A MESSAGE TRANSFER FRAMEWORK FOR ENHANCED RELIABILITY IN DELAY-AND DISRUPTION-TOLERANT NETWORKS

FARZANA YASMEEN

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE AND ENGINEERING
YORK UNIVERSITY
TORONTO, ONTARIO

SEPTEMBER 2015

Abstract

Many infrastructure-less networks require quick, ad hoc deployment and the ability to deliver messages even if no instantaneous end-to-end path can be found. Such networks include large-scale disaster recovery networks, mobile sensor networks for ecological monitoring, ocean sensor networks, people networks, vehicular networks and projects for connectivity in developing regions such as TIER (Technology and Infrastructure for Emerging Regions) [38]. These types of networks can be realized with delay-and disruption-tolerant network (DTN) technology [28]. Generally, messages in DTNs are transferred hop-by-hop toward the destination in an overlay above the transport layer called the "bundle layer". Unlike mobile ad hoc networks (MANETs), DTNs can tolerate disruption on end-to-end paths by taking advantage of temporal links emerging between nodes as nodes move in the network. Intermediate nodes store messages before forwarding opportunities become available. A series of encounters (i.e., coming within mutual transmission range) among different nodes will eventually deliver the message to the desired destination.

The message delivery performance (such as delivery ratio and delay) in a DTN highly depends on time elapsed between encounters (inter-contact time) and the time two

nodes remain in each others communication range once a contact is established (contact-duration). As messages are forwarded opportunistically among nodes, it is important to have sufficient contact opportunities in the network for faster, more reliable delivery of messages.

In this thesis, we propose a simple yet efficient method for increasing DTN performance by increasing the contact duration of encountered nodes (i.e., mobile devices). Our proposed sticky transfer framework and protocol enable nodes in DTNs to collect neighbors' information, evaluate their movement patterns and amounts of data to transfer in order to make decisions of whether to "stick" with a neighbor to complete the necessary data transfers. Nodes intelligently negotiate sticky transfer parameters such as stick duration, mobility speed and movement directions based on user preferences and collected information. The sticky transfer framework can be combined with any DTN routing protocol to improve its performance. Our simulation results show that the proposed framework can improve the message delivery ratio by up to 38% and the end-to-end message transfer delay by up to 36%.

Acknowledgements

I would like to thank:

My supervisor, Prof. U.T., for providing me every opportunity and resources to move forward with my research work. I am grateful for her patient guidance; the valuable time she spent for numerous discussions to progress my work and for reading and correcting my articles many times to make them acceptable for submission.

The members on the thesis advisory committee, Prof. Sebastian Magierowski, and Prof. Baoxin Hu, for their time, valuable suggestions and comments to improve the quality of this thesis.

Prof. Shigeki Yamada at the National Institute of Informatics (NII), Tokyo, for introducing me to the world of DTNs, for helping me envision this work, and teaching me about the systematic ways of doing research.

Prof. Cristian Borcea at the New Jersey Institute of Technology (NJIT), New Jersey, for providing valuable insight and suggestions on the sticky transfer framework.

Prof. Nurul Huda, Ryerson University, Toronto, for his time to discuss and reason various aspects of my work and develop parts of the code for the sticky transfer framework.

The members on the DTNRG and TheONE mailing list for responding to inquiries on various questions related to DTN and the ONE simulator.

York University and the Faculty of Graduate Studies for their financial support.

My fellow lab members for their suggestions and support regarding both research and life in general. I wish each of them success in their on-going and future endeavors.

To my parents, and family members for their encouragement and understanding.

Last but not least, I would like to acknowledge the main source of my inspiration, my two troopers, Adrian and Alex. Their endurance, co-operation, and support kept me on track to complete this work.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The continuously increasing popularity of devices equipped with wireless network interfaces has introduced new demands in wireless access technology trends. An increasing number of mobile device users today want affordable communication capabilities anytime, everywhere. Advent of short range wireless communication technologies, such as infrared, Bluetooth and Wi-Fi have brought about the concept of opportunistic networks [1, 2], where information can be stored and passed by taking advantage of device mobility, or forwarded over a wireless link when an appropriate contact is met. Opportunistic networks can be realized by delay-tolerant networks (DTNs) [3].

The use of device mobility for data transmission over intermittent links requires fundamentally different networking protocols than those in mobile ad-hoc networks (MANETs) [4], which rely on contemporaneous connectivity between source and destination endpoints. In sparse, mobile, infrastructureless environments, frequent network partitions and link instability render conventional MANET routing protocols [5–7] ineffective, due to high

numbers of route requests, timeouts in the underlying transmission protocol and most importantly, the requirement of a-priori end-to-end connections between the source and final destination before data transfers. DTN protocols [8–16], on the other hand, can handle long, frequent, and intermittent link conditions in the network by using the store-and-forward mechanism [68] and opportunistic contacts for routing, given that the sequence of connectivity graphs over a time interval forms a virtual end-to-end path between a source and a destination. The tradeoff is generally high delivery latency.

Suitable applications for DTNs are non-real time, such as file transfer, sensor data collection, and buffered messaging. Examples of DTNs in practice include (but are not limited to) sensor-based networks using scheduled intermittent connectivity, terrestrial wireless networks that cannot ordinarily maintain end-to-end connectivity, satellite networks with moderate delays and periodic connectivity, underwater acoustic networks with moderate delays and frequent interruptions due to environmental factors, vehicular networks with cyclic but non-deterministic connectivity, disaster recovery situations, rural area communication scenarios, wildlife monitoring systems, and battlefield operations.

1.1 Motivations

The message delivery performance (such as delivery ratio and end-to-end delay) in a mobile DTN highly depends on the time elapsed between encounters (the *inter-contact time*) and the time two nodes remain in each others communication range once a contact is established (*contact duration*) as node contacts are opportunistic and limited. Node

mobility may cause nodes to move out of each others transmission range in the middle of a transmission, interrupting the transmission and wasting the resource consumed by the failed transfer. In addition, many other messages which have been processed and ready for transmission cannot be forwarded. These messages will stay longer in buffers of limited sizes, which may eventually be discarded due to buffer overflow, wasting node resources. The end result is low message delivery ratio and long end-to-end delay. The above problems are exacerbated in DTNs with highly mobile nodes that must handle large messages, such as vehicular networks [17, 18].

Inter-contact time depends primarily on node mobility and node density in the network. In sparse networks, the inter-contact time can be reduced by introducing special components, such as ferries [19] or data mules [20], that move at relatively faster speeds on predefined routes and therefore increase contact opportunities.For example, in disaster recovery operations, such as those performed after earthquakes or tsunamis, where communication infrastructure has been damaged, mobile devices in that area may use a temporarily deployed DataMule [20] to carry data to and from a sink remotely connected to the Internet.

The contact duration is the length of time during which pair-wise nodes remain within the transmission range of each other in the network. The contact duration directly influences the capacity of opportunistic networks (e.g., DTNs) as it limits the amount of data that can be transferred successfully between nodes.

The performance of DTNs can be improved by reducing the inter-contact time and

increasing the contact duration. There exist routing algorithms that can reduce the inter-contact time of nodes [19, 20]. However, irrespective of the forwarding technique used by a routing algorithm, the actual time to transfer messages between two nodes is limited by the contact duration of the nodes, which depends inherently on node mobility.

Thus, when a node is forwarding a message to a neighbor node, if the expected contact duration between the two nodes is not sufficient for the entire message to be transmitted (e.g., due to node mobility), the message transfer will fail. This may cause the transport protocol (e.g., TCP) to retransmit the message, thus wasting valuable bandwidth, resources and opportunistic contacts in the network.

One way to tackle the above problem is to transmit smaller sized messages, i.e. message fragmentation, to enable successful forwarding within shorter contact durations. However, as we will discuss in Section 2.4.3, message fragmentation may not lead to an optimal routing solution. A small (large) fixed-length fragmentation size can waste valuable bandwidth during a contact opportunity, as smaller (larger) fragments can under-utilize (exceed) the bandwidth. Also, assuming a message m is split into n fragments, these n fragments could, in the worst case, take n possible time-varying paths to reach the final destination. Thus, reconstruction of message m at the destination can become increasingly difficult with a growing number of fragments, as fragments can be lost while being forwarded to the final destination due to being dropped from buffers or because of transmission errors. Also, some fragments of the message may not reach the destination in a timely fashion, and can cause the TTL (time-to-live) of message m to expire. Zhuo

et al. [22] propose a packet-level replication protocol, which uses erasure coding to encode large messages into smaller packets, to address the problem of limited contact duration. However, this technique requires replication of packets at each node, which is expensive in a DTN.

Our research objective is to present a solution for maximizing node contacts for message transfers in delay-tolerant networks. The goal is to assist routing protocols by providing a solution that can be implemented, irrespective of the routing protocol being used, to maximize utilization of the contact duration in order to meet the required message transfer duration. Our proposed framework requires no additional mechanism nor infrastructure other than the simple beaconing mechanism [80] which has been used for many other purposes in wireless ad hoc networks. To the best of our knowledge, our work is the first to propose the concept of 'sticky' message transfers to extend contact durations.

In the proposed sticky message transfer framework, nodes send out periodic beacons for neighbor discovery. Once a neighbor is detected with which a node can perform sticky transfers, the nodes exchange information such as their mobility speed and direction, current location, transmission range, available buffer size, the amount of data to be sent and the final destination, using our proposed sticky transfer protocol within the framework. Based on the received information, a node A calculates the needed contact duration based on the amount of data to be exchanged with or transferred to a neighbor B, determines the required mode of movement (e.g., slowing down or stopping in order

to stay in contact), and negotiates an agreement to stick with B (e.g., negotiating the mode of movement and contact duration). After the sticky transfer is over, nodes A and B resume their original movement behavior.

Sticky transfers can be used to improve the network performance of many applications. For example, robots in a region survey application may be programmed to stick with each other longer when needed to improve message delivery ratio and delay. Emergency response team members could be asked to stop or follow each other when necessary to improve the network performance. A network of mobile sensors engaged in ecological monitoring could use sticky transfers to enable faster message delivery to the sink.

The sticky transfer mechanism is optional; nodes may choose not to run the protocol or ignore sticky transfer requests from other nodes.

In the remainder of this thesis, *"messages"* denotes any type of payload content, including but not limited to text, audio, video, and image. We also use the terms "messages", "packets" and "bundles" interchangeably.

1.2 Methodology

The proposed sticky transfer scheme requires temporary changes in movements of nodes upon sticky transfer agreements. It is useful where the required change in node movement can be applied, such as in human-and robot-assisted mobility scenarios. As changing node movement depends upon user cooperation, we implement a user's agreement to modified movement by defining a list of user preferences for each user in the network.

A user's preference consists of an ordered list of acceptable sticky modes. The order defines the priority of user preferences, with higher priority modes coming first in the list. Sticky modes set in preferences represent how the users would respond to a sticky transfer request. We define five sticky modes: *Stop (STP), Follow me (FLW1), Follow you (FLW2), Slow down (SLW)* and *No stick (NO_STK)*. Users can set one or more modes in their preference settings to be used for implementing sticky transfer decisions.

Furthermore, sticky transfers are technically possible when the preferences of two nodes are compatible. For example, FLW1 and FLW2 are compatible (i.e., one node agrees to follow the other), while STP and FLW1 are not compatible (i.e., one node wants to stop while the other node wants to be followed). Based on the five proposed sticky modes, we design rules for compatibility between any two nodes. These rules govern sticky transfer compatibility based on user preference settings. For example, if both user's preference allows STP, then the modes are compatible and they can engage in sticky transfers. Also, two modes may or may not be compatible based on their speed limitations and movement direction. For example, when node A allows SLW and B allows $FLW1$ then they are compatible if B's speed is slower than A's speed. They are not compatible if B's speed is faster than A's speed or they are not moving in the same direction.

We also propose the *sticky transfer protocol* which defines a sequence of operations for the sticky transfer of messages. This protocol involves several steps before the actual message transfer, such as calculating the expected contact duration between nodes A and

B, determining compatible stick modes between the nodes, determining messages to send and receive (exchange), and calculating the stick time between the nodes.

To complete the framework we show a conceptual overview of the sticky transfer protocol and components integrated in a layered DTN architecture. To evaluate the effectiveness of the proposed sticky transfer framework, we performed simulations using a city-based network topology and the Spray-and-Wait [10], PRoPHET [9], and Epidemic [8] opportunistic routing protocols. We evaluated the performance of each routing protocol with and without sticky transfers. We performed simulations on the Opportunistic Network (ONE) simulator, a simulation environment capable of routing messages between nodes using various DTN routing algorithms [30].

1.3 Contributions

In summary, our contributions include:

1. A novel framework called the sticky transfer framework that enables mobile nodes in a DTN to modify their mobility and prolong contact durations to enhance successful message transfers.

2. A sticky transfer protocol within the framework that governs how two mobile nodes will "stick" to each other for a negotiated period of time in order to complete the transmission of messages.

We also demonstrate the seamless integration of the sticky transfer framework with the existing DTN network management modules. Sticky transfers can increase the delivery

ratio and shorten delivery delay by ensuring faster forwarding and reducing the number of message transfer aborts. Our framework is especially beneficial for large message sizes and/or high mobility situations which lead to short contact time between nodes and thus low message delivery ratios.

Our simulation results show a significant improvement in the performance of the routing protocols in the presence of sticky transfers. The message delivery ratio is increased by as much as 38%, while the end-to-end delay is shortened by as much as 36% with sticky transfers.

1.4 Organization of the Thesis

The remainder of the thesis is organized as follows. We discuss background information and related work in Chapter 2. We provide an overview of the state-of-the-art in DTNs, and a review of issues in DTNs such as routing, replication scheme trade-offs, and delay characteristics. In Chapter 3, we describe our proposed sticky transfer framework and sticky transfer protocol. In Chapter 4, we present simulation results and an analysis of our proposed framework and protocol. We conclude the thesis in Chapter 5 with a summary of our findings and an outline of future work.

Chapter 2

Background and Related Work

Research on *delay-and disruption-tolerant networks* (a.k.a., *delay-tolerant networks*) (DTNs) and subsequently the standardization of the DTN architecture [43] stems from two different areas: (i) interplanetary networks where interoperability among dissimilar proprietary protocols is necessary; and (ii) 'challenged' terrestrial networks [46] where sufficient infrastructure is not available. Hence existing reliable end-to-end protocols such as, TCP/IP-based solutions, and wireless local area network (WLAN) solutions can not apply. Furthermore, due to the lack of any end-to-end connectivity between the source and destination of a packet and the intermittent nature of the path between a source and a destination in a DTN, traditional mobile ad-hoc network (MANET) routing protocols [4], [5], [6], [7] can not be used either.

In this chapter, we look into the background of DTNs, discuss the DTN architecture, and the DTN bundle protocol (BP), which is the central component of the DTN overlay layer. We then discuss routing in a DTN by presenting several DTN routing protocols

and distinguish DTN routing form MANET routing. Finally, we state issues related to our thesis objective and show existing related work.

The DTN architecture provides an extra layer between an application and a network, allowing data to be sent reliably over intermittent or long-haul links. It can be seen as an overlay on top of a number of diverse regional networks, including the Internet. Within a DTN, the regional networks may have varying connectivity, delay and loss characteristics, and may employ different lower-layer technologies. The DTN overlay accommodates these different network characteristics and provides a service that works regardless of the underlying network environment.

Before we present an overview of the DTN architecture, we will briefly review the Internet architecture to make the comparison between the two architectures more clear and to identify why a new DTN architecture is necessary.

2.1 Overview of the TCP/IP Protocol Suite

The Internet interconnects communication devices across the globe by using a homogeneous set of communication protocols, called the TCP/IP protocol suite. Communication on the Internet is based on packet switching. Packets are pieces of a complete block of user data (e.g., pieces of an email message or a web page) that travel independently from source to destination through a network of links connected by routers. The source, destination, and routers are collectively called 'nodes'.

Each packet that makes up a message can take a different path through the network.

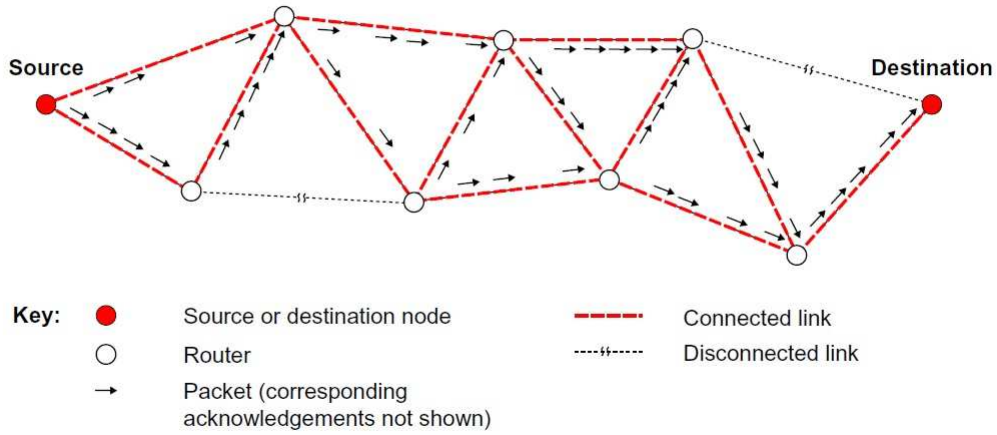


Figure 2.1: Packet-switching network

If one link is disconnected, packets take another link (Fig. 2.1). Packets contain both application-program user data (the payload part) and a header (the control part). The header contains a destination address and other information that determines how the packet is switched from one router to another. The packets in a given message may arrive out of order, but the destinations transport mechanism reassembles them in correct order.

The usability of the Internet depends on some important assumptions [51]:

- *Continuous, Bidirectional End-to-End Path:* A continuously available bidirectional connection between source and destination to support end-to-end interaction.

- *Short Round-Trips:* Small and relatively consistent network delay in sending data packets and receiving the corresponding acknowledgement packets.

- *Symmetric Data Rates:* Relatively consistent data rates in both directions between source and destination.

- *Low Error Rates:* Relatively little loss or corruption of data on each link.

Messages are moved through the Internet by protocol layers, a set of functions performed by network nodes on data communicated between nodes. Hosts (computers or other communicating devices that are the sources or destinations of messages) usually implement at least five protocol layers, which perform the following functions:

- *Application Layer:* Generates or consumes user data (messages).

- *Transport Layer:* Source-to-destination (end-to-end) segmentation of messages into message pieces and reassembly into complete messages, with error control and flow control. On the Internet, the Transmission Control Protocol (TCP) is used.

- *Network Layer:* Source-to-destination routing of addressed message pieces through intermediate nodes, with fragmentation and reassembly if required. On the Internet, the Internet Protocol (IP) [58] is used.

- *Link Layer:* Link-to-link transmission and reception of addressed message pieces, with error control. Common link-layer protocols include Ethernet for Local-Area Networks (LANs) and Point-to-Point Protocol (PPP) for dial-up modems or very high-speed links.

- *Physical Layer:* Link-to-link transmission and reception of bit streams. Common

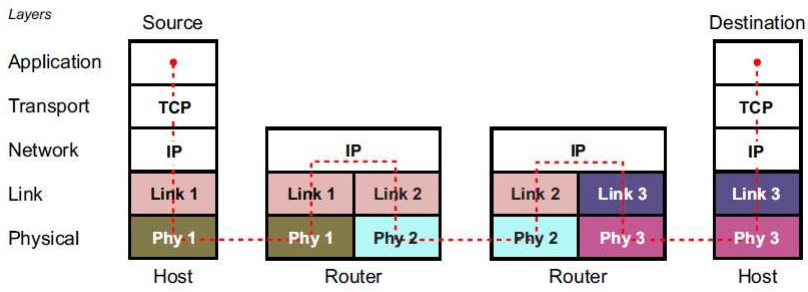


Figure 2.2: Routing in the Internet with TCP/IP

physical media include category 5 (cat5) cable, unshielded twisted pair (UTP) telephone cable, coaxial cable, fiber-optic cable, and RF.

Figure 2.2 shows the basic mechanism or routing in the Internet. Routers implement only the lower three protocol layers. However, routers also implement the higher layers for routing-table maintenance and other management purposes. Each hop on a path can use a different link-layer and physical-layer technology, but the IP protocol runs on all nodes and the TCP protocol runs only on source and destination end points. Several other Internet protocols and applications are also used to provide routing-path discovery, path selection, name resolution, and error recovery services.

The term *packet* is applied to the objects actually sent over the physical links of a network. They are often called IP packets because the IP protocol, the only protocol used by all nodes on the path, is primarily responsible for directing them, node-by-node, from source to destination along their entire path.

Packets consist of a hierarchy of data-object encapsulations that are performed by the

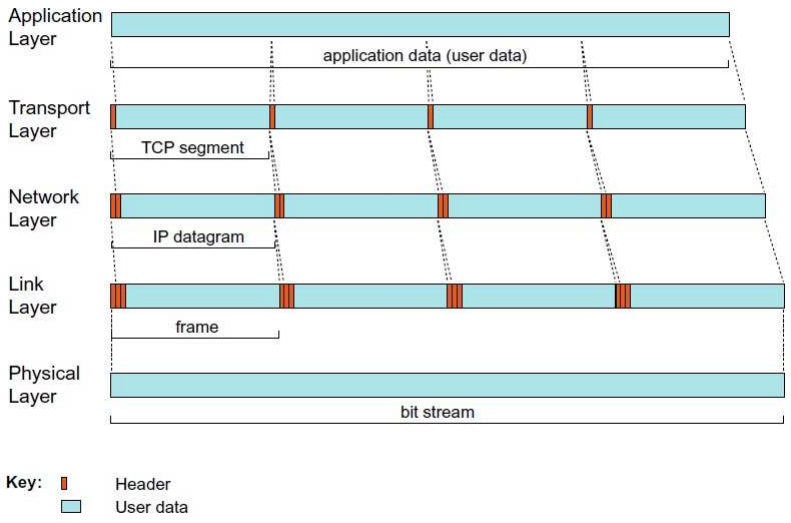


Figure 2.3: Data Encapsulation in the Internet

protocol layers (see Figure 2.3). During transmission, higher-level data and its header are encapsulated in a lower-layer data object, which is given its own header. The headers are used by their respective protocol layers to control the processing of the encapsulated data. Successive headers are added at the source as user data moves down the protocol stack from application to physical layer. Headers are removed at the destination end as data moves up the stack to the destination application.

TCP breaks user data into pieces called *segments*. IP encapsulates the TCP segments into datagrams, and it may break the segments into pieces called fragments (not shown in the figure below). The link-layer protocol encapsulates IP datagrams into frames. The physical layer then transmits and receives a sequence of frames as a continuous bit stream [51].

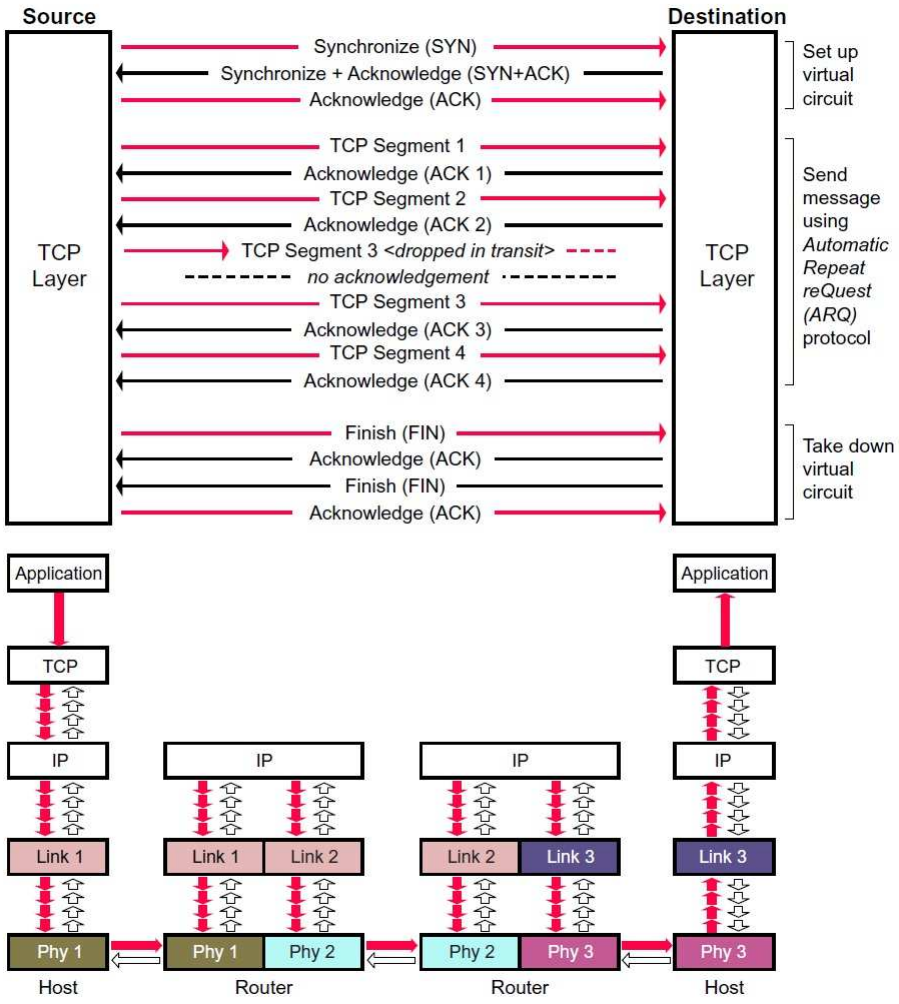


Figure 2.4: TCP connection establishment

The TCP protocol is said to be conversational (interactive), because a complete one-way message involves many source-to-destination signaling round-trips:

- Set Up: A three-way "Hello" handshake.
- Segment Transfer and Acknowledgement: Each TCP segment (or a few segments) sent by the source is acknowledged by the destination.

