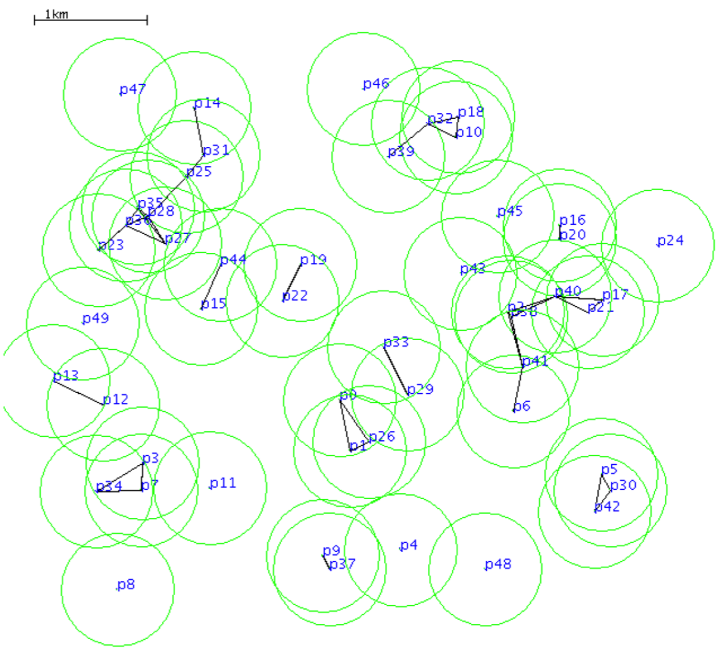


FIGURE 6.4 – Terrestrial network scenario.

Three mobility models were used in the scenario, namely Random Walk, Random Way Point, and Shortest Path Map-Based Movement. In Random Walk (RW) (CAMP *et al.*, 2002), a

node moves from current location to a new location for a time interval t by randomly choosing a direction and speed from the predefined ranges. At the end of the time interval a new direction and speed are calculated. The Random Way Point (RWP) (CAMP *et al.*, 2002) model is a generalization of RW. In this model a mobile node stays at a given location for a certain period of time; then, the node moves to a new random destination at a random speed chosen from [0-MAXSPEED]. The Shortest Path Map-Based Movement (SPMBM) (KERANEN; OTT, 2007) model uses Dijkstra's shortest path algorithm to calculate the shortest path from the current location to a randomly selected destination. Destinations can be chosen in a random way or from a set of Points of Interests (POIs). When a node reaches its destination, it waits for the time interval t and then heads toward a new destination by using the shortest path. In SPMBM, nodes do not move in a completely random way: they follow the shortest paths to reach their destinations.

In our simulations, for the RWP and SPMBM mobility models the minimum and maximum wait time after a node reaches the destination are 0 and 120 seconds, respectively.

6.3.4.2.2 Interplanetary Network Scenario

Our IPN (Interplanetary Network) scenario features high latency (because of astronomical distances) and scheduled contacts, i.e., node encounters that are known a priori. There are five nodes, representing a Base Station on the surface of the Earth which sends data to two rovers on Mars through two satellites located near Mars (see Figure 6.5). Simulation parameters are set to correspond to realistic conditions. As such, links between the satellites and the rovers on the Martian surface are set to 1s propagation delay while the links between the base station on Earth and the satellites at Mars are set to 240s propagation delay.

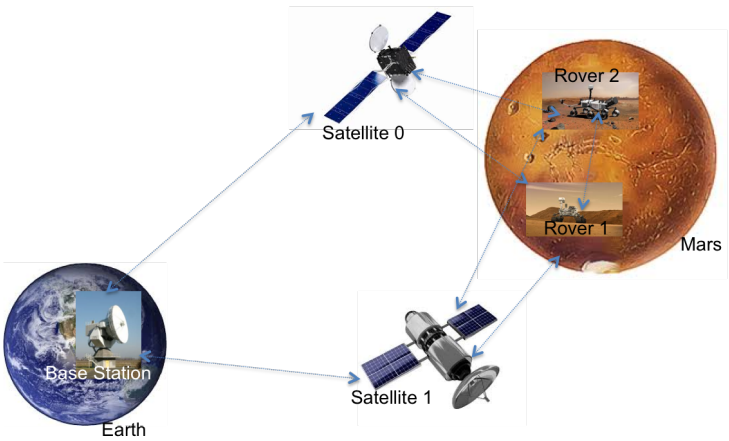


FIGURE 6.5 – Interplanetary network scenario.

We mainly focus on testing how congestion is affected by the inter-contact time and the duration of contact. Thus a scheduled contact table was created (see Table 6.4) which records

the time when a connection is created from one node to another. Also, it records the time when a connection drops between two nodes. The Up and Down connection times are asserted a priori in the scheduled contact table. Note that the communication channel is asymmetric. Although Table 6.4 does not indicate when the connection from *Satellite 0* to *Base Station* is Up and Down, we assume that the state of the connection from *Satellite 0* to *Base Station* is always the same as the state of the connection from *Base Station* to *Satellite 0*. For example, according to the first line of Table 6.4, at time $2000s$, the connection between *Base Station* and *Satellite 0* is "up" but goes"down" at time $3000s$. Then, the same nodes are again in contact at time $10000s$ (line 5) for $1000s$ (line 6).

Time (s)	Identifier	Initial Node	End Node	State
2000	CONN	Base Station	Satellite 0	up
3000	CONN	Base Station	Satellite 0	down
6000	CONN	Base Station	Satellite 1	up
7000	CONN	Base Station	Satellite 1	down
10000	CONN	Base Station	Satellite 0	up
11000	CONN	Base Station	Satellite 0	down
17000	CONN	Base Station	Satellite 1	up
18000	CONN	Base Station	Satellite 1	down
20000	CONN	Rover 2	Rover 1	up
21000	CONN	Rover 2	Rover 1	down
20000	CONN	Satellite 1	Rover 2	up
21000	CONN	Satellite 1	Rover 2	down
21100	CONN	Satellite 1	Rover 1	up
22100	CONN	Satellite 1	Rover 1	down
25000	CONN	Base Station	Satellite 1	up
26000	CONN	Base Station	Satellite 1	down
30000	CONN	Satellite 0	Rover 2	up
31000	CONN	Satellite 0	Rover 2	down
33000	CONN	Satellite 1	Rover 1	up
34000	CONN	Satellite 1	Rover 1	down
38000	CONN	Satellite 0	Rover 2	up
39000	CONN	Satellite 0	Rover 2	down

TABLE 6.4 – Example of scheduled contact table.

We use 5 different scheduled contact tables that differ in inter-contact time. We named the five different contact schedules as Contact1, Contact2, Contact3, Contact4, and Contact5. We increase the inter-contact time by $1000s$ as we go from Contact$i$ to Contact$i+1$.

Parameters		
Name	Description	Value
Scenario.endTime	simulation time	43200 seconds
btInterface.transmitSpeed	bandwidth	2.5 Mbps
Group.router	routing protocol	[EpidemicRouter, ProphetRouter, SprayAndWaitRouter (10 msg copies)]
Group.movementModel	mobility model	StationaryMovement
Group.bufferSize	node buffer size	4000 KB
Group.bufferThreshold	percentage value that indicates if the node is Non-Congested	[50, 60, 70, 80, 90]%
Group.alphaEWMA	the weight assigned to the current observation at EWMA function	[0.05, 0.20, 0.40, 0.60, 0.80]
Group.gammaGlearning	the discounted estimated future value at Q-learning function	[0.20, 0.40, 0.60, 0.80, 1.0]
Group.actionSelectionMethod	action selection methods	[boltzmann, wolf]
Group.gammaMin	minimum learning rate for WOLF action selection method	0.1
Group.gammaMax	maximum learning rate for WOLF action selection method	0.9 -$\rho_{a_i}^{t-1}$
Group.msgTTL	message time to live	30000 seconds
Group.nrofHosts	number of nodes in network	5
Movimentmodel.worldSize	area where simulation takes place	6km x 6km
Events1.size	message size	{1, 100} KB
Events1.interval	Creation interval in seconds, i.e. one new message every 1 to 100 seconds	[1-100, 1-200, 1-300, 1-400, 1-500] seconds

TABLE 6.5 – Simulation parameters and their values for the IPN scenario

Nodes generate messages according to the $Events1.interval$ parameter (see Table 6.5) where its value is $[x_i, x_j) = x_i \leq$ *next-event-time* $< x_j$. Uniformly distributed value between x_i (inclusive) and the x_j value (exclusive). We vary the message generation rate according to

the values listed in Table 6.5 to show how this parameter affects the performance of the different congestion control schemes.

6.3.4.3 Results and Analysis

In this section, we evaluate our framework, named DTN-learning using simulation in different scenarios (terrestrial and IPN scenarios), e.g., different values of buffer threshold, buffer sizes, different inter-contact times etc., and also we show the performance of learning process which DTN-learning is based on. Futhermore, we compare our framework with the state-of-the-art DTN congestion control mechanisms in the literature. We mainly compare DTN-learning with four other DTN congestion controls: AFNER, CCC, RRCC and SR (see Chapter 4 for more details about these mechanisms). Moreover, a deeper study of DTN-learning evaluation under the effect of different mobility models and different routing protocols is provided.

As is evident that in IPN scenario the learning time can be a constraint mainly due to the excessive long inter-contact times, we also present use cases for IPN scenario where DTN-learning is combined with supervised learning.

6.3.4.3.1 Simulation Results for Terrestrial Scenario

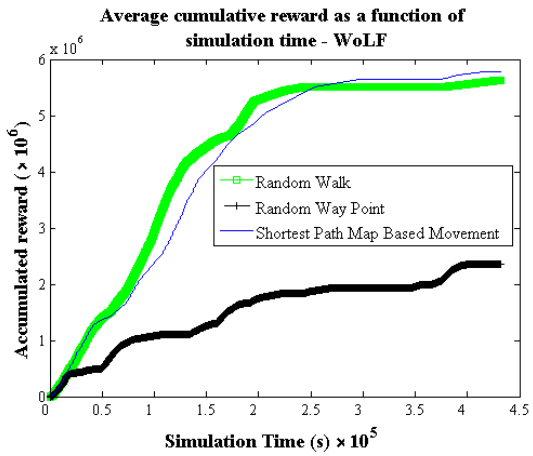
In this section, we present simulation results from our terrestrial scenarios which are characterized by a great number of opportunistic contacts and consequently shorter inter-contact times.

6.3.4.3.2 DTN-Learning Algorithm Evaluation

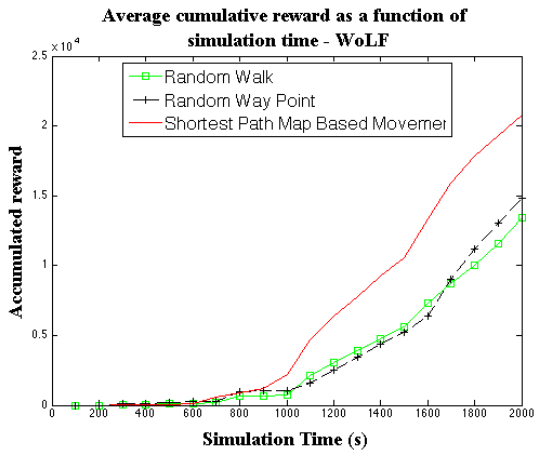
We judge a reinforcement learning algorithm by how good a policy it finds and how much reward it receives while acting in the environment (YADAV; SHRIVASTAVA, 2010) (INSA-CABRERA *et al.*,). Which is more important depends on how the node will be deployed. If there is enough time for the node to learn safely before it is deployed the final policy is more important. If the node has to learn while being deployed, it may never get to the stage where it has learned the optimal policy, and the reward it receives while learning may be what it wants to maximize.

Figure 6.6 shows the cumulative reward for different mobility models as a fucntion of simulation time. For this domain, the cumulative rewards depend on whether the node learns to visit the congestion control action when it is in one of the states: Congested, PCongested or DCongested. The cumulative reward therefore tends to be positive if the node makes a transition for either NonCongested or DCongested states. As can be seen from the plots, the accuracy of DTN-learning using Q-learning algorithm based on reinforcement learning is satisfying. Figures 6.6a and 6.6b show the cumulative rewards when using WoLF action selection method

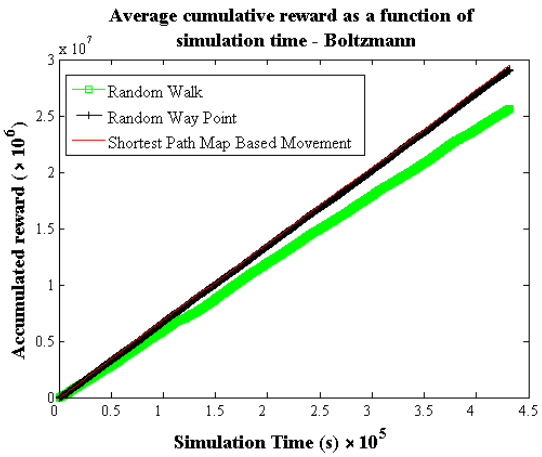
as a function of the simulation time. Figure 6.6b captures the cumulative reward between $0s$ and $2000s$ in order to have a better view of the learning performance for DTN-learning. Note that for different mobility models the node using our approach have a learning period that ends around $0.25 \times 10^5 s$ (see Figure6.6a) and it clearly seems to learn when simulation time increases. However, RW and SPMBM have a higher accumulated reward when comparing with RWP. It is interesting to point out here that both mobility models RW and SPMBM are different and they have similar cumulative reward. On the other hand, when RWP is used the accumulated reward is smaller and as it is known RWP is a RW instance. Thus this result confirms that DTN-learning works independent of the mobility since we are interested in observe if the cumulative reward increases when the simulation time grows.



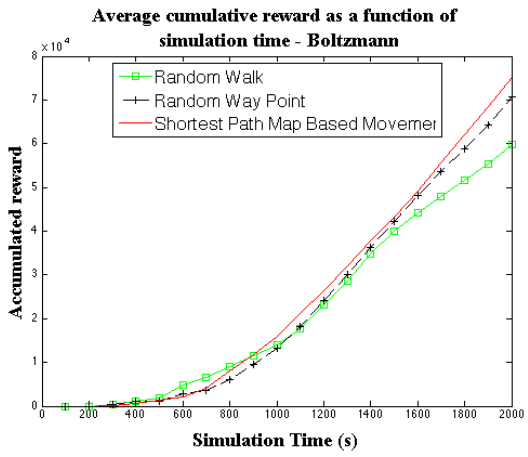
(a) WoLF



(b) WoLF - Snapshot with Simulation Time From $0s$ To $2000s$



(c) Boltzmann



(d) Boltzmann - Snapshot with Simulation Time From $0s$ To $2000s$

FIGURE 6.6 – Average cumulative reward as a function of the simulation time (Terrestrial Scenario, Simulation Time $12h$, Buffer Size $4000kB$, Buffer Threshold 60%, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, α EWMA 0.80, γ Q-learning 0.2, Message Generation Period $300s$, Random Way Point Mobility, Epidemic Routing).

As we can see in Figure 6.6c for results using Boltzmann action selection method for dif-

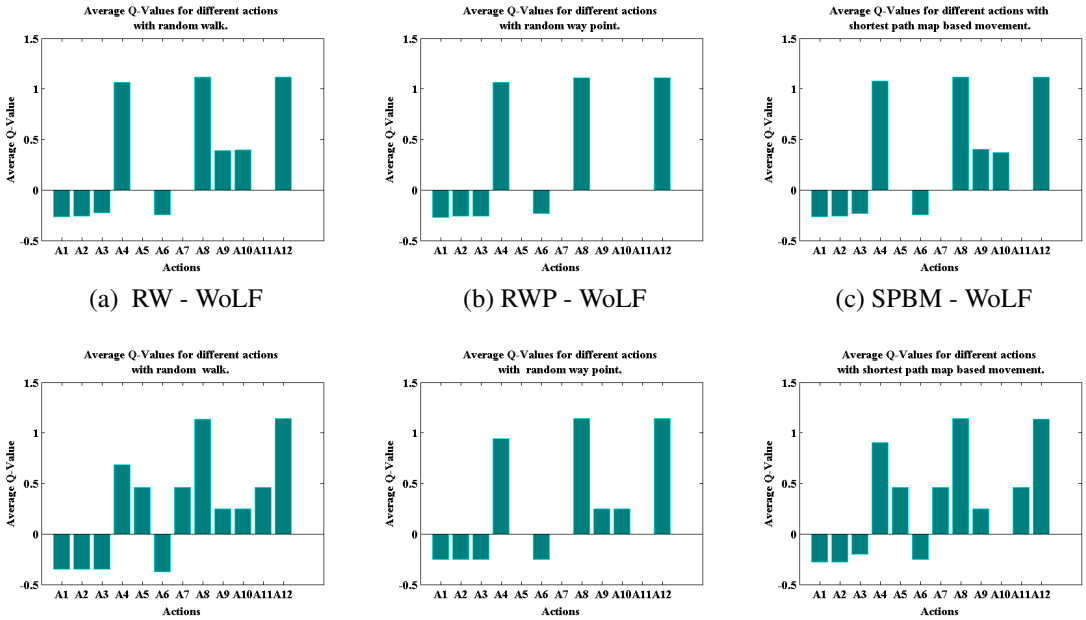
ferent mobility models the cumulative reward is similar. Therefore the cumulative reward is directly impacted by the action selection method been used. We can note that the cumulative reward in terrestrial scenario with WoLF method is greater than when using Boltzmann method. It is important to point out that WoLF is more explorative than Boltzmann. This happens because in WoLF the node gradually redistributes the probability of chosing a profitable action among all actions. As a result the node does more exploration. Our experiments uses a constant temperature $T = 1$ for Boltzmann method.

TABLE 6.6 – Action set and their descriptions.

Actions	
ID	**Description**
$A1$	Increase message generation period
$A2$	Broadcast CNB
$A3$	Discard expired message
$A4$	Discard oldest message
$A5$	Discard random message
$A6$	Discard message that will expire before the next contact arises
$A7$	Migrate messages
$A8$	Receive messages
$A9$	Decrease message generation period
$A10$	Broadcast DCNB
$A11$	Discard oldest message until space available
$A12$	Forward message

Figure 6.7 shows the average *Q-value* for different actions and mobility models (RandomWalk, Random Way Point and Shortest Path Based Map Moviment) for the terrestrial scenario. The *Q-value* expresses the action choice preferences. Table 6.6 shows the description of each action. These results indicate, for each action under a given selection method, which was its average *Q-value*. Note that a high *Q-value* is equivalent to a good action. A good action is expected to return a positive reward. Thus the node learns through the action's *Q-value* which are the good actions to mitigate congestion. We can note that actions $A4$, $A8$ and $A12$ have the highest *Q-values* for different action selection methods. The actions $A8$ and $A12$ allows to maximize the network utility receiving and forwarding messages. This result shows that DTN-learning is able to keep network utilization and in the same time that it mitigates congestion. The action $A4$ refers to discarding of oldest messages either when congestion takes place or to prevent congestion. As the Epidemic routing protocol was used several messages copies exist in the network. Discarding oldest messages is equivalent to drop messages that were already delivered or some copies have already spread in the network. Therefore this action reduces buffer occupancy and increases the delivery ratio. As a result the node receives a positive reward. Observe also that the actions $A1$, $A2$, $A3$ and $A6$ have the lower *Q-values*. This happens because actions $A1$ and $A2$ do not have immediate impact on buffer occupancy. Since $A1$ reduces the message generation frequency that depends on the event generation time and $A2$ broadcasts a congestion notification to neighbors which may not be able to listen. This situation leads to

delayed reward which we do not consider in this work. In addtion, actions $A3$ and $A6$ consider message expiration time to discard message. As we set the message's TTL to $30000s$ given a simulation time of $43200s$ the chances of having messages close to expire is small. Thus, the action of discarding expired messages is not able to discard any messages consequently the node receives a negative reward and reduces the action's *Q-value*.