- Take Down: A four-way "Goodbye" handshake.

The use of positive or negative acknowledgments to control retransmission of lost or corrupt segments is called an Automatic Repeat reQuest (ARQ) [60]. The TCP 3-way handshake protocol to establish a connection between the source and destination nodes is depicted in Figure 2.4.

The most common protocols used in the Internet today, e.g. TCP and UDP, are designed for a network with continuous end-to-end paths between source and destination, with relatively high bandwidth and low delays, providing for short round-trip times of packets. However, as discussed in Section 2.2.1 and Section 2.2.2, there exists challenged applications where a nodes or links may be unavailable for very long periods of time, generating long delays for data waiting to be sent. Besides sporadic loss of connection, a fluctuating link can also cause high error rates in transmissions when data gets dropped or not transferred correctly. Long delays and intermittent connectivity between nodes causes problems for ordinary transport protocols, especially connection-oriented ones such as TCP, that need to establish a connection with the destination before sending data. The problem of a TCP session timing out caused by too long delays is also an issue when dealing with asymmetric data rates [51].

The DTN architecture address these concerns with highly challenged networks in which application layer"sessions" lack contemporaneous end-to-end connectivity and are not possible to address using the traditional Internet architecture. It provides a means for transporting data across dissimilar lower-layer protocols with an overlay between the ap-

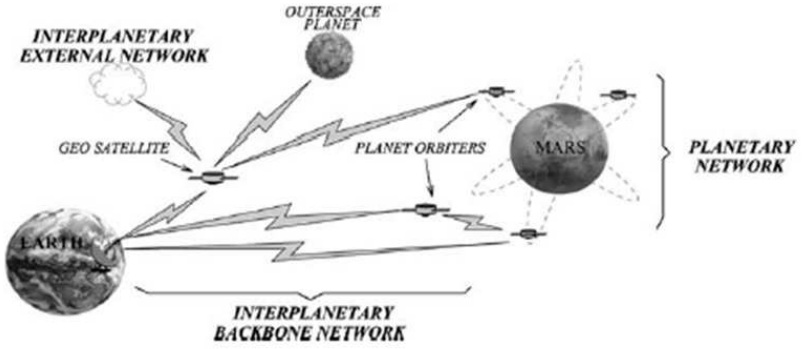


Figure 2.5: Interplanetary network

plication and network layer, and uses a store-carry-forward mechanism to store messages on intermediate hops for extended durations to address intermittent connectivity.

In the next section we discuss the origination of delay-tolerant networks before proceeding to discuss the architecture in detail.

2.2 History of Delay-Tolerant Networks (DTNs)

Research on DTNs and subsequently the standardization of the DTN architecture stems from two different network and data communication areas: (i) interplanetary networks; and (ii) 'challenged' terrestrial networks. Following are their contributions to the origination of DTNs.

2.2.1 Interplanetary Networks

In early 1998, the DARPA Next Generation Internet initiative funded the InterPlaNetary Internet (IPN) Project [49], [53], [54], [55] with the aim to define a communication

system together with Internet-like services across interplanetary distances. The project was originally conceived to support deep space exploration with the vision of a stable interplanetary backbone network among planets, satellites, asteroids, robotic spacecrafts, and possible crewed vehicles, which later gave rise to other new challenging applications from the mining of asteroids to the deployment of space-based hotels for tourism support (see Fig. 2.5).

The case of interplanetary networks (IPN) is an extreme scenario where long distances are involved between communicating nodes (e.g., satellites, base stations), the level of noise is substantial and the node distribution is sparse. Communications in deep space environments are characterized by [56]:

- *High propagation delays and round trip times:* distances between planets are enormous and lead to propagation delays of tens of minutes of magnitude (the round trip time at the speed of light, between Earth and Mars, for example, ranges between 8 and 40 minutes). This makes unfeasible utilization of transport protocols that rely on long handshakings between peer nodes (for example TCP) because they would take too long to complete a single data transfer.

- *Low data rates:* the radio signal easily degrades and attenuates over long distances.

- *Intermittent connectivity:* a complete path between the communicating parties is likely not to be available due to the movements of the planets, the occultation of satellites during planet-orbiting, and so on. In addition, distances between planets

change over time and cause variations in delay, transmission capacity, connectivity, and topology. Intermittent connectivity results in long periods of network partitioning, and discontinuities in the capabilities of adjacent networks.

- *Asymmetric links:* transmissions from Mars to Earth, for example, may be received at 100 kilobits/second while Mars-based systems may only receive from Earth at 1 kilobit/second.

- *High bit error rate links:* links are error-prone.

- *Low available bandwidth:* data rates range from hundreds of kilobits per second to few megabits per second and will probably remain unmodified in the next few decades. This is due to the combined effect of large distances, expense and difficulty in deploying large antennas to distant planets, and difficulty in generating power in space.

- *Need for special equipment:* transmissions are generally very expensive over the deep space because they need special instrumentation like large antennas, high-performance receivers, etc. This further worsens the end-to-end delay experienced during transmissions because of the sharing of Earth-based resources that introduces scheduling and queuing delays. An efficient use of the communication channel allows more information to be carried per unit of transmitted power.

Transmission distances for deep space data communication currently span from Earth-orbiting satellites to the Mars exploration mission, with no intermediate nodes for data

store-and-forwarding [47]. The basic idea of an IPN is to try to make data communications between Earth and (very) remote spacecraft seem almost as easy as that between two people on different sides of the world. As it happens, before a network node can send any application data using the Transmission Control Protocol (TCP) [48], a three-way handshake is required that consumes 1.5 round-trip times (RTTs). Theres also a generic, two-minute timeout implemented in most TCP stacks: if no data is sent or received for two minutes, the connection breaks. Putting these facts together, its easy to see that once a spacecraft is more than a minute away (in terms of light-trip time), every attempt to establish a TCP connection will fail, and no application data will ever be transmitted. In the case of Mars, for example, at its closest approach to Earth, the RTT is roughly eight minutes, with a worst-case RTT of approximately 40 minutes. Thus, normal TCP cant work at all for Earth to Mars communications [50]. The level of space and radiation noise adds another obstacle to reliable communication, as does the intermittent connectivity when there is no line of sight between the communication entities or when the interference duration is beyond temporary [47].

It becomes clear from the above characteristics and observations on communications in deep space that *non-chatty* communication protocols, which use paradigms similar to the Postal and Pony Express systems [57], are more suitable for deep space environment than protocols such as TCP. In addition, more suitable protocols for deep space communication should also pack as much data as possible per single transmission to minimize the number of round-trips needed. For example, a protocol for file transfers should send

an entire file with the relative control data all together in a single atomic transaction. The total transmission would thus end faster within the IPN and make more efficient use of terrestrial high-speed networks [56].

Furthermore, until recently, it has been common practice among space missions to: (i) operate each spacecraft in isolation of others via direct-to-Earth communication links, (ii) manually schedule, review and revise communication contacts between all spacecraft and ground station antennas on Earth, (iii) inspect incoming data as it arrives and sends commands to spacecraft in order to request retransmission of data either missing or received in error, and (iv) often deploy customized and incompatible protocol stacks tailored to accommodate each mission's specific needs. This anachronistic model of operation poses significant limitations on total communication time provided to spacecraft, requires tedious and error-prone human intervention and, essentially, forces mission designers to "reinvent the wheel" every time an ad hoc communication system and protocol stack needs to be deployed on some spacecraft. Likewise, space communications rely heavily on proprietary protocols and confidential products, which often leads to incompatible communication protocols among different agencies.

To address the challenges and problems mentioned earlier, the Delay-Tolerant Networking (DTN) architecture [43] appeared as an emergent solution to automated space inter-networking and quickly gained wide acceptance in the space community [32]. The DTN architecture emerging from the IPN project consists of a network of internets, i.e., a collection of independent internets eventually interconnected by a system of (IPN) gate-

ways. Single internets are located apart from each other (e.g., on the surface of planets or satellites, or in spacecrafts) forming distinct (IPN) regions. The internets are independent to each other in that each one has its own protocols to rely on. Protocols are chosen to best suit the particular infrastructure, communication means, and technologies available in the particular internets region and they may differ from the protocols of the other IPN regions internets. A novel overlay protocol is added on top of the traditional transport layers at all the (IPN) nodes to manage the end-to-end data transfers among the (IPN) regions. Two nodes that are adjacent in the overlay space may be many hops apart in the context of the underlying network topology [56].

In [32] the authors do admit that communicating with spacecraft will never be as easy as using the terrestrial Internet because many other difficulties must be overcome. For example, the radio antenna may be frequently on the wrong side of the planet. At least in terms of networking, however, they envision that good progress can be made compared to how spacecraft data communications currently occur, which is to essentially be manually scheduled on a mission-by-mission basis.

These characteristics of the IPN project's DTN architecture have gradually found their way into applications on earth, as we will describe in section 2.2.2. Today, delay-tolerant networking has come to generally describe the concept of networking in unreliable environments where the communication with endpoints is likely to be impaired due to external factors. The architecture forms a message-oriented overlay that offers message relay services across potentially long propagation delays or even network disruptions. It

employs persistent storage to withstand network interruptions, and includes a hop-by-hop transfer for reliable delivery [43]. We will shortly discuss the DTN architecture in detail.

2.2.2 Terrestrial Networks

A growing number of communicating devices are mobile and/or operate on limited power. This is becoming much more common in terrestrial applications among mobile wireless communication devices. The last few years have assisted to the deployment of Mobile Ad hoc NETworks (MANETs) in many application environments. Originally conceived for military applications and aimed at improving battlefield communications and survivability, MANETs have lately begun pervading many civil scenarios. Precision agriculture, for example, makes use of sensor ad hoc networks to collect information about soil conditions and spatial and temporal variability in crop yield and quality. Data collected is exploited to fine-tune seeding, fertilizer, chemical and water use, thus potential increase production, lower costs, and reduce pollution. Lately supported by the development of the Zigbee technology, which allows low data rate and low power consumption exchanges devices; sensor ad hoc networks are also used by biologists for wildlife tracking, i.e., to gather biological and position information from wild species like zebras, whales, and penguins, to study their habits, mobility patterns, migratory phenomena, and so on. Intelligent highways are being developed by exploiting the possibility of ad hoc vehicle-to-vehicle and vehicle-to-roadside communications and provide driving assistance, improve driving safety (e.g., with emergency warnings) and comfort, and also allow some leisure activities

(e.g., with interactivity games for passengers). Personal Area Networks (PANs) are used for health care applications. Specifically, by distributing biomedical sensor nodes on the human body, continuous monitoring of vital functions is possible that may help prevent acute crisis in healthy subjects, give assistance to the elderly, or even avoid hospitalization for long time after an operation. Ad hoc networks may also serve small communities of colleges or conferences, or can be used to deploy emergency-response or post-disaster recovery networks, for example, to provide temporarily services like data sharing [56].

When communicating nodes are in motion, links can be obstructed by intervening bodies. When nodes must conserve power or preserve secrecy, links are shut down. These events cause intermittent connectivity. When no path exists to connect a source with a destination, a network partition occurs. Also, when a node or a link is unavailable for a very long period of time, it incurs long delays in data waiting to be sent. Besides sporadic loss of connection, a fluctuating link can also cause high error rates in transmissions when data gets dropped or not transferred completely.

The existing TCP/IP-based Internet operates on a principle of providing end-to-end inter-process communication through a concatenation of dissimilar link-layer technologies. End-to-end connectivity is enabled by the standardization of the IP protocol [58] and its mapping into link-layer data frames at each router as required. A number of key assumptions are made regarding the overall performance characteristics of the underlying links in order to achieve smooth operation:

- end-to-end path exists between a data source and its peer,

- the maximum round-trip time between any node pairs in the network is not excessive,
- and the end-to-end path loss probability is small.

On the Internet, intermittent connectivity, such as the ones that may arise in 'challenged' terrestrial applications, as mentioned above, causes loss of data. Packets that cannot be immediately forwarded are usually dropped (discarded), and TCP may retransmit them with slower retransmission timing. If packet-dropping is too severe, TCP eventually ends the session, which can cause applications to fail.

However, when can we in tangible terms call a network "challenged"? The author in [41] defines *challenged networks* as those which deviate significantly from the performance experienced in the Internet. "Significantly" can be defined informally as anything an order of magnitude larger (or smaller) than the comparable metric in the Internet.

Table 2.1: Typical path performance in challenged networks

	Wired Internet	**Internet (Satellite)**	**Oceanic Acoustic**	**Deep Space**
e2e OTT	< 2s	< 5s	< 1min	> 4 min
assym	< 30	< 300	1	to 2000
loss	< 5%	< 30%	< 10%	< .01%

Here; **e2e OTT** = end-to-end (one-way) latency; **assym** = raw bandwidth asymmetry; **loss** = message or bit loss probability.

Table 2.1 [41] compares several metrics, placing a number of networks into the chal-

lenged category. The table summarizes typical bounds for these metrics seen by users in a number of network settings. The data for this table is collected from a number of sources cited in [41]. The strikingly low loss rate for deep space links is due to extensive use of forward error correction (FEC) and high power levels. Each network has found its place in the table for different reasons. For some wireless networks (e.g. tactical military), high loss rates, long queuing delay (due to competition from telecom traffic) and mobility can lead to significantly different performance than experienced in the wired (or LAN-based wireless) Internet. For acoustic-based networks in water, the channel provides up to about 15KHz of bandwidth with a speed-of-sound propagation of about 67s/km and typical communication distances of up to 5km using acoustic modems. This slow signal propagation rate, in combination with errors created from interference of various natural and man-made sources, makes this type of network challenged from both a latency and error point of view.

With tolerance for delays and intermittent connectivity, the DTN architecture supports communication between intermittently connected nodes and unstable links, and can handle nodes or links being unavailable for several days by isolating delay with a store-and-forward buffering technique.

2.3 The DTN Architecture

The DTN architecture adds a layer between an application and a network, allowing data to be sent reliably over intermittent or long-haul links. It can be seen as an overlay on top

of a number of diverse regional networks, including the Internet. In such an overlay, delays and disruptions (i.e., intermittency) can be handled at each DTN "hop" in a path between a sender and a destination. Nodes on the path can then provide the storage necessary for application data before forwarding that to the next node on the path. For example, any required retransmissions in an ARQ scheme may come from an intermediate node, and no end-to-end connection is required between the sender and destination. Thus, the main benefit of protocols implementing the DTN architecture is that they do not require the contemporaneous end-to-end connectivity that TCP and other standard Internet transport protocols require in order to reliably transfer application data.

2.3.1 Message Switching

DTNs overcome the problems associated with intermittent connectivity, long or variable delay, asymmetric data rates, and high error rates by using store-and-forward message switching. As such, DTNs are often referred to as store-carry-forward networks. Message switching is the precursor of packet switching, where messages are routed in their entirety, one hop at a time. It was first built by Collins Radio Company, Newport Beach, California, during the period 1959 –1963 for sale to large airlines, banks and railroads. Message switching is still used for telegraph traffic and a modified form of it, known as packet switching, is used extensively for data communications [68].

In a message switching network:

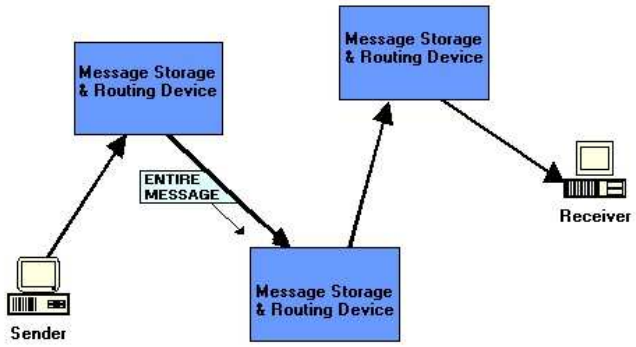


Figure 2.6: Concept of message switching.

- Messages are routed in their entirety.
- Each message is treated as a separate entity.
- Each message contains addressing information (in case of broadcast, to all recipients). This information is read and at each node and the transfer path to the next node (towards the destination) is decided (a function of routing).
- Each message is stored (usually on persistent storage) before being transmitted to the next node.

The advantages to using message switching are:

- Data channels are shared among communication devices, improving the use of bandwidth.
- Messages can be stored temporarily at message switches, when network congestion becomes a problem.

- Priorities may be used to manage network traffic.

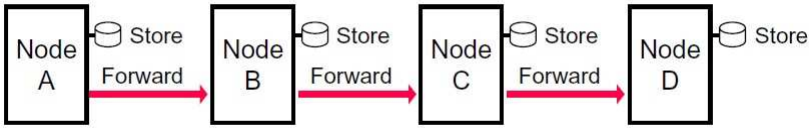
In a message switching network (see Figure 2.6), no physical path is established in advance between sender and receiver. When a message consisting of a block of data is ready to be sent, it is stored in the first node (i.e. router) and then forwarded later when it is convenient to do so to the next node, one hop at a time. As such the data is not transferred in real-time. Each block is received in its entirety, inspected and later retransmitted to the next hop. A network using this technique is referred to as a *store-and-forward network* (as shown in Figure 2.7(a) [51]).

Since message switching stores each message at intermediate nodes in its entirety before forwarding, messages experience an end to end delay which is dependent on the message length, and the number of intermediate nodes. Each additional intermediate node introduces a delay which is at minimum the value of the minimum transmission delay into or out of the node. Note that nodes could have different transmission delays for incoming messages and outgoing messages due to different technology used on the links. The transmission delays are in addition to any propagation delays which will be experienced along the message path.

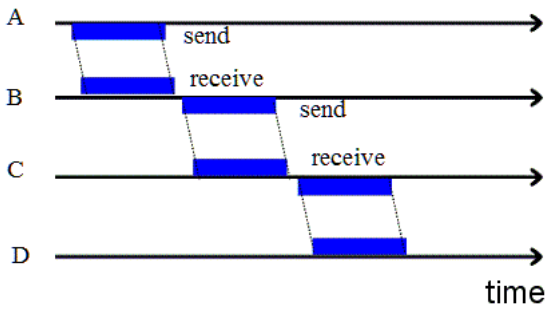
In a message-switching node an incoming message is not lost when the required outgoing route is busy. It is stored in a queue with any other messages for the same route and retransmitted when the required link becomes free. Message switching is thus an example of a delay system or a queuing system.

Figure 2.7 shows the concept of store-and-forward message switching in a DTN. Here,

a node is any entity in the network capable of forwarding bundles (i.e., messages), implementing the BPL stack. Figure 2.7(b) [68] shows the forwarding of bundles from source node A to destination node D over time.



(a) Message switching at each router.



(b) Message switching over time.

Figure 2.7: Store-and-Forward Message Switching

Implementations of the DTN architecture use persistent storage to hold bundles/messages. Persistent storage (such as, hard disks) can hold messages indefinitely, as opposed to very short-term storage provided by memory chips. Internet routers use memory chips to store (queue) incoming packets for a few milliseconds while they are waiting for their next-hop routing table lookup and an available outgoing router port.

During bundle transmission along the path between the source node and the des-

tination node, it may happen that a next hop is not available for forwarding, i.e., no connection is available to the next hop (intermittent connectivity). This may happen because the resources to be used for transmission are temporarily busy in transmitting high priority traffic or because the next hop is only reachable through a scheduled connection (e.g., when a satellite is visible). Bundles therefore need for buffering while waiting for forwarding.

DTN routers need persistent storage for their queues for one or more of the following reasons [51]:

- A communication link to the next hop may not be available for a long time (sometimes in the range of hours, days, or even weeks).
- One node in a communicating pair may send or receive data much faster or more reliably than the other node.
- A message, once transmitted, may need to be retransmitted if an error occurs at an upstream (toward the destination) node or link, or if an upstream node declines acceptance of a forwarded message.

By moving whole messages (or fragments thereof) in a single transfer, the message-switching technique provides DTN nodes with immediate knowledge of the size of messages. By combining a-priori knowledge of messages awaiting delivery with network topology and performance information; admission control, storage allocation and message routing and scheduling can be dynamically computed relatively early in the lifetime

of a message. By matching pending messages to network capacity (which may be intermittent), more sophisticated path selection algorithms beyond shortest path may allow for the use of multiple delivery paths simultaneously. Also, given an appropriate encapsulation, message switching easily supports multi-message multiplexing or"bundling" together to form an aggregate useful for message scheduling and routing [41].

2.3.2 The Bundle Protocol

The key part in the DTN architecture is the bundle protocol (BP) described by the Delay-Tolerant Network Architecture [43] and the Bundle Protocol Specification [33]. The basic unit of data in the BP is a "bundle", which is a message that carries application layer protocol data units (APDUs), sender and destination names, and any additional data required for end-to-end delivery. The bundle protocol allows hosts that normally cannot communicate with each other (due to network partitioning or because they do not have the same protocol set) to be able to communicate. This is done using message switching, which means that only adjacent nodes need to share the same protocol set, and multiple protocol sets are only required in nodes bridging protocol borders. The protocol uses existing transport protocols for data transmission but also acts as a transport protocol to applications, making it non-compliant with the traditional layer model for Internet communication. Instead of categorizing the BP as a transport layer protocol, a new layer, called the convergence layer, is added between the application and transport layers. This also implies that applications need to be adapted to use the bundling services. Figure 2.8

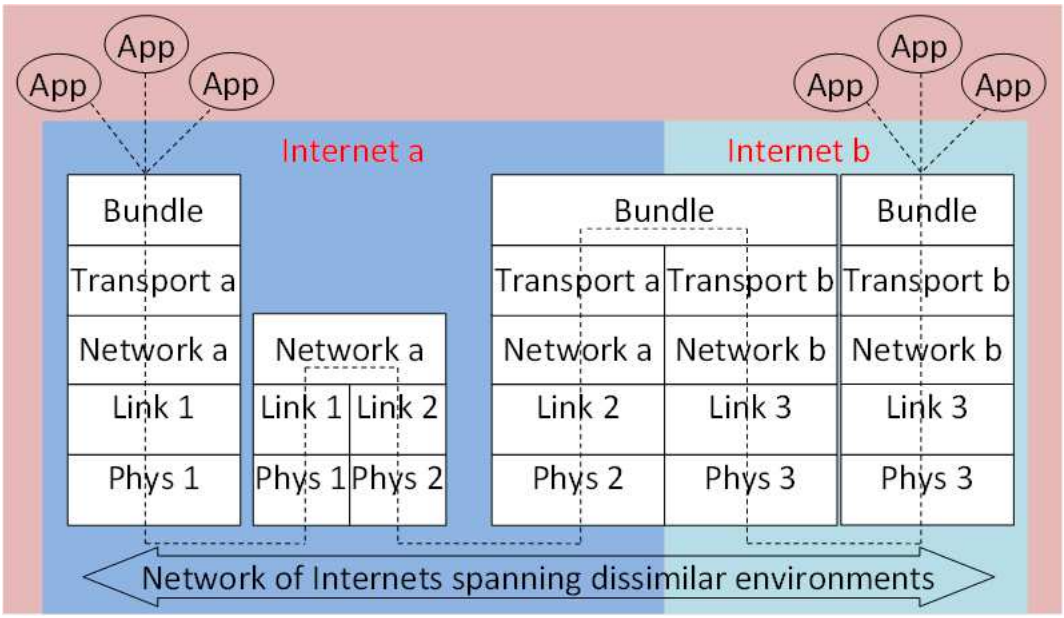


Figure 2.8: The Bundle Protocol

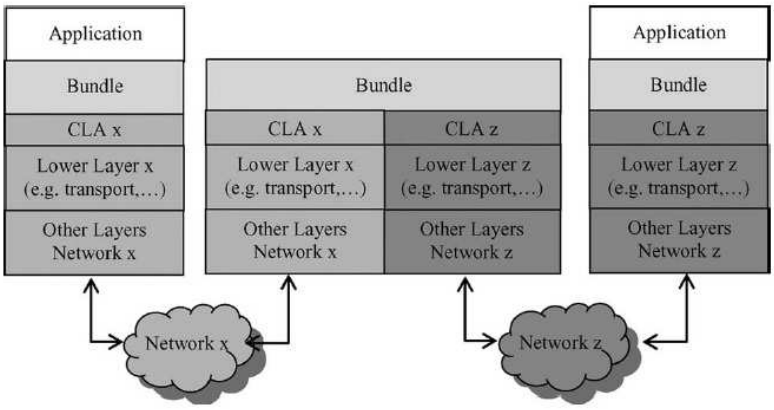


Figure 2.9: The DTN architecture

shows the bundle protocol as a transport layer for applications.