(a) RW - WoLF (b) RWP - WoLF (c) SPBM - WoLF

(d) RW - BOLTZMANN (e) RWP - BOLTZMANN (f) SPBM - BOLTZMANN

FIGURE 6.7 – Average *Q-value* for each action for different mobility models (Terrestrial Scenario, Simulation Time $12h$, Buffer Size $4000kB$, Buffer Threshold 60%, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, α EWMA 0.80, γ Q-learning 0.80, Message Generation Period $300s$, Random Way Point Mobility Model, Epidemic Routing Protocol, WoLF Action Selection Method).

Note that in Figure 6.7, the Boltzmann action selection method performs more exploration than WoLF for different mobility models. When using WoLF there are more actions with *Q-values* close to zero (e.g.,see Figure 6.7b) than Boltzmann (e.g., see Figure 6.7e). This result shows that WoLF makes less exploration than Boltzmann in the terrestrial scenario. This happens because in WoLF the node gradually redistributes the probability of chosing a profitable action among all actions but still mantain the preference for the one with high probability. In WoLF method, each node keeps an updated probability distribution on the actions and stochastically picks one based on these probabilities. Thus the WoLF method is more conservative when the node learns which actions have higher *Q-value* since it continues to explore even reducing *Q-values* of some actions. In particular, WoLF uses the received reward to calculate the action's probability. In this case if the node receives a positive reward the action's probability increases by γ_{max}. On the other hand, the behavior of Boltzmann distribution is regulated by the temperature T. If T goes to zero the probability of selecting the strictly highest *Q-valued* action goes to 1(greedy distribution), while the selection probability of other's goes to 0. Note that pure greedy selection does not happens in our case since we use constant temperature equal

1. Because the nodes have zero initial estimatives and also the same probability of being picked in a specific state. So the greedy selection is desirable only at a later stage. As we can see by the results Boltzmann method makes more exploration than WoLF. It is important to point out that the balance between exploration and exploitation is, however, a difficult task and need to be addressed in more details.

6.3.4.3.3 Q-learning Evaluation for DTN-learning Performance

In the context of Q-learning, the value of a state is defined to be the maximum *Q-value* in the given state. Given this definition it is easy to derive Equation 6.18 which is presented again for the purpose of discussion. This equation includes a discount factor γ which is a number in the range of $[0 \ldots 1]$ and is used to weigh near term reinforcement more heavily than distant future reinforcement. The closer γ is to 1 the greater the weight of future reinforcements. In this section we vary the discount factor to analyze its impact on DTN-learning's performance.

$$Q_n^*(s,a) = (1-\eta_n)Q_{n-1}^*(s,a) + \eta_n(r + \gamma V_{n-1}^*(s^{'})) \tag{6.18}$$

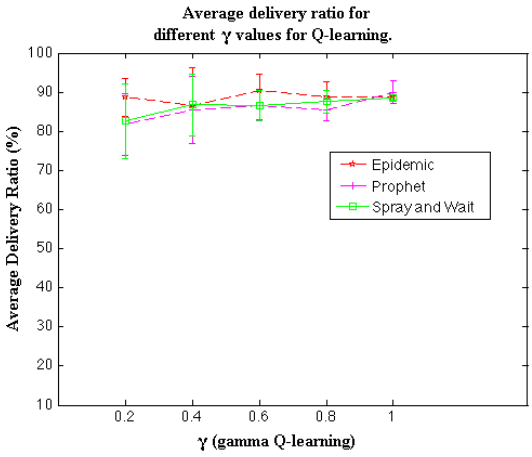
$$where$$

$$\eta_n = \frac{1}{1 + visits_n(s,a)} \tag{6.19}$$

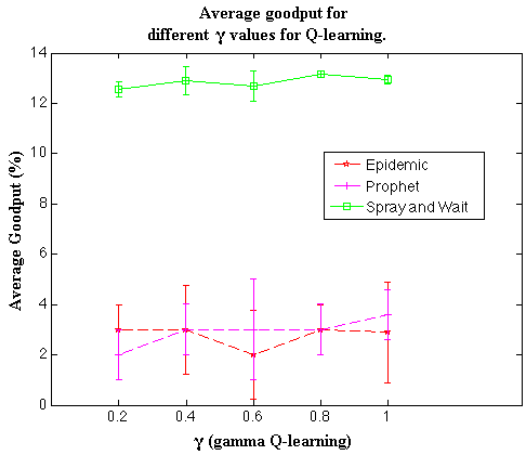
$$and$$

$$V_{n-1}^*(s^{'}) = max_{a'}[Q_{n-1}^*(s^{'},a^{'})] \tag{6.20}$$

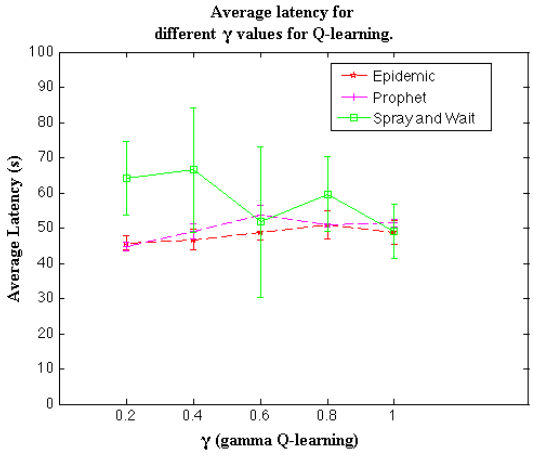
Figure 6.8 shows the average delivery ratio (Figure 6.8a), the average goodput (Figure 6.8b) and the average latency (Figure 6.8c) for the terrestrial scenario as a function of the discount factor γ in the *Q-value* function for different routing protocols (Epidemic, Prophet, Spray and Wait). The discount factor makes rewards earned earlier more valuable than those received later. Note that when increasing γ value the three performance metrics fluctuate independent of the routing protocol. Thus γ does not directly impact these metrics. By definition the γ value is used to weight the value of a node's state which is based on the reward received. Low γ means pay little attention to the future. High γ means that potential future rewards have a major influence on decision now. Thus we may be willling to trade short term loss for long-term gain. But we need to be aware that DTN environments are completely dynamic and consider the future cannot be a good strategy. Note that for $\gamma < 1$, the value of the future rewards always eventually becomes negligible. In terms of latency, note that as expected Spray and Wait perform worse compared to the other protocols. Since it is quota-based protocol, it limits the number of message copies in the network, minimizing network resource usage. Therefore it contributes to reduce the network's congestion level. However the message can take longer to arrive in its destination when we compare with either Epidemic or Prophet.



(a) Delivery Ratio



(b) Goodput



(c) Latency

FIGURE 6.8 – Average delivery ratio, goodput and latency as a function of γ for different routing protocols (Terrestrial Scenario, Simulation Time $12h$, Buffer Size $4000kB$, Buffer Threshold 60%, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, α EWMA 0.80, Message Generation Period $300s$, Random Way Point Mobility Model, WoLF Action Selection Method).

6.3.4.3.4 EWMA Approach Evaluation for DTN-learning Performance

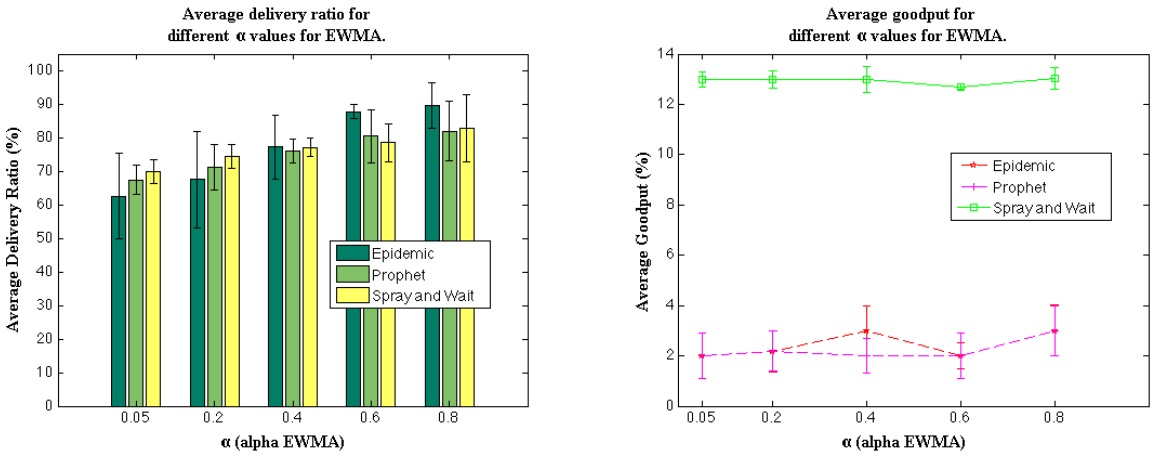
As we mentioned the buffer occupancy monitoring and the contact prediction is performed by EWMA. DTN-learning monitors the buffer's occupancy to predict when the buffer's occupancy is either growing or decreasing. This buffer's occupancy prediction allows the node autonomous monitoring of its buffer and depending on the prediction value it makes a transition to DCongested or PCongested state. To track a node's encounter DTN-learning allows each node to maintain local information about the last contact. Using EWMA[3] it predicts the future encounter time. The EWMA requires a decay factor α. The larger α, the more the average is biased toward recent history. The alpha must be between 0 and 1.

To explore the effect of different levels of α on the amount of message delivered we perform

[3]The EWMA function is $z_t \leftarrow \alpha x_t + (1-\alpha)z_{t-1}$, see Appendix A for more details.

simulations on each of the three routing protocols in both scenarios. To illustrate how each of the protocols react to changes in α, we analyze two performance metrics: delivery ratio and goodput.

Figure 6.9 shows the average delivery ratio and the average goodput for the terrestrial scenario. Delivery ratio and goodput are depicted as a function of the decay factor α for different routing protocols. In terms of delivery ratio, as α increases, the delivery ratio slightly grows (see Figure 6.9a). This result depicts the impact of the weight of current buffer occupancy in the message delivery ratio. As α grows, the more the average is biased toward recent buffer occupancy. Note that for different routing protocols the delivery ratio tends to be close when α varies strongly favoring DTN-learning performance in terms of interoperability. Additionaly, the goodput is constant for different α values. This happens because the tradeoff between the number of message received and messages transferred is small. As expected, the goodput is significantly greater when the number of copies is small, as shown in Figure 6.9b for Spray and Wait routing protocol.



(a) Delivery Ratio (b) Goodput

FIGURE 6.9 – Average delivery ratio and goodput as a function of α of EWMA function for different routing protocols (Terrestrial Scenario, Simulation Time $12h$, Buffer Size $4000kB$, Buffer Threshold 60%, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, γ Q-learning 0.2, Message Generation Period $300s$, Random Way Point Mobility Model, WoLF Action Selection Method).

6.3.4.3.5 Buffer Threshold Evaluation

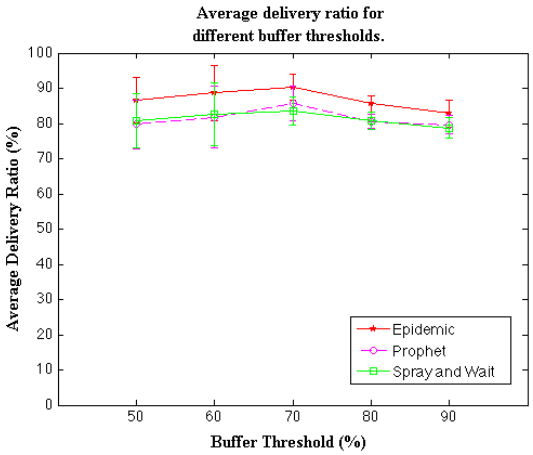
Since DTN-learning operates over the buffer occupancy to determine the node's current state as presented in Section 6.3.1, we evaluate it against itselft using different buffer thresholds. We should remark that in order for the node to be able to identify the NonCongested state, a buffer threshold was defined. Once the node knows that its buffer's occupancy ratio is below this threshold it also knows that the best state was reached. The basic idea is to use the "buffer threshold" to determine the urgency of each node, being aware about the necessity of making

congestion control accordingly. In other words, this number decides how many buffers to utilize from the common pool before the node starts to prevent congestion. To be less conservative we varied the buffer threshold values to evaluate its impact on message delivery ratio.

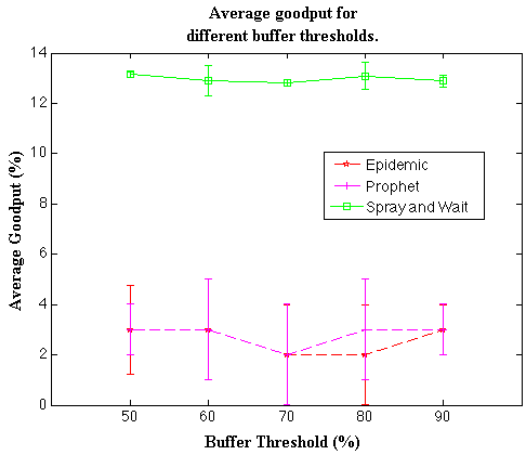
First, we reveal the influence of the buffer threshold in the average delivery ratio on the network performance. Figure 6.10 shows the average delivery ratio (Figure 6.10a), the average goodput (Figure 6.10b) and the average latency (Figure 6.10c) for the terrestrial scenario, varying the buffer threshold, for different routing protocols. In terms of average delivery ratio (see Figure 6.10a) when increasing the buffer threshold DTN-learning starts to produce higher delivery ratio. However, lets analyze this behavior. Starting at the beginning, the buffer fills linearly because the node generates messages by itself and also recieves message from its encounters. At the time t the node's buffer reaches the buffer threshold. At this point the DTN-learning starts to have more awareness with regard to the possibility of buffer congestion and acts proactively to make congestion control. If the node is still learning how to better control congestion, the delivery ratio can decrease. On the other hand, the node can mitigate congestion, because it already learned and it can act increasing the delivery ratio. If the buffer threshold's value is increased the node can accumulate more messages and still remain in the NonCongested state. In this case if the node's buffer becomes full the probability of having negative feedback because of the congestion is greater. Therefore, the figure shows that the delivery ratio can fluctuate as the buffer threshold grows larger. However, this fluctuation has a tendency to produce a lower delivery ratio. Obviously with a larger buffer threshold, the node can buffer more messages in the NonCongested state and probably a poor viewing experience in the future. Determining the correct buffer threshold depth requires to weigh resource utility and high delivery ratio. When goodput and latency are considered the behavior is the same. The average goodput (see Figure 6.10b) is greater for Spray and Wait because it limits the number of messages copies in the network. The average latency (see Figure 6.10c) is higher for Spray and Wait and Prophet since they do not send messages to all nodes as performed by Epidemic.

From the Figure 6.11, it can be stated that as buffer size of node increases, delivery probability increases and overhead ratio decreases for all routing protocols. Buffer provides space for messages while movement and messages will not be dropped due to buffer overflow as buffer size increases.

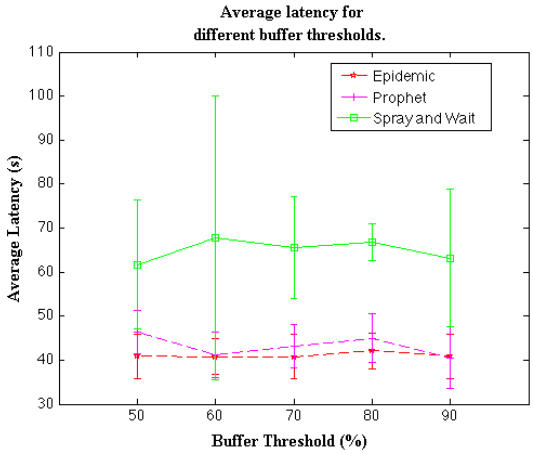
Specifically, Figure 6.11 shows how DTN-learning reacts to different buffer sizes in terms of average message delivery ratio, goodput, latency and overhead considering three kinds of routing protocols in terrestrial scenario. As shown in Figure 6.11a, when the buffer size increases, the average delivery ratio increases. This is because larger buffer space increases the chance for the node to find a next hop to forward the message. On the other hand, with small buffer size, new messages may replace the old undelivered messages, resulting in packets drops and low delivery ratio. It can be seen that varying buffer size Spray and Wait routing protocol performs better (high delivery ratio and goodput) as compared to both Epidemic and Prophet.



(a) Delivery Ratio



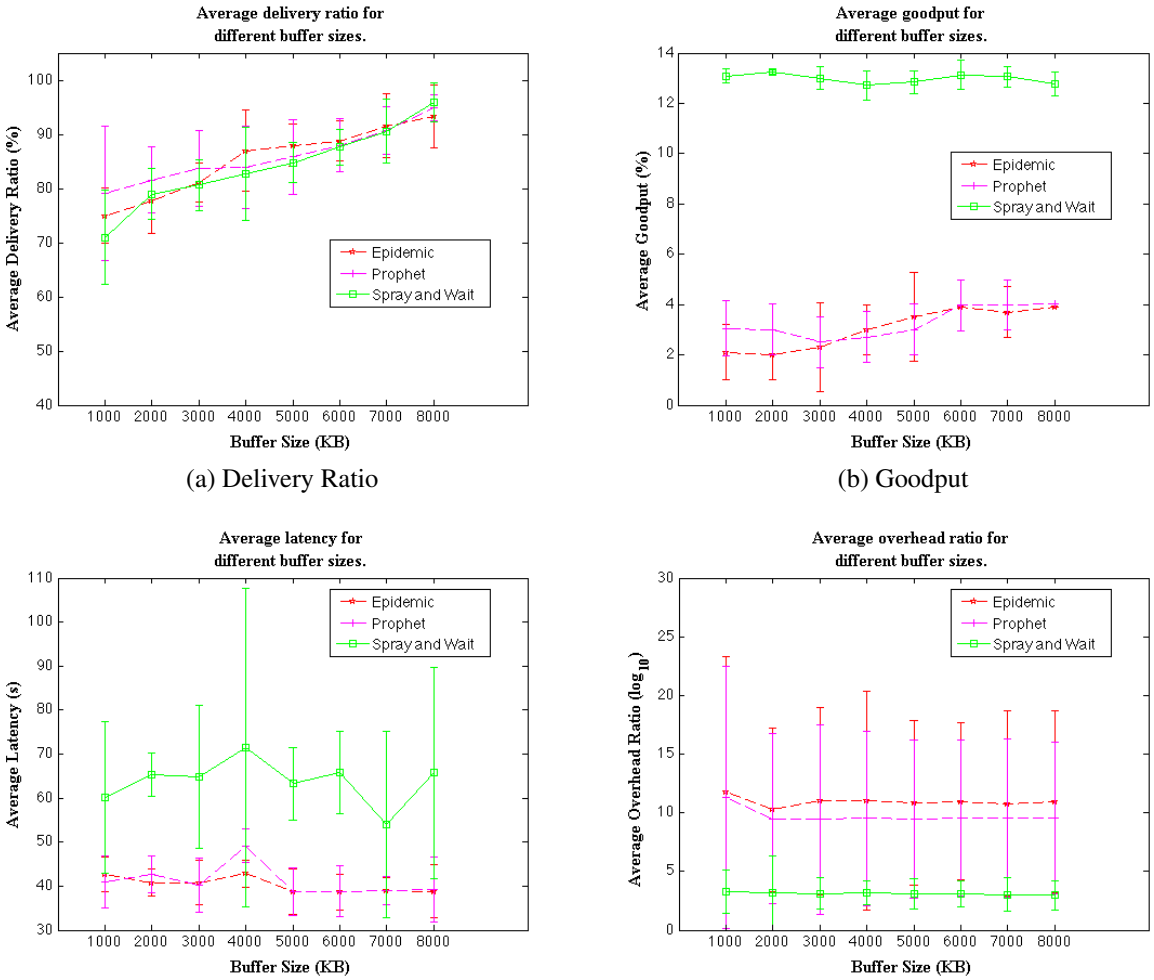
(b) Goodput



(c) Latency

FIGURE 6.10 – Average delivery ratio, goodput and latency as a function of the buffer threshold for different routing protocols (Terrestre Scenario, Simulation Time $12h$, Buffer Size $4000kB$, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, γ Q-learning 0.2, α EWMA 0.80, Random Way Point Mobility Model, Message Generation Period $300s$, WoLF Action Selection Method).

Since it limits the number of message copies and it does not experience a lot of congestion. The delivery ratio of Epidemic steadily increased because unlimited replication benefited from the increasing buffer size. As it is illustrated in Figure 6.11d, the lowest transmission cost for delivering a message in terms of network overhead is achieved by Spray and Wait. This is because there is a bound on the total number of relay copies of a message. The overhead for Epidemic and Prophet decreases with increase in buffer size up to some extent and then it decreases because more messages can be stored in the buffer and DTN-learning does not select action that discards messages. Therefore, more messages are relayed through the network. Prophet has lower overhead than Epidemic because Prophet sends messages only to reliable nodes, while Epidemic sends messages to all possible nodes. As expectecd the latency (see Figure 6.11c) has a tendency to be constant when the buffer size increases.



(a) Delivery Ratio (b) Goodput

(c) Latency (d) Overhead

FIGURE 6.11 – Average delivery ratio, goodput and latency for different buffer sizes and different routing protocols (Terra Scenario, Simulation Time $12h$, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, γ Q-learning 0.2, α EWMA 0.80, Random Way Point Mobility Model, Message Generation Period $300s$, Buffer Threshold 60%, WoLF Action Selection Method).

6.3.4.3.6 Simulation Results for IPN Scenario

In this section, we present simulation results from ours IPN scenarios. Under conditions of high propagation delay and very high inter-contact times DTN-learning presents good performance.

6.3.4.3.7 DTN-Learning Algorithm Evaluation

Figure 6.12 shows the cumulative reward for different routing protocols in the IPN scenario for different action selection methods. From these figures we get a general picture which is consistent with the performance of reinforcement learning algorithm mentioned above. However, we clearly note that the learning process takes longer in IPN scenario (see Figures 6.12b

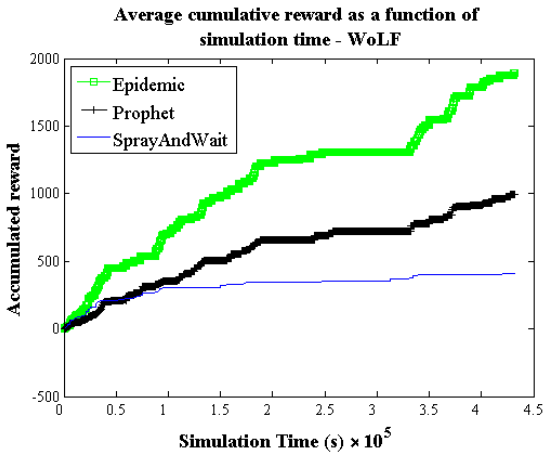
and 6.12d) than terrestrial. As can be seen DTN-learning has a degradation between $0s$ and $400s$ when the accumulated reward reaches a negative value. We believe that this is due to the sensitivity to the noise present in the IPN environment. Compared to the terrestrial scenario, this happens because the contacts in terrestrial scenario are more opportunistic. Therefore, nodes can exchange more messages and makes transitions to Congested, PCongested or DCongested frequent. As a result nodes in terrestrial scenarios are able to experience more trial-and-error situations decreasing the learning time. On the other hand, in the IPN scenario the nodes are subject to scheduled contacts that take longer to arise, consequently reducing the nodes chance of making frequent transition to Congested, PCongested or DCongested states. Observe that for different routing protocols the cumulative reward has great fluctuations. As mentioned in previous chapters, these routing protocols (Epidemic, Prophet, Spray and Wait) were not designed to operate in IPN environment which makes them unreliable.

Alhough we have used different action selection methods in IPN scenarios. We can note that the cumulative reward has a tendency to behavior similar for different action selection methods. Figures 6.12a and 6.12c show that the cumulative reward tends to be constant when using Prophet or Spray and Wait routing protocols. This happens because both routing protocols have a more restrictive forwarding strategy that results in lower congestion level. Consequently, the number of trial-and-errors is reduced and the cumulative reward tends to being constant.

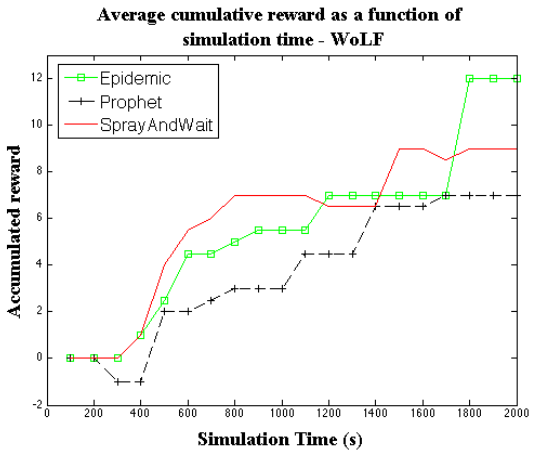
It is important to highlight two statistics of the result presented in this section. First, the minimum of the curve shows how much reward must be sacrificed before the nodes start to improve. Second, the zero crossing shows how long the nodes take until the algorithm has recovered their cost of learning. Note that the cumulative reward refers to the total reward measured for each node, yet DTN-learning optimizes discounted reward at each step.

We should point out that, in our framework, we are not only interested in nodes that reach a convergence in their behaviors, but also to make them adaptable to environmental changes, even if this implies that sometimes they will choose suboptimal actions. Furthermore, we are interested in how the decisions of the nodes at each simulation step affect both their local buffer occupancy as well as the aggregate performance of the entire network.

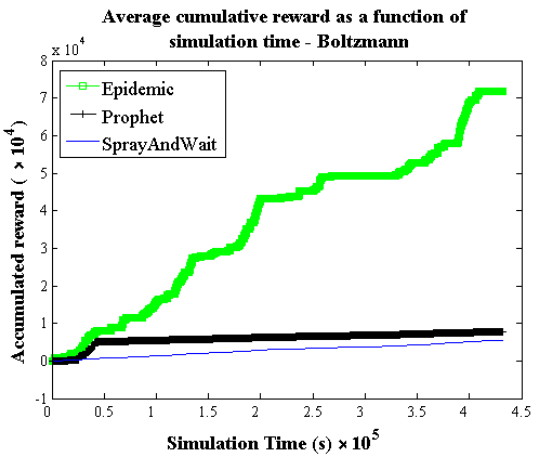
Figure 6.13 shows the average *Q-value* for different actions and routing protocols (Epidemic, Prophet and Spray and Wait) for the IPN scenario. Note that in the IPN scenario, the WoLF selection method makes more exploration when moving among different routing protocols. As we earlier discussed, Boltzmann distribution action spends great part of the time on exploration and thus the node starts to use its knowledge only at the end of the period. In the case of IPN scenarios this is critical since sometimes the nodes could not have enough time to learn due to the characteristics of this kind of DTN environment, for example long inter-contact times. This can be explained by the fact that both methods are based on exploration. We should point out here that the node's decision making policy can be modified by varying the value of the control parameter in the determined temperature range marked by temperature bounds in the



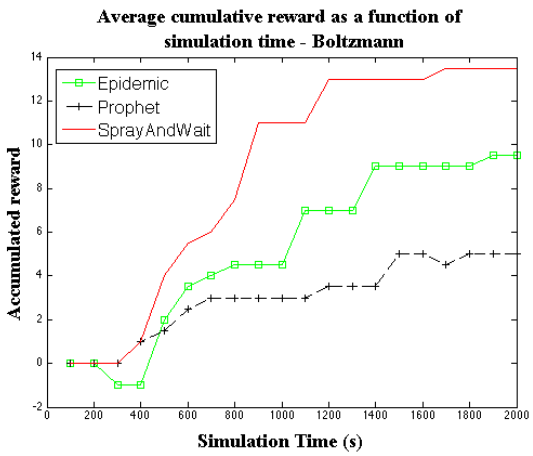
(a) WoLF



(b) WoLF - Snapshot with Simulation Time From $0s$ To $2000s$



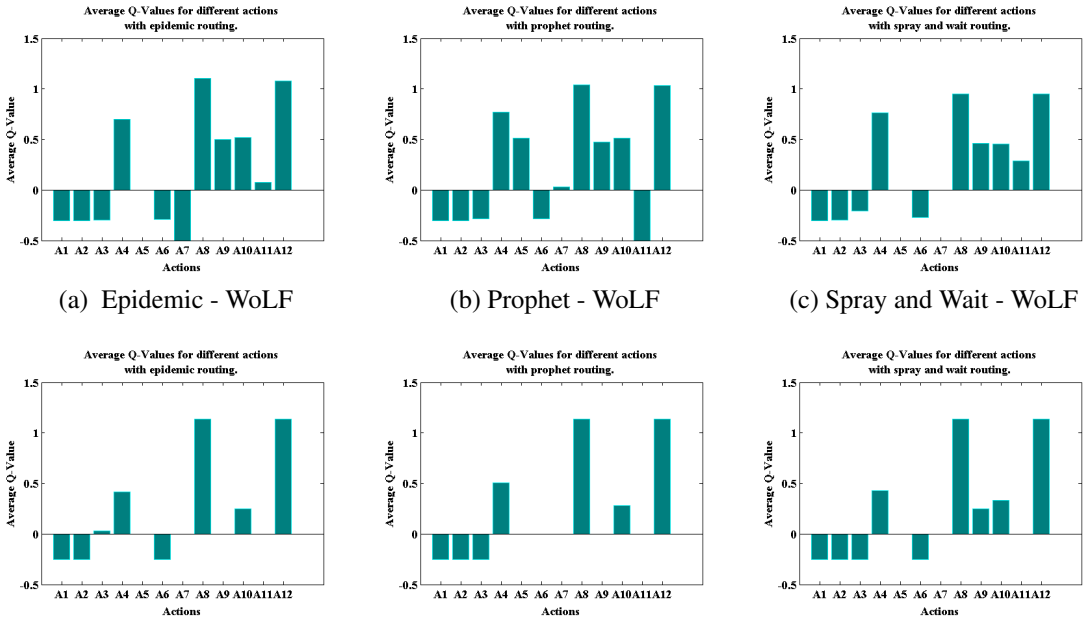
(c) Boltzmann



(d) Boltzmann - Snapshot with Simulation Time From $0s$ To $2000s$

FIGURE 6.12 – Average cumulative reward as a function of the simulation time (IPN Scenario, Simulation Time $12h$, Buffer Size $4000kB$, Buffer Threshold 60%, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, α EWMA 0.80, γ Q-learning 0.2, Message Generation Period $300s$, Epidemic Routing, Inter-Contact Time $1000s$ and Contact Duration $1000s$).

case of Boltzmann action selection method. Note that action $A4$ presents high *Q-value* in both scenarios. As said before this action discards messages that are expected being either delivered or forwarded. In addition action $A11$ has a good performance in IPN scenario because it dicards oldest message to receive a new arriving message. This action benefits from IPN scenarios because easier to discard old message than expired messages. Usually, message stay longer in IPN scenario and they may not expire soon. Therefore, discarding the oldest messages could be a better alternative. Actions $A9$ and $A10$ also present good performance in IPN scenario since they perform a more local control. As IPN scenario is subject to high inter-contact time these two actions allow the node to react proactively to make congestion control keeping the network utility level.



(a) Epidemic - WoLF (b) Prophet - WoLF (c) Spray and Wait - WoLF

(d) Epidemic - BOLTZMANN (e) Prophet - BOLTZMANN (f) Spray and Wait - BOLTZMANN

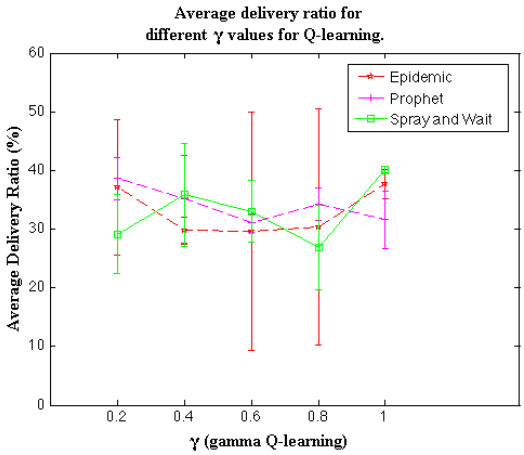
FIGURE 6.13 – Average *Q-value* for each action for different mobility models and routing protocols (IPN Scenario, Simulation Time $12h$, Buffer Size $4000kB$, Buffer Threshold 60%, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, α EWMA 0.80, γ Q-learning 0.2, Message Generation Period $300s$, Inter-contact time of $1000s$ and Contact Duration of $1000s$, WoLF Action Selection Method).

6.3.4.3.8 Q-learning Evaluation for DTN-learning Performance

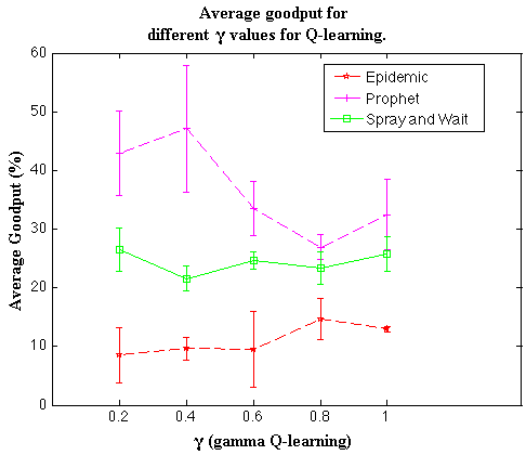
In this section we vary the discount factor to analyze its impact on DTN-learning's performance under the IPN scenario. Figure 6.14 shows the average delivery ratio (see Figure 6.14a, the average goodput (see Figure 6.14b) and the average latency (see Figure 6.14c) as a function of the discount factor γ in the *Q-value* function for different routing protocols. Observe that in the IPN scenario, similar to the terrestrial scenario, the γ values variation does not show significant impact on the DTN-learning performance. The average delivery ratio presents some fluctuations (see Figure 6.14a) but it is not directly affected for different γ values. In Figure 6.14b, we can note that the average goodput is better for Prophet since it benefits from the schedule contacts. Therefore sending messages only for nodes that have high probability of delivering the message. As a result, the number of message copies is small increasing the goodput value, as expected. In terms of average latency (see Figure 6.14c), DTN-learning performs much worse than in the terrestrial scenario. However high latencies are expected in IPN environments. Thus DTN-learning still presents good performance.

6.3.4.3.9 EWMA Approach Evaluation for DTN-learning Performance

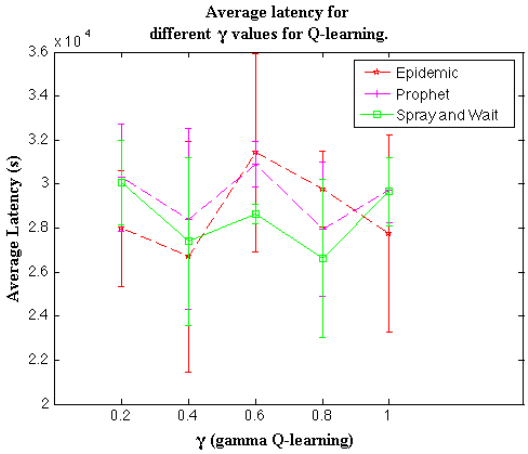
To evaluate the impact of different levels of α on the amount of message delivered we perform simulations on each of the three routing protocols in IPN scenarios. Similarly to the



(a) Delivery Ratio



(b) Goodput



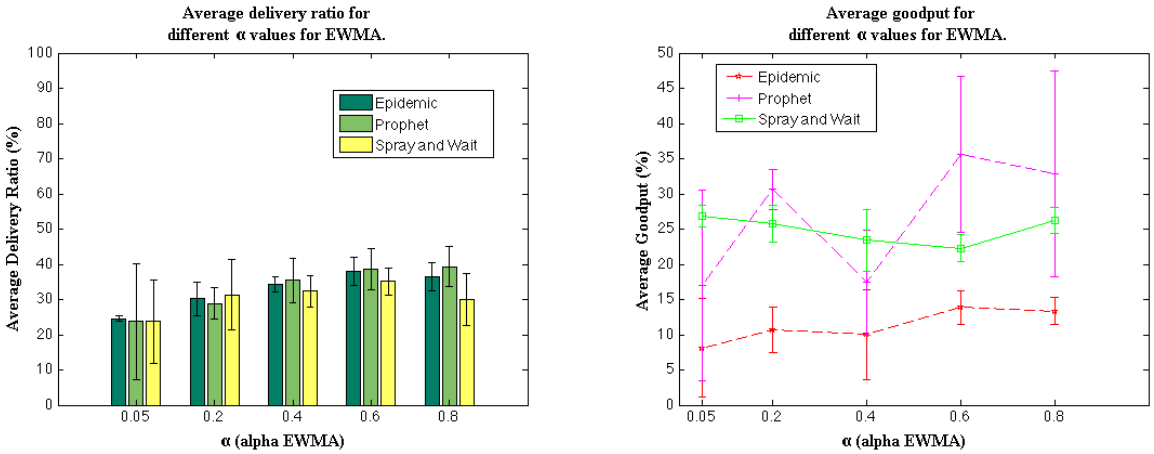
(c) Latency

FIGURE 6.14 – Average delivery ratio, goodput and latency as a function of γ for different routing protocols (IPN Scenario, Simulation Time $12h$, Buffer Size $4000kB$, Buffer Threshold 60%, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, α EWMA 0.80, Message Generation Period $300s$, Inter-Contact Time $1000s$, Contact Duration $1000s$, WoLF Action Selection Method).

terrestrial scenario, as α increases, more weight is considered in the current buffer occupancy ratio. Therefore, higher delivery ratio are achieved with higher α as Figure 6.15a shows. The average goodput is greater when Prophet or Spray and Wait are considered, as expected.