The BP can interface with different lower layer (usually transport) protocols through

convergence layer adapters (CLA's) as shown in Fig. 2.9. Various CLAs have been defined for transport layer protocols, including for TCP [45], UDP [61], and the Licklider transmission protocol (LTP) [62], [63]. Additional CLAs including NORM [64], DCCP [65], Bluetooth, and raw Ethernet have been implemented in the most commonly used open-source implementation of the BP, called DTN2 [66]. With the BP, each DTN node on a path may use whatever CLA is best suited for the next forwarding operation [59].

2.3.2.1 Bundles

In the DTN architecture, in the DTN overlay, variable-length protocol data units (PDUs) are called "bundles" (a.k.a., "messages"), and may carry application data along with information needed to deliver them to final destinations. Key DTN functionality is that each bundle is kept in memory in its entirety, and is deleted upon receipt of acknowledgment for its successful delivery to the next node on the path to the destination. However, the DTN bundle protocol specification does not limit the bundle size or specify content of bundles. Indeed, a bundle may: (i) contain a single file (e.g., a photograph), multiple files (such as, small-sized engineering telemetry files), or a file segment (possibly part of high-quality video); (ii) be of fixed size determined by an application type or network management procedures; (iii) be of variable size set by the application, and containing a coherent bundling of application data, e.g., a set of data records that can be independently processed; (iv) be self-contained and include self-described metadata useful for the application at the receiving end-system.

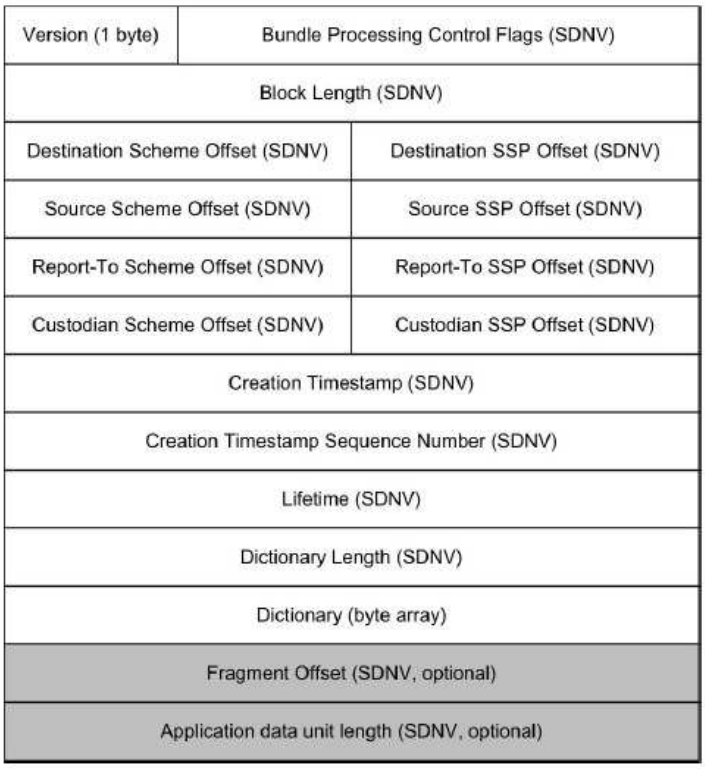


Figure 2.10: The structure of the primary block of a bundle.

Figure 2.10 depicts the structure of a primary bundle block. Each bundle consists of one or more headers, stacked after each other. The first or primary block (or, header) of each bundle contains the DTN equivalents of the data typically found in an IP header on the Internet: version, source and destination IDs (called EIDs, as we will discuss shortly), length, processing flags, and (optional) fragmentation information. It can also contains some additional fields, for delivery options and handling: report-to EID, current custodian EID, creation timestamp and sequence number, lifetime and a dictionary.

The primary header contains delivery options and references to addresses stored in a

succeeding dictionary header. Other possible headers can follow in a non-specific order, with the only exception that the payload header is placed at the very end. The payload is placed last to allow for dynamic fragmentation in case of a link failure during transmission, which means that in case of a link drop-out at the end of a transmission only the last part needs to be resent. This can be used to always maximize link usage. Because of this, the payload header has no information about any trailing header, whereas other headers include this next-header information. A node sending a bundle can request reports of what happens to the bundle during its journey to the destination. These so called status report are placed in the payload of a new bundle, as a different payload type, and sent to a specified report-to address. Most fields in the primary bundle block are variable in length, and use a relatively compact notation called self-delimiting numerical values (SDNVs) [33], discussion of which is beyond the scope of this thesis.

The name "bundle" derives from considering protocols that attempt to minimize the number of round-trip exchanges required to complete a protocol transaction, and dates back to the original IPN work. By "bundling" together all information required to complete a transaction (e.g., protocol options and authentication data), the number of exchanges can be reduced, which is of considerable interest if the round trip time is hours, days or weeks [28].

The following definitions are taken from the Bundle Protocol Specification, RFC - 5050 [33].

2.3.2.2 Bundle Nodes

A bundle node (or, in the context of this thesis, simply a "node") is any entity that can send and/or receive bundles. In the most familiar case, a bundle node is instantiated as a single process running on a general-purpose computer, but in general the definition is meant to be broader. A bundle node might alternatively be a thread, an object in an object-oriented operating system, or a special-purpose hardware device.

In the context of the operation of a bundle node, a bundle is an instance of some bundle in the network that is in that node's local memory. Multiple instances of the same bundle (the same unit of DTN protocol data) might exist concurrently in different parts of a network, possibly in different representations, in the memory local to one or more bundle nodes and/or in transit between nodes.

Each bundle node has three conceptual components: a "bundle protocol agent", a set of zero or more "convergence layer adapters", and an "application agent".

- Bundle protocol agent: The bundle protocol agent (BPA) of a node is the node component that offers the BP services and executes the procedures of the bundle protocol. The manner in which it does so is wholly an implementation matter. For example, BPA functionality might be coded into each node individually; it might be implemented as a shared library that is used in common by any number of bundle nodes on a single computer; it might be implemented as a daemon whose services are invoked via inter-process or network communication by any number of bundle nodes on one or more computers; it might be implemented in hardware.

The bundle protocol agent of each node is expected to provide the following services to the node's application agent (AA):

- commencing a registration (registering a node in an endpoint);
- terminating a registration;
- switching a registration between Active and Passive states;
- transmitting a bundle to an identified bundle endpoint;
- canceling a transmission;
- polling a registration that is in the passive state;
- delivering a received bundle.

We will discuss bundle endpoints, registration and delivery shortly.

- Convergence layer adapters: A convergence layer adapter (CLA) sends and receives bundles on behalf of the BPA, utilizing the services of some 'native' internet protocol that is supported in one of the internets within which the node is functionally located. The manner in which a CLA sends and receives bundles is wholly an implementation matter, exactly as described for the BPA.

- Application agent: The application agent (AA) of a node is the node component that utilizes the BP services to effect communication for some purpose. The application agent in turn has two elements, an administrative element and an application-specific element. The application-specific element of an AA constructs, requests

transmission of, accepts delivery of, and processes application, specific application data units; the only interface between the BPA and the application-specific element of the AA is the BP service interface. The administrative element of an AA constructs and requests transmission of administrative records (status reports and custody signals), and it accepts delivery of and processes any custody signals that the node receives.

In addition to the BP service interface, there is a (conceptual) private control interface between the BPA and the administrative element of the AA that enables each to direct the other to take action under specific circumstances. In the case of a node that serves simply as a "router" in the overlay network, the AA may have no application-specific element at all. The application-specific elements of other nodes' AAs may perform arbitrarily complex application functions, perhaps even offering multiplexed DTN communication services to a number of other applications. As with the BPA, the manner in which the AA performs its functions is wholly an implementation matter; in particular, the administrative element of an AA might be built into the library or daemon or hardware that implements the BPA, and the application-specific element of an AA might be implemented either in software or in hardware.

2.3.2.3 Bundle Endpoint IDs (EIDs)

A bundle endpoint (or simply "endpoint") is a set of zero or more bundle nodes that all identify themselves for BP purposes by some single text string, called a "endpoint ID" (EID). Each endpoint ID conveyed in any bundle takes the form of a Uniform Resource Identifier (URI) [67]. Each endpoint ID can be characterized as having this general structure:

<scheme name>:<scheme-specific part, or ‘‘SSP’’>

As used for the purposes of the bundle protocol, neither the length of a scheme name nor the length of an SSP may exceed 1023 bytes.

Using URIs as identifiers brings several advantages [28]. First, they can encode names or addresses taken from many namespaces. For example, we might refer to a host by its Ethernet address as ether://00-1c-c0-eb-0f-aa but also refer to it using some distinguished hierarchical name like dns://myhost.foo.ca. While in the Internet, the scheme specifier also tends to suggest the protocol stack used (e.g., *http* is typically http/TCP/IP) to contact remote node(s), this need not be the case for DTN; we can use the bundle protocol, or some other combination of protocols.

Next, using strings for representing EIDs opens up the possibility of creating interesting DTN forwarding policies using string matching. For example, a wildcarded string match could be be used in directing a DTN forwarder to cause any traffic destined for York University to be directed to some particular next hop:

```
dtn://*.yorku.ca.dtn ->

ether://00-1c-c0-eb-0f-aa
```

This example illustrates that the addressing format for a DTN next hop (at a DTN forwarding node) 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.

Using URIs can also help to support application layer gateways by piggybacking on a number of pre-existing URI schemes. For example, the URIs:

```
http://www.slashdot.org

dtn:http://www.slashdot.org
```

are syntactically legitimate bundle protocol EIDs. The full extent to which this capability may be useful remains to be seen, as few such application layer gateways using existing schemes have been constructed, but the naming compatibility is clear.

The special case of an endpoint that never contains more than one node is termed a "singleton" endpoint; every bundle node must be a member of at least one singleton endpoint. Singletons are the most familiar sort of endpoints, but in general the endpoint notion is meant to be broader. For example, the nodes in a sensor network might constitute a set of bundle nodes that identify themselves by a single common endpoint ID and thus form a single bundle endpoint. Likewise, a given bundle node might identify itself by multiple endpoint IDs and thus be a member of multiple bundle endpoints.

2.3.2.4 Bundle States

A bundle may be in one of the following states:

- *Registration* - A registration is a given node's membership in a given endpoint. Any number of registrations may be concurrently associated with a given endpoint, and any number of registrations may be concurrently associated with a given node. Any single registration must at any time be in one of two states: Active or Passive.

- *Forwarding* - When the bundle protocol agent of a node determines that a bundle must be "forwarded" to an endpoint, it causes the bundle to be sent to all of the nodes that the bundle protocol agent currently believes are in the "minimum reception group" of that endpoint. The nature of the minimum reception group for a given endpoint can be determined from the endpoint's ID.

  The minimum reception group of an endpoint may be any one of the following:

  (a) ALL of the nodes registered in an endpoint that is permitted to contain multiple nodes, in which case forwarding to the endpoint is functionally similar to "multicast" operations in the Internet, though possibly very different in implementation;

  (b) ANY N of the nodes registered in an endpoint that is permitted to contain multiple nodes, where N is in the range from zero to the cardinality of the endpoint, in which case forwarding to the endpoint is functionally similar to "anycast" operations in the Internet; or

(c) THE SOLE NODE registered in a singleton endpoint, in which case forwarding to the endpoint is functionally similar to "unicast" operations in the Internet.

- *Transmission* - A transmission is a sustained effort by a node's bundle protocol agent to cause a bundle to be sent to all nodes in the minimum reception group of some endpoint (which may be the bundle's destination or may be some intermediate forwarding endpoint) in response to a transmission request issued by the node's application agent. Any number of transmissions may be concurrently undertaken by the bundle protocol agent of a given node.

- *Deliverability, Abandonment* - A bundle is considered "deliverable" subject to a registration if and only if (a) the bundle's destination endpoint is the endpoint with which the registration is associated, (b) the bundle has not yet been delivered subject to this registration, and (c) delivery of the bundle subject to this registration has not been abandoned. To "abandon" delivery of a bundle subject to a registration is simply to declare it no longer deliverable subject to that registration; normally only registrations' registered delivery failure actions cause deliveries to be abandoned.

- *Discarding, Deletion* - A bundle protocol agent "discards" a bundle by simply ceasing all operations on the bundle and functionally erasing all references to it. The specific procedures by which this is accomplished are an implementation matter.

"Retention constraints" are elements of the bundle state that prevent a bundle from being discarded. A bundle cannot be discarded while it has any retention constraints. A bundle protocol agent "deletes" a bundle in response to some anomalous condition by notifying the bundle's report-to endpoint of the deletion and then arbitrarily removing all of the bundle's retention constraints, enabling the bundle to be discarded.

- *Fragmentation* - DTN supports fragmentation and reassembly of bundles in order to improve the efficiency of transfers. Fragmentation allows the bundle protocol to fully utilize the available link capacity and to avoid retransmissions of partially sent bundles.

  There are two forms of DTN fragmentation: proactive and reactive. Proactive fragmentation is performed when a DTN node knows in advance (or predicts) that sending multiple fragments, rather than a single large bundle, is more likely to succeed. For example, a node might know that a regularly occurring downstream contact is always of such a short duration that the entire bundle cannot be sent entirely. In this case the node would proactively fragment the bundle and transmit the fragments during consecutive contacts.

  Reactive fragmentation occurs when a lower layer indicates that some of the transmitted bytes were successfully transferred, but the entire bundle was not. The previous hop node may then retransmit the missing portion. With both fragmentation types, the fragments are only reassembled at the final destination. Bundle

fragments may also be further fragmented along the way, either proactively or reactively.

If an application wants to prevent fragmentation it can set a "do not fragment" flag in the bundle. This may be useful, for example, when the integrity of a bundle is protected by a digital signature.

The above components of the DTN bundle layer have been presented from a systems perspective, as defined by the protocol specification. Any networked software system requires an implementation architecture to guide the choice of abstractions used to represent the corresponding concepts in the network architecture. In [29] authors present one implementation of the DTN bundle protocol. Another one is also presented in [28]. Figure 2.11 (courtesy of [29]) is a block diagram enumerating the major components of the DTN Bundle forwarding system. As can be seen from the diagram, the bundle router module represents the most central component of the implementation; in general, it requires the most detailed information regarding the state of the system upon which to base routing decisions. Decisions made by the router are passed as a set of instructions (actions) to the forwarder which is responsible for executing the actions. The functionality of each major component has been described in [29], and we refer the reader to the paper if they wish to get a sense of the overall operation of the system.

As seen in Figure 2.11, implementations of the DTN architecture use persistent storage for holding in-transit bundles. In standard networks, which assume continuous connectivity and short delays, routers perform non-persistent (short-term) storage and information

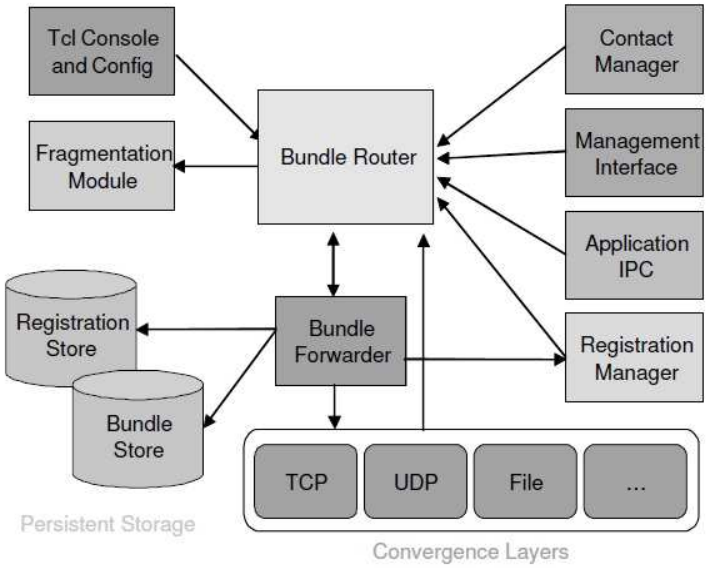


Figure 2.11: An implementation of the Bundle protocol system components.

is persistently stored only at end nodes, i.e., outside the network core. This is because, dealing with reliable transmission, information is supposed to be easily retrieved directly from the source. This may not be the case in challenged networks. Therefore, to deal with long RTTs and channel disruptions, and to cope with the extreme case of the absence of end-to-end connectivity, in DTN networks information is persistently (long-term) stored at intermediate DTN nodes [59] and forwarded from node to node (i.e. hop by hop) using message switching.

2.3.3 Custody Transfer and Congestion

In some DTN use cases, the original sender of a bundle will never have the opportunity to retransmit the application data, for example, due to physical movement away from the

network, or for power management reasons (if the sender will be powered off until after the bundle expires). DTNs support node-to-node retransmission of lost or corrupt data at both the transport layer and the bundle layer. However, because no single transport-layer protocol (the primary means of reliable transfer) operates end-to-end across a DTN, end-to-end reliability can only be implemented at the bundle layer. The bundle layer supports node-to-node retransmission by means of custody transfers. Custody transfers are an optional service.

In the BP, a sending node can request that other nodes on the path take custody by signaling this in the bundle header. Any node on the path can take custody of the bundle. If a node chooses to take custody of a bundle, it takes over all responsibilities regarding the bundle, such as retransmission, and related resources can be released from the previous custodian.

The node that first sends a bundle keeps a copy of it in its persistent memory after transmission until receipt of an acknowledgement from the next hop. It is the first custodian of the bundle because it keeps the only reliable copy of the bundle and will use it for retransmissions, if necessary. When the bundle custodian sends the bundle to its next hop node, it requests the custody transfer for that bundle and starts a time-to-acknowledgement retransmission timer. If the bundle layer of the next hop accepts custody of the bundle, it returns an acknowledgement to the sender. If the bundle custodian receives no acknowledgement before the senders time-to-aknowledgement expires, it transmits the bundle again to the next hop. The bundle custodian stores the bun-

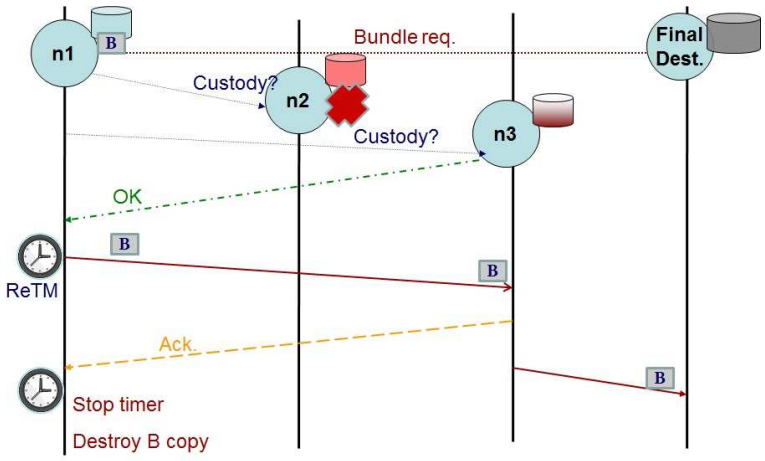


Figure 2.12: Custody transfer for bundle forwarding reliability in a DTN.

dle until either another node accepts custody of the bundle or the bundle time-to-live expires. Then, either a new bundle custodian exists or the bundle has no more reason to exist. Thus, the copy of the bundle is discarded by the (obsolete) bundle custodian. Figure 2.12 shows an abstraction of the custody transfer mechanism, from node $n1$ to the $finaldestination$ delivery of bundle B, as node $n3$ subsequently becomes a custodian of the bundle. The time-to-live of a bundle is obviously much longer than the time-to-acknowledgement of any custodian. The custody option increases reliability and is particularly useful whenever the sender has limited memory and/or power resources, as in sensor networks, or has good reasons not to keep in its memory sensitive information, such as in military applications.

Node-to-node reliability is perceived by local retransmissions managed at the transport layer inside single DTN regions. This is different from what happens in the TCP protocol where retransmissions are handled end-to-end and the source node retransmits

the data in case of a missed acknowledgement from the destination node. This approach is unfeasible in an DTN environment because of the high end-to-end delays involved, whereas retransmissions inside single DTN regions are faster and more efficient. So, reliable transport layer protocols and custody transfers are used by the bundle layer to move points of retransmissions progressively forward towards the destination node.

In case the source node desires final notification of delivery, the destination node should send a separate return receipt to the source after receiving the bundle. The return receipt is transmitted as a new bundle, and is subject to the same custody transfers as the original transmission. The return receipt is similar to those utilized within the postal system [69].

Not every node in a DTN needs to offer custody transfer. A node may refuse to accept custody for messages for implementation or policy reasons, because not enough free storage space is currently available, or for other reasons. However, many users of DTN networks wish to lose no data, so every node and every bundle operates using custody transfer or some equivalent capability. This may be adequate for a stable network with sufficient storage resources, but is not when the source rate exceeds the network delivery rate beyond the networks buffering capability. This is, in essence, the main problem of DTN congestion.

DTN congestion occurs when storage resources become scarce due to the presence of too much bundle data or too many bundle fragments. A node experiencing these situations has several options to mitigate the situation, in the following order of preference:

drop expired bundles, move bundles somewhere else, cease accepting bundles with custody transfer, cease accepting regular bundles, drop unexpired bundles, and drop unexpired bundles for which the node has custody. Very few papers on DTN congestion control have been published ([71], [72]) and research in this area is on-going.

2.3.4 Regions, Gateways, and Naming

DTN is an "overlay" architecture in that it is expected to operate above the existing protocol stacks present in other network architectures. For example, in the Internet the overlay may operate over TCP/IP, in the space context it may operate over CFDP/CCSDS, and in sensor/actuator networks it may operate over a network composed of a yet-to-be-standardized sensor transport protocol with some specialized routing (e.g. Epidemic, Diffusion, DSDV). Each of these networking environments have their own specialized protocol stacks and naming semantics developed for their particular application domain.

As such, a large DTN network can consist of nodes from several different network topologies, each with a different addressing scheme. The use of different addressing schemes is usually a reason why nodes from different networks are unable to communicate.

DTN solves this problem by defining a *region* [43] as a group of nodes in a network, using the same protocol set for communication.

Achieving interoperability between regions is accomplished by special DTN gateways located at their interconnection points. Interoperability gateways (see Figure 2.14 [41])

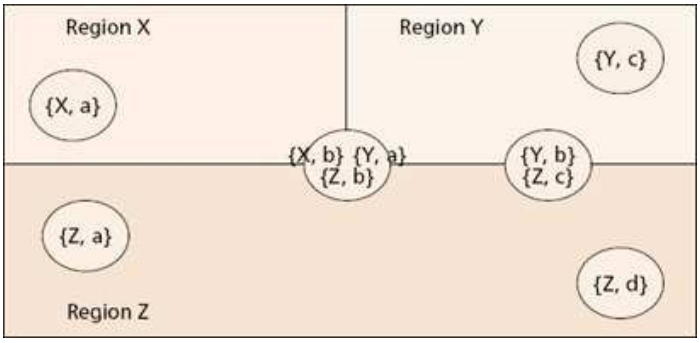


Figure 2.13: DTN regions in a large network of networks.