6.3.4.3.10 Buffer Threshold Evaluation

In this section we evaluate the DTN-learning in the IPN scenario against different buffer threshold values. Figure 6.16 shows the average delivery ratio (Figure 6.16a), the average goodput (Figure 6.16b) and the average latency (Figure 6.16c) varying the buffer threshold for different routing protocols. As discussed, the proactive congestion control imposed by DTN-learning points out the tradeoff between resource usage and delay, delivery ratio or goodput. The IPN results show similar behavior reported on the terrestrial scenario. Although it is not



(a) Delivery Ratio (b) Goodput

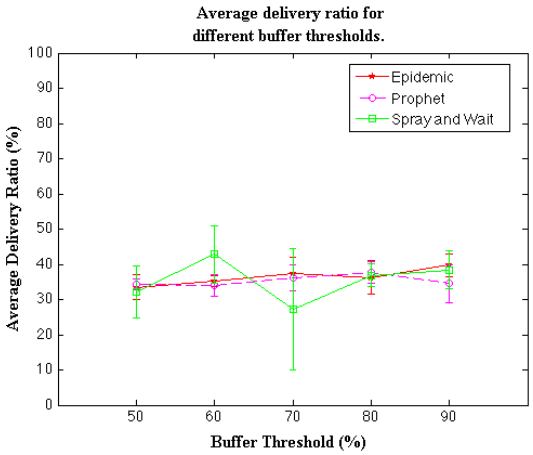
FIGURE 6.15 – Average delivery ratio and goodput as a function of α of EWMA function for different routing protocols (IPN Scenario, Simulation Time $12h$, Buffer Size $4000kB$, Buffer Threshold 60%, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, γ Q-learning 0.2, Inter-Contact Time of $1000s$ and Contact Duration of $1000s$, Message Generation Period $300s$, WOLF Action Selection Method).

easy to select the optimal threshold values, we can see that whe buffer threshold value varies the delivery ratio fluctuates. We can find that DTN-learning shows best performance when *buffer threshold* is 60% for all routing protocols (see Figures 6.16a), which approximately corresponds to 60% of the buffer size. For comparison purposes, in Figure 6.10a which refer to terrestrial scenario, as buffer threshold increases, DTN-learning presents good performance around 70%. Thus, we believe that a buffer threshold of 60% is reliable, because the node have the opportunity of preventing from congestion. In addition, if a standard deviation of 10% is considered both scenario will be include which is sufficiently large to obtain a good control policy.

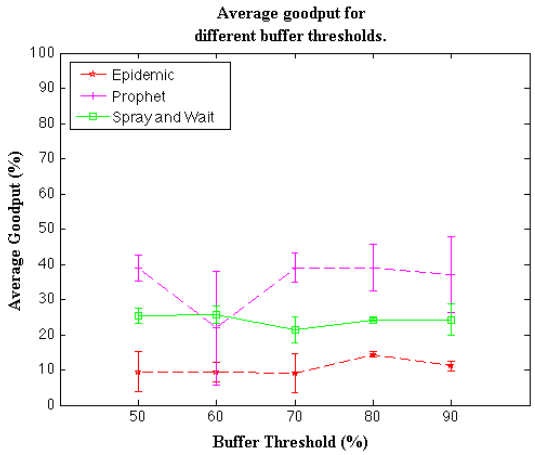
6.3.4.3.11 IPN Use Cases

Since IPN scenario is different of terrestrial scenarios mainly because it does not consider opportunistic contacts and it is subject to very high propagation delay, we perform some experiments where our framework provides as input for the next episode[4] the *Q-values* learned in the past. Following this strategy will allow a better understanding of how DTN-learning is working in the IPN scenario. Thus to demonstrate the effectiveness of DTN-learning and to improve the reinforcement process in IPN scenario, we design a use case where the framework keeps track of the *Q-value* table after each $12h$ of simulation. Then this table is provided as input for the next $12h$ of simulation. In this case, we are using supervised learning since we are feeding the environment with lessons previously learned. Each $12h$ of experiment, we called one "episode". A series of 9 episodes were simulated.

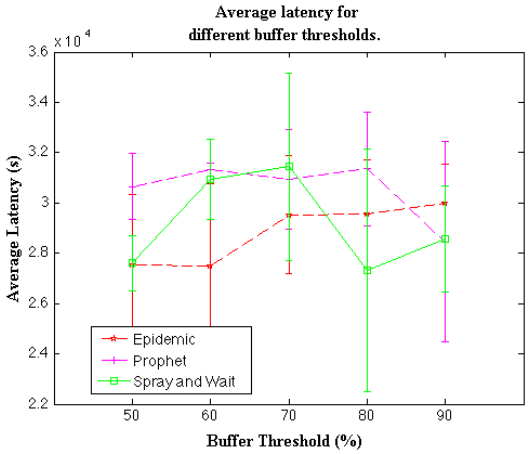
[4]An episode is a trajectory started in a simulation time equal to $0s$ and ending in $43200s$.



(a) Delivery Ratio



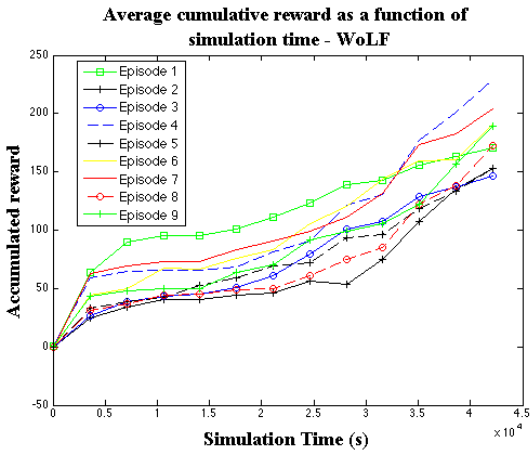
(b) Goodput



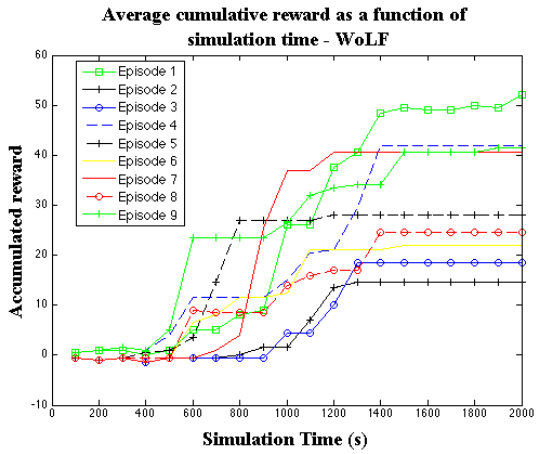
(c) Latency

FIGURE 6.16 – Average delivery ratio, goodput and latency as a function of the buffer threshold for different routing protocols (IPN Scenario, Simulation Time $12h$, Buffer Size $4000kB$, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, γ Q-learning 0.2, α EWMA 0.80, Inter-contact time of $1000s$ and Contact Duration of $1000s$, Message Generation Period $300s$, WoLF Action Selection Method).

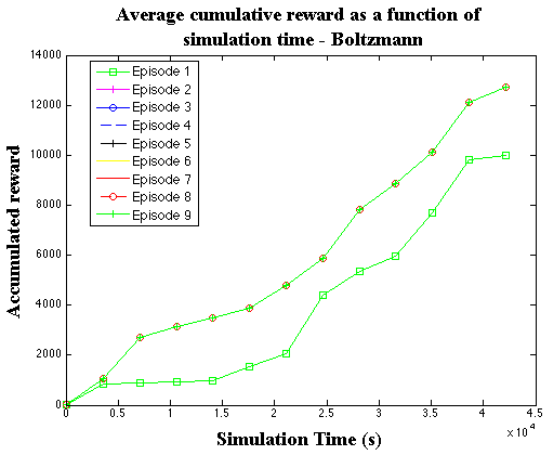
Figure 6.17 shows the accumulated reward as a function of simulation time for WoLF (see Figure 6.17a and 6.17b) and Boltzmann (see Figure 6.17c and 6.17d)) action selection methods. Figures 6.17b and 6.17d show a snapshot of the accumulated reward values between $0s$ and $2000s$. Note that in Figure 6.17a and 6.17b, the cumulative reward fluctuates for different episodes using WoLF selection method. This happens because WoLF considers that besides of the node learns the environment it is still doing exploration. As shown by the plots the cumulative reward values for episode 1 remain above the other episodes throughout the simulation time interval from $0s$ to $3.2 \times 10^4 s$. In comparison with Figures 6.17c and 6.17d, where Boltzmann action selection method is used, the cumulative reward values for episode 1 is underneath the other episodes. Note that the cumulative rewards are similar from episode 2 to episode 9. The results reveal that the combination of DTN-learning based on RL with supervised learning allows the node to perform well in IPN scenario.



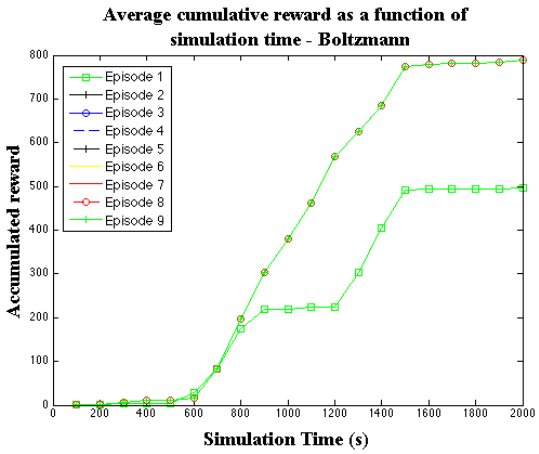
(a) WoLF



(b) WoLF - Snapshot with Simulation Time From $0s$ To $2000s$



(c) Boltzmann

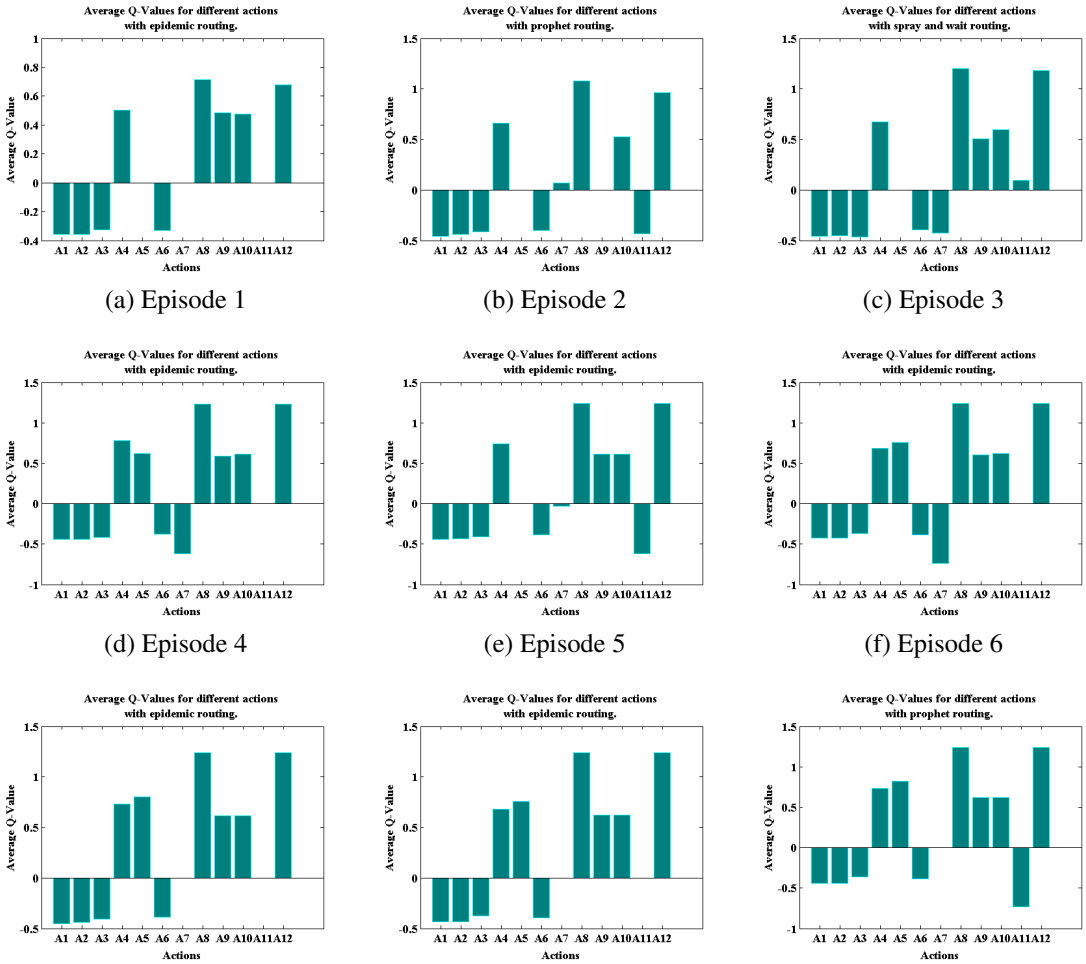


(d) Boltzmann - Snapshot with Simulation Time From $0s$ To $2000s$

FIGURE 6.17 – Average cumulative reward as a function of the simulation time (IPN Scenario, Simulation Time $12h$, Buffer Size $1000kB$, Buffer Threshold 60%, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, α EWMA 0.80, γ Q-learning 0.2, Message Generation Period $100s$, Message's TTL $30000s$, Epidemic Routing Protocol, Inter-Contact Time $1000s$ and Contact Duration $1000s$).

Figure 6.18 shows the average *Q-values* for different actions and episodes. As can be seen, when the number of episodes goes from 1 to 9 the number of different selected actions tends to increase. This shows that the learning processs is more advanced and the node tends to choose actions with great *Q-values* to mitigate congestion. Note that the nodes keep choosing actions $A4$, $A8$, $A10$ and $A11$. Gradually, actions $A5$ and $A9$ are included in the set of good actions (set of actions with positive *Q-values*). This feature confirms that DTN-learning propagates information learned in previous episodes to episodes ahead.

For comparison purpose, Boltzmann action selection method is applied to the same context of DTN-learning with supervised learning (see Figure 6.19). Observe that after episode 1, DT-learning only select actions into the set of profitable actions. This may be because the node



(a) Episode 1 (b) Episode 2 (c) Episode 3

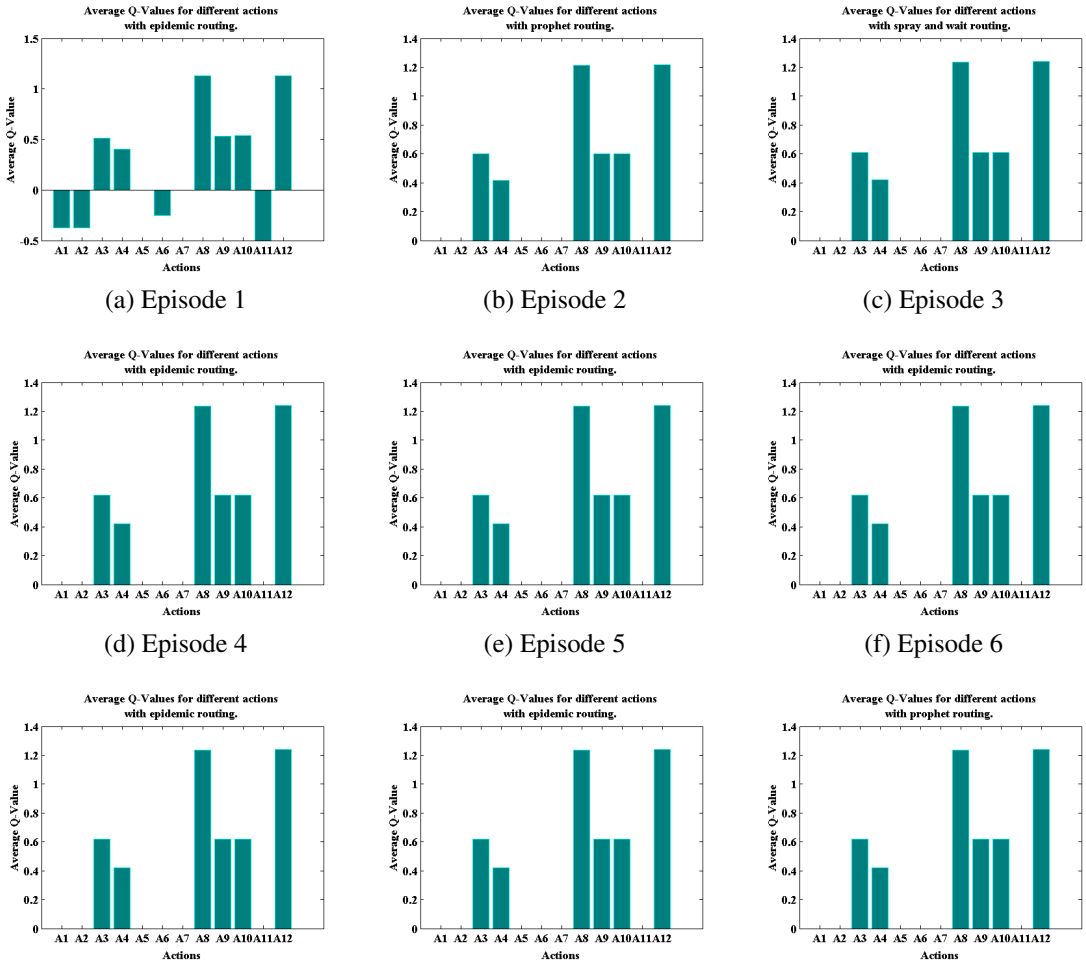
(d) Episode 4 (e) Episode 5 (f) Episode 6

(g) Episode 7 (h) Episode 8 (i) Episode 9

FIGURE 6.18 – Average *Q-values* for each action for different episodes (IPN Scenario, Simulation Time $12h$, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, γ Q-learning 0.2, α EWMA 0.80, Message Generation Period $100s$, Buffer Threshold 60%, Buffer Size $1000kB$, Message's TTL $30000s$, Epidemic Routing Protocol, WoLF Action Selection Method).

using Boltzmann method understands that it has done enough exploration and then it starts to choose among more profitable actions. We need to be aware that a tradeoff exists between exploration and exploitation. In this case, Boltzmann may not be a good option for some DTN environments.

Figure 6.20 shows the accumulated reward as a function of the simulation time for different action selection methods. To illustrate the DTN-learning's performance over IPN scenario where message's TTL is still longer but more messages are able to expire before the simulation end time, we design an use case to capture the impact of a small message's TTL on actions that depend on message's expiration. This use case considers Prothet routing protocol and a message's TTL of $10000s$. We set up Prophet routing protocol in this use case because we believe that it benefits from IPN scenarios and this can contribute for the learning process' performance. As shown in Figures 6.20a, 6.20b, 6.20c and 6.20d, DTN-learning reaches a good learning per-



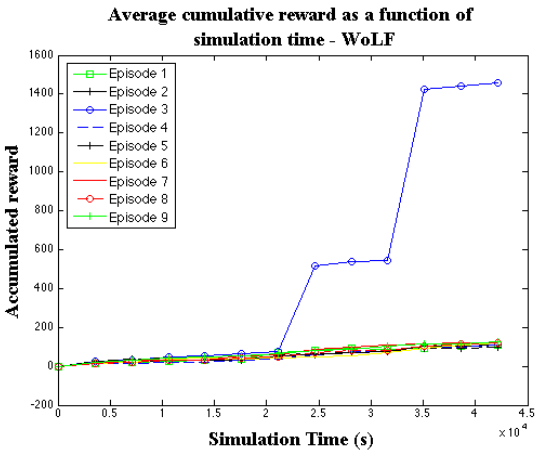
(a) Episode 1 (b) Episode 2 (c) Episode 3

(d) Episode 4 (e) Episode 5 (f) Episode 6

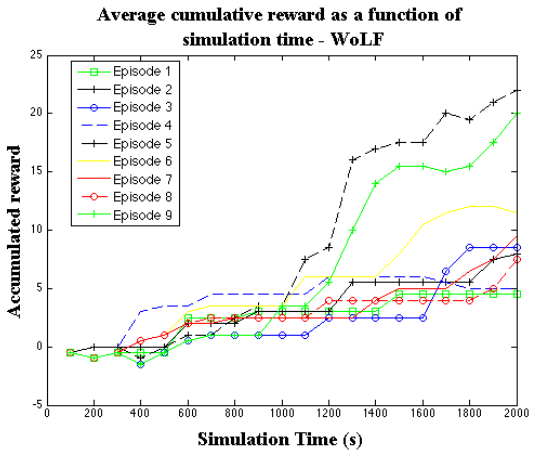
(g) Episode 7 (h) Episode 8 (i) Episode 9

FIGURE 6.19 – Average *Q-values* for each action for different episodes (IPN Scenario, Simulation Time $12h$, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, γ Q-learning 0.2, α EWMA 0.80, Message Generation Period $100s$, Buffer Threshold 60%, Buffer Size $1000kB$, Message's TTL $30000s$, Epidemic Routing Protocol, Boltzmann Action Selection Method).