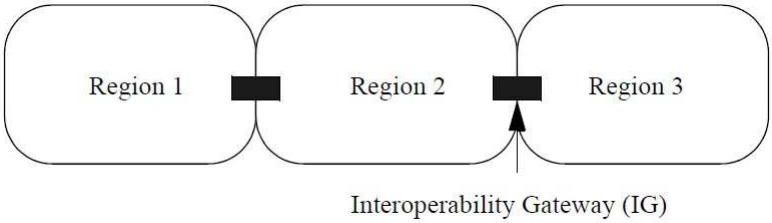


Figure 2.14: DTN gateway for interoperability among dissimilar protocol stacks

are DTN forwarders interconnecting regions running potentially incompatible protocol stacks. By operating above the transport protocols in use on the incident networks, they provide message switching, in-network retransmission, and name mapping, allowing the use of globally-interoperable names to be mapped to region-local names as required by the adjacent region's delivery semantics [41].

Figure 2.13 shows six DTN nodes in three regions, each having a region id and a local id. Abstaction of how gateways reside at two region endpoints to provide interoperability is depicted in Figure 2.14.

Since nodes in different regions often use different protocol sets, consequently using

different addressing schemes; DTN defines a naming scheme comprising of a tuple with two variable-length portions of the form R, L where R is a hierarchical $regionID$ and L contains a name local to region R, and is called an $entityID$.

Routing *between* regions is based only on region IDs, which are bound to their corresponding addresses throughout the DTN. Routing *within* regions is based only on entity IDs, which are bound to their corresponding addresses only within that region.

An entity may be a host (a DTN node), an application instance, a protocol, a URL, a port (used to find the bundle service on a host) and potentially a token (used to find a particular application instance that is using the bundle service), or something else.

To express the scope of the naming scheme we could have the following tuple, for example:

`internet.earth.sol.int,`

`‘‘http://www.cse.yorku.ca/grad/mastersGuidelines.html’’`

The first portion identifies a region and is interpreted by DTN forwarders to find the path to one or more interoperability gateways at the edge of the specified region, and the second portion identifies a name within the specified region. As a message transits across a potentially long and heterogeneous collection of regions, only the region part of the endpoint ID is meaningful until the bundle has arrived somewhere within its destination region. Once a bundle reaches the edge of the destination region, the name information is locally-interpreted, and translated if necessary, into a name appropriate to the containing region.

Region IDs use the same name-space syntax as the Internets Domain Name System (DNS); i.e. a tree structure with the root last.

In summary, routing in a DTN is primarily done based on the region part of the eidpoint and then according to local rules used by each network topology; despite the regional networks having varying connectivity, delay and loss characteristics, or employing different lower-layer technologies.

2.3.5 Contact Types

The DTN architecture is targeted at networks where an end-to-end routing path cannot be assumed to exist. Rather, routes are comprised of a cascade of time-dependent contacts (communication opportunities) used to move messages from their origins toward their destinations.

Contacts are parameterized by their start and end times, capacity, endpoints, and direction. In addition, a measure of a contacts predictability can help to choose next-hop forwarders for message routing as well as select the next message to be sent. The predictability of a route exists on a continuum ranging from completely predictable (e.g. wired connection or a periodic connection whose phase and frequency are well-known) to completely unpredictable (an "opportunistic" contact in which a mobile message router has come into communication range with another DTN node).

For example, in disaster recovery networks the future location of communicating entities, such as emergency responders, may not be known. These types of contacts are

known as *intermittent or opportunistic contacts*. On the other hand, DataMules carrying data to certain infrastructure endpoints on a routine basis are examples of a *scheduled or predictable contact*.

2.3.6 Routing

For most IP router implementations, routing protocols maintain a routing information base (RIB) that maps destinations to a set of potential next-hop links (and/or other information), and often a system-wide forwarding information base (FIB) stores the current best route for each destination. Packet arrivals trigger lookups in the FIB structure, resulting in forwarding or a drop. The job of the routing algorithm is generally confined to maintaining the RIB.

A DTN router also requires state about next-hop contacts and networks reachable through those contacts, but a large number of other factors come into play. Given the store-and-forward nature of DTN, a router will likely consider its own storage state and perhaps the storage state of a peer node when making forwarding decisions. Unroutable messages are typically not dropped immediately, but rather queued in persistent storage until they either expire or an appropriate next-hop peer becomes available. To handle custody transfer based reliability, a router may still need to buffer a message even after it has been transmitted. Finally, the uncertain nature of some networks may cause the router to maintain historical contact state and make future predictions about contact arrivals for scheduling purposes.

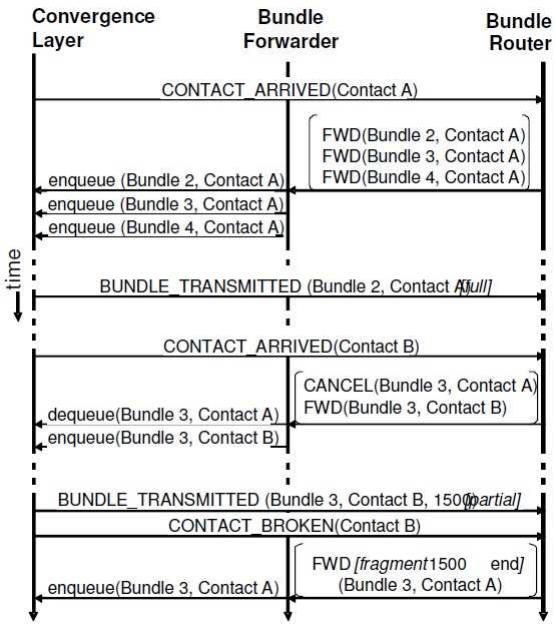


Figure 2.15: An example event-action flow of Bundle routing.

In considering the list of potential inputs to the routing decision function, it is clear that traditional RIB/FIB design for a router is insufficiently expressive. Furthermore, due to the wide range of situations in which DTN may be applied, it is likely that policy decisions may affect multiple system layers, and may vary widely from deployment to deployment. As such, current implementations of the BP push all policy related decision making to event handler routines in the BP that comprise the router implementation. The advantages of doing so have been discussed in [29].

Figure 2.15 shows an example of this event exchange between the Convergence Layer, Bundle Forwarder and the Bundle Router (see Figure 2.11). The Bundle Forwarder converts lists of actions into queue manipulation options to the convergence layer. Based on

the arrival of a new contact opportunity, the router schedules three bundles for transmission on the contact, and is then notified when one of the transmissions completes. The arrival of a second contact causes the router to change an earlier scheduling decision to use the new contact. Finally, when that contact is broken, the router is again notified, triggering a fragmentation, and re-scheduling the unsent portion of the bundle back on the first contact.

In this section we have discussed the hop-by-hop characteristics of DTN routing with concentration on the BP layer.

Currently, no routing protocol has been defined to be used in conjunction with the bundle protocol, to provide a complete end-to-end solution to bundle routing. In Section 2.4 we discuss the end-to-end routing characteristics of a delay-tolerant network and also discuss several DTN routing protocols that have been proposed.

2.3.7 Security

Since the resources of a DTN network can be limited, steps need to be taken to ensure that only the intended data is accepted into the network.

In a DTN, standard Internet security mechanisms, such as TLS, IPsec and variants do not perform well, or at all, because of long delays, possible disruptions, and the possible lack of a continuous end-to-end connectivity. The DTN architecture addresses this problem with new security tools. At present, the definition of BP security is still in progress but many specifications are already contained in the bundle security protocol

(BSP) [70], which defines a set of BP extensions to support hop-by-hop and end-to-end authentication, integrity validation, and confidentiality [59].

The bundle structure consists of a series of elements called blocks (see Figure 2.10). Additional security blocks are defined in the BSP [70]. We discuss the *payload security header* and the *authentication header*.

The payload security header is used to verify the payload integrity, and also can be used to authenticate the original sender on an end-to-end basis. Verification is done by calculating a hash of the payload and comparing it to the hash supplied int he header, whereas the authenticity is verified by checking a supplied signature of the hash. Hashing and signing are done using common algorithms and asymmetric keys. The services provided by the payload header can also be implemented in applications allowing for more flexibility regarding new functionality [52].

The authentication header, on the other hand, is used to verify the entire bundle on a hop-by-hop basis, using the same types of hashes and signatures as in the payload verification. The use of a hop-by-hop verification has several benefits, such as early discard of non-authenticated or damaged bundles and simplified key exchange. the benefits of the simpler key exchange is achieved because a node will only need to know the keys of adjacent nodes and any end node it communicates with, greatly reducing the number of keys a node needs to store and also the time and resources used on a key update [52].

2.4 Routing in DTNs

Delay-tolerant networks (DTNs) have the potential to connect devices and areas that are can not be fully served by traditional network solutions, such as TCP/IP. One application area of DTNs is terrestrial networks consisting of wireless, mobile nodes where issues such as excessive delivery latency, high error rates in transmissions and consequently low throughput may arise because of frequent, long-duration partitions in the network. Both links and nodes may be inherently unreliable due to node mobility, geographic dispersion density (or, lack thereof) of nodes, insufficient communication infrastructure, and power conservation issues.

Presumably, in such networks establishment of an end-to-end connection between a traffic source and its destination, prior to message forwarding is unattainable due to link fluctuations, rendering many tested mobile ad hoc network (MANET) routing protocols [4], [5], [6], [7] impracticable. We establish the non-applicability of MANET routing protocols to DTNs environments in the following section.

2.4.1 MANET vs. DTN Routing

A *wireless network* is any type of computer network that uses wireless data connections for connecting network nodes.

A *wireless ad hoc network* (WANET) is a decentralized type of wireless network. The network is ad hoc because it does not rely on pre-existing infrastructure, such as routers in wired networks or access points in managed (infrastructure) wireless networks.

A *mobile ad hoc network* (MANET) is a continuously self-configuring, infrastructure-less network of mobile devices connected without wires. They are a kind of wireless ad hoc network that usually has a routable networking environment on top of a Link Layer ad hoc network. Each node in a MANET is free to move independently in any direction, and will therefore change its links to other devices frequently. Each node must forward traffic unrelated to its own use, and thus be a router as well. Once two nodes are in wireless communication range, they can communicate in a point-to-point fashion.

While the nodes in MANETs are mobile, it is generally assumed that end-to-end, possibly multi-hop paths between node pairs exist most of the time. Routing protocols designed to operate in MANETs assume that these paths are formed by a set of wireless links that exist contemporaneously [4], [5], [6], [7]. It is also assumed that if these paths are disrupted because of node mobility, then this disruption is only temporary and the same or alternate paths are restored relatively quickly.

We briefly discuss the AODV routing protocol as an example of MANET routing. The AODV routing protocol uses the following three types of packets to establish, maintain and update routes:

- *Sending out route request (RREQ)*:

  If a node using the AODV routing protocol [5] desires to send a message to a destination node for which it does not have a valid route to, it initiates a route discovery to locate the destination node. The source node broadcasts a route request (RREQ) packet to all its neighbors, which then forwards the request to their neighbors and

so on until either the destination or an intermediate node with a "fresh enough" route to the destination listed in the RREQ is located. Nodes keep track of the RREQ's source IP address and broadcast ID. If they receive a RREQ which they have already processed, they discard the RREQ and do not forward it.

AODV makes use of sequence numbers to ensure that the routes are loop free. Each node maintains its own sequence number, and a broadcast ID. The sequence number is incremented whenever there is a change in the neighborhood of a node and the broadcast ID is incremented for every route discovery the node initiates. Along with its own sequence number and the broadcast ID, the source node also includes the most recent sequence number it has for the destination node. Intermediate nodes may reply to the RREQ if they have a route to the destination with a destination sequence number equal to or more than the one listed in the RREQ (a.k.a, a fresh enough route).

- *Receiving route reply (RREP)*:

  When the RREQ reaches the destination or an intermediate node having a fresh enough route to the destination, it responds by sending a route reply (RREP) packet to the source. As the RREP propagates back to the source, nodes set up forward pointers to the destination. Once the source node receives the RREP, it may begin to forward data packets to the destination. If the source later receives a RREP containing a greater sequence number or contains the same sequence number with a smaller hopcount, it may update its routing information for that destination and

begin using the better route.

- *Maintaining routes with route error message (RERR)*:

  As long as the route remains active, it will continue to be maintained. A route is considered active as long as there are data packets periodically traveling from the source to the destination along that path. Once the source stops sending data packets, the links will time out and eventually be deleted from the intermediate node routing tables.

  Periodic HELLO broadcasts are used in AODV by the nodes in the network to inform each mobile node of other nodes in its neighborhood. These broadcasts are used to maintain local connectivity. If a link break occurs while the route is active (node along the route moves, or powers down), its upstream neighbor notices the move and propagates a link failure notification/route error message (RERR) to each of its active upstream neighbors for the source node to inform it of of the now unreachable destination(s). After receiving the RERR, if the source node still desires a route to the destination, it can reinitiate route discovery.

Hence, a pre-established end-to-end path from the source to the destination (that is, links on an end-to-end path) should exist contemporaneously in MANETs for messages to be successfully forwarded to intended recipients. However, in DTNs this pre-establishment is not possible due to the nature of the nodes in the network, or the application itself. Suitable applications for DTN environments are non-real time, such as

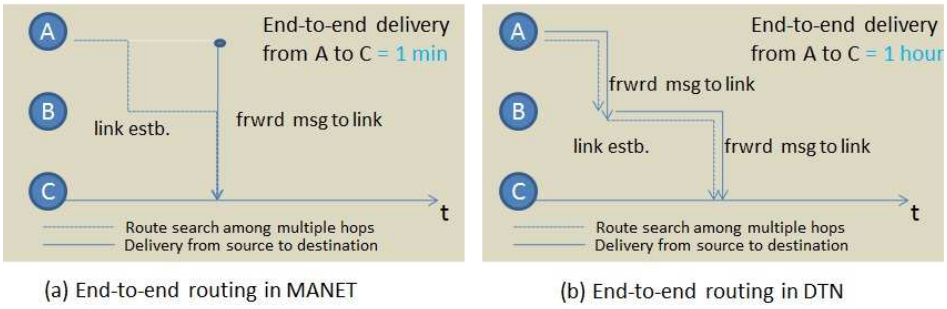


Figure 2.16: Routing in a MANET versus store-(carry)-forward routing.

file transfer, sensor data collection and messaging. We have mentioned in section 2.2.2 that there exists numerous application areas for DTN's, including those from deep sea (e.g. underwater acoustic networks) to those in deep space (e.g. IPN).

For example, in underwater acoustic networks [72], which is usually a deployment of wireless sensors on certain carriers (e.g. whales, underwater anchors, buoys) to collect data, the sensors are in movement due to the motion of water or the carriers themselves. The deployment and monitoring area is usually kept as large as possible as to not hamper the natural habitat. It is easy to see how sensors may not always be within the wireless transmission range of each other due to geographically being dispersed and only periodically, on certain opportunities, coming within the communication range of each other (i.e. when two whales meet). Hence, the above AODV protocol would always fail, as neighborhood discovery would fail and it would also timeout before receiving a RREP from the destination.

Another example is vehicular networks, where nodes are moving continuously which would cause AODV to keep on receiving RERRs on paths as soon as they were established,

as nodes from the path have moved away. Any new route initiation would fail for the same reason.

In disaster recovery operations, such as those performed after earthquakes or tsunami's, where communication infrastructure has been damaged, mobile devices in that area may use a temporarily deployed DataMule [20] to carry data to and from a sink remotely connected to the Internet. Due to the sparse nature of the mobile nodes locations and the periodic, but continuously moving nature of the DataMules route, AODV routing protocol would not be able to establish an end-to-end path for data communication.

However, for all the (DTN) network environments just mentioned, their primary distinction from MANETs is the fact that links on an end-to-end path may not exist contemporaneously, intermediate nodes may need to store data waiting for opportunities to transfer data towards destinations, and applications may need to tolerate very long delays in data delivery.

Routing protocols for DTN's implement message switching with persistent storage, utilizing the store-(carry)-forward mechanism to endure delays and preserve data packets on intermittent links, allowing above such applications a solution to routing.

In figure 2.16, we show side-by-side the routing link establishment phases, and subsequent data forwarding of MANET and DTNs respectively among three nodes, where A is the source and C is the destination. In MANETs, the end-to-end link must be established a priori. Hence, when the actual data is transfered on the pre-established path, the data packet may be received with very little delay (almost instantaneously). However, in the

DTN, as links are established hop-by-hop, opportunistically between A to B, the B to C, the data packets must endure longer delays as the total end-to-end delay will include both the delay to encounter the contacts B and C, and also forward the actual data. The delay times presented in the figures are fictional and not extracted from any real data.

2.4.2 DTN Network Model

Authors in [3] take a graph theory approach to model a DTN network to help address DTN routing issues. Their model provides a set of definitions and a framework for evaluating DTN routing algorithms. We present the model here, courtesy of [3]:

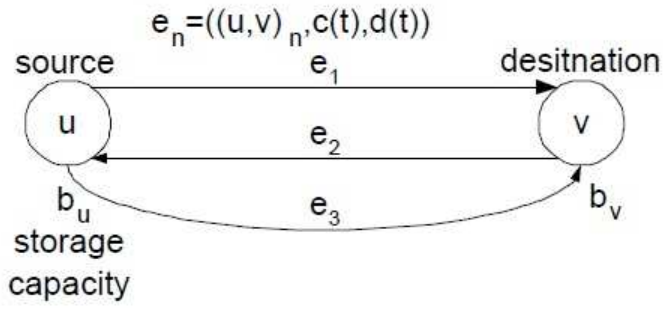


Figure 2.17: A graph representation of a DTN network.

The challenges in DTN routing stem largely from the fact that the DTN network model is not simply a graph, as in most present networking systems, but instead is a time varying multigraph, as is presented in figure 2.17 [3]. To represent the multigraph properties, nodes may be connected by multiple edges, representing different physical links. Each node j performs store-and-forward routing, and has finite storage capacity (b_j). To represent time variance, an edge is parameterized by its source and destination

nodes plus a capacity ($c(t)$) and delay function ($d(t)$).

Definitions of the graph components are as follows:

- *Nodes and Edges*: The DTN graph is a directed multi-graph, in which more than one edge (also called link) may exist between a pair of nodes (figure 2.17). The reason for using a multigraph is straightforward: it may be possible to select between two distinct (physical) connection types to move data between the same pair of nodes. Furthermore, the link capacities (and to a lesser extent, propagation delay) are time-dependent (capacity is zero at times when the link is unavailable). Thus, the set of edges in the graph must capture both time-varying capacity and propagation delay as well as multiple parallel edges.

- *Contact*: A contact is an opportunity to send data over an edge. More precisely, it is a specific edge and a corresponding time interval during which the edge capacity is strictly positive.

- *Messages*: Communication demands are represented by messages. A message is a tuple $(u; v; t; m)$, where u is the source of the message, v is the destination, t is the time at which the message is injected into the system and m is its size (messages can be of arbitrary size). The set of all messages is called the traffic demand.

- *Storage*: The nodes in a DTN have finite long-term storage (buffers) used for holding in-transit data or data waiting to be consumed by the application at a destination node. In our model, the storage is exclusively used for holding in-transit data.

Destination nodes are assumed to have sufficient capacity for holding data to be consumed by an application.

- *Routing*: Routing occurs in a store and forward fashion. The routing algorithm is responsible for determining the next edge(s) that a message should be forwarded along. Messages not immediately forwarded wait until they are assigned to contacts by the routing algorithm.

2.4.3 Considerations in Designing DTN Routing Protocols

There are several characteristics of a network and its components a DTN routing protocol must take into consideration [73].

The first consideration is if information about future contacts is readily available. We have discussed briefly about contact types in section 2.3.5 where contacts may be either scheduled, opportunist, or some variance of these two. A 'contact' has been defined as a duration during which one node can send to another with a certain bandwidth expectation.

The second consideration is if mobility can be exploited and, if so, which nodes are mobile. There are three major cases, classifying the level of mobility in the network. First, it is possible that there are no mobile entities. In this case, contacts appear and disappear based solely on the quality of the communication channel between them. For instance, in interplanetary networks, large objects in space, such as planets, can block communicating nodes for a set period of time. Second, it is possible that some, but not all, nodes in

the network are mobile. These nodes, sometimes referred to as Data Mules, [19] [20] are exploited for their mobility. Since they are the primary source of transitive communication between two non-neighboring nodes in the network, an important routing question is how to properly distribute data among these nodes. Third, it is possible that the vast majority, if not all, nodes in the network are mobile. In this case, a routing protocol will most likely have more options available during contact opportunities, and may not have to utilize each one [18], [27], [10], [23] An example of this type of network is a disaster recovery network where all nodes (generally people and vehicles) are mobile [13], [**?**]. A second example is a vehicular network where mobile cars, trucks, and buses act as communicating entities [18].

The third consideration is the availability of network resources. Many nodes, such as mobile phones, are limited in terms of storage space, transmission rate, and battery life. Others, such as buses on the road, may not be as limited. Routing protocols can utilize this information to best determine how messages should be transmitted and stored to not over-burden limited resources. Only recently has the scientific community started taking resource management into consideration, and this is still an active area of research.

The fourth consideration is whether to send message in their entirety or to split messages into smaller sizes. A message is split when it is forwarded in such a way that different parts (fragments) are routed along different paths (or across different contacts on the same path). This technique may reduce the delay or improve load balancing among multiple links. It is particularly relevant as messages can be arbitrarily large and may

not fit in a single contact. However, splitting complicates routing because, in addition to determining the sizes of the fragments, corresponding paths for the fragments also need to be determined.

2.4.4 Routing Objectives and Challenges

The routing objective of traditional routing schemes has been to select a path which minimizes some simple metric (e.g. the number of hops). For DTN networks, however, the most desirable metric to optimize/address during routing is not immediately obvious and varies depending on the application. An important objective for DTN, in general, is to increase the probability of bundle delivery, but reducing the delivery delay is also usually important for applications, due to mission deadlines or the validity period of the message itself (TTL). Storage management is also related to routing, as is energy efficiency. A critical challenge for DTNs is determining routes through the network without potentially having an end-to-end connection between the sender and receiver. To make communication possible, intermediate nodes temporarily store the data being transferred and forward it as the opportunity arises. Both links and nodes may be inherently unreliable (nodes may change their routes randomly) and disconnections may be long-lived. Moreover, buffer and bandwidth restrictions may force routing protocols to send discovery and topology information as sparingly as possible to avoid consuming resources. Routing protocol design must take these factors into consideration, all at the same time balancing the outcome of the performance expectations (i.e., delivery ratio, end-to-end delay, and

so on) of the network.

Furthermore, DTN routing algorithms need to know:

- When to send/forward a message - (opportunistically/periodically)
- Where to send a message - (next hop selection strategy)
- Which message to send/forward - (queuing and forwarding strategy)
- Which message to delete - (message priority)

2.4.5 Goals of DTN Routing Protocols

Despite the strategy a DTN routing protocol implements to accomplish the task of routing for a particular application, all DTN protocols have, more or less, the following common goals:

- *Low latency*: Latency is the time taken by the packet to reach its destination from its source.
- *Low latency jitter*: Latency jitter is the variation in latency, for real time applications such as streaming video, the requirement for low latency jitter is more important than the requirement of low latency.
- *High throughput*: Throughput can be defined as the number of data packets delivered per second. Throughput is affected by packets being dropped, and protocol data units that are used by protocols to set up communication with peers.

- *Low packet loss or High Reliability*: Packet loss causes decrease in throughput and increases latency.

- *Low convergence*: time in case of changes in network topology. It is necessary for routing algorithm to adapt to changes in network as quickly as possible, so that utilization of network resources is maximized.

- *Low routing overhead*: Routing overhead is caused by the update packets that are exchanged by routing protocols to convey network information to its peers. Routing overhead decreases throughput.

It is not possible for a routing protocol to achieve all the goals. Also, some of the above stated goals are conflicting in nature, for example to achieve low convergence time in case of change in topology certainly requires high routing overhead which in turn reduces the throughput for end to end communication.

2.4.6 Routing Strategies

Based on the above considerations, objective and goals of a DTN, there are several strategies routing protocol designs can take into consideration and choose to implement, accordingly:

- **Deterministic vs. stochastic**: Deterministic approaches to routing make use of exact and known mobility patterns to perform scheduling of messages over time-varying links. An example of deterministic DTNs are interplanetary satellite com-

munication networks where the exact times of communication opportunities between devices can be calculated due to known device movement. The network dynamics are deterministic and routing can be pre-computed. Stochastic approaches work based on probabilities that e.g. describe contacts of devices, or recurrence of devices to specific geographic areas. Dissemination-based protocols, as we will discuss shortly, are an example of the stochastic approach to DTN routing.

- **One-copy vs. n-copy**: DTN routing protocols differ in replication strategies, i.e., how many copies of a messages they create for forwarding. This, in turn, has a direct impact on the load incurred on the network. Some protocols generate just a single copy [16], others a fixed number limited by the sender [10], while others create an "infinite" number of messages [8].

- **Knowledge vs. replication**: The more knowledge of nodes there is available, or can be gathered and utilized, the less replication necessary. However obtaining knowledge of the network, especially for networks with large number of nodes in high load scenarios, imposes problems in case of communication overhead and resource utilization (regarding knowledge maintenance; e.g. when metrics are fresh or stale).

- **Random mobility vs. structured mobility**: Protocols may take into consideration the characteristics of the mobile nodes, which may follow random mobility patterns such as Random Waypoint (RWP) or; may follow structured mobility patterns such as Community-based models (CMM) or Map-based mobility models

(MBM).

2.4.7 Overview of Opportunistic Routing

In recent years numerous routing protocols have been proposed for DTNs [8] - [26], each with solutions depending on, but not limited to, the amount of tolerable delay by the application, link characteristics, contact types, and resource availability. Different mechanisms are applied depending on whether the network is primarily of mobile ad-hoc nature (e.g., mobile devices carried by humans) or is based upon a (fixed or mobile) infrastructure (e.g., space networks, bus networks). Mixed networks exist as well (e.g., mobile users supported by infrastructure nodes).

Numerous papers, articles and journals have been published regarding the classification of DTN routing protocols, where authors have categorized the protocols based on criteria they independently deem important. In our review of DTN routing protocols, we mainly focus on a category of DTN networks called ***opportunistic networks*** [1] (as opposed to scheduled DTNs, which employ comparatively trivial routing solutions).

In opportunistic networks, no assumption is made with regard to the existence of a complete path between two nodes wishing to communicate. Source and destination nodes might never be connected to the same network, at the same time. Furthermore, nodes may not possess or acquire any knowledge about the network topology (e.g. hop count, fresh routes), which is necessary in traditional MANET routing protocols. Routes are built dynamically, while messages are en route between the sender and the destination(s),

and any possible node can opportunistically (i.e. when the chance arises) be used as a next hop. An "opportunity" refers to the possibility to transmit to an intermediate node (next hop) which is nearer to the destination node with respect to the source node. Figure 2.18 [8] shows an example of hop-by-hop opportunistic forwarding. In the figure, a source, S, wishes to transmit a message to a destination but no connected path is available in part (a). Carriers, $C1 - C3$ are leveraged to transitively deliver the message to its destination at some later point in time as shown in (b).

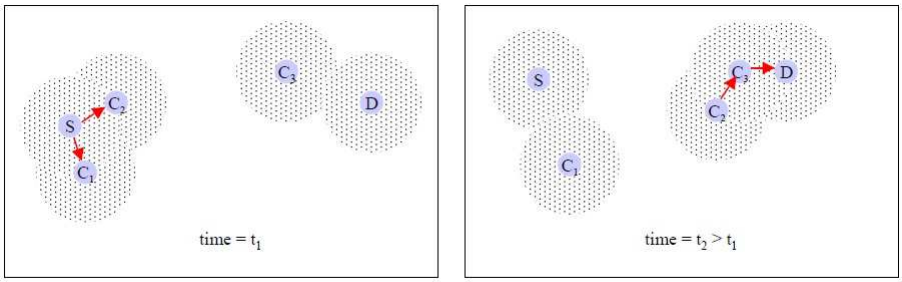


Figure 2.18: Routing in opportunistic networks.

The terms "opportunistic networks" and "delay-tolerant networks" are sometimes used interchangeably. However [1] cites opportunistic networks to correspond to a more general concept, which includes DTNs.

The distinction is as follows:

- Routes in DTNs are typically computed via legacy-Internet techniques by taking into consideration the link unavailability as another component of link cost. In opportunistic networks, routes are computed at each hop while a packet is forwarded. Each node receiving a message for an eventual destination exploits local knowledge

to decide which is the best next hop, among its current neighbors, to reach the eventual packet destination. When no forwarding opportunity exists (e.g., no other nodes are in the transmission range, or the neighbors are evaluated not suitable for that communication), the node stores the message and waits for future contact opportunities with other devices to forward the information.

- DTNs assume the knowledge of Internet-like topologies, in which some links between gateways could be available just at certain (possibly unspecified) times. In opportunistic networks, as it is not mandatory to have a priori knowledge about the network topology, each single node acts as a gateway.

This makes opportunistic networks a more flexible environment than DTNs, and calls for a more radical revision of legacy routing approaches designed for the Internet or for well-connected MANETs.

Researchers have implemented a number of real-application scenarios (projects) in opportunistic networks. Among these are ZebraNet [27], DakNet [75], Haggle [2], and SWIM [26].

- **ZebraNet** [27]: is a wildlife tracking application aimed at monitoring wild species (zebras, wearing special collars) in unmanned scenarios (its deployment scenario is the vast savanna area of central Kenya). The base station consists of a mobile vehicle for the researchers, which periodically moves around in the savanna and collects data from the zebras encountered. Two alternative protocols have been considered for data collection in ZebraNet.

The first one is simple *flooding* - each collar sends all its data to each encountered neighbor until the data eventually reach the base station.

The second one, a *history-based protocol* - proposes that each node selects only one of its neighbors as relay for its data. The selected node is the one with the highest probability to eventually encounter the base station. Each node is thus assigned a hierarchy level (initially zero) that increases each time it encounters the base station, and conversely decreases after not having seen the base station for a certain amount of time. When sending data to a relay (intermediate) node, the neighbor to be selected is the one with the highest hierarchy level.

- **DakNet** [75]: to provide intermittent Internet connectivity to rural and developing areas where they typically represent the only affordable way to help bridge the digital divide. Kiosks are built up in villages and equipped with digital storage and short-range wireless communications.

  Periodically, mobile access points (MAPs) mounted on buses, motorcycles, or even bicycles pass by the village kiosks and exchange data with them wirelessly. MAPs can upload any sort of request or data stored at the kiosks, and download them to the Internet when passing by an access point (AP) in a nearby town. Similarly, MAPs may download, from the Internet, the requested information and bring it to villages.

  DakNet has the potential to support Internet/Intranet messaging (e.g., email,audio/video messaging, and mobile e-commerce), distribution of information (e.g., public health

announcements, community bulletin boards, news, and music), and collection of information (e.g., environmental sensor information, voting, health records, and census).

- **Haggle** [2], [76]: is a project funded by the European Commission for studying the properties of Pocket Switched Networks (PSNs) [2], and to measure and model pair-wise contacts between devices (e.g., cell phones and PDAs that users carry in their pockets).

  Pair-wise contacts between users/devices can be characterized by the means of two parameters: *contact durations* and *intercontact times*. For the purpose of the project, the duration of a contact is the total time that a tagged couple of mobile nodes are within reach of each other, and thus have the possibility of communicating. Intercontact time is the time in between two contact opportunities between the same couple of tagged devices.

  To characterize contact durations and intercontact times occurring in real-world environments, different sets of traces were collected and analyzed. Traces were inferred from the logs collected by the APs of some university campuses; directly logged by Bluetooth devices carried by students and researchers in their university and laboratories; and also by the participants of some international conferences. The analysis of *all* the traces led to an important result stating that both intercontact times and contact durations were characterized by heavy-tailed distribution functions approximately following power laws. The implications of the results are

further discussed in [76] and also in [56].

- **SWIM** [26]: similar to ZebraNet, whales (with special tags) are the wild species to be monitored. Data is replicated at each pair-wise contact between whales (similar to what happens in the flooding protocol of ZebraNet) and finally arrives to special SWIM stations that can be fixed (on buoys) or mobile (on seabirds). Hence, both whale-to-whale and whale-to-base-station communications are allowed. From the SWIM stations, data are eventually forwarded onshore for final processing and utilization. No real deployment currently exists for SWIM. however, extensive simulations have been performed and show that mobile SWIM stations have shown better performance than fixed SWIM stations.

In opportunistic networks, the concepts of routing and forwarding are mixed together, since routes are actually built while messages are forwarded. In remainder of this thesis we will use the terms "routing" and "forwarding" interchangeably.

2.4.8 Classification of Opportunistic Routing Protocols

The design of efficient routing strategies for opportunistic networks is generally complicated due to the absence of knowledge about the topological evolution of the network. Routing performance improves when more knowledge about the expected topology of the network can be exploited. Unfortunately, this kind of knowledge is not easily available in opportunistic networks.

A common technique used to maximize the probability of a successful message transfer

is to replicate/disseminate/flood many copies of the message in hopes that one will succeed in reaching its destination. This is feasible only on networks with large amounts of local storage and inter node bandwidth relative to the expected traffic. In many common problem spaces, this inefficiency is outweighed by the increased efficiency and shortened delivery times made possible by taking maximum advantage of available unscheduled forwarding opportunities. In others, where available storage and inter node throughput opportunities are more tightly constrained, a less indiscriminate algorithm is required. The storage duration in opportunistic networking is typically longer than in classical routing approaches because nodes have to wait for a communication opportunity to occur. The mobility of nodes helps create communication opportunities and reduce the storage duration and the end-to-end delay of transmissions.

Figure 2.18 [1] shows a categorized view of the routing/forwarding algorithms in opportunistic networks. For brevity, we will only discuss the dissemination/flooding-based routing protocols in detail. We will discuss each of the branches briefly, but refer to the respective citations for further details.

Figure 2.19 classifies opportunistic networks into algorithms designed for ad hoc networks *without infrastructure*, and algorithms in which the ad hoc networks exploit some form of *infrastructure* to opportunistically forward messages.

Infrastructure-less approaches can be further divided in *dissemination based* and *context based* approaches. *Dissemination-based* algorithms are essentially forms of controlled flooding, and differentiate themselves for the policy used to limit flooding. *Context-based*

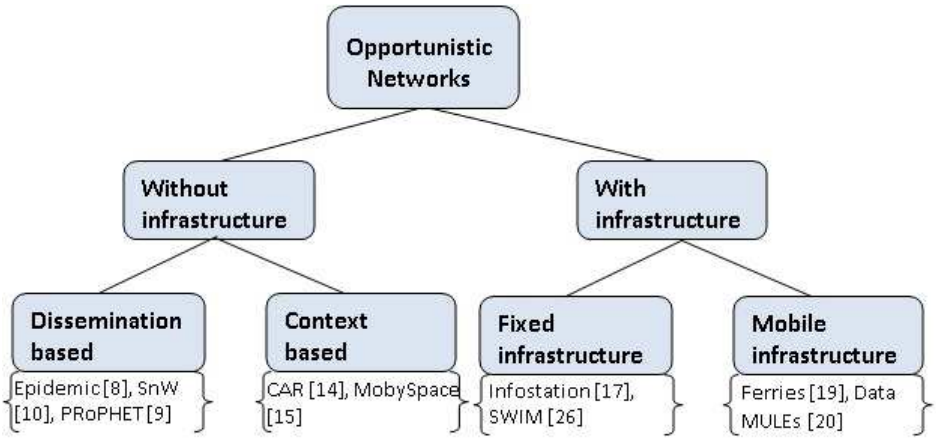


Figure 2.19: A classification of routing techniques for opportunistic networks.

approaches usually do not adopt flooding schemes, but use knowledge of the context that nodes are operating in to identify the best next hop at each forwarding step.

Algorithms that exploit some form of infrastructure can be divided (depending on the type of infrastructure they rely on) into *fixed infrastructure* and *mobile infrastructure* approaches. In both cases the infrastructure is composed by special nodes that are more powerful with respect to the other nodes commonly present in the ad hoc network. They have high storage capacity and hence they can collect messages from many nodes passing by, even for a long time. They also have high levels of energy, and may be seen as "always on" nodes. Nodes of a *fixed infrastructure* are located at specific geographical points whereas nodes of a *mobile infrastructure* move around in the network following either pre-determined known paths or completely random paths.

We discuss opportunistic routing protocols with infrastructure first, then proceed to describe infrastructure-less ones. Infrastructure-less routing protocols can be further

divided into context-based versus replication based approaches. As we envision our proposed framework to be efficient under scenarios suitable for dissemination based routing, we will discuss three popular dissemination-based protocols: Epidemic [8], ProPHET [9] and Spray-and-Wait [10], in detail.

2.4.8.1 Routing with Infrastructure

Infrastructure, with consideration to opportunistic routing, can be divided into fixed infrastructure or mobile infrastructure schemes.

- **Fixed Infrastructure** In infrastructure-based routing, a source node wishing to deliver a message generally keeps it until it comes within reach of a base station belonging to the infrastructure, then forwards the message to it. Base stations are generally gateways towards less challenged networks, e.g., they can provide Internet access or they can be connected to a LAN. Hence, the goal of an opportunistic routing algorithm is to deliver messages to the gateways, which are supposed to be able to find the eventual destination more easily.

  Two variations of the protocol are possible. The first one works exactly as described above, and only node-to-base-station communications are allowed. As a result, messages experience fairly high delays. The classic example of this approach is the Infostation model [17]. A second version of the protocol allows both node-to-base-station and node-to-node communications. The Shared Wireless Infostation Model (SWIM) [26] is an example of this approach.

- **Mobile Infrastructure** This category of opportunistic routing is also known as *carried-based* algorithms as base stations actively participate as carriers of messages.

  In carrier-based routing, nodes of the infrastructure are mobile data collectors. They move around in the network area, following either pre-determined or arbitrary routes, and gather messages from the nodes they pass by. These special nodes are referred to as *carriers, supports, forwarders, MULEs*, or *ferries*. They can be the only entities responsible for message delivery, when only node-to-carrier communications are allowed, or they can simply help increasing connectivity in sparse networks and guaranteeing that also isolated nodes can be reached. In the latter case, delivery of messages is accomplished both by carriers and ordinary nodes, and both node-to-node and node-to-carrier communication types are allowed.

2.4.8.2 Routing without Infrastructure

In the case of little or no infrastructure availability, due to environmental, geographical or other reasons, infrastructureless routing schemes can be divided into context-based or dissemination-based routing, and are described below.

- **Context-based Routing** Context-based routing [14], [15] exploits information about the context that nodes are operating in to identify suitable next hops towards the eventual destinations e.g., the home address of a user is a valuable piece of context information to decide the next hop. The usefulness of a host as next hop for a message is hereafter referred to as utility of that host. Context-based rout-

ing techniques are generally able to significantly reduce the messages duplication with respect to dissemination-based techniques. On the other hand, context-based techniques tend to increase the delay that each message experiences during delivery. This is due to possible errors and inaccuracies in selecting the best relays. Moreover, utility-based techniques have higher computational costs than disseminationbased techniques. Nodes need to maintain a state to keep track of the utility values associated to all the other nodes in the network (i.e., all the possible destination nodes), and hence need storage capacity for both state and messages. Finally, the cost to hold and update the state at each node should also be considered in the overall protocol overhead.

- **Dissemination-based Routing** Routing techniques based on data dissemination (a.k.a., *flooding*) perform delivery of a message to a destination by simply diffusing it all over the network. The heuristic behind this policy is that, since there is no knowledge of a possible path towards the destination nor of an appropriate next-hop node, a message should be sent everywhere. It will eventually reach the destination by passing node by node. Dissemination-based techniques obviously work well in highly mobile networks where contact opportunities, which are needed for data diffusion, are very common. They tend to limit the messages delay, but they are also very resource hungry. Due to the considerable number of transmissions involved, dissemination-based techniques suffer from high contention and may potentially lead to network congestion. To increase the network capacity, the spreading radius

of a message is typically limited by imposing a maximum number of relay hops to each message, or even by limiting the total number of message copies present in the network at the same time. When no relaying is further allowed, a node can only send directly to destination when/in case met.

In our proposed framework, we evaluate the following three dissimenation-based routing protocols: Epidemic, PRoPHET, and Spray-and-Wait (SnW). Following are the descriptions of these protocols.

Epidemic [8]

In the Epidemic Routing protocol messages diffuse in the network by means of pair-wise contacts between individuals/nodes. Each node possesses a buffer to store both the messages it generates and the incoming messages from other nodes that need forwarding. Messages are ordered in a list which is accessible through a hash table and is generally managed according to a FIFO policy. A summary vector describes which entries of the hash table correspond to actual messages and further contains a compact representation of them all.

Each node has its own IDentifier (ID). Whenever two nodes come into communication range of each other (i.e., a pair-wise contact occurs) the one with the smaller ID, say node A, delivers its summary vector to the other node with the greater ID, say node B. Node B contrasts the received summary vector with its own summary vector and builds up a list with those messages stored by node A that it has not received yet (messages are

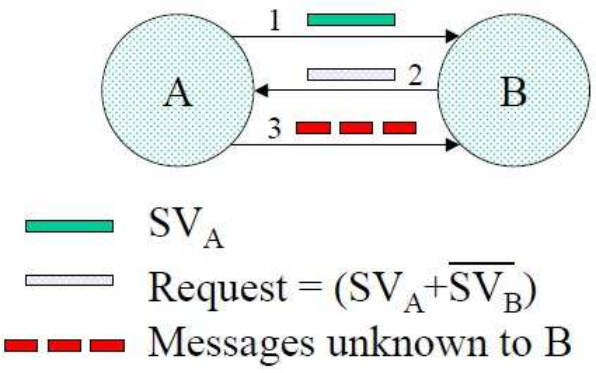


Figure 2.20: Message forwarding in the Epidemic routing protocol

discriminated by the means of unique identifiers and hop counts). Node B then makes request of the messages it misses to node A and afterwards waits for node A to send them to it. After messages have transferred from node A to node B, node B sends its proper summary vector to node A that repeats the same procedure as node A. This process has been depicted in Figure 2.20.

The choice of messages to ask for to the other node is based on considerations on the total buffer size available. A more intelligent choice may also keep into account the destination node of the announced messages. So, a node will preferentially request to the encountered node those messages that have as destination a node which is most frequently encountered. To avoid repeating this procedure uselessly, nodes are recommended to maintain a list of the most recently met nodes and to start exchanges only with nodes that do not belong to this list.

Dissemination throughout the network guarantees that messages eventually arrive to their actual destinations. The dissemination process is somehow bounded because each

message when generated is assigned a hop count limit giving the maximum number of hops that that message is allowed to traverse till the destination. Moreover, the delivery latency is more likely to be low. When the hop count limit is imposed to be low, a message cannot be frequently forwarded and mostly moves from point to point thanks to the mobility of the nodes that store it (delivery exploits mobility more than wireless transmissions).

Epidemic routing allows delivery of messages in disconnected ad hoc networks where traditional routing algorithms generally fail. However, this protocol tends to consume great deal of resources, specifically storage capacity, network bandwidth for transmissions, and energy for both message storing and sending. Clearly, the efficacy of epidemic routing, as originally conceived, is severely limited by the scarcity of the storage capacity available at nodes. After generation of a message, many copies of it are spread all over the network and continue to be stored even after arrival of one of the replicas to the destination. This obviously leads to memory wastage.

PRoPHET [9]

In the Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET), each node holds a delivery predictability table with the probabilities of successful delivery towards each known node in the network. Also in this case, the probability to deliver a message to a certain destination node increases whenever it comes within sight, and decreases over time in case no meeting occurs.

The *Delivery Predictability* at any node a, for each know destination node b is indicated as $P_{(a,b)} \in [0, 1]$. This is a probabilistic metric that indicates how likely it is that node a will be able to deliver a message to destination b.

The operation of PROPHET is similar to that of Epidemic Routing. When two nodes meet, they exchange *summary vectors* which in this case also contain the delivery predictability information stored at the nodes. This information is used to update the internal delivery predictability vector as described below, and then the information in the summary vector is used to decide which messages to request from the other node based on the forwarding strategy used.

The delivery predictability metric, $P_{(a,b)}$ has three properties, based on which it is (re)calculated. The three properties are *update*, *aging*, and *transitivity*. The calculation of the delivery predictability based on these properties are as follows:

- **Update**:

  Every time two nodes a and b meet each other, their delivery predictabilities increase as follows (Eq. 2.1):

$$P_{(a,b)} = P_{(a,b)_{old}} + (1 - P_{(a,b)_{old}}) \times P_{init} \tag{2.1}$$

  $P_{(a,b)_{old}}$ is the last known delivery-predictability value, whereas $P_{init} \in [0, 1]$ is an initialization constant. This *update* is done so that nodes that are encountered more often have a higher delivery predictability.

- **Aging**:

  If a pair of nodes does not encounter each other in a while, they are less likely to be good forwarders of messages to each other, thus the delivery predictability values must age, being reduced in the process. The aging equation is shown in Eq. 2.2:

$$P_{(a,b)} = P_{(a,b)_{old}} \times \gamma^k \tag{2.2}$$

  $\gamma \in [0, 1)$ is the aging constant, and k is the number of time units that have elapsed since the last time the metric was aged. The time unit used can differ, and should be defined based on the application and the expected delays in the targeted network.

- **Transitivity**:

  The *transitive* property is based on the observation that if node a frequently encounters node b, and node b frequently encounters node c, then node c probably is a good node to forward messages destined for node a. The transitivity equation is shown in Eq. 2.3:

$$P_{(a,c)} = P_{(a,c)_{old}} + (1 - P_{(a,c)_{old}}) \times P_{(a,b)} \times P_{(b,c)} \times \beta \tag{2.3}$$

  $\beta \in [0, 1]$ is a scaling constant that decides how large impact the transitivity should have on the delivery predictability.

Every time two nodes are within transmission range of each other, they exchange the summary vectors of the messages they store. They add to each message entry their

delivery predictability related to the destination node of that message (i.e., how likely it is that it can successfully forward the message). Both nodes, then, decide which messages to request to the other, based on these probabilities. Specifically, node a requests a message from node b which is destined to node c only if the delivery predictability from node a to node c is higher than the delivery predictability from node b to node c.

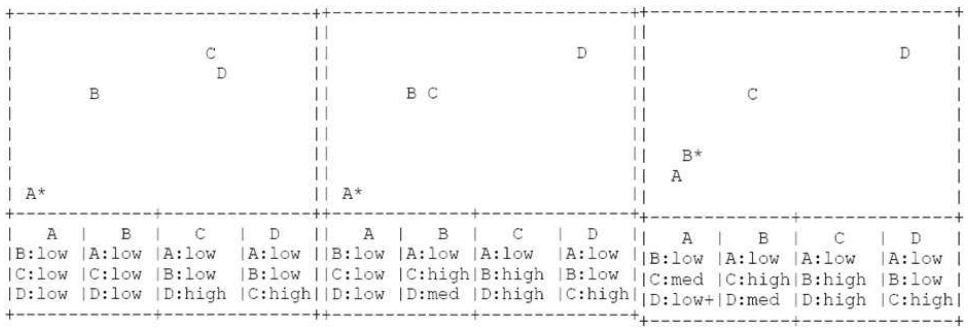


Figure 2.21: Message forwarding in the PRoPHET routing protocol

Figure 2.21 shows the delivery predicitbilty tables of four mobile nodes in the network, using the PRoPHET protocol for message forwarding. The source A needs to send the message to destination D. Each node has a delivery predictability stored for other nodes in the networks. The values (high, low, medium) used for $P_{(a,b)}$ in this particular example are used to give a conceptual view of the probability a has of delivering a message to b. A will choose to forward to node B if it has a higher value stored for sending the message to D. In this example, A forwards the message to B when it is encountered, as upon exchanging summary vectors A finds that $P_{(B,D)} > P_{(A,D)}$, thus determining node B to be a better candidate for delivering the message to the eventual destination D.

PRoPHET outperforms Epidemic routing in terms of both delivery success rate and delay experience of messages. It also introduces far less *traffic overhead* than Epidemic routing to manage forwarding. However, for large number of nodes in high load scenarios, the exchange and maintenance of delivery predictability vectors imposes problems in case of communication overhead and resource utilization.

Spray-and-Wait [10]:

Spray and Wait (SnW) protocol is an n-copy routing protocol. This routing algorithm consists of two phases:

- *Spray phase*: for every message M originating at a source node, K copies of the message are initially spread (forwarded by the source and possibly other nodes receiving a copy) to "K" distinct relays (i.e. intermediate nodes).