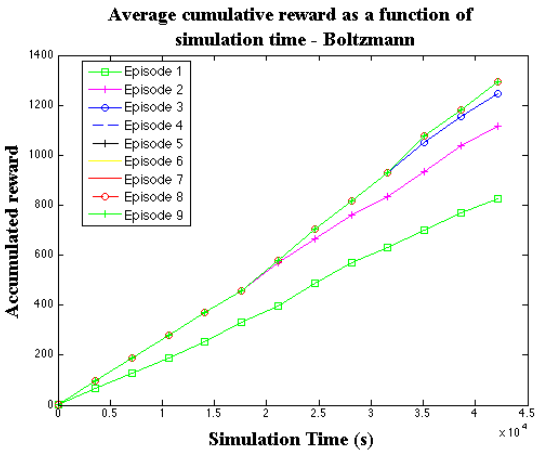
formance after spending some times with exploration. However, the best performance is obtained with Boltzmann where the plotting for episode 1 is underneath of the plotting for episode 9 (see Figures 6.20c and 6.20d). This behavior can not be observed in Figures 6.20a and 6.20b for WoLF. We believe this happens because the WoLF's principle has a simple intuition of learn quickly while losing and slowly while winning. As a result, the cumulative reward fluctuates and seems to converge faster since it tries to alternate among different selected actions. In other words, WoLF policy remains rational, since only the speed of learning is alterated. The results of applying WoLF policy at IPN scenario do not show good performance because the learning process in an environmnet with scheduled contact take longer. Thus this method does not benefit from the node contacts. Note that in Figure 6.20b WoLF quickly begins to oscilate around the equilibrium, with ever decreasing amplitude. On the other hand, Boltzmann tends being more linear without any hint of converging.



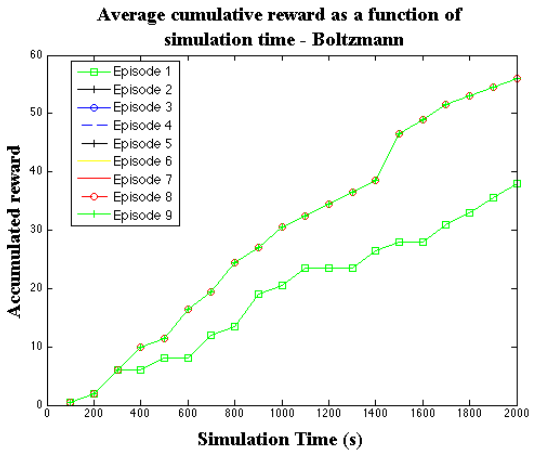
(a) WoLF



(b) WoLF - Snapshot with Simulation Time From $0s$ To $2000s$



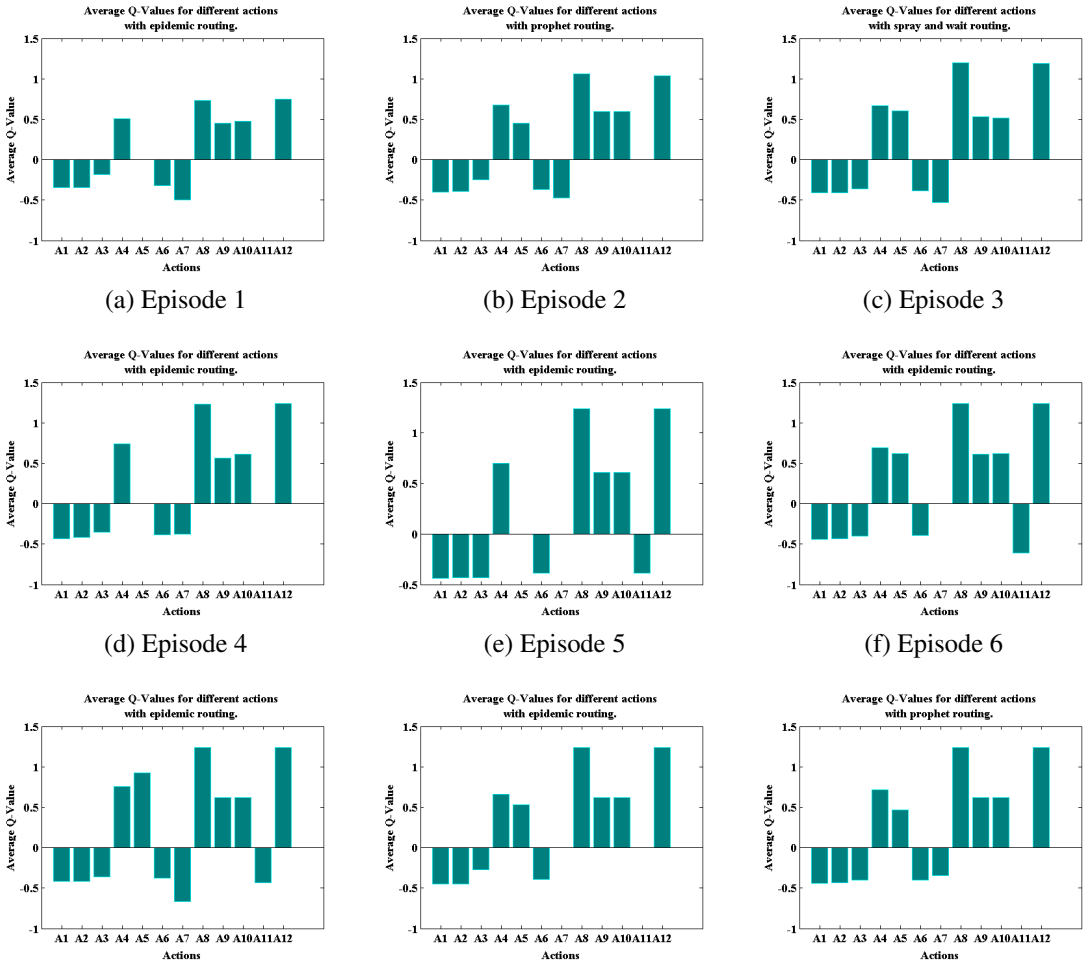
(c) Boltzmann



(d) Boltzmann - Snapshot with Simulation Time From $0s$ To $2000s$

FIGURE 6.20 – Average cumulative reward as a function of the simulation time (IPN Scenario, Simulation Time $12h$, Buffer Size $1000kB$, Buffer Threshold 60%, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, α EWMA 0.80, γ Q-learning 0.2, Message Generation Period $100s$, Message's TTL $10000s$, Prophet Routing Protocol, Inter-Contact Time $1000s$ and Contact Duration $1000s$).

Figure 6.21 shows the *Q-values* for different actions and episodes using WoLF action selection method. As can be seen, the actions $A3$ and $A6$ which use message's TTL expiration to mitigate congestion are selected in all episodes. However they do not present good performance since their *Q-values* are negative as depicted by the plottings for different episodes. Note that the exact effect of the WoLF principle can be seen by different *Q-values* for all actions when we move from episode 1 to episode 9 using either the larger (γ_{max}) or smaller (γ_{min}) learning rate. The combination of DTN-learning with RL and supervised learning can be clearly seen in Figures 6.21a, 6.21b, 6.21c, 6.22d, 6.21e, 6.21f, 6.21g, 6.21h and 6.21i. Observe that in Figure 6.21a episode 1, the set of good actions are $A4$, $A8$, $A9$, $A10$ and $A11$ when using supervised learning these actions are maintained for all next episodes. This happens because the supervised learning strategy is being used.



(a) Episode 1 (b) Episode 2 (c) Episode 3

(d) Episode 4 (e) Episode 5 (f) Episode 6

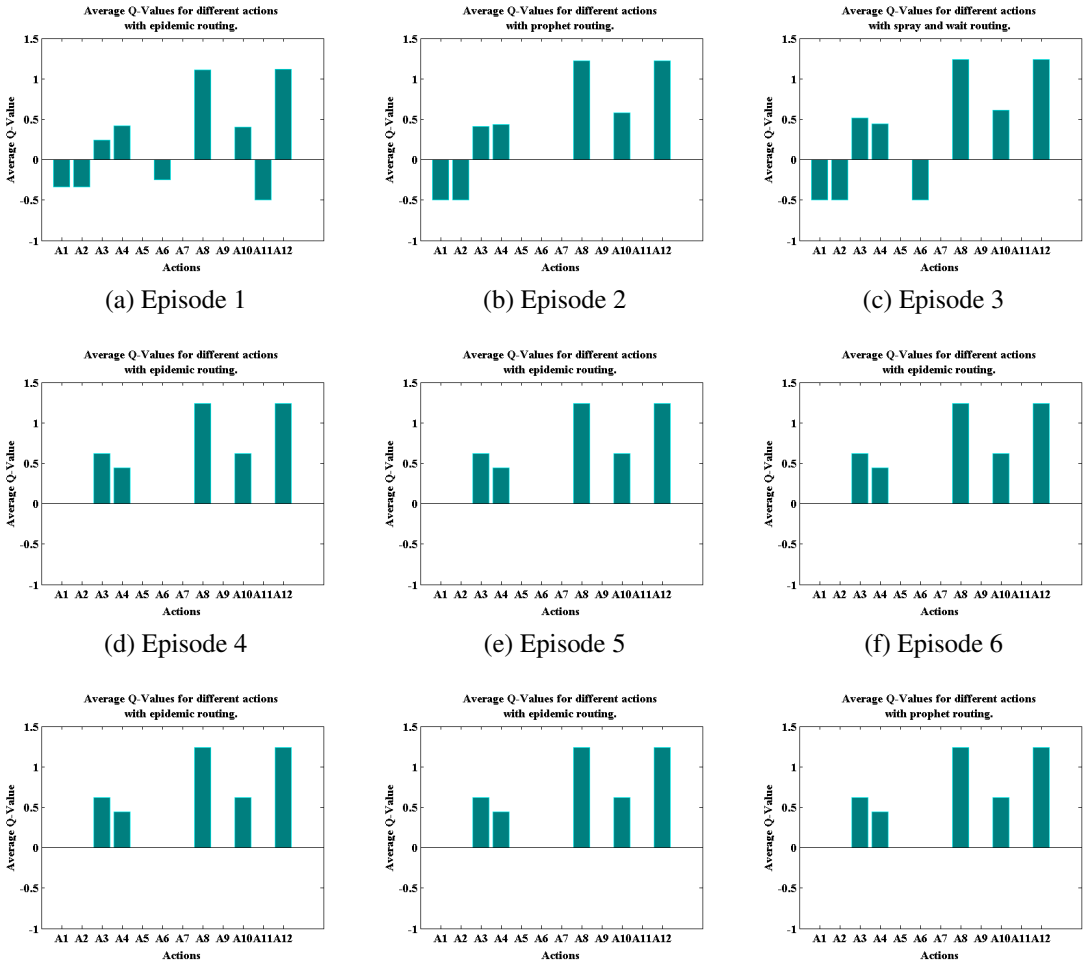
(g) Episode 7 (h) Episode 8 (i) Episode 9

FIGURE 6.21 – Average *Q-values* for each action for different episodes (IPN Scenario, Simulation Time $12h$, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, γ Q-learning 0.2, α EWMA 0.80, Message Generation Period $100s$, Buffer Threshold 60%, Buffer Size $1000kB$, Message's TTL $10000s$, Prophet Routing Protocol, WoLF Action Selection Method).

Figure 6.22 shows the *Q-values* for different actions and episodes using Boltzmann action selection method. Similar to Figure 6.21, the results shown in Figure 6.22 also select actions $A3$ and $A6$ that depend on message's TTL. However, once the Boltzmann strategy learns that $A6$ has been returning negative reward, consequently negative *Q-values* it stops to select this action after episode 3. On the other hand, it continues to select $A3$. Therefore, DTN-learning is learning by both RL and supervised learning when the use case prioritizes actions that depend on message's TTL.

6.3.5 Comparative Analysis of DTN-learning

The goal of our evaluation is to show that DTN-learning achieves a high message delivery ratio and good latency, while maintaining extremely low overhead. To demonstrate the



(a) Episode 1 (b) Episode 2 (c) Episode 3

(d) Episode 4 (e) Episode 5 (f) Episode 6

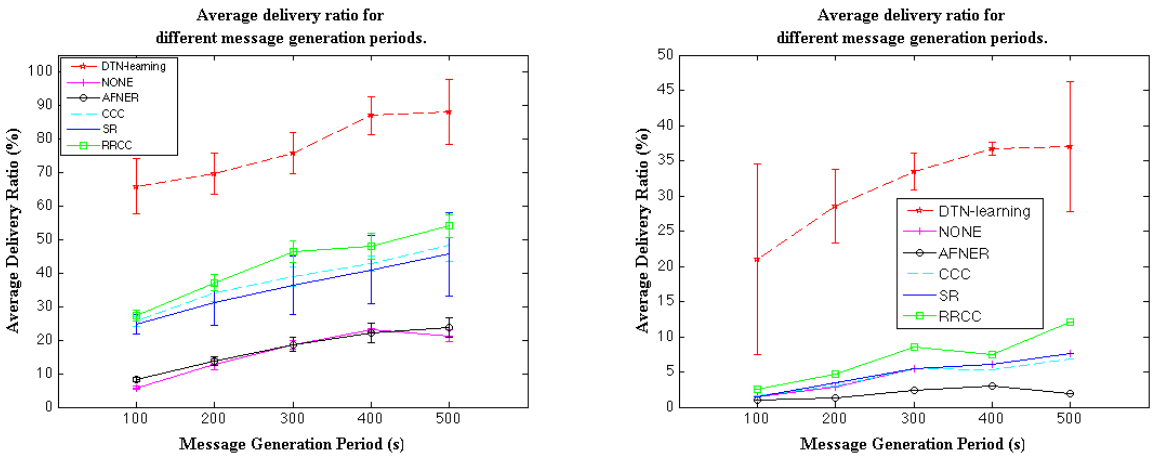
(g) Episode 7 (h) Episode 8 (i) Episode 9

FIGURE 6.22 – Average *Q-values* for each action for different episodes (IPN Scenario, Simulation Time $12h$, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, γ Q-learning 0.2, α EWMA 0.80, Message Generation Period $100s$, Buffer Threshold 60%, Buffer Size $1000kB$, Message's TTL $10000s$, Prophet Routing Protocol, Boltzmann Action Selection Method).

effectiveness of DTN-learning, we evaluate it against four other DTN congestion control mechanisms (AFNER (YUN *et al.*, 2010), CCC (LEELA-AMORNSIN; ESAKI, 2010), RRCC (THOMPSON *et al.*, 2010) and SR (SELIGMAN *et al.*, 2006)) which were described in Chapters 3 and 4. Some of the AFNER results are omitted from our evaluation due to the fact it has the worst performance when comparing it with the other mechanisms (CCC, RRCC and SR). More details about the results reported by AFNER experiments can be found in Chapter 4.

Figure 6.23 shows average delivery ratio for DTN-learning, CCC, RRCC and SR as a function of the message generation period in both scenarios: terrestrial and IPN. For the terrestrial scenario (see Figure 6.23a), we use a buffer threshold of 60%, buffer size of $1000KB$, Epidemic routing protocol and Random Way Point mobility model. For the IPN scenario (see Figure 6.23b), we use a buffer thresholf of 60%, buffer size of $4000KB$, Epidemic routing protocol, inter-contact time of $1000s$ and contact duration of $1000s$. The average delivery ra-

tio for DTN-learning is better than those of CCC, RRCC and SR due to less frequent buffer overflows. Although CCC, RRCC and SR also can avoid congestion, the performance of DTN-learning is better because CCC just tries to mitigate congestion by discarding messages, RRCC tries to do replication management and SR adopts the strategy of message migration. All these strategies are ineffective since messages that would be delivered can be discarted, replication management can fail due to the intermittent connectivity and the migration process which does not work when all nodes are congested. DTN-learning performs a congestion control based on experience. Thus the nodes are able to learn the best action to either prevent or react to congestion. Therefore, DTN-learning is congestion aware and it is able to easily adapt to any DTN environment. As a result DTN-learning reduces the drop ratio and increases the delivery ratio. Note that DTN-learning does not use any global information to perform congestion control as AFNER. Because of this AFNER performs worst than the other mechanisms.



(a) Terrestrial Scenario (b) IPN Scenario

FIGURE 6.23 – Average delivery ratio as a function of message generation period.

Figure 6.24 shows the average latencies for terrestrial scenario using different DTN congestion control mechanisms (AFNER, SR, RRCC, CCC and DTN-learning) under different mobility models (RW, RWP and SPMBM). Specifically the three mobility models were chosen to encompass a wide variety of terrestrial DTN environments. The number of messages relayed in SPMBM compared to the other two mobility models is usually higher. The number of messages delivered is also expected to be high in case of SPMBM; a factor that makes this result would be the selection of the next location of the node in the network area. SPMBM uses an algorithm that searches for the shortest path that a node should follow. This makes nodes visit most of the map locations within the given time period. In addition, the difference in the number of contacts in SPMBM is relatively higher compared to that of other mobility models. This is because when node movements are constrained to follow a certain path a greater probability of encounter occurs. Due to these factors the average message delivery latency is smaller in the SPMBM (see Figure 6.24c) when comparing to RW or RWP. Note that the average latency

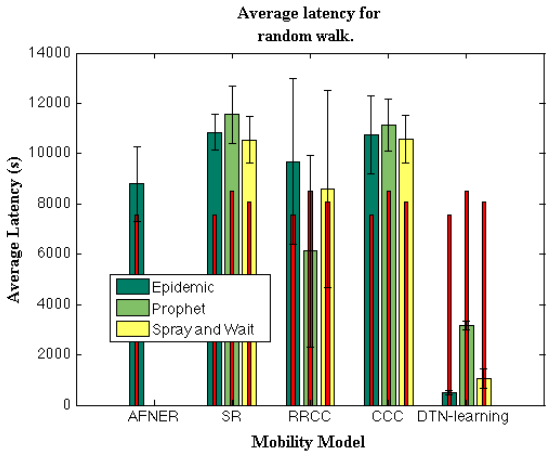
values fluctuate for different routing protocols. The difference is notable when moving from RW to either RWP or SPMBM. This happens because the mobility model regulates the number of contacts available for each routing protocol. This number of contacts can determine the message delivery delay. Thus for less uniform nature of the nodes, as in the case of RW, the latency has a tendency to increase. In terms of congestion control, as the network load increases the message delivery delay is easily impacted. In this point, we notice that DTN-learning performs at the highest level independent of either the mobility model and the routing protocol (see Figures 6.24a, 6.24b and 6.24c). Independent of the mobility model DTN-learning presents an average latency of $230s$ which is not depicted in the figures due to the scale adopted to show all latency values per mechanisms. We believe that the high performance of DTN-learning is due to the capability of reacting proactively to the congestion. Mainly, this happens when it monitors the buffer occupancy ratio allowing the node to know if its buffer occupancy ratio is growing or not. Thus the node can intelligently prevent network overload and mantain acceptable latency levels.