- *Wait phase*: if the destination is not found in the spraying phase, each of the K nodes carrying a message copy performs direct transmission (i.e. will forward the message only to its destination).

This definition of Spray and Wait leaves open the issue of how the K copies are to be spread initially. A number of different "spraying" heuristics can be envisioned. Two modes have been proposed by the authors of the protocol.

- *Normal mode*: In this case, the sender node replicates a message to all nodes that are encountered. Only K nodes get copies as there are K message copies available.

- *Binary mode*: In this case, out of K copies, $K/2$ copies are stored at the sender node and the remaining copies are forwarded to all first encountered nodes. These $K/2$ stored copies are then relayed in the same binary manner until a single copy is left and the last copy is forwarded to the final destination.

Authors in the paper [10] discuss models to optimize and choose the K value in SnW to meet performance constraints, for example:

- Choosing K to achieve a required expected delay
- Estimating K when network parameters are unknown

Spray and Wait routing decouples the number of copies generated per message, and therefore the number of transmissions performed, from the network size.

Spray and Wait could be viewed as a trade-off between single and multi-copy schemes. It combines the speed of epidemic routing with the simplicity and thriftiness of direct transmission.

Simulation results on SnW shows its performance is better with respect to both number of transmissions and delay than all other practical single and multi-copy dissemination-based routing schemes, in most scenarios considered.

2.5 Related Work

It can be concluded from the discussion in the previous section that opportunistic networking in DTNs utilize node mobility to achieve message delivery. Nodes forward messages

only when they encounter the appropriate relay or the destination node. Due to this dependence on the mobility of nodes participating in the network, understanding mobility characteristics such as inter-contact times and contact duration's of mobile nodes plays an important role in the analysis of routing algorithms and the overall performance of the network.

Routing algorithms can exploit mobility, but they can not essentially change the mobility characteristics inherent of a networks participating nodes. Routing algorithms build solutions "around" these characteristics. Our proposed sticky transfer framework derives motivation from this fact.

We discuss the inter-contact time, contact duration and delay characteristics of DTNs next.

2.5.1 Delay and Encounter Characteristics

One of the most important performance measures of a data communication network is the average delay required to deliver a packet from origin to destination. Furthermore, delay considerations strongly influence the choice and performance of network algorithms, such as routing. Delays of traditional data and communication networks mostly focus on packet delay within the communication subnet (i.e., the network layer). The delay is usually measured as the sum of delays on each subnet link traversed by the packet. However, in the absence of stationary infrastructure or connected network components, such as in opportunist DTNs, delay measurements are redefined.

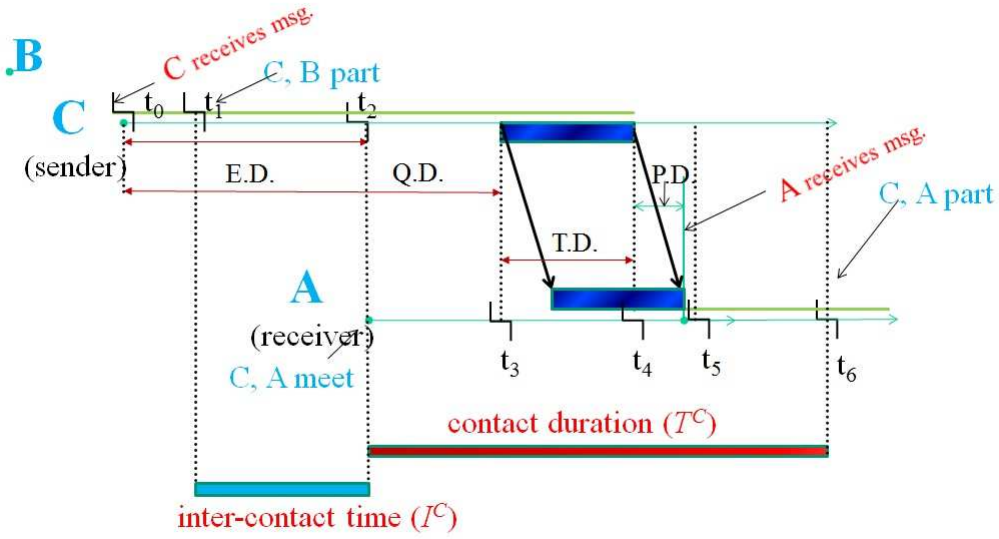


Figure 2.22: Delay characteristics of a DTN

Figure 2.22 depicts delay parameters in an opportunist DTN and the characteristics of inter-contact time and contact duration based on those parameters.

We observe the four delay components of Figure 2.22, where C is the sender of a message and A is the receiver and each discrete event is placed on an ordered timeline from t_0 to t_6.

The delay parameters are:

1. *Encounter Delay* (E.D.): The encounter delay is the time between when a message is received (if intermediate hop)/generated at a mobile host to the time the mobile host carrying the message comes into contact with another mobile host. This 'coming into contact' is an encounter. In Figure 2.22 node C receives a message at time t_0

and comes into contact with node A at time t_2; (t_0 - t_0) is the encounter delay.

2. *Queuing Delay* (Q.D.): The queuing delay is the time between when the message is received (if intermediate hop)/generated at the node and assigned to a queue for transmission to the time it starts being transmitted. During this time, the message waits in the nodes buffers while other messages in the transmission queue are transmitted. In Figure 2.22, the transmission of the message from C to A begins at time t_3; the queuing delay is (t_3 - t_0). Queuing delay also includes the E.D.

   Transmission delay (T.D.) and propagation delay (P.D) are the same as in opportunistic networks as in existing networks based on packet switching.

3. *Transmission Delay* (T.D.): The transmission delay is the amount of time required to put an entire message into the communication medium (in our case, the wireless link). It is essentially the delay caused by the data-rate of the link, and can be It can be computed by the following equation (Eq. 2.4):

$$T.D. = \frac{L}{R} \tag{2.4}$$

   where, L is the length of a packet in bits and R is the transmission rate in bits per time unit.

   In Figure 2.22 (t_4 - t_3) is the transmission delay.

4. *Propagation Delay* (P.D.): Propagation delay is defined as the amount of time it

takes for a certain number of bytes to be transferred over a (wireless) medium. Propagation delay is the distance between the two routers/nodes divided by the propagation speed, and can be defined by the following equation (Eq. 2.5):

$$P.D. = \frac{d}{s} \tag{2.5}$$

where, d is the distance from the node to the next node and s is propagation speed of the medium.

In Figure 2.22 (t_5 - t_4) is the propagation delay.

In opportunistic DTNs, the encounter delay can be quite large, especially if the network is sparse. The success of message delivery highly depends on the likeliness of nodes encounters and the time elapsed between encounters greatly influences other performance metrics of the network, such as delay, storage occupancy, and so on. We define the two main contributing factors [21] that determine the performance of an opportunistic network based on encounters:

- *Average Inter-contact Time, I^C*: The *average inter-contact* time measures how frequently nodes encounter other nodes in the network. Specifically, it is the duration from the moment a node A moves out of the transmission range of node B until node A encounters another node (which could be B again). Inter-contact time depends primarily on node mobility and node density in the network. In sparse networks, the inter-contact time can be reduced by introducing special components, such as

ferries [19] or data mules [20], that move at relatively faster speeds on predefined routes and therefore increase contact opportunities.

- *Average Contact Duration,* T^C: The *average contact duration* is the length of time during which pair-wise nodes remain within the transmission range of each other on average in the network. The contact duration directly influences the capacity of opportunistic networks (e.g., DTNs) as it limits the amount of data that can be transferred successfully between nodes. By using the proposed sticky transfer framework, nodes can intelligently and cooperatively increase their contact durations to improve the capacity of the network by agreeing to brief, temporary modifications in their movement patterns and speeds.

In Figure 2.22, node C losses contact with node B at time t_1 and comes into contact with node A at time t_2; (t_2 - t_1) is the inter-contact time. Subsequently, node C looses contact with node A at time t_6. Therefore, the contact duration between C and A is (t_6 - t_2).

We can also observe from the above definitions and in Figure 2.22, the inter-contact time and contact duration directly impacts the throughput of the network. We can characterize the average throughput (THP), which is the maximum data rate that can be sent between two nodes as:

$$THP = \frac{T^C \cdot R}{(T^C + I^C)} \tag{2.6}$$

where, R is the transmission rate in bits per time unit.

Thus, the performance of DTNs can be improved by reducing the inter-contact time and increasing the contact duration, which is observed from Eq. 2.6.

Now, we address the following question:

Why is it important to address the encounter characteristics just discussed and how do they affect the performance of the DTN?

Routing algorithms have been proposed which can reduce the *inter-contact time* (meeting time in-between contacts) of nodes. However, irrespective of the forwarding technique used by routing algorithms, the actual time to transfer messages between two nodes is limited by the *contact duration* (duration of the contact) of the nodes, which depends inherently on node mobility.

Thus, when messages are being forwarded to pair-wise contacts, if the expected contact duration is not sufficient for the entire message to be transmitted, which can change at any moment due to node mobility, messages tend to fail to be forwarded to the next hop. This can cause the routing protocol to retransmit the message, thus wasting valuable bandwidth, resources and "opportunities" in the network.

For example, when using the PRoPHET routing protocol, referring back to Figure 2.21, A may encounter B. As determined by their delivery predectabilities, B is a suitable forwarder for A's message to D. However, if the contact duration when A and B are exchanging the message is not sufficient, the message transfer will be aborted. In this case, A may have to retransmit (see Section 2.3.3) the message. However, the next

encountered node may not be B (due to B moving out of the transmission range of A in the meantime), and may be some other node E, who's delivery probability to D is lower than that of A or perhaps 'NULL'. Hence, A has to postpone the forwarding of the message. This not only requires additional bandwidth consumption (and hence resource wastage on A's part) but also prolongs the delivery of the message to destination D.

Note that the PRoPHET routing protocol in this case can help node A to decide 'who' to forward to, but has no way to 'ensure' that A will actually be able to forward the message to the most suitable next hop.

Next, we address the question:

Why are we proposing our sticky message transfer framework and what will it solve regarding encounter delays and routing performance in DTNs?

One way to tackle the above problem is to allow smaller sized messages; i.e. fragment/split the message, to allow for forwarding within smaller contact duration's. However, as mentioned in Section 2.4.3, splitting complicates routing because, in addition to determining the sizes of the fragments, corresponding paths for the fragments also need to be determined. Moreover, if the message fragmentation size is fixed then the actual size of the splits may not be optimal to 'fit' within the contact duration.

We see the necessity to assist routing protocols in enchaining the performance of the DTN by providing a solution that can be implemented irrespective of the routing protocol being used to extended the contact duration to meet the required message transfer

duration.

Our proposed sticky transfer framework addresses the following:

- *Contact Duration vs. Required Transfer Duration*:

  In an opportunistic DTN, forwarding opportunities can be lost due to:

  – Lack of buffer space at the next hop: message gets dropped.

  – Limited bandwidth: there is not enough time to forward all messages in the queue while the two nodes are in range (contact duration).

  – MAC contention: more than one nodes within range are trying to access the media at the same time.

  – Interference: ongoing communications in the surrounding area contribute to the noise level [21].

  Any MACAW (Multiple Access with Collision Avoidance for Wireless) [77] protocol, such as the IEEE 802.11 RTS/CTS, could be used to cope with collisions and hidden/exposed terminal problems. Due to the rapidly declining cost of data storage [78], limited buffer is an issue that is becoming non-existent in practical scenarios. In this paper we deem it more important to address the issues of limited bandwidth, since it depends on a fixed property of the wireless channel.

- *Contact Duration vs. Number of Contacts*:

  The average delay of message delivery in DTNs can be decreased by increasing the contact duration. Providing enough time for messages to be exchanged in their

entirety eliminates the number of aborted messages (i.e., message exchanges that start, but do not complete because nodes move out of each others transmission range), thus reducing the necessity for retransmitting the message to subsequent encounters. Less intuitively, increasing the contact time reduces drops due to full buffers or expiring TTLs. Sticky transfers use coordination between nodes to achieve a contact duration long enough to ensure all data that needs to be transferred between pair-wise nodes are transferred before the contact is broken.

2.5.2 Packet-level Replication

Zhuo et al. [22] propose a packet-level replication protocol, which uses erasure coding to encode large messages into smaller packets, to address the problem of limited contact duration. However, this technique requires replication of packets at each node, which is expensive in a DTN. Our sticky transfer protocol requires no additional mechanism nor infrastructure other than the simple beaconing mechanism which has been used for many other purposes in wireless ad hoc networks. To the best of our knowledge, our work is the first to propose the concept of 'sticky' message transfers to extend contact durations.

Our proposed sticky transfer framework is independent of, but can function with, any DTN routing protocol. DTN routing protocols use different strategies in forwarding messages at each encounter. For example, dissemination/replication based routing protocols [8, 10, 23] create and forward multiple copies of a message when an encounter happens. In replication-based protocols, any node A can receive a copy of a message

from any other node B when they come in contact. Other protocols try to improve the performance by using information (such as encounter history) to intelligently forward a limited number of copies [9, 18, 25–27].

In our simulations, we show the performance of three replication based DTN routing protocols: Epidemic [8], PRoPHET [9], and Spray-and-Wait [10]. Our results have shown the proposed method to improve the performance of each protocol and enhance the reliability of the network, compared to each DTN routing protocols performance standalone.

2.6 Chapter Summary

In this chapter, we review the DTN architecture, opportunistic network routing protocols, and work related to our proposed sticky transfer framework.

We looked into the background of DTNs, discussed the contrast between the DTN architecture and the Internet architecture. We described the DTN bundle protocol (BP), which is the central component of the DTN overlay and presented an implementation of the bundle (BP) layer modules and their dependencies. We then discussed message-switched routing in a DTN by presenting several DTN routing protocols and distinguished DTN routing form MANET routing. We presented a classification of opportunist DTN routing protocols and discussed three prominent replication-based DTN protocols in depth. Finally, we stated issues related to our thesis objective and presented existing related work.

In the next chapter, we discuss our proposed framework in detail.

Chapter 3

The Proposed Sticky Transfer Framework and Protocol

In opportunistic networks such as DTNs, the message delivery performance, such as delivery ratio and end-to-end delay, highly depends on the time elapsed between encounters (i.e., inter-contact time) and the time two nodes remain in each other's communication range once a contact is established (i.e., contact duration). Inter-contact time and contact duration have been discussed in section 2.5.1. Limitations in the actual time to transfer data between nodes result from the inherent mobility of hosts, as illustrated by the example shown in Figure 3.1.

In this example, at time t node A has three variable-length messages {#1, #2, #3} queued in it's buffer to send to the next encountered node using a flooding-based routing strategy for message forwarding (see section 2.4.8.2). At time $t+\delta t$, A is presented with contact opportunity B as node B comes within the wireless communication range of A.

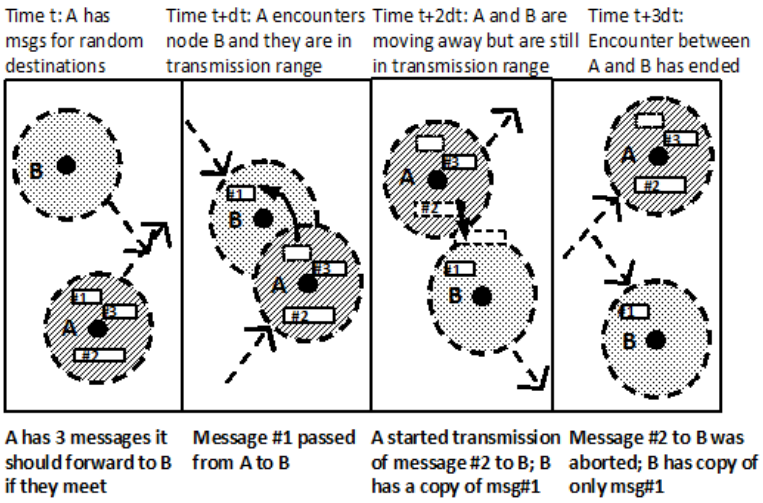


Figure 3.1: Data transfer limitations due to natural movement

A successfully transmits message #1 to B, and retains metadata of the copy of message #1 sent to B. At time $t+2\delta t$, A proceeds to transfer message #2 to B, while A and B are gradually moving out of each other's communication range. Due to their velocity, the contact duration to transmit message #2 is insufficient. Hence at time $t+3\delta t$, node A was unsuccessful to transmit message #2 to B, which may have been a better candidate (next hop node) for delivery on the time-varying path to the message's eventual destination. Also, node A now requires to re-transmit the message to the next viable encounter.

Moreover, the DTN Bundle Protocol specification [33] does not limit bundle size or specify content of bundles. Indeed, a bundle may: (i) contain a single file (e.g., a photograph), multiple files (such as small-sized engineering telemetry files), or a file segment (possibly part of high-quality video); (ii) be of fixed size determined by application type or network management procedures; (iii) be of variable size set by application, and contain-

ing a coherent bundling of application data. Key DTN functionality is that each bundle is kept in memory in its entirety, and is deleted upon receipt of acknowledgment for its successful delivery to the next node on the path to the destination (see section 2.3.6).

Fragmentation allows smaller sized messages to be forwarded within smaller contact duration's. However, as mentioned in Section 2.4.3, splitting complicates routing because, in addition to determining the sizes of the fragments, corresponding paths for the fragments also need to be determined, which is complicated in opportunistic networks. Moreover, if the message fragmentation size is fixed then the actual size of the splits may not be optimal to 'fit' within the contact duration.

When nodes move out of each other's transmission range, data transfer is discontinued. Thus, many messages which are ready for forwarding cannot be forwarded. Additionally, if a message happens to be in transit during the disconnection, the transmission is disrupted, which can result in an unsuccessful transmission. Therefore, insufficient contact duration during opportunistic communications results in underutilized contact opportunities (i.e. wasted contact time and bandwidth), which limits the capacity of the network. Furthermore, messages will stay longer in limited buffers, which may lead to message lifetime (TTL) expiration. Such problems are exacerbated in highly mobile DTNs (e.g. vehicular networks) that must handle large message sizes [17], [18].

To solve the above problem, we propose a novel framework called the *sticky transfer framework* for maximizing node contacts for message transfers in DTNs . In the proposed framework, to *'stick'* means mobile devices remain within the communication

range of each other longer than the contact duration that would be expected between them without using the framework. By using the proposed sticky transfer scheme, nodes can cooperatively increase the contact duration to improve the capacity of the network by agreeing to brief, temporary modifications in their movement patterns. Nodes adjust mobility at the instance of an encounter, and return to normal movement behaviors after the encounter is over. Thus, by encouraging cooperative and voluntary mobility changes, the sticky transfer mechanism can substantially improve network performance.

As part of the framework, we also propose a *sticky transfer protocol* which governs how nodes engage and participate in sticky message transfers. The protocol allows nodes to exchange information, upon encounter, and calculate a compatible mode and duration for message exchanges. The agreement is made based on information gathered about current network parameters and *user preferences* (discussed in section 3.3.2). Based on this mutual decision, nodes may remain within the transmission range of each other until message transfers are completed. The amount of messages that are transferred depends on the underlying routing protocols message selection and forwarding strategy and the agreed stick-time. Ideally, all the messages that the routing protocol decides to transfer.

In this chapter, we first state the definitions and assumptions, and then describe the concept and components of the sticky message transfer framework and protocol. We also explain how the proposed framework can be seamlessly integrated into the existing DTN architecture.

3.1 Definitions and Assumptions

The *natural contact duration* T_C is the length of time during which two nodes are expected to remain within the transmission range of each other, and can be estimated as follows. Consider two nodes A and B that are in contact (i.e., within the transmission range W of each other) and moving on a plane at angles of θ_A and θ_B ($0 \leq \theta_A, \theta_B \leq 2\pi$), and at speeds of v_A and v_B ($v_A, v_B > 0$), respectively. Let (x_A, y_A) and (x_B, y_B) be the coordinates of A and B, respectively. By projecting the speeds and directions of the two nodes along their movements, the natural contact duration T_C of the two nodes can be estimated to be:

$$T_C = \frac{-(ab+cd) + \sqrt{(a^2+c^2)W^2 - (ad-bc)^2}}{a^2+c^2} \tag{3.1}$$

where $a = v_A \cos\theta_A - v_B \cos\theta_B$, $b = x_A - x_B$, $c = v_A sin\theta - v_B sin\theta_B$, and $d = y_A - y_B$. This estimated value is the expected contact duration between the two nodes. Note that when $v_A = v_B$ and $\theta_A = \theta_B$, T_C approaches ∞. That is, A and B will always stick to each other.

We make the following assumptions regarding Equation 3.1:

– We assume that every node is equipped with a GPS, which helps to determine it's speed and direction of movement.

– All nodes have the same transmission range, W. This can easily be modified to include nodes with heterogeneous transmission ranges [40].

– A node is in the transmission range of only one node. If at time t_0, A comes into the transmission range of B and moves away from B at time t_1, then $T_C(A, B) = t_1 - t_0$.

If multiple nodes are in the transmission range of each other, we assume the mutual encounter sequence comes naturally from the order in which a nodes receives beacon messages [80] from other nodes. For example, if at time t_0, A comes into the transmission range of B and C and receives B's beacon message first, A will finish sticky transfers with B first and then begin sticky transfers with C (assuming that C has not already moved away). Assuming that at time t_1, A moves away from the transmission range of B and C, then $T_C(A, B) + T_C(A, C) = t_1 - t_0$. However, if C has already moved out of range while A is transferring messages to B then $T_C(A, C) = 0$.

On the other hand, the time required for A to complete transferring all messages scheduled by the routing protocol to B is the *required transfer duration* T_R. Let R be the transmission rate of the nodes. (The calculation can easily be extended to nodes transmitting at different rates). If node A has p messages to send to node B, B has q messages to send to A, and M_i denotes the size of message i, then the required transfer duration between A and B is

$$T_R = \frac{\sum_{k=1}^{p} M_k + \sum_{l=1}^{q} M_l}{R} \tag{3.2}$$

Assuming that the message transfer starts immediately after nodes encounter each other, if $T_C(A, B) < T_R(A, B)$ then message aborts are probable and not all of the

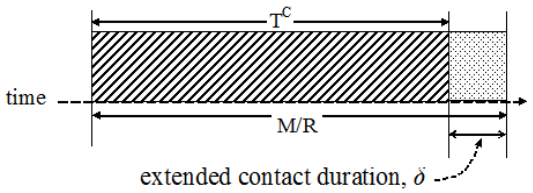


Figure 3.2: Extending the contact duration

messages that A wants to forward to B can be transferred within the estimated contact time.

We envision the sticky transfer protocol to be a part of the bundle layer (see Section 2.3.2.4), and hence any lower layer technology for each layer in the protocol stack can be implemented. For example,

– At the MAC layer, any MACAW (Multiple Access with Collision Avoidance for Wireless) [77] protocol, such as IEEE 802.11 RTS/CTS, could be used to cope with collisions and hidden/exposed terminal problems.

– At the transport layer, we can assume a point-to-point reliability protocol such as *Saratoga* [81], [82]. Saratoga is a UDP-based protocol, supporting a Selective Negative Acknowledgment (SNACK) with holes-to-fill strategy [81] for reliable retransmissions. Additionally, for reliable transfer, data packets are sent using UDP with checksums turned on.

– 802.11b, or 802.11g can be used at the physical layer.

3.2 The Concept of Sticky Message Transfers

To maximize the use of valuable contact opportunities, the sticky transfer mechanism allows nodes to come to a mutual agreement on the time during which they will remain in each other's communication range. Once all message transfers are complete, nodes may "unstick" and resume their natural mobility patterns. Assuming that T_C is the natural but insufficient contact duration for the message transfer, the additional time the nodes should remain in contact beyond the natural contact duration is $\delta = T_R - T_C$, where $T_R = M/R$ and M is the total size of all the messages to be transferred between the two nodes (Fig. 3.2). We call δ the *stick duration*, which is calculated by nodes using the sticky transfer protocol described later on in section 3.5.

We expect this mechanism to improve the performance of the network in two ways.

First, sticky transfers will be able to deliver messages faster in the network, hence minimize the end-to-end delay. For example, if a routing protocol wants to forward K copies of a message to different nodes in order to improve delivery, the sticky transfer mechanism ensures that the K copies are forwarded faster; in fact, they will be forwarded during the first K contacts with encountered nodes. This leads to lower latencies and higher delivery ratios, which improves network performance.