Figure 6.25 shows message delivery ratio for different inter-contact times[5] where the slim bars refer to delivery ratio values when no congestion control is used. As expected, for longer inter-contact times the delivery ratio decreases. DTN-learning clearly outperforms all the other mechanisms in terms of delivery ratio as we can see in Figure 6.25a. Basically, DTN-learning is able to maintain the average delivery ratio approximately 3 times greater than the other mechanisms when the inter-contact time grows from $1000s$ to $5000s$. This is significant since DTN-learning is more adaptable to changes in the environment than the other mechanisms.

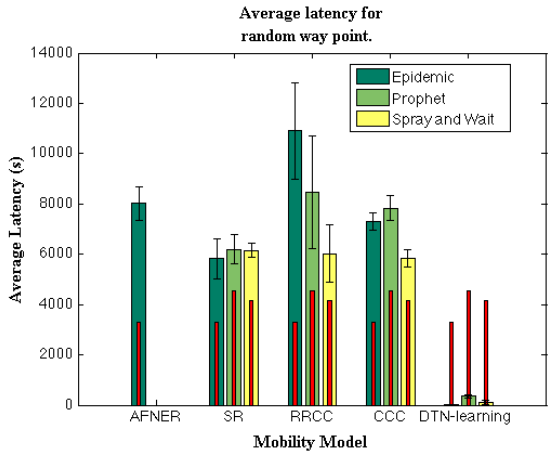
Figure 6.26 shows message delivery ratio for different contact durations [6]. We can note that for longer contact duration the nodes can forward more messages. Consequently, the chance of the messages being delivered increases. The CCC and SR gives slightly worse delivery ratio than the DTN-learning and RRCC because they adopt a reactive congestion control. Since the nodes have more time to tranfer messages the node's buffer becomes full faster and thus CCC and SR start to discard messages reducing the delivery ratio. The SR does not work when all neighbors are congested. In this case it randomly discards messages decreasing delivery ratio. On the other hand, DTN-learning and RRCC have the capability of proactively acting to prevent congestion. Thus they avoid a great number of messages being discarded. However, RRCC tries to prevent congestion by doing replication management. Nodes exchange information about message drops upon encountering other nodes in order to determine the message replication limit. Once the nodes have enough time to exchange messages, the message drop information may be outdated. As a result the replication limit can incorrectly be set up, for example, increasing the replication limit when it is expected to reduce the replication limit. As a result messages are dropped. In DNT-learning among several congestion control actions

[5]An inter-contact time between two nodes is defined as the length of the time interval over which the two nodes are not in contact and are in contact at the end points of this interval.

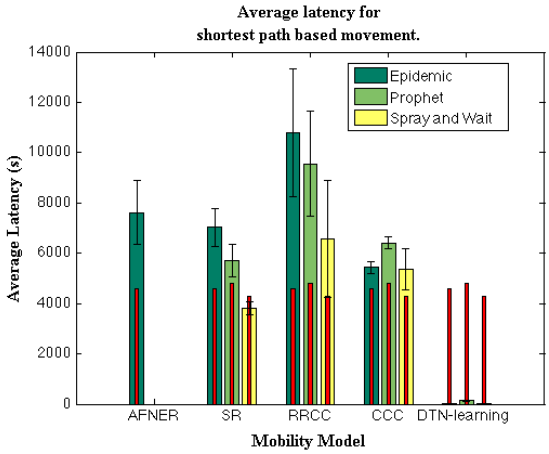
[6]Contact duration is time during which a node pair is in contact.



(a) Random Walk



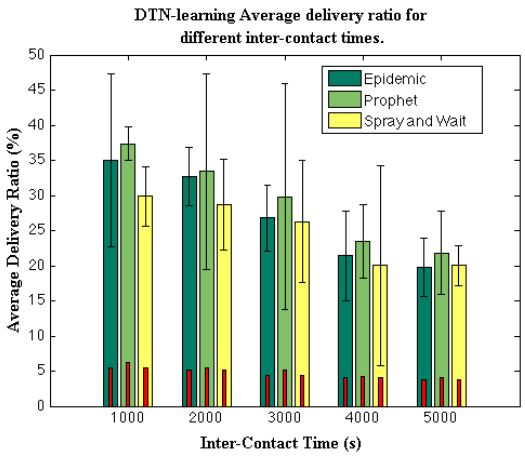
(b) Random Way Point



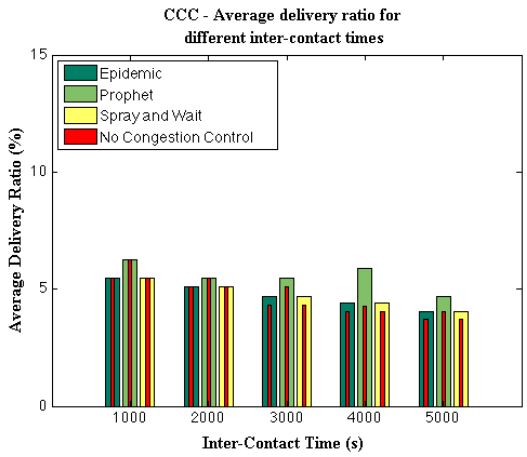
(c) Shortest Path Map Based Movement

FIGURE 6.24 – Average latency for different mobility models and routing protocol (Terrestrial Scenario, Buffer Size of $500KB$, Message Generation Period of $300s$, Buffer Threshold for DTN-learning of 60%, WoLF Action Selection Method for DTN-learning).

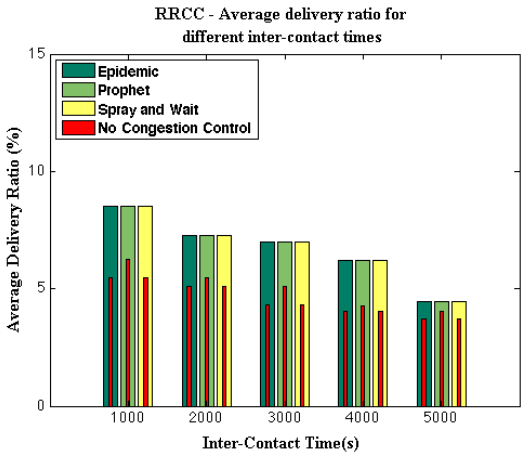
available, it includes one similar action to RRCC replication limit. DTN-learning allows the node to manage its local message generation rate without using any neighbor information. In addition, DTN-learning allows the node to learn which better action to use since it has several possibilities available. Because of this, DTN-learning achieves the highest message delivery ratio. Since DTN-learning provides a smart way to control congestion, it can adapt to different DTN environments over either different contact duration or inter-contact time.



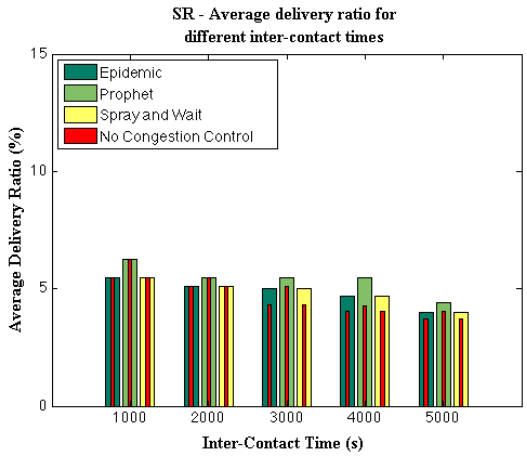
(a) DTN-learning



(b) CCC

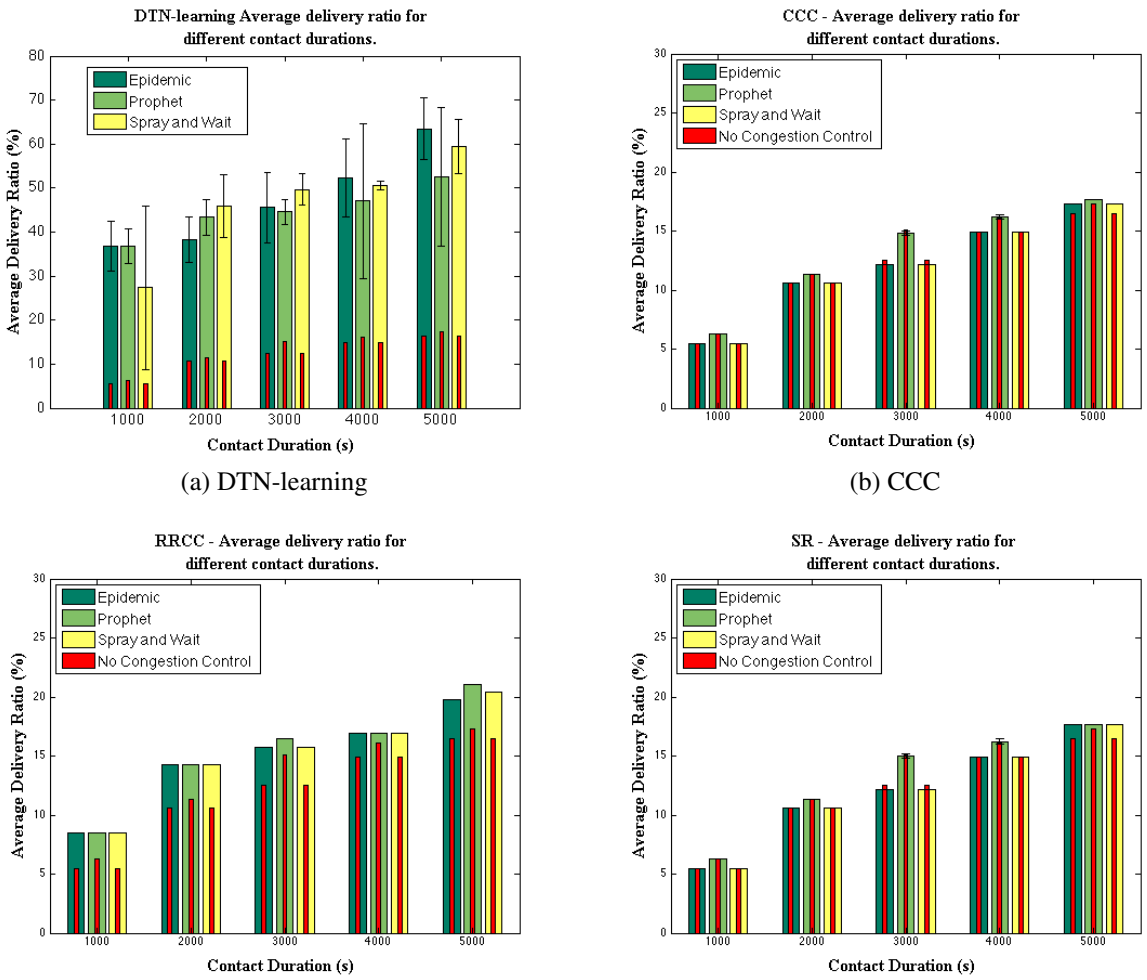


(c) RRCC



(d) SR

FIGURE 6.25 – Average delivery ratio for different inter-contact times (IPN Scenario, Buffer Size of $4000KB$, Contact Duration of $1000s$, Message Generation Period of $300s$, Buffer Threshold for DTN-learning of 60%).



(a) DTN-learning (b) CCC

(c) RRCC (d) SR

FIGURE 6.26 – Average delivery ratio for different contact durations (IPN Scenario, Buffer Size of $4000KB$, Inter-Contact Time of $1000s$, Message Generation Period of $300s$).

6.4 Conclusion

In delay tolerant networks, the high mobility of nodes and their dramatically changing topologies lead to intermittent connectivity. The store-carry-forward principle is used by most routing protocols to forward messages. With limited storage space, excessive copies of messages can easily lead to buffer overflow, especially when the bandwidth is also limited and the message sizes differ. In this situation, the question of how to allocate network resources becomes important. In this chapter, a novel congestion control framework for delay and disruption tolerant network based on reinforcement learning, called DTN-learning, was presented. This strategy, which aims to improve the delivery ratio, allows the node to perform congestion control by applying a machine learning technique intelligently and autonomously. Using this approach the node has the capability to learn how to perform congestion control by trial-and-error. The most important, DTN-learning can be applied in any DTN environment since the nodes can adapt to dynamic environments. We conducted simulations in ONE under two scenarios: terrestrial and IPN. The simulation results show that, compared to other DTN congestion control mechanisms, DTN-learning significantly improves the delivery ratio independent of the routing protocol.

Our framework is independent of the nodes' mobility model, and we considered the case when the mobile nodes move according to the random waypoint, random walk and shortest path map based movement models in the terrestrial scenario. Furthermore the framework considers different routing protocols (Epidemic, Prophet, Spray and Wait) in both scenarios terrestrial and IPN. Further, we provided two different action selection techniques based on our reinforcement learning algorithm: Boltzmann and WoLF which are based on the probability of executing an action in a given node's state (Congested, Prospective-Congested, Decrease-Congested, Non-Congested). The results shown that the terrestrial and IPN scenario present better learning performance when using Boltzmann. However, in IPN scenario the learning performance present some fluctuation as expected. After combining DTN-learning with supervised learning in IPN scenario the learning performance linearly improve. The chapter showed that our DTN congestion control framework based on reinforcement learning outperforms the simulation based approaches in the literature (e.g., AFNER, CCC, RRCC, SR). There are still some open questions that we leave for future work such as design real testbed. Additionally, we present some more future work ideas in the next chapter.

7 Conclusion and Future Directions

7.1 Conclusion

Our research on design and developing an effective congestion control framework for delay tolerant networks (DTNs) addresses the open technical issues of how to efficiently increase message delivery ratio in such disconnected and congested networks that in general suffer from lack of continuous network connectivity. We specifically focused on congestion issues in DTNs. This thesis studied various DTN congestion control techniques, and proposed a new framework that improves message delivery probability of the DTNs using the idea of autonomous control implemented through reinforcement learning.

We found that the design of a more general congestion control mechanism for DTN depends on three important factors towards increasing the delivery performance. First it needs to operate independently of the routing protocol that makes it suitable for different DTN scenarios. Second it must embrace a hybrid approach being able to prevent congestion that takes place (proactive control) and promptly react to the congestion when it happens (reactive control). Since messages can be buffered for long periods of time before being acknowledged, buffer saturation becomes highly common resulting in excessive latencies and messages losses. This is aggravated by the fact that, in this context, reactive congestion control would be severely impacted by delayed control decisions. To solve this problem a proactive strategy is extremely important. Because of this our framework provides a hybrid smart approach.

As a third factor, a new congestion control mechanism must consider both strategies open- and closed-loop where an open-loop scheme is fundamentally important because it takes into account policies that carefully looks for minimizing congestion in the first place since a closed-loop scheme is not alway adequate for DTN. Once the closed-loop scheme uses feedback information, in the context of DTNs, such feedback is not immediate and may take some time before nodes are informed. This surely has a negative impact on network's performance. Thus the usage of closed-loop feedback control to maintain consistent state at the source and destination cannot work.

We have combined the three factors previously discussed with an online learning algorithm. In particular, our work considers a straighforward application of Q-learning to a congestion con-

trol scheme. This resulted in a novel congestion control framework, called DTN-learning. The DTN-learning allows nodes without having to know in advance the network topology and traffic patterns, and without the need for any centralized control and global information being able to discover efficient congestion control policies in a dynamically changing network. We built a simple framework where nodes adapt their behaviors based on feedback received from the environment, and stochastically select actions accordingly. The modeled feedback function takes into the node's states where the transition for one of the states considered terminal (Decrease-Congested or Non-Congested) provides a positive reward while the transition to Congested or Prospective-Congested states provides a negative reward. Essentially, the task of the node is to learn from this indirectly, to choose sequences of actions that produce the greatest cumulative reward. We implemented if the reward is positive or negative based on how the chosen action affects the node's state.

In the view of the congestion phenomenon caused by excessive consumption of storage space brought by store-carry-and-forward paradigm, incoming messages may be dropped due to buffer overflow. This increases the drop ratio and overhead as the forwarder inefficiently ends up consuming precious resources to transmit messages that are dropped. It is important to point out here that in our framework, when a node becomes congested or prospective congested it has an intelligent way to cope with this problem. The congested or prospective congested node informs its neighbors by broadcasting a congestion notification that informs the neighbors of either the buffer occupancy rate or the time for the buffer fill up since it is imminet to buffer exhaustion. Thus using this information, the neighbors have the option of delaying transmission of messages to that node since the message has a high probability of being discarded. This decreases message drop and avoid unnecessary consuming of resources.

It is important to highlight here that due to short contacts and large intercontact intervals, some stored unexpired messages may not have enough residual lifetimes for a contact to occur. In particular, in deep space scenarios, messages may expire while propagating to their intended receivers. DTN-learning provides the nodes the action to discard early those messages that significantly reduce buffer occupancy.

We evaluated DTN-learning and its performance thoroughly via comprehensive simulations. As part of our study we presented and implemented a variety of controlling actions in order to evaluate the performance of our protocol under various scenarios. Our studies indicate that our approach can achieve competitive message delay and delivery rates while it outperforms existing state-of-the-art DTN congestion control mechanisms in the literature.

Preliminary experiments show that, by learning to take more conservative actions in the presence of congestion the delivery ratio increases, and more aggressive actions, too, but at the expense of increasing drop ratio. Nodes using our proposed approach reach high delivery ratio that those nodes simulated using non-adaptive congestion control schemes.

Although the experiments described here are not fully realistic from the standpoint of actual DTN, we belive this work has shown that adaptive congestion control scheme is a natural domain for reinforcement learning. The key point here is that our framework can be easily combined with any congestion control strategy that allows it to be interoperable in a range of DTN applications. In this direction algorithms based on DTN-learning, but specifically tailored to the DTN congestion control domain, will likely perform even better.

The novel DTN-learning framework is a step towards achieving our vision of local and autonomous DTN congestion control. The key contribution here is that the nature of our framework which is based on reinforcement learning allows it to effectively increase delivery ratio while being robust enough to cope with the highly dynamic and unpredictable environments in DTNs, being aware of the learning time cost.

7.2 Future Directions

During our research, we identified the following topics of particular interest to DTNs:

1. **DTN-learning Enhancement and Optimization.** Several studies presented in this thesis were done in order to optimize and enhance DTN-learning functionality and performance. There are still open areas that we believe could lead to an increase in the performance of DTN-learning. For example, utilizing an annealing schedule technique where in each learning phase the node determines what proportion of time is spent on exploration and how much time it intends to exploit its actual knowledge to maximize the sum of expected rewards. We belive this can make it more robust under the dynamic environment as well as increasing its performance.

2. **Security Issues.** In DTN-learning we assumed that all mobile nodes cooperate in message delivery. However, in a real application, there might be situations where selfish nodes may deny distribution of other mobile nodes' messages. In other situations, malicious nodes can dramatically affect the performance of DTN-learning. Malicious nodes may attempt to reduce network connectivity by pretending to be cooperative but in effect not delivering any data messages. This situation can be intensified if they attempt to overload relays' buffer with their own messages threatening the delivery performance of the framework. In our future work, we propose different mechanisms to effectively detect malicious nodes and find methods to balance their side effects.

3. **Real Life Applications and Scenarios.** This thesis was based on the hypothetical applications and scenarios which are in general provided in the literature. In order to compare simulation results to real-life measurements, we are looking to make a real world application in which real observations would be studied. We intend to incorporate our framework

into the Interplanetary Overlay Network (ION) open-source implementation. Currently, ION congestion control is essentially done manually and its actions are set a priori based on scheduled node contacts given by a "contact graph." By utilizing our congestion control framework, ION will be able to also consider opportunistic and probabilistic contacts which is expected to alleviate congestion and improve delivery reliability in IPNs. It will also allow ION to be used in other DTN applications including sensor networks, emergency response, etc. We will design an ION testbed to deploy and validate the proposed congestion control framework.