Second, sticky transfers will minimize the number of message aborts, improving message delivery ratio and network resource utilization, as it allows the atomic transfer of messages.

The sticky transfer scheme is optional. Any user may opt-in or opt-out from exchanges

using the sticky transfer protocol at any time. Several applications exist where sticky transfers can be used to improve performance: (1) robots in a region survey application may be programmed to stick with each other for a certain stick time to improve message transfers, (2) emergency response team members could be asked to stop or follow each other for a while to improve the network performance, or (3) people may be asked to stick with each other to help in a socially-aware forwarding protocol, thus helping with data delivery in their community.

3.3 The Proposed Sticky Transfer Framework

The sticky transfer framework consists of three components: *sticky modes*, *user preferences*, and *compatibility lists*. Sticky modes are modes of operation defining users agreement to modify their natural movement. A *user preference* consists of an ordered list of acceptable sticky modes. Modes set as preferences represent how the users would respond to a sticky transfer request. A compatibility list, stored on each device/node/user, defines if two modes chosen by two encountered nodes allow for the engagement in a sticky transfer. A schematic of the framework is shown in Fig 3.3. We discuss each of these components as follows.

3.3.1 Sticky Modes

Two neighbor nodes can "stick" to each other by reducing their relative speed so that they remain within the transmission range of each other for the required transfer duration. The

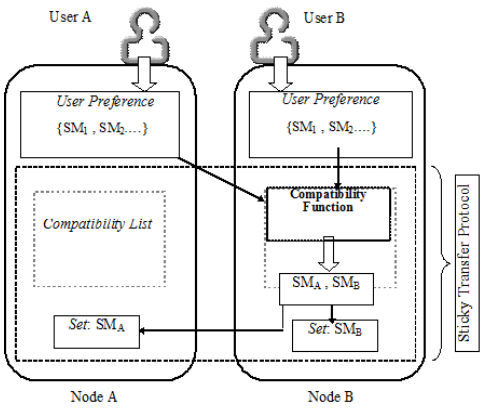


Figure 3.3: The sticky transfer framework

relative speed of the two nodes can be reduced by changing the speed and/or movement direction of one or both nodes. We define five sticky modes: *Stop, Follow me, Follow you, Slow down* and *No stick*.

The *Stop* (STP) mode is implemented by changing the relative speed of two nodes to zero. One way to achieve this is to change both of the nodes velocity to zero, i.e., stopping the nodes. Another way of achieving zero relative speed is to allow the two nodes to move at the same speed as the other in the same direction, i.e., one node follows the other node. The mode of the node which is followed by the other node is called *Follow me* (FLW1) mode. The mode of a node that adjusts its speed and direction to the other node's speed and direction in order to follow it is called *Follow you* (FLW2). When a node reduces its speed to match that of a slower moving node, its mode is called *Slow down* (SLW). Finally, a node may not agree to stick for message transfers. This mode is called *No Stick* (NO_STK).

Users (e.g., network administrators) can set one or more sticky modes in mobile nodes (e.g., sensors or robots), which make decisions based on the pre-defined modes and collected information (e.g., mobility speeds, movement directions, buffer sizes, and message sizes).

3.3.2 User Preferences

When setting sticky modes in mobile nodes, a user (e.g., network administrator) may not be able to select some of the modes at all. For example, a robot performing region surveys may not be able to use FLW2 mode due to its fixed route and schedule, but it may be able to use SLW mode for a very short duration. On the other hand, emergency response team members in a disaster stricken area may choose to accommodate all modes and set a low priority for NO_STK mode to ensure the highest level of cooperation among team members. We assume that nodes will have sticky mode preferences set according to the application before engaging in a mission.

A user preference consists of an ordered list of acceptable sticky modes. The order defines the priority of user preferences, with higher priority modes coming first in the list. In the framework (Fig. 3.3), users A and B have input and stored their preferred sticky modes (i.e. SM_1, SM_2 , and so on). When setting preferences, some nodes may not be able to select some of the stick modes at all. For example, a robot performing region surveys may not be able to use FLW2 mode due to its fixed route and schedule but it may be able to use SLW mode for a very short duration. On the other hand, emergency

Table 3.1: Sticky mode compatibility

SM_A	SM_B			
	STP	SLW	FLW1	FLW2
Stop (STP)	√	√	×	√
Slow down (SLW)	√	√	√/×	√
Follow me (FLW1)	×	√/×	×	√/×
Follow you (FLW2)	√	√	√/×	×

response team members in a disaster stricken area may prefer all modes and set a low priority for NO_STK, to ensure cooperative rescue operations. We assume that nodes will have sticky preferences set according to applications before engaging in any mission.

3.3.3 Compatibility List

Among the five sticky modes defined above, modes may or may not be compatible depending on the speeds and movement directions of the nodes involved in the negotiation. Sticky transfers are technically possible when nodes have *compatible* modes that allow for the mobility of each to mutually extend the contact. Two modes may or may not be compatible based on their speed limitations and movement direction. For example, when A's mode allows SLW and B's mode allows FLW1 then they are compatible if B's speed is slower than A's speed. They are not compatible if B's speed is faster than A's speed or both are not moving in the same direction.

For this reason, we construct a table that determines the compatibility between any two sticky modes (Table 3.1). In the table, $\surd$ indicates mode compatibility, $\times$ indicates incompatibility and $\surd/\!/\times$ indicates the modes may sometimes not be compatible due to user limitations. When implementing the framework, each node has a copy of the compatibility table. Among compatible modes, the most preferred modes are used during the sticky transfer. If compatible modes cannot be found, then sticky transfers are not possible and the nodes will exchange messages in the normal transfer mode, possibly with limited contact time.

3.4 The Sticky Transfer Framework within the DTN Overlay Architecture

The DTN architecture (section 2.3) was designed as an overlay to accommodate not only network connection disruption, but also to provide a framework for dealing with heterogeneity [28]. In Fig. 3.4 we show a conceptual overview of the sticky transfer

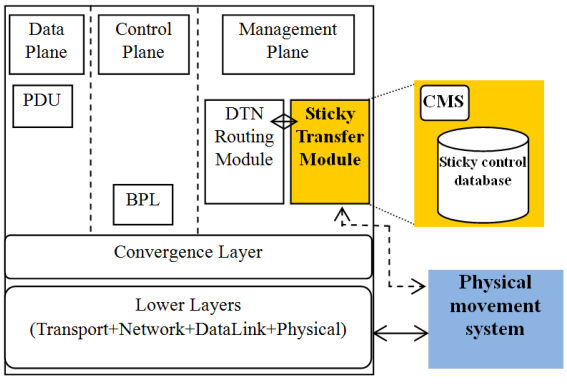


Figure 3.4: Sticky transfer framework integration in the DTN architecture

protocol and components integrated in a layered DTN architecture. The overlay spans several planes where the sticky transfer module resides in the management plane along with the DTN routing module. As DTN can use a multitude of different delivery protocols including TCP/IP, raw Ethernet, serial lines, or hand-carried storage drives for delivery, the convergence layer provides the functions necessary to carry DTN protocol data units (PDUs) on each of the corresponding protocols at lower layers. The bundle protocol (BPL) is the central component in the overlay; it requires detailed information of the state of the system upon which to base routing decisions [29]. The management plane handles providing such information to the BPL. The sticky transfer module interacts with the routing module and adds components to existing routing management features to implement sticky forwarding decisions. The sticky control database contains compatibility information which is used by the control management system to calculate sticky modes. The sticky control management system communicates sticky transfer decisions to physical movement systems; such as - to the motion sensors on an environmental monitoring robot/vehicle. Essentially, when sticky transfer is combined with a specific DTN routing protocol, the overall network performance will depend on the performance of the routing protocol and the benefit of sticky transfers.

In the following section we describe the sticky message transfer protocol in detail.

3.5 The Proposed Sticky Transfer Protocol

We assume nodes have user preferences P and status information I consisting of movement vectors V (i.e., speed, direction, and current location), transmission range W, transmission rate R, free buffer size Buf, and message vectors μ (i.e., containing the message size and ID). Here, $V = \{v_j, \theta_j, (x_j, y_j)\}$ and $\mu = \{(\mu_1, id_1), (\mu_2, id_2), (\mu_3, id_3), \ldots, (\mu_k, id_k)\}$, $j = 1, \ldots, n$ and $k = 1, \ldots, m$, where n is the number of nodes in the network and m is the number of messages to be transferred from node j. The set of messages in μ is decided for node j by the routing protocol strategy [8–10]. Suppose that nodes A and B have just come into each other's transmission range and have a number of messages to exchange. Fig. 3.5 shows the protocol sequence diagram for the sticky transfer of messages, assuming A sends the request first.

First, A sends a sticky transfer request to B along with its status information I_A and user preferences P_A. Sticky requests can be included in the beacon messages to reduce

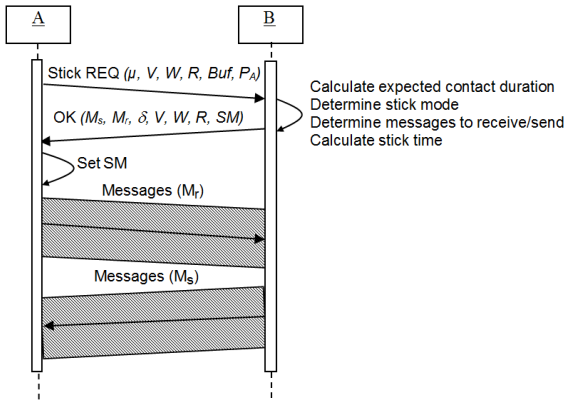


Figure 3.5: The sticky transfer protocol sequence diagram

the number of messages exchanged.

After receiving the stick request from A, B first calculates the expected contact duration T_C between A and B using Eq. 3.1, the status information in I_A and its own status information I_B. B then determines the messages it needs from A by removing from μ_A the messages B already received. B then records the IDs of the messages it needs from A in a 'receive' vector M_r. B could remove some messages from M_r if it does not have enough buffer for all messages in M_r. Next, if B has messages to send to A from its own message vector μ_B, it will use A's free buffer space information Buf_A to determine the messages it wants to send to A without overflowing A's buffer and records their IDs in a 'send' vector M_s. B then calculates an upper bound on the required transfer duration T_R using the total size of the messages recorded in M_r and M_s, the transmission rate and Eq. 3.2. B can compare T_C and T_R to determine if the natural contact duration is sufficient. B also determines if a compatible stick mode exists between A and B using user preferences P_A and P_B.

If T_C is sufficient for completing the message exchange, sticky transfers are not necessary. B will notify A through a reply with the stick mode SM set to NO_STK and stick duration $\delta = 0$. B would also send a NO_STK message to A if A's and B's sticky preferences are not compatible. On the other hand, if compatible sticky modes exist between them and sticky transfers are necessary (i.e., $T_C < T_R$), B will update its own sticky mode (e.g., *Follow you*) and record the mode it expects A to use in a variable SM_A. B then sends an OK message to A, which contains the following information:

message vectors μ_B, M_r and M_s, stick duration δ, status information I_B, and stick mode SM_A.

A receives the OK message from B and sets its stick mode as defined in SM_A. Next it updates vector M_s by removing from M_s the messages it had received. A then sends to B the data messages B has indicated in vector M_r. A also sends the updated vector M_s to B by piggybacking M_s onto some of the data messages.

After receiving M_r and the updated vector M_s, B will send messages indicated in M_s to A to complete the transfer. After completing the message transfers, the nodes will resume their natural movements.

Since the estimation of the required transfer time may be slightly lower than necessary or nodes may opt-out from the stick transfer agreement, a limited number of aborts are still possible. As previously mentioned in section 3.1, during sticky transfers if there are in fact multiple nodes in range, the requester (e.g., A) selects one particular sequence of mutual encounters. This sequence may come naturally from the order in which A hears the advertisement messages of other nodes.

3.6 Setting the Speed in Sticky Modes

To complete message transfers, as described in section 3.1, it is necessary to extend the natural contact duration, T_C when it is smaller than the required transfer duration, T_R. This extension can be achieved by reducing the relative speed of the two encountered

nodes. Let the initial relative speed of the two nodes be v_1 and the required relative speed of the two nodes for completing the transfer be v_2 (Figure 3.6). If the relative distance traveled by the two nodes before going out of contact is d, then

$$v_2 = \frac{d}{M/R} \tag{3.3}$$

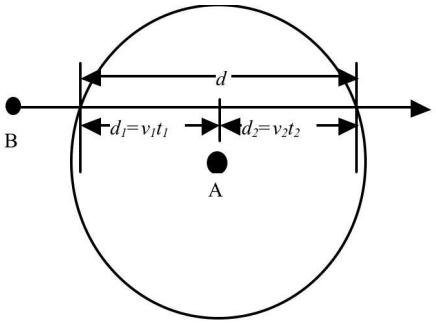


Figure 3.6: Relative speed in the sticky transfer framework

Here, M is the message size and R is the transmission rate. If they are going in the same direction the required relative speed can be achieved by either increasing the speed of the slower node or by reducing the speed of the faster moving node. But if they are going in opposite directions, this can be achieved by slowing down one or both nodes. Also, extension of the contact duration can be achieved by reducing the relative speed of the two nodes to zero, i.e., stopping them or by one node following the other at matching velocities. In that case, the time T_2 for which they should stop or follow will be

$$T_2 = \frac{M}{R} - \frac{d}{v_1} \tag{3.4}$$

3.7 Performance Limitations of Sticky Transfers

Nodes using the sticky transfer protocol may experience transitory changes to their natural movement in order to successfully engage in the sticky transfer of messages. An effect on the network performance, which depends on node mobility, is expected when sticky transfer is used, as mobility itself is know to increase the capacity wireless ad hoc networks [79]. Generally in DTN, both the end-to-end delay and delivery ratio can be improved by reducing pair-wise inter-contact times. A shorter inter-contact time means getting a forwarding opportunity faster, which results in less delivery delay. Sticky transfer is achieved by reducing nodes movement speed which may increase the inter-contact time in the network. Thus, if sticky transfers are used persistently (e.g., all nodes engage in sticky transfers with a high probability) in highly loaded network conditions, the effectiveness of sticky transfers may be reduced due to lower mobility (i.e., nodes stopping for long periods to finish their exchanges). Lower mobility can also be detrimental from the point of view of the mission. For example, it may take longer for the robots to map an area, or it takes longer for the first responders to do their job). Finally, sticky transfers may lead to incessant buffer overflows when the network load is very high and the routing protocols make many copies (see section 2.4.8.2) of each message.

3.8 Chapter Summary

In this chapter we presented our proposed sticky transfer framework by stating definitions and assumptions, and then describing the concept and components of the sticky transfer

framework. We then also described the proposed sticky transfer protocol that governs the exchange of messages within the framework. We explained how the proposed framework can be seamlessly integrated into the existing DTN architecture. We also discussed how the sticky transfer framework may impose limitations on the the performance of the DTN network due to its assumption of modified mobility behavior.

In Chapter 4, we will present simulation results to demonstrate the effectiveness and performance of the proposed sticky transfer framework.

Chapter 4

Performance Evaluation

In this chapter, we provide performance evaluations of the sticky transfer framework using the Opportunistic Network (ONE) simulator, a simulation environment capable of routing messages between nodes using various DTN routing algorithms and sender/receiver types [30]. We compare the performance of DTN routing protocols with and without sticky transfers. In particular, we evaluate the performance gain resulting from the sticky transfer mechanism with the following DTN routing protocols: Epidemic [8]; Spray-and-Wait (SnW) [10]; and PRoPHET [9].

In our simulations, we assume that a node agrees to a sticky transfer request with a probability value of SP (stick probability), where $0 \leq SP \leq 1$. A value of $SP =$ 0, 0.5, and, 1 indicates that a node does not agree, agrees to 50% of the requests, or always agrees to sticky transfer requests, respectively. Our study mainly observes the trade-offs of various SP values on parameters such as delivery reliability, traffic overhead, and memory utilization. The SP of a node does not change during the duration of a

simulation run. Also, we assume that all nodes in the network will have the same stick probability. In future work, we will develop algorithms to enable nodes to determine the optimal stick probability when receiving a stick request based on the collected information (e.g., mobility speed, direction, and message sizes) and network conditions (e.g., network density).

We implement the "STOP" (STP) mode as the mode for sticky agreements from the possible sticky transfer modes described in Section 3.3.1. The implementation choice is based on the observation that the "STOP" mode has the most inhibiting effect on the natural mobility of nodes, as not only the relative but also the actual speed of the two nodes agreeing to participate in the sticky transfer becomes 0 m/s. They must momentarily *stop* on their natural movement path(s) until sticky transfers are completed. Implementation of the "STOP" mode will provide a worst-case scenario analysis of the effect of our proposed sticky transfer framework. When using alternative modes such as "FOLLOW" and "SLOW DOWN" (SLW) only the relative speeds of both nodes become 0 m/s. Nodes participating in the sticky transfer do not have to stop moving. An alteration (i.e. change in direction) on one or both of the nodes natural movement paths might be required. For example when participating in the sticky transfer with the mode set to "FOLLOW", the node using the "FOLLOW ME" ($FLW1$) mode may not have to change the direction of its natural movement path, but only adjust its speed to accommodate and enable the other node to follow it. The node using the "FOLLOW YOU" ($FLW2$) mode might have to change both direction and speed to enable the

sticky transfer to be successful. In a realistic case, this may happen when one node's (node in "FOLLOW ME" mode) message has a higher delivery priority than that of other nodes in the network and delivering that package may be critical for the mission. Hence, other nodes will have incentive to engage in the "FOLLOW YOU" mode. Nodes engaging in sticky transfers in the "SLOW DOWN" (SLW) mode will also have a greater advantage than nodes engaging in sticky transfers in the "STOP" mode, as they only need to move at a relative speed of 0 m/s and do not require a change in direction of their natural movement path. These observations leads us to believe that implementation of the "STOP" (STP) mode will be the most effective for observing the lower-bound of sticky transfer performance. The implement and comparison of other sticky modes are part of our future work.

In our implementation of the *sticky transfer protocol* (see Section 3.5), we ignore the time for calculating the stick mode agreement at nodes during the initiation of the protocol as this overhead is negligible compared to the overall message transfer time, which depends on the wireless transmission capabilities of nodes.

4.1 Performance Metrics

As discussed in Section 2.4.8.2, replication-based DTN routing protocols such as Epidemic [8] and PRoPHET [9] forward multiple copies of a message to increases the probability of message delivery to the destination. Replication-based routing is also know as multi-copy forwarding, or flooding. In this process, a source node generates multiple copies of a

message m in its buffers and forwards them to nodes/contacts that have not yet 'seen' (i.e. received) m. The specific number of copies depend on each individual routing protocol strategy. Flooding-based routing guarantees a higher delivery probability compared to other DTN routing strategies at the expense of higher energy consumption and lower scalability. We design/select performance metrics with multi-copy message forwarding in mind and take into account the multiple copies of messages in our analysis.

We use the following performance metrics to evaluate the effectiveness of the proposed framework.

Average message delivery ratio. The message delivery ratio of a destination d is the ratio of the number of unique messages for d generated by its source(s) and successfully received by d to the total number of unique messages created. The *average* message delivery ratio is the average of the delivery ratios of all the destinations in the network.

Average end-to-end delay. The end-to-end delay of every message successfully received at every destination is recorded. Once the copy of a particular message is received at the final destination, subsequent copies are disregarded. The average over all the messages received is then computed.

Average buffer time. The buffer time for a message m is the interval between the time the message is saved in the buffer at a node to the time it is removed from the buffer of that node. Essentially, it is the time the message was queued in the node's buffer. The *average* buffer time is the average taken over all messages stored at every node in

the network. [1] Average buffer time significantly impacts performance as: (a) it is an indicator of how fast messages are being forwarded at nodes, and thus are propagating through the network, as longer buffer time leads to longer end-to-end delays; and (b) the longer messages are stored in buffers the more buffer overflows will occur, leading to higher loss rates.

Average overhead ratio. The overhead ratio is defined as follows:

$$\frac{H - h_d}{h_d} \tag{4.1}$$

where H is the number of hops a message m and all copies of $m$traverse in the network before a copy is successfully delivered to the intended recipient, and h_d is the number of hops the delivered message (or its copy) had traversed before reaching the final destination. It represents the quantity of excessive relays required to deliver the message and quantifies the bandwidth and energy consumption of the routing protocol. The *average* overhead ratio is the total number of transmissions required over all messages successfully delivered to their destinations.

Average number of disrupted transmissions. A successful message transfer from one hop (sender) to another (receiver) entails that a message is transmitted in its entirety (all bits) from the senders queue and all bits of the message are received at the receivers

[1]Note that messages may be removed from nodes either because the message was relayed to the next encountered node or it was dropped from the current node due to buffer overflow. Hence, the average buffer time can be higher than the average end-to-end delay, as end-to-end delay considers successfully delivered messages only.

queue (explained in section 2.5.1). Some messages may be received at the destination with bit errors, which are not counted as disrupted messages. Unsuccessful/disrupted transmissions in our DTN network environment can be caused by nodes moving out of the transmission range of each other in the middle of message transfers, or by packet collisions. Obviously, these cases can not be avoided. The number of disrupted transmissions is an implicit measure of the wasted bandwidth in the network. The *average* number of disrupted message transmissions is the sum of the number disrupted transmissions observed by each sender over the total number of nodes in the network.

Average number of contact opportunities per hour. We define a 'contact opportunity' as two nodes coming within the wireless transmission range of each other, presenting an opportunity to transfer messages. The *average* number of contact opportunities per hour is the total number of contact opportunities for the duration of a simulation run divided by the total simulated time (in hours). To be counted as a contact opportunity, nodes do not necessarily have to use that contact opportunity to transfer messages. The number of contact opportunities in the network is important, particularity for DTN's where the routing strategy is multi-copy forwarding because more contact opportunities increase the delivery probability by providing the opportunity to spread more message copies in the network.

Average stick time. We define 'stick time' as the time after two nodes come within the wireless communication range of each other, and use the contact opportunity to engage in message transfers using one of the sticky transfer modes. This is measured at each

node by the δ-value introduced in section 3.2. The *average* stick time is the sum of the stick time of pair-wise contacts during each simulation run over the total number of pair-wise contact opportunities used for sticky transfers in that simulation run. In our results, we represent the stick time as a percentage of the total simulation time for one run. For example, lets assume the stick time was measured as 3,000 simulated seconds. The stick time in this case would be represented as 10% given the total time for each experiment run was 30,000 simulated seconds.

4.2 Simulation Parameters

We choose a map of the Helsinki downtown area of size 4500 m x 3400 m as our network area. Mobile nodes move according to the Shortest Path Map Based Movement model [37] [38]. Nodes randomly choose next locations from eleven disconnected points of interest (POIs). POIs are places on the map used to model office buildings, tourist attractions, shops, restaurants and parks. Nodes move to selected POIs with random speeds between 1 m/s to 25 m/s and pause for a period with a value randomly distributed between 1 second and 50 seconds before selecting the next POI. We use 5 to 30 mobile nodes in the network. In the graphs, we refer to nodes as hosts or mobile hosts (MH).

We consider a uniform traffic model where all mobile nodes have data to send to destinations selected randomly. Each simulation runs for 30,000 seconds in simulated time. Messages are generated on average every 5 seconds after a warm-up period of 1,000 seconds. Sources stop generating messages after 21,000 seconds to allow for all message

Table 4.1: Default simulation parameters

Parameter	**Value**
Network size (map area)	20 mobile nodes in 4500m x 3400m
Points of interest (POI's)	11
Movement model	Shortest Path Map Based Movement [31]
Pause time at POI's	random between 1 to 50 seconds
Movement speed	Total of 20 nodes 4 nodes (pedestrians): 1.25 - 1.53 m/s 6 nodes (cyclist): 2.5 - 8.33 m/s 10 nodes (cars): 20 - 25 m/s
Traffic model at sources	0.2 messages/second
Message generation rate	5 seconds on average
Simulation time	30,000 seconds of simulated time
Message generation start time	at 1,000 simulated seconds
Message generation stop time	at 21,000 simulated seconds
Message size	100 Kbytes ; 20 Mbytes
Time-to-live (TTL) of messages	∞
Buffer size of nodes	1GB
Transmission range	100 meters
Transmission rate	11 Mbps (modeling IEEE 802.11b) 54 Mbps (modeling IEEE 802.11g)
DTN routing protocols	Epidemic [8] PRoPHET [9] Spray-and-Wait [10]
Number of runs per data point	10

copies already generated in the network a chance to propagate and be distributed fairly compared to messages previously generated. The average message generation rate in the network is 0.2 messages/second, as messages are generated randomly at the i^{th} second of every 5 second generation interval.