4. **Second order derivative.** Another interesting future direction is exploring the effects of using a second order derivative in terms of buffer occupancy growth. Currently, DTN-learning considers the current and past rate of buffer occupancy and average this rate using exponentially weighted average to account for both older and newer data. If DTN-learning used a second order derivative, it would consider the change in rate of buffer occupancy over time. This trend could be used to set an appropriate node's state. In addition we can compare the results of both of them.

In order to stimulate real-world applications of RL, it is necessary to demonstrate that RL methods can be efffectively used in the control of physical systems. The successful real-time learning control results presented in this thesis are in our view an important step in this direction.

Bibliography

AHMAD, S.; MUSTAFA, A.; AHMAD, B.; BANO, A.; HOSAM, A.-S. Comparative study of congestion control techniques in high speed networks. **International Journal of Computer Science and Information Security**, v. 6, n. 2, p. 222–231, 2009.

AKYILDIZ, I.; AKAN, O.; CHEN, C.; FANG, W. S. J. The state of the art in interplanetary internet. **IEEE Communications Magazine**, v. 42, n. 7, p. 108–118, July 2004.

AN, Y.; LUO, X. MACRE: A novel distributed congestion control algorithm in DTN. **Trans Tech Publications, Advanced Engineering Forum**, v. 1, p. 71–75, 2011.

ANDALORA, C. **Q-learning and The Impact of Exploration Policies**. November 2007. `http://www.cs.rit.edu/~cja9200/files/ArtificialIntelligence_ResearchPaper.pdf`.

ARANITI, G.; BISIO, I.; SANCTIS, M. D. Interplanetary networks: Architectural analysis technical challenges and solutions overview. In: **IEEE ICC 2010 Proceedings**. [S.l.: s.n.], 2010. p. 1–5.

AYUB, Q.; RASHID, S. T-drop: An optimal buffer management policy to improve QoS in DTN routing protocols. **Journal of Computing**, v. 2, n. 10, p. 46–50, October 2010.

AYUB, Q.; RASHID, S.; ZAHID, M. S. M. Buffer scheduling policy for opportunistic networks. **International Journal of Scientific & Engineering Research**, v. 2, n. 7, July 2011.

BARTO, A. G. **Some Learning Tasks From a Control Perspective**. [S.l.], 1991.

BASAGNI, S.; CONTI, M.; GIORDANO, S.; STOJMENOVIC, I. **Mobile Ad Hoc Networking**. first. [S.l.]: Wiley-IEEE Press, 2005.

BASU, P.; GUHA, S.; SWAMI, A.; TOWSLEY, D. Percolation phenomena in networks under random dynamics. In: **Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on**. [S.l.: s.n.], 2012. p. 1–10.

BISIO, I.; CELLO, M.; COLA, T. de; MARCHESE, M. Combined congestion control and link selection strategies for delay tolerant interplanetary networks. In: **GLOBECOM 2009**. [S.l.: s.n.], 2009.

BOWLING, M.; VELOSO, M. Rational and convergent learning in stochastic games. In: **In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence**. [S.l.: s.n.], 2001. p. 1021–1026.

BOWLING, M.; VELOSO, M. Multiagent learning using a variable learning rate. **Artificial Intelligence**, v. 136, p. 215–250, 2002.

BOX, G. E. P.; JENKINS, G. **Time Series Analysis, Forecasting and Control**. [S.l.]: Holden-Day, Incorporated, 1990. ISBN 0816211043.

BOX, G. E. P.; JENKINS, G. M.; MACGREGOR, J. F. Some recent advances in forecasting and control. **Journal of the Royal Statistical Society. Series C (Applied Statistics)**, v. 23, n. 2, p. 158–179, 1974.

BROADBENT, S.; HAMMERSLEY, J. Percolation processes i. crystals and mazes,. **Cambridge Philosophical Society**, v. 53, p. 629–641, 1957.

BURGESS, J.; GALLAGHER, B.; JENSEN, D.; LEVINE, B. N. Max-prop: Routing for vehicle-based disruption tolerant networks. In: **25th IEEE Int. Conf. on Computer Communications**. [S.l.: s.n.], 2006. p. 1–11.

BURLEIGH, S. Interplanetary overlay network: An implementation of the dtn bundle protocol. In: **In 4th IEEE Consumer Communications and Networking Conference**. [S.l.: s.n.], 2007. p. 222–226.

BURLEIGH, S. **Interplanetary Overlay Network - Design and Operation**. [S.l.], 2013.

BURLEIGH, S.; FALL, K.; CERF, V.; DURST, B.; SCOTH, K.; WEISS, H. Delay-tolerant networking: An approach to interplanetary internet. **IEEE Communication Magazine**, v. 41, n. 6, 2003. `http://trs-new.jpl.nasa.gov/dspace/handle/2014/40636.` Acessed in May 2012.

BURLEIGH, S.; HOOKE, A.; TORGERSON, L.; FALL, K.; CERF, V.; DURST, B.; SCOTT, K.; WEISS, H. An approach to interplanetary internet. **IEEE Communication Magazine**, July 2003.

BURLEIGH, S.; JENNINES, E.; SCHOOLCRAFT, J. Autonomous congestion control in delay tolerant network. Jet Propulsion Laboratory, American Institute of Aeronautics and Astronautic. 2007.

BURLEIGH, S.; JENNINGS, E.; SCHOOLCRAFT, J. Autonomous congestion control in delay tolerant networks. In: **AIAAA SpaceOps'06**. [S.l.: s.n.], 2006.

BUŞONIU, L.; BABUŠKA, R.; SCHUTTER, B. De. A comprehensive survey of multi-agent reinforcement learning. **IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews**, v. 38, n. 2, p. 156–172, mar. 2008.

CAINI, C.; CRUICKSHANK, H.; FARRELL, S.; MARCHESE, M. Delay and disruption tolerant networking (DTN): An alternative solution for future satellite networking applications. In: **IEEE**. [S.l.: s.n.], 2011. v. 99, n. 11, p. 1980–1997.

CAMP, T.; BOLENG, J.; DAVIES, V. A survey of mobility models for ad hoc network research. **Wireless Communications and Mobile Computing**, v. 2, n. 5, p. 483–502, 2002.

CASETTI, C.; MEO, M. Anew approach to model the stationary behaviour of tcp connections. In: **INFOCOM 2000 Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies**. [S.l.: s.n.], 200.

CAVENDISH, D. A control theoretical approach to congestion control in packet networks. **IEEE/ACM Transactions on Networking**, v. 12, n. 5, p. 893–906, 2004.

CELLO, M.; GNECCO, G.; MORCHESE, M.; SONGUINETI, M. A model of buffer occupancy for ICNs. **IEEE Communication Letters**, v. 16, n. 6, 2012.

CERF, V.; BURLEIGH, S.; HOOKE, A.; TARGERSON, L.; DURST, R.; SCOTT, K.; FALL, K.; WEISS, H. Delay tolerant networking architecture. Internet RFC4838. April 2007.

CERF, V.; BURLEIGH, S.; HOOKE, A.; TARGERSON, L.; DURST, R.; SCOTT, K.; FALL, K.; WEISS, H. Delay tolerant networking architecture. Internet RFC4838. April 2007.

CHANDRASEKARAN, B. **Survey of Network Traffic Models**. `http://www.cse.wustl.edu/~jain/cse567-06/ftp/traffic_models3.pdf`. Accessed in March 2015.

CHAUHAN, A.; KUMAR, S.; KUMAR, V.; MUKHERJEE, S.; BHUNIA, C. T. **Implementing Distributive Congestion Control in a Delay Tolerant Network for Different Types of Traffic**. 2012. `www.ietf.org/mail.../web/.../docYrH2LrS75z.doc`. Department of Computer Science and Engineering, Indian School of Mines, Deemed University. Accessed in June 18, 2012.

CHEN, C.; CHEN, Z. Towards a routing framework in ad hoc space networks. **International Journal of Ad Hoc and Ubiquitous Computing**, v. 5, n. 1, 2010.

CHEN, T. M. The handbook of computer networks. In: ____. [S.l.]: Hossein Bidgoli (ed.), Wiley, 2007. cap. Network Traffic Modeling.

CHENG-JUN, W.; ZHENG-HU, G.; YONG, T.; ZI-WEN, Z.; BAO-KANG, Z. CRSG: A congestion routing algorithm for security defense based on social psychology and game theory in DTN. **Journal of Central South University**, v. 20, n. 2, p. 440–450, February 2013. Springer.

CLARK, D. D.; PARTRIDGE, C.; RAMMING, J. C.; WROCLAWSKI, J. T. A knowledge plane for the internet. In: **Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications**. New York, NY, USA: ACM, 2003. (SIGCOMM '03), p. 3–10. ISBN 1-58113-735-4. Disponível em: <http://doi.acm.org/10.1145/863955.863957>.

COE, E.; RACHAVENDRA, C. Token based congestion control for DTN. In: **Aerospace Conference**. [S.l.]: IEEE, 2010.

DAVIS, J. A.; FAGG, A. H.; LEVINE, B. N. Wearable computers as packet transport mechanism in highly-partitioned ad-hoc networks. In: **International Symposium on Werable Computing**. Zurich: [s.n.], 2001. p. 141–148.

DEMMER, M.; BREWER, E.; FALL, K.; JAIN, S.; HO, M.; PATRA, R. **Implementing Delay Tolerant Networking**. [S.l.], 2004.

DTNRG. **Delay-Tolerant Networking Research Group**. July 2014. `https://sites.google.com/site/dtnresgroup/home`.

DURST, R. C.; MILLER, G. J.; TRAVIS, E. J. TCP extensions for space communication. In: **Proceedings of the 2nd annual international conference on Mobile computing and networking**. [S.l.: s.n.], 1996. p. 15–26.

FALL, K. A delay tolerant netwok architecure for challenged internets. In: **ACM SIGCOMMM**. [S.l.: s.n.], 2003.

FALL, K. **A Delay-Tolerant Network Architecture for Challenged Internets**. [S.l.], 2003.

FALL, K.; FARRELL stephen. DTN: an architecture retrospective. **IEEE Journal on Selected Areas in Communications**, v. 26, n. 5, 2008.

FALL, K.; HONG, W.; MADDEN, S. **Custody Transfer for Reliable Delivery in Delay Tolerant Networks**. [S.l.], 2003. IRB-TR-03-030.

FARRELL, S. **A Delay and Disruption Tolerant Transport Layer Protocol**. Tese (Doutorado) — University of Dublin, Trinity College, September 2008.

FARRELL, S.; CAHILL, V.; GERAGHTY, D.; HUMPHREYS, I. When TCP breaks: Delay and disruption-tolerant networking. **IEEE Internet Computing**, v. 10, n. 4, 2006.

FLOYD, S. **Metrics for the Evaluation of Congestion Control Mechanisms**. March 2008. `http://www.ietf.org/rfc/rfc5166.txt`. Network Working Group, Accessed in November 06.

GALANT, D. C. **Queuing Theory Models for Computer Networks**. [S.l.], 1989. NASA Technical Memorandum.

GERLA, M.; KLEINROCK, L. Vehicular networks and the future of the mobile internet. **Computer Networks, Elsevier**, v. 55, n. 2, p. 457–469, 2011.

GODOY, J.; KARAMOUZAS, I.; GUY, S. J.; GINI, M. Online learning for multi-agent local navigation. In: **CAVE Workshop at AAMAS**. [S.l.: s.n.], 2013.

GRAVER, W.; TIPPER, D. Design and operation of survivable networks. **Journal of Network and Systems Management - Special Issue on Designing and Managing Networks on Service Reliability**, v. 13, n. 1, p. 7–13, 2005.

GRIMMETT, G. **Percolation**. [S.l.]: Springer, 1999.

GRUNDY, A.; RADENKOVIC, M. Promoting congestion control in opportunistic networks. In: **IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications**. [S.l.: s.n.], 2010. p. 324–330. Department of Computer Science, University of Nottingham.

GRUNDY, A. M. **Congestion Framework For Delay-Tolerant Communications**. Tese (Doutorado) — School of Computer Science, University of Nottingham, July 2012.

GUHA, S.; BASU, P. **Applications of Directed Percolation in Wireless Networks**. 2012. `http://www.academia.edu/2840955/Applications_of_Directed_Percolation_in_Wireless_Networks`. Raytheon BBN Technologies.

GULLAPALLI, V. **Reinforcement Learning and Its Application to Control**. Tese (Doutorado) — Department of Computer and Information Sciences, University of Massachusetts, 1992.

HARMON, M. E.; HARMON, S. S. **Reinforcement Learning: A Tutorial**. 1996.

HARRAS, K. A.; ALMEROTH, K. C.; BELDING-ROYER, E. Delay tolerant mobile networks (DTMNs): Controlled flooding schemes in sparce mobile networks. In: **IFIP Networking**. [S.l.: s.n.], 2005.

HEMMINGER, S. **Network Emulation with NetEm**. April 2005. Linux Conference Australia - LCA2005.

HERBERTSSON, F. **Implementation of a Delay-Tolerant Routing Protocol in the Network Simulator NS-3**. Tese (Doutorado) — Departament of Computer and Information Science, Linkopings Universitet, 22/12/2010.

HOLTON, G. J.; BRUSH, S. G. **Physics, The Human Adventure: From Copernicus to Einstein and Beyond**. [S.l.]: Rutgers University Press, 2001.

HUA, D.; DU, X.; CAO, L.; XU, G.; QIAN, Y. A DTN congestion avoidance strategy based on path avoidance. In: **2nd International Conference on Future Computer and Communication, IEEE**. [S.l.: s.n.], 2010. v. 1, p. 855–860.

HYYTIA, E.; OTT, J. Criticality of large delay tolerant networks via directed continuum percolation in space time. In: **INFOCOM2013**. [S.l.: s.n.], 2013. p. 320–324.

IBRAHIM, M. **Routing and Performance Evaluation of Disruption Tolerant Networks**. Tese (Doutorado) — UniversitÃ© de Nice - Informatique - Sophia AntÃpolis, November 2008.

INDGREN, A.; PHANSE, K. S. Evaluation of queuing policies and fowarding strategies for routing in intermittently connected networks. In: **IEEE COMSWARE**. [S.l.: s.n.], 2006. p. 1–10.

INSA-CABRERA, J.; L.DOWE, D.; HERNÃ¡NDEZ-ORALLO, J. Evaluating a reinforcement learning algorithm with a general intelligence test. In: LOZANO, J. A.; GÃ¡MEZ, J. A.; MORENO, J. A. (Ed.). **CAEPIA**. [S.l.]: Springer. (Lecture Notes in Computer Science, v. 7023), p. 1–11. ISBN 978-3-642-25273-0.

JACOBSON, V. Congestion avoidance and control. In: REVIEW, A. C. C. (Ed.). **SIGCOMM**. Stanford, CA: [s.n.], 1988. v. 18, p. 314–329.

JAIN, S.; FALL, K.; PATRA, R. Routing in a delay tolerant network. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 34, n. 4, p. 145–158, ago. 2004. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1030194.1015484>.

JAIN, S.; FALL, K.; PATRA, R. Routing in delay tolerant network. In: **Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Communications**. [S.l.: s.n.], 2004.

JONES, E. P.; WARD, P. A. Routing strategies for delay-tolerant networks. **ACM Computer Communication Review**, Citeseer, 2006.

JORDAN, M. I.; RUMELHART, D. E. **Forward models: Supervised Learning With a Distal Teacher**. 1990. Center for Cognitive Science, Massachusetts Institute of Technology.

JPL-NASA. **Deep Space Network**. 1998. `http://deepspace.jpl.nasa.gov`. Jet Propulsion Laboratory, California Institute of Technology. Last access in April 2015.

KERANEN, A.; OTT, J. **Increasing Reality for DTN Protocol Simulation**. [S.l.], July 2007.

KERäNEN, A.; OTT, J.; KäRKKäINEN, T. The ONE Simulator for DTN Protocol Evaluation. In: **Proceedings of the 2nd International Conference on Simulation Tools and Techniques**. New York, NY, USA: ICST, 2009.

KHABBAZ, M. J.; ASSI, C. M.; FAWAZ, W. F. Disruption-tolerant networking: A comprehensive survey on recent developments and persisting challhenges. In: **IEEE Communications Survey & Tutorials**. [S.l.: s.n.], 2011.

KIM, D.; PARK, H.; YEOM, I. Minimize the impact of buffer overflow in DTN. In: **International Conference on Future Internet Technologies**. [S.l.: s.n.], 2008.

KONG, Z.; YEH, E. M. **Connectivity, Percolation, and Information Dissemination in Large-Scale Wireless Networks with Dynamic Links**. 2009. `http://arxiv.org/abs/0902.4449`. Cornell University Library.

KONLER, E.; HANDLEY, M.; FOYD, S. **Datagram congestion control protocol (DCCP)**. [S.l.], 2006. RFC4340.

KRIFA, A.; BARAKAT, C.; SPYROPOULOS, T. Optimal buffer management policies for delay tolerant networks. In: . [S.l.: s.n.], 2008.

LABORATORY, J. P. **ION: Interplanetary Overlay Network**. 2013. `https://ion.ocp.ohiou.edu/`. Accessed in January 2013.

LAKKAKORPI, J.; PITKANEN, M.; OTT, J. Using buffer space advertisements to avoid congestion in mobile opportunistic DTNs. In: **9th IFIP TC 6 International Conference on Wired/Wireless Internet Communications**. [S.l.: s.n.], 2011. p. 386–397.

LEE, T. B.; FIELDING, T.; MASINTER, L. **Uniform Resource Identifier (URI): Generic Syntax**. [S.l.], 2005.

LEE, U.; ZHOU, B.; GERLA, M.; MAGISTRETTI, E. Mobeyes: smart mobs for urban monitoring with a vehicular sensor network. **Wireless Communications, IEEE**, v. 13, n. 5, p. 52–57, October 2006.

LEELA-AMORNSIN, L.; ESAKI, H. Heuristic congestion control for massage deletion in delay tolerant network. In: **Smart Spaces and Next Generation Wired/Wireless Networking**. [S.l.: s.n.], 2010. Third Conference on Smart Spaces and 10th International Conference.

LI, Y.; QIAN, M.; JIN, D.; SU, L.; ZENG, L. Adaptive optimal buffer management policies for realistic DTN. In: **IEEE GLOBECOM 2009**. [S.l.: s.n.], 2009.

LINDGREN, A.; DORIA, A. Probabilistic routing in intermittently connected networks. Internet-Draft, draft-irtf-dtnrg-prophet-01. 2008.

LINDGREN, A.; DORIA, A.; AL et. Probabilistic routing in intermittently connected networks. **Computer Communication Review**, v. 7, p. 19–20, 2003.

LINDGREN, A.; DORIA, A.; SCHELEN, O. Probabilistic routing in intermittently connected networks. In: **SIGMOBILE Mobile Computing and Communication Review**. [S.l.: s.n.], 2003. v. 7, n. 3, p. 19–20.

LO, S. C.; LU, C. L. A dynamic congestion control based routing for delay tolerant network. In: **9th International Conference on Fuzzy Systems and Knowledge Discovery**. [S.l.: s.n.], 2012. p. 2047–2051.

MAHADEVAN, S.; CONNELL, J. Automatic programming of behavior based robots using reinforcement learning. **Artificial Intelligence**, v. 55, p. 311–365, 1992.

MAMATAS, L.; HARKS, T.; TSAOUSSIDIS, V. Approaches to congestion control in packet networks. **Internet Engineering**, v. 1, n. 1, 2007.

MATARIC, M. J. Reinforcement learning in the multi-robot domain. **Autonomous Robots**, v. 4, n. 1, p. 73–83, 1997.

MATHIS, M.; SEMKE, J.; MAHDAVI, J.; OTT, T. The macroscopic behaviour of thetcp congestion avoidance algorithm. **Computer Communication Review**, v. 27, n. 3, 1997.

MATSUDA, T.; TAKINE, T. (p,q)-epidemic routing for sparsely populated mobile ad hoc networks. **Selected Areas in Communications, IEEE**, p. 783–793, 2008.

MENDEL, J. M.; MCLAREN, R. W. A prelude to neural networks. In: MENDEL, J. M. (Ed.). Upper Saddle River, NJ, USA: Prentice Hall Press, 1994. cap. Reinforcement-learning Control and Pattern Recognition Systems, p. 287–318. ISBN 0-13-147448-0. Disponível em: <http://dl.acm.org/citation.cfm?id=190521.187334>.

METRICS for the Evaluation of Congestion Control Mechanism. August 2006. Http://www.ietf.org/mail-archive/web/ietf-announce/current/msg04676.html. Internet-DRAFT. Acessed in May 2012.

MISRA, V.; GONG, W. B.; TOWSLEY, D. Fluid-based analysis of a network of AQM routers supporting tcp flows with application to RED. In: **ACM SIGCOMM'00**. [S.l.: s.n.], 2000. p. 151–160.

MUKHERJEE, J. **Routing Over The Interplanetary Internet**. Tese (Doutorado) — The Graduate College at the University of Nebraska, 2012.

NELSONA, S. C.; BAKHTAND, M.; KRAVETS, R.; HARRIS, A. F. Encounter-based routing in dtns. **ACM SIGMOBILE Mobile Computing and Communications Review**, v. 13, n. 1, p. 56–59, 2009.

PADHYE, J.; FIROIU, V.; TOWSLEY, D.; KUROSE, J. Modeling tcp throughput: A simple model and its empirical validation. In: **ACM SIGCOMM98**. [S.l.: s.n.], 1998. p. 303–314.

PARSHANI, R.; DICKISON, M.; COHEN, R.; STANLEY, H. E.; HAVLIN, S. Dynamic networks and directed percolation. **A Letters Journal Exploring The Frontiers of Physics**, 2010.

PELUSI, L.; PASSARELLA, A.; CONTI, M. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. **IEEE Communication Magazine**, v. 44, n. 11, p. 134–141, November 2006.

PERES, Y.; SINCLAIR, A.; SOUSI, P.; STAUFFER, A. Mobile geometric graphs: Detection, coverage and percolation. In: **the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms**. [S.l.: s.n.], 2011. p. 412–428.

PSARAS, I.; WOOD, L.; TAFAZOLLI, R. **Delay-/Disruption-Tolerant Networking: State of the Art and Future Challenges**. [S.l.], 2009.

RADENKOVIC, M.; GRUNDY, A. Congestion aware forwarding in delay and social opportunistic networks. In: **Eight International Conference on Wireless on Demand Network Systems and Services**. [S.l.: s.n.], 2011. p. 60–67.

RANGO, F. D.; TROPEA, M.; LARATTA, G. B.; MARANO, S. Hop-by-hop local flow control over interplanetary networks based on DTN architecture. In: **Communications, 2008. ICC'08. IEEE International Conference on**. [S.l.: s.n.], 2008. p. 1920–1924.

RASHID, S.; ABDULLAH, A. H.; ZAHID, M. S. M.; AYUB, Q. Mean drop and effectual buffer management policy for delay tolerant network. **European Journal of Scientific Research**, v. 70, n. 3, p. 396–407, 2012.

RASHID, S.; AYUB, Q. Effective buffer management policy DLA for DTN routing protocols under congestion. **international Journal of computer and Network Security**, v. 2, n. 9, p. 118–121, 2010.

RASHID, S.; AYUB, Q.; ZAHID, M. S. M.; ABDULLAH, S. H. E-DROP: An effective drop buffer management policy for DTN routing protocols. **International Journal of Computer Applications**, v. 13, n. 7, January 2011.

ROBERTS, S. W. Control chart tests based on geometric moving averages. **Technometrics**, v. 1, p. 239–250, 1959.

RUSSEL, S. J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 2. ed. [S.l.]: Pearson Education, 2003. ISBN 0137903952.

SCOTH, K.; BURLEIGH, S. **Bundle Protocol Specification**. November 2007. `http://www.rfc-editor.org/rfc/rfc5050.txt`. Request for Comments: RFC 5050.

SELIGMAN, M.; FALL, K.; MUNDUR, P. Alternative custodians for congestion in delay tolerant networks. In: **SIGCOMM'06 Workshops**. [S.l.: s.n.], 2006.

SELIGMAN, M.; FALL, K.; MUNDUR, P. Storage routing for DTN congestion control. In: **Wireless Communications and Mobile Computing**. [S.l.: s.n.], 2007. v. 7, p. 1183–1196.

SILVA, A. P. da; BURLEIGH, S.; HIRATA, C. M.; OBRACZKA, K. A survey on congestion control for delay and disruption tolerant networks. **Elsevier Ad Hoc Networks**, 2014.

SINGH, S. P.; BARTO, A. G.; GRUPEN, R.; CHRISTOPHER; CONNOLLY, C. Robust reinforcement learning in motion planning. In: **Advances in Neural Information Processing Systems 6**. [S.l.]: Morgan Kaufmann, 1994. p. 655–662.

SKINNER, B. How to teach animals. In: **Scientific American**. [S.l.: s.n.], 1951. p. 26–29.

SOELISTIJANTO, B.; HOWARTH, M. P. Transfer reliability and congestion control strategies in opportunistic networks: A survey. **IEEE Communications Surveys And Tutorials**, 2013.

SPYROPOULOS, T.; PSOUNIS, K.; RAGHAVENDRA, C. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In: **ACM SIGCOMM workshop on Delay-tolerant Networking**. [S.l.: s.n.], 2005. p. 259.

SPYROPOULOS, T.; PSOUNIS, K.; RAGHAVENDRA, C. S. An efficient routing scheme for intermittently connected mobile networks. In: **2005 ACM SIGCOMM workshop on Delay-tolerant Networking**. [S.l.: s.n.], 2005. p. 252–259.

SPYROPOULOS, T.; RAIS, R. N. B.; TURLETTI, T.; OBRACZKA, K.; VASILAKAS, A. DTNs: Protocols and aplications. In: ____. [S.l.]: CRC Press, 2011. cap. DTN Routing: Taxonomy and Design.

SRIKANT, R. **The Mathematics of Internet Congestion Control**. [S.l.]: Birkusera Boston, 2004.

STAUFFER, A. **Space Time Percolation and Detection by Mobile Nodes**. [S.l.], 2012. `http://arxiv.org/abs/1108.6322`.

STERBENZA, J. P.; HUTCHISONB, D.; CETINKAYAA, E. K.; JABBARA, A.; ROHRERA, J. P.; SCHOLLERC, M.; SMITHB, P. Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. **Computer Networks, Elsevier**, v. 54, n. 8, p. 1245–1265, June 2010.

STRAUSS, C.; SAHIN, F. Autonomous navigation based on a q-learning algorithm for a robot in a real environment. In: **3rd IEEE International Conference on System of Systems Engineering, SoSE 2008, Singapore, 2-4 June 2008**. [S.l.: s.n.], 2008. p. 1–5.

SUTTON, R. S. Learning to predict by the methods of temporal differences. In: **MACHINE LEARNING**. [S.l.]: Kluwer Academic Publishers, 1988. p. 9–44.

SUTTON, R. S. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In: **In Proceedings of the Seventh International Conference on Machine Learning**. [S.l.]: Morgan Kaufmann, 1990. p. 216–224.

SUTTON, R. S.; BARTO, A. G. **Introduction to Reinforcement Learning**. 1st. ed. Cambridge, MA, USA: MIT Press, 1998. ISBN 0262193981.

THE Network Simulator ns2. 2013. `http://nsnam.isi.edu/nsnam/index.php/User_Information`. Accessed in March 2013.

THOMPSON, N.; KRAVETS, R. Understanding and controlling congestion in DTNs. In: **Mobile Computing and Communications**. [S.l.: s.n.], 2010. v. 13, n. 3.

THOMPSON, N.; NELSON, S. C.; BAKNT, M.; ABDELZAHER, T.; KRAVETS, R. Retiring replicants: Congestion control for intermittently - connected networks. In: **INFOCOM'2010, IEEE**. [S.l.: s.n.], 2010. p. 1–9. University of Illinois at Urbana-Champaing, Department Of Computer Science.

THRUN, S. The role of exploration in learning control. In: WHITE, D.; SOFGE, D. (Ed.). **Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches**. Florence, Kentucky 41022: Van Nostrand Reinhold, 1992.

UTHER, W. T. B.; VELOSO, M. M. **Generalizing Adversarial Reinforcement Learning**. [S.l.], 1997.

VAHDAT, A.; BECKER, D. **Epidemic Routing for Partially-Connected Ad Hoc Networks**. [S.l.], 2000.

VAHDAT, A.; BECKER, D. **Epidemic Routing for Partially Connected Ad Hoc Networks**. [S.l.], July 2000.

VIRTAMO, E. H. J.; LASSIL, P.; OTT, J. Continuum percolation threshold for permeable aligned cylinders and opportunistic networking. **IEEE COMMUNICATIONS LETTERS**, v. 16, n. 7, 2012.

VOYIATZIS, A. G. A survey od delay and disruption tolerant networking applications. **Internet Engineering**, v. 5, n. 1, 2012.

WANG, C.; ZHAO, B.; PENG, W.; WU, C.; GONG, Z. Following routing: An active congestion control approach for delay tolerant networks. In: **2012 15Th International Conference on Network-Based Information Systems**. [S.l.: s.n.], 2012. p. 727–732.

WANG, C.; ZHAO, B.; YU, W.; WU, C.; GONG, Z. SARM: An congestion control algorithm for dtn. In: **9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing**. [S.l.]: IEEE Computer Societ, 2012. p. 869–875.

WANG, R.; TALEB, T.; JAMALIPOUR, A.; SUN, B. Protocols for reliable data transport in space internet. **IEEE Communications Surveys & Tutorial**, v. 11, n. 2, 2009.

WATKINS, C. J. C. H. **Learning from Delayed Rewards**. Tese (Doutorado) — King's College, Cambridge, UK, May 1989. Disponível em: <http://www.cs.rhul.ac.uk/˜chrisw/new_thesis.pdf>.

WATKINS, C. J. C. H.; DAYAN, P. Q-learning. **Machine Learning**, v. 8, n. 3-4, p. 279–292, 1992.

WYDROWSKI, B. P. **Techniques in Internet Congestion Control**. Tese (Doutorado) — Electrical and Electronic Engineering Department, The University of Melbourne, February 2003.

XU, Y.; WANG, X. Fundamental lower bound for node buffer size in intermittently connected wireless networks. In: **INFOCOM2011**. [S.l.: s.n.], 2011. p. 972–980.

YADAV, A. K.; SHRIVASTAVA, S. K. Evaluation of reinforcement learning techniques. In: **Proceedings of the First International Conference on Intelligent Interactive Technologies and Multimedia**. New York, NY, USA: ACM, 2010. (IITM '10), p. 88–92.

YIN, L.; LU hui-mei; CAO, Y. da; GAO, J. min. An incentive congestion control strategy for DTNs with mal-behaving nodes. In: . [S.l.: s.n.], 2010. p. 91–94.

YUN, L.; XINJIAN, C.; QILIE, L.; XIANOHU, Y. A novel congestion control strategy in delay tolerant networks. In: **Second International Conference on Future Networks**. [S.l.: s.n.], 2010.

ZEEPHONGSEKUL, P.; BEDFORD, A.; BROBERG, J.; DIMOPOULOS, P.; TARI, Z. Queuing theory applications to communication systems: Control of traffic flows and load balancing. **Springer Handbook of Engineering Statistics**, 2006.

ZHANG, G.; LIU, Y. Congestion management in delay tolerant networks. In: **WICON**. [S.l.: s.n.], 2008. p. 1–9.

ZHANG, L.; CAI, L.; PAN, J. Connectivity in two-dimensional lattice networks. In: **INFOCOM, 2013 Proceedings IEEE**. [S.l.: s.n.], 2013. p. 2814–2822.

ZHANG, L.; ZHOU, X. Hop-by-hop dynamic congestion control with contact interruption probability for intermittently connected deep space information networks. **Wireless Personal Communication**, p. 399–424, 2013.

ZHANG, W.; DIETTERICH, T. G. A reinforcement learning approach to job-shop scheduling. In: **In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence**. [S.l.]: Morgan Kaufmann, 1995. p. 1114–1120.

ZHANG, Z. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges. **Communications Surveys & Tutorials, IEEE**, v. 8, 2006.

Appendix A - Exponetially Weighted Moving Average

A.1 Exponentially Weighted Moving Average - EWMA

First introduced by Roberts (ROBERTS, 1959) it is a way to continuously compute a type of average for a series of numbers, as the numbers arrive. After a value in the series is added to the average, its weight in the average decreases exponentially over time. This biases the average towards more recent data. EWMAs are useful for several reasons, mainly their inexpensive computational and memory cost, as well as the fact that they represent the recent central tendency of the series of values. The EWMA also is known to have optimal properties in some forecasting and control applications (BOX *et al.*, 1974) (BOX; JENKINS, 1990) (NELSONA *et al.*, 2009). In this work we focus on the predictability of DTN node's buffer occupancy rate in a DTN.

The primary purpose of tracking the buffer occupancy rate of node is to intelligently decide how to mitigate DTN congestion. To track a node's buffer occupancy rate, every node maintains two pieces of local information: a buffer occupancy rate value (BORV) and a current buffer occupancy rate value (CBORV). BORV represents the node's past rate of buffer occupancy as an exponentially weighted moving average, while CBORV is used to obtain information about the buffer occupancy rate in the current time interval. BORV is periodically updated to account for most recent CBORV in which rate of buffer occupancy information was obtanined. Updates to BORV are computed as follows by Equation A.1.

$$BORV = \alpha CBORV = (1 - \alpha)BORV \tag{A.1}$$

The EWMA algorithm requires a decay factor α. The large α, the more average is biased towards recent history. The α must be between 0 and 1. In this work we use different values of α.

This exponentially weighted moving average places an emphasis proportional to α on the most recent CBORV. Updating CBORV is straightforward: for each time step, the CBORV is

calculated based on the number of messages in the node's buffer at time step t.

Since BORV represents a prediction of the future rate of buffer occupancy for each node per time step, the node can know if its buffer occupancy rate is increasing or decreasing. This is equivalent to positive and negative derivative respectively.

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO	2. DATA	3. DOCUMENTO Nº	4. Nº DE PÁGINAS
TD	25 de março de 2015	DCTA/ITA/TC-018/2015	175

5. TÍTULO E SUBTÍTULO:

A Novel Congestion Control Framework For Delay and Disruption Tolerant Networks

6. AUTOR(ES):

Aloizio Pereira da Silva

7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES):

Instituto Tecnológico de Aeronáutica – Divisão de Engenharia Eletrônica e Computação – ITA/IEC

8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:

Computação; Redes; Performance

9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:

Computação; Redes; Performance

10. APRESENTAÇÃO: **(X) Nacional** ( ) **Internacional**

ITA, São José dos Campos. Curso de Doutorado. Programa de Pós-Graduação em Engenharia Eletrônica e Computação. Área de Informática. Orientador: Prof. Dr. Celso Massaki Hirata. Coorientadora: Prof[a]. Dr[a]. Katia Obraczka. Defesa em 05/07/2015. Publicada em 25/07/2015.

11. RESUMO:

Redes tolerantes a atrasos e desconexões, também conhecidas como DTNs (*Delay and Disruption Tolerant Networks*) são redes capazes de lidar com atrasos longos e desconexões intermitentes, diferentemente das redes tradicionais, tais como Internet TCP/IP. Outra característica que diferencia DTNs das redes convencionais é que não existe nenhuma garantia de uma conectividade fim-a-fim entre fonte e destino. Essas características apresentam uma série de desafios técnicos no projeto de funções chaves da rede tais como mecanismos de roteamento e mecanismos de controle de congestionameto. Detectar e tratar o congestionamento em DTN é particularmente um desafio e extremamente importante no contexto de performance da rede. Atualmente o controle de congestionamento fim-a-fim em redes tradicionais é tratado pelo TCP (Transmission Control Protocol) que previne a rede de entrar em colapso mas não evita que o desempenho da rede degrade quando em presença de congestionamento. TCP/IP funciona bem quando não existe interrupção da comunicação fim-a-fim. Em particular, DTN endereça alguns ponto fracos do TCP, por exemplo implementando mecanismos de controle de congestionamento onde cada roteador pode tomar decisão considerando a informação local (e.x. aceitando dados de outro roteador). Essa informação local pode incluir disponibilidade de armazenamento e o risco de aceitar os dados baseado em experiências anteriores as quais podem ter tido efeitos adverso na rede resultando por exemplo em perda de dados. Mecanismos de controle de congestionamento DTN existentes tipicamente tentam usar alguma informação global da rede ou foram projetados para operar em um cenário especifico sujeito a uma estratégia especifica de encaminhamento de dados. Como resultado, estes mecanismos não tem boa performance quando eles são usados em diferentes cenários com diferentes protocolos de roteamento.

Neste trabalho, nós primeiramente revisamos os desafios em DTNs e exploramos o dominio dos mecanismos de controle de congestionamento nestas redes. Além disso, nós propomos uma taxonomia para classificar mecanismos de controle de congestionamento DTN discutindo pontos fracos e fortes no contexto de suas suposições e aplicabilidade. Nós também apresentamos uma analise quantitativa de alguns mecanismos de controle de congestionamento DTN para avaliar como eles se comportam no cenário de comunicação espacial considerando que eles foram projetados para operar em cenários terrestres. Nós avaliamos estes mecanismos extensivamente através de simulação, usando duas aplicações diferentes e três protocolos de roteamento e padrões de mobilidade. Os resultados mostraram que os mecanismos selecionados não tiveram boa performance no cenário espacial. Portanto, do ponto de vista das caracteristicas DTN, para estudar novos mecanismos de controle de congestionamento e entender o impacto do congestionamento em DTN nós modelamos o problema de congestionamento DTN usando a teoria da percolação. Nós deixaremos mais claro ao longo deste trabalho que a formulação do problema de congestionamento DTN como um processo de percolação acontece naturalmente e o modelo de percolação resultante é simples e facil de derivar. Além disso, outra caracteristica importante do modelo de percolação proposto é o fato de que ao invés de requerer o uso de informação global da rede, ele depende exclusivamente de informação local, por exemplo, informação relacionada ao nó e aos seus vizinhos dele. A principal vantagem do nosso modelo matematico é permitir validar experimentos reais e simulação. Consequentemente, o modelo proposto pode ser usado para predizer e controlar congestionamento em DTNs.

Dado que longe das redes tradicionais, DTN é um novo tipo de rede derivada de pesquisas realizadas no contexto de comunicação espacial e considerando a importância do controle de congestionamento que diretamente afeta a performance da rede. O desenvolvimento de redes DTN deve depender de um mecanismo de controle de congestionamento que garanta confiabilidade, estabilidade e a extensão da rede. Objetivando melhorar a confiabilidade da entrega de mensagens nestas re-

12. GRAU DE SIGILO:

(X) **OSTENSIVO** ( ) **RESERVADO** ( ) **CONFIDENCIAL** ( ) **SECRETO**