Message sizes vary between 0.1 Mbytes to 30 Mbytes with infinite lifetime (TTL) values. These sizes cover different types of messages such as text, photos, or short videos. The sticky transfer mechanism is expected to show its effectiveness especially for larger message sizes, which are desirable in DTNs. Given the same amount of data to be sent, larger messages mean less data units to be transmitted, and thus lower overhead incurred by data unit headers at different layers of the TCP/IP protocol stack. The buffer size of each mobile node is 1 GB. The transmission range of nodes are 100 meters with transmission rates of 11 Mbps (modeling IEEE 802.11b) and 54 Mbps (modeling IEEE 802.11g).

Table 4.1 summarizes the simulation settings; the default values are used unless otherwise stated. Under these settings, we will evaluate the performance of the sticky transfer mechanism with the following routing protocols: Epidemic [8]; Spray-and-Wait (SnW) [10]; and PRoPHET [9].

In the Spray-and-Wait [10] protocol, the source of a message initially starts replication with K copies of the message. When any node with $K>1$ message copies (source or relay) encounters another node with no copies of the message, it transfers $\lfloor K/2 \rfloor$ copies to the other node and keeps $\lceil K/2 \rceil$ copies for itself. When a node is left with only one copy,

Table 4.2: Parameter settings for experiments

Function of	Parameter	Value
Different Transmission Rates	Transmission rate	11 Mbps ; 54 Mbps
	Message size	100 KB ; 20 MB
	No. of nodes	20
	Movement speed	default (Table 4.1)
	TTL	∞
Different Node Densities	Transmission rate	11 Mbps
	Message size	20 MB
	No. of nodes	30 ; 50
	Movement speed	30 nodes: 4 pedestrians, 6 cyclist, 20 cars (Table 4.1) 30 nodes: 10 pedestrians, 10 cyclist, 30 cars (Table 4.1)
	TTL	∞
Different Node Mobility Speeds	Transmission rate	11 Mbps
	Message size	20 MB
	No. of nodes	20
	Movement speed	5 m/s ; 20 m/s
	TTL	∞
Different TTL Values	Transmission rate	11 Mbps
	Message size	20 MB
	No. of nodes	20
	Movement speed	default (Table 4.1)
	TTL	1 simulated hour 5 simulated hours
Different Message Sizes	Transmission rate	54 Mbps
	Message size	5 MB ; 30 MB
	No. of nodes	20
	Movement speed	default (Table 4.1)
	TTL	∞
Supporting Message Sizes	Transmission rate	54 Mbps
	Message size	100 KB, 5 MB, 20 MB, 30 MB
	No. of nodes	20
	Movement speed	default (Table 4.1)
	TTL	∞
	Routing protocol	Spray-and-Wait

it waits to meet the destination and directly delivers the message upon encounter. The performance of Spray-and-Wait depends largely on the number of initial copies, K at the source. In our experiments we implement SnW in binary mode (see section 2.4.8.2) with K=4. The K value for SnW was chosen after reviewing existing literature on the protocol and observing standard values from simulations performed in the DTN research community.

Each simulation was run 10 times. Each data point plotted on graphs is an average value over 10 runs.

We conducted a total of five sets of experiments to provide in-depth insight into the performance of the sticky transfer mechanism. We performed the experiments on three DTN routing protocols: Epidemic, ProPHET, and SnW. In each experiment we varied the stick probability (SP) and one of the following parameters:

- *Transmission rate at the physical layer* [Section 4.3.1]: The first set of experiments was performed by varying the transmission rates of nodes on both small and large message sizes. We measured several performance metrics for this experiment set mentioned in Section 4.1. The results depict the performance of the sticky transfer mechanism under networks that support different transmission rates at the physical layer.

- *Node density* [Section 4.3.2]: The second set of experiments was performed by varying the number of nodes in the network area. We measured two performance metrics for this experiment set: (i) the average message delivery ratio, and (ii) the average

end-to-end delay. In this experiment set, we show the performance of the sticky transfer mechanism with varying node densities at 5 mobile hosts(MH) to represent a low density network and 30 mobile hosts(MH) to represent a higher density. The transmission rate of nodes is 11Mbps and the message size is 20MB. We measure two performance metrics: average delivery ratio and average end-to-end delay.

- *Node speed* [Section 4.3.3]: The third set of experiments was performed by varying the speed of nodes in the network. We measured two performance metrics for this experiment set: (i) the average message delivery ratio, and (ii) the average end-to-end delay. In this experiment set, we present results for 5m/s and 20m/s node speeds with varying stick probabilities for 20MB message sizes with nodes transmitting messages at 11Mbps. In order to observe the performance improvement with sticky transfers for different node movement speeds, we set the same speed for all 20 nodes per experiment, as opposed to the default simulation setting in table 4.1.

- *Time-to-live (TTL) value* [Section 4.3.4]: The fourth set of experiments was performed by varying the TTL value of messages in the network. We measured two performance metrics for this experiment set: (i) the average message delivery ratio, and (ii) the average end-to-end delay. We run experiments for 20MB message sizes at 11Mbps transmission rates on two time-to-live (TTL) values for messages: 1 hour and 5 hours, respectively,

- *Message size* [Section 4.3.5]: The fifth set of experiments was performed by varying

the size of the messages generated in the network. We measured two performance metrics for this experiment set: (i) the average message delivery ratio, and (ii) the average end-to-end delay. We show the performance improvement using sticky transfers varying the message size at 5MB and 30MB message sizes. In each experiment 20 nodes are in the network transmitting messages at 54Mbps. We choose a higher transmission rate for this set of experiments as it involves larger message sizes. By having a faster transmission rate we could reduce the simulated time required to run the experiments, yet still observe the effect of the sticky transfer protocol.

We summarize the set of parameters used in the five experiments in Table 4.2. Note that the sixth column in the table, 'supporting message sizes', is an accumulated analysis from previous experiments and did not require new simulation runs.

4.3 Simulation Results and Analysis

We present simulation results in Fig 4.1 to Fig 4.12.

When analyzing simulation results we note the following:

- due to the nature of the protocol design (i.e., message dissemination strategy), the Spray-and-Wait (SnW) protocol incurs lower network traffic than the Epidemic and PRoPHET protocols, as the number of copies of a message that can exist in the network is limited by the K parameter in SnW. In PRoPHET and Epidemic, in worst-case scenarios, it is theoretically possible for each node to be carrying a

copy of every message generated in the network, leading the number of copies of a message m to be equal to the number of nodes N in the network.

- traffic in the network impacted by the message forwarding strategy of the routing protocol. For example, the Epidemic protocol incurs higher loads than the PRoPHET or Spray-and-Wait protocol as it is a flooding protocol and copies of messages are forwarded to every encountered node.

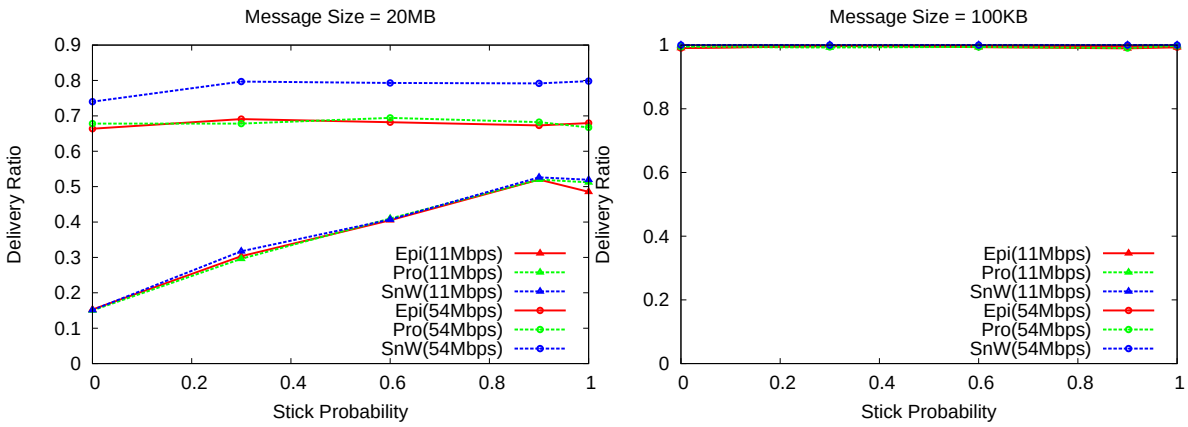
4.3.1 Different Transmission Rates

In the first set of results we measure the performance of the sticky transfer protocol using two different transmission rates at the physical layer: 11 Mbps and 54 Mbps. The results presented in this section show the effect of using the sticky transfer protocol for delivering messages in the network considering several performance metrics as defined in section 4.1 and presented as follows:

4.3.1.1 Average Message Delivery Ratio

Fig. 4.1 presents the delivery ratio as a function of the stick probability (SP). The graph in Fig. 4.1(a) depicts the performance of the average message delivery ratio with a message size of 20 MBytes and messages being transmitted at 11 Mbps and 54 Mbps transmission rates, respectively.

At 54 Mbps, a high delivery ratio was achieved even without sticky transfers due to the shorter transfer time of messages. Gradually increasing SP values increased the



(a) Message size = 20MB.

(b) Message size = 100KB.

Figure 4.1: Average Delivery Ratio as a function of Stick Probability.

delivery ratio of SnW up to 6% but did not significantly increase the delivery ratio of flooding based protocols (Epidemic, PRoPHET) because of buffer overflows.

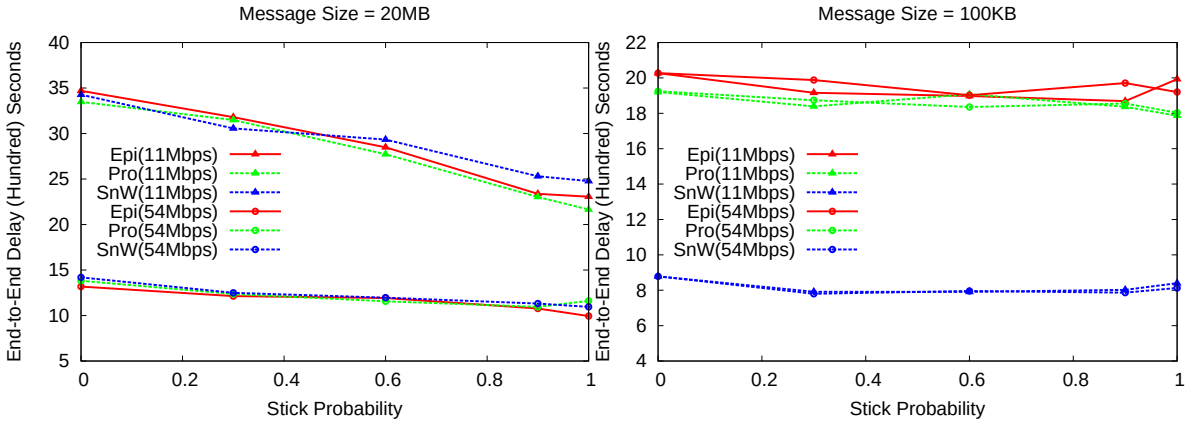
At 11 Mbps, the delivery ratio increased up to 38% with increasing SP values as more messages could be exchanged upon encounters due to mobile hosts' increasing willingness to stick for transfers.

At the lower transmission rate (11 Mbps) when buffers were not a limitation, the maximum delivery ratio was achieved at SP=1.0 in SnW. However, in Epidemic and PRoPHET, the maximum delivery ratio was achieved at a SP value around 0.9. In these two algorithms, the delivery ratio decreased as SP increased from 0.9 to 1 because nodes always agreed to stick for message transfers (i.e., the stop mode), which reduced the node movement. The above difference is due to the fact that SnW forwards fewer copies of a message than the Epidemic and PRoPHET protocols.

The graph in Fig. 4.1(b) depicts the performance of the average message delivery ratio for smaller message sizes (e.g., 100KB). On average, a 100% delivery ratio was achieved in all three protocols even without the sticky transfer mechanism (i.e. at SP=0) at both transmission rates due to the smaller size of messages and sufficient contact opportunities (see Fig. 4.6(b)).

In summary, the benefit of sticky transfers is clearly evident in scenarios with larger message sizes and lower transmission rates, which indicates that DTN networks with these characteristics can benefit from sticky transfers.

4.3.1.2 Average End-to-End Delay



(a) Message size = 20MB.

(b) Message size = 100KB.

Figure 4.2: Average End-to-End Delay as a function of Stick Probability.

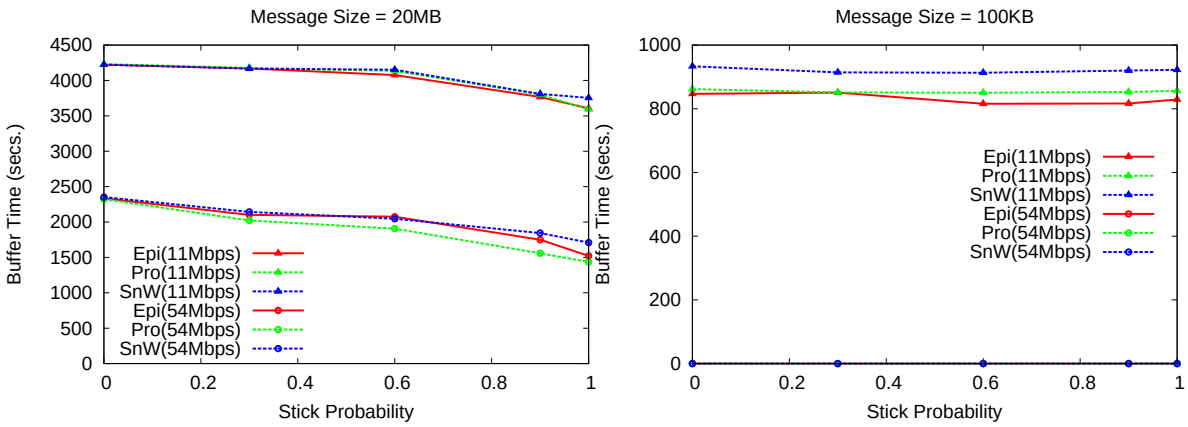
Fig. 4.2 presents the average end-to-end delay of messages as a function of the stick probability (SP). In general, the end-to-end delay decreased gradually with increasing

SP. Fig. 4.2(a) shows the end-to-end delay with increasing stick probabilities for 20 MB message sizes. The reduction of the delay using sticky transfers was significant especially at the lower transfer rate of 11 Mbps. A maximum of up to 25% and up to 36% reduction in the end-to-end delay was observed at 54 Mbps and 11 Mbps, respectively. As expected, the delays for 54 Mbps were much lower than those for 11Mbps due to the higher transmission rate.

Fig. 4.2(b) shows the end-to-end delay with increasing stick probabilities for 100 KB message sizes. The improvement on the delay performance was about 10% overall because in most cases sticky transfers was not used due to the nature of contact duration's available. However, this does show that increasing the stick probability despite the delivery ratio being almost 100% (see Fig. 4.1(b)) for all stick probabilities did improve the end-to-end delay of those delivered messages. Note that the delay is minimum around SP=0.9, instead of at SP=1.0 as would be expected, for flooding based protocols (e.g., Epidemic) because persistent stick restricts movement in high loads.

4.3.1.3 Average Buffer Time

Fig. 4.3 shows the average amount of time messages spend in node buffers before being forwarded to the next node. In Fig. 4.3(a), as the SP was gradually increased, the average buffer time of messages decreased (by 38%) compared to the no-stick transfer case (SP=0). With increasing SP values, messages spent less time in buffers and message copies were forwarded faster in the network, thus reducing delay significantly. The average



(a) Message size = 20MB.

(b) Message size = 100KB.

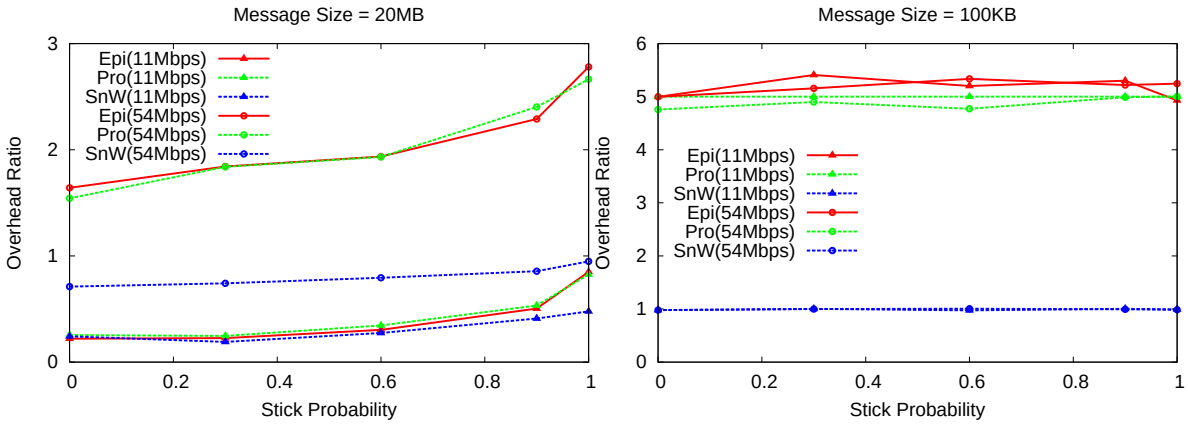
Figure 4.3: Average Buffer Time as a function of Stick Probability.

time spent in buffers is lower for 54 Mbps than 11 Mbps due to faster transmission rates. In both cases at higher SP values (0.9 and 1.0), SnW has a higher average buffer time as it goes into the direct delivery (i.e., waiting) mode once all its copies are forwarded; thus messages remained longer in node buffers just prior to the final delivery. Also, for Epidemic and PRoPHET, we can expect that more message drops occurred.

A similar trend was observed for the three routing protocols at a transmission rate of 11 Mbps for smaller messages sizes of 100KB as shown in Fig. 4.3(b). However, we observe in this graph that at the faster transmission rate of 54 Mbps, the buffer occupancy time is negligible across all SP values. This is of course due to the smaller messages along with faster transmission speeds and nodes having sufficient contact durations upon encounters. Also comparing Fig. 4.3(a) and Fig. 4.3(b), we can see that the average time messages spent in buffers was much less for messages sizes of 100KB compared to the

larger messages of 20MB overall.

4.3.1.4 Average Overhead Ratio



(a) Message size = 20MB.

(b) Message size = 100KB.

Figure 4.4: Average Overhead Ratio as a function of Stick Probability.

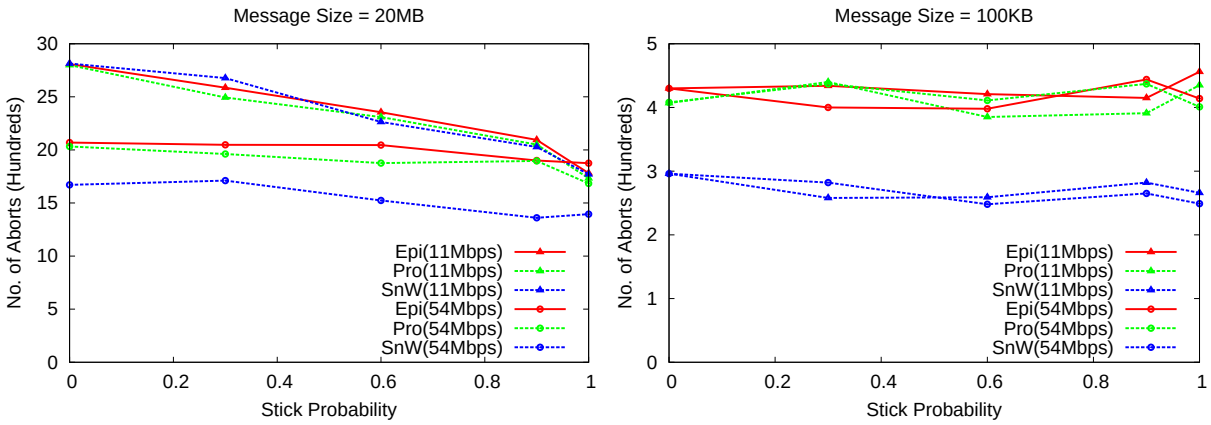
Fig. 4.4 shows the average overhead ratio which signifies the number of 'extra' (additional) hops a messages requires for end-to-end delivery. We consider direct delivery as a 0-hop delivery. Direct delivery is when the source itself meets the destination and forwarding to an intermediate node is not necessary. When there is one intermediate node (relay) involved in the end-to-end delivery, we consider it a 1-hop delivery. Direct or 0-hop delivery is considered the ideal case. Any additional hop after a 0-hop delivery is considered as overhead. Hence, if the message required 2 relays to be delivered, the overhead is 2. As mentioned in section 4.1, the overhead ratio quantifies the bandwidth and energy consumption of the routing protocol.

In Fig. 4.4(a), for Epidemic and PRoPHET, each time nodes encounter each other, copies of messages are forwarded in the network until any one of the copies reaches the final destination. For SnW once all 4 copies are spread, nodes go into direct delivery mode. So, there is naturally less overhead for SnW and the number of extra hops a copy is spread before being finally delivered is less than 1 (for both 54Mbps and 11Mbps).

At gradually higher SP values and 54Mbps transfer rates, more copies are spread faster due to nodes always successfully forwarding copies to other encounters because of sticky agreements. This fulfills the strategy of flooding protocols at a cost of overhead. At SP=1.0, this overhead can cause 'thrashing'. Thrashing is a condition where due to buffer limitations messages are dropped incessantly to make room for new ones. This can lead to reduced network performance.

Fig. 4.4(b) shows the average overhead ratio for 100KB sized messages. Thrashing was not noticed in this scenario as buffers were sufficient to accommodate messages. Hence, the overhead ratio was higher as when messages were forwarded hop-by-hop, they could be accommodated and not dropped from the buffers. SnW incurred the least overhead due to the design of the protocol as there is the initial 'spray' phase which accounts for 1 hop, then the 'wait' phase commences and messages wait in buffers until encountering the final destination (i.e. intended recipient), aligning with the results of buffer occupancy being higher for SnW (Fig. 4.3). Epidemic and PRoPHET's overhead ratio was much higher on average, as all copies forwarded are accounted for in the overhead ratio.

4.3.1.5 Average Number of Disrupted Message Transmissions



(a) Message size = 20MB.

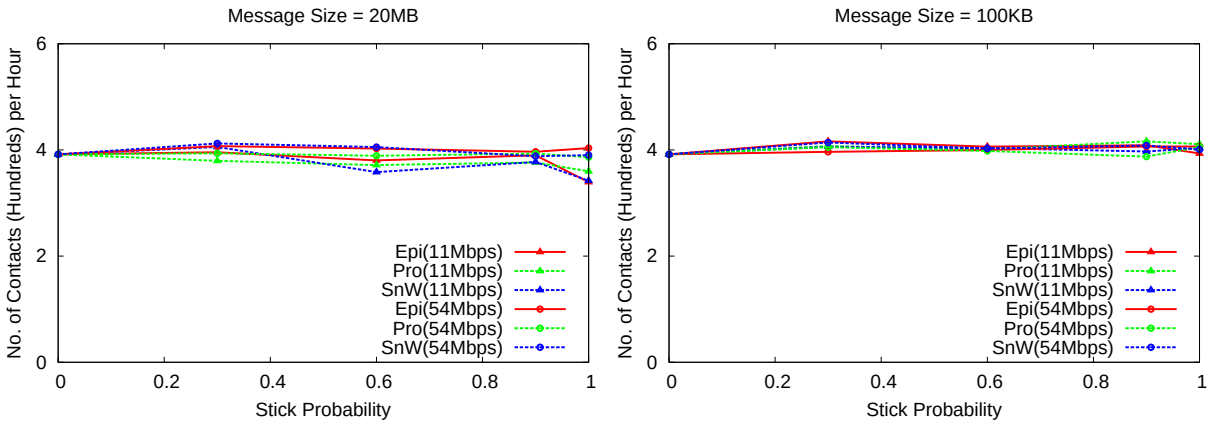
(b) Message size = 100KB.

Figure 4.5: Average Number of Disrupted Message Transmissions.

Fig. 4.5 shows the average number of message aborts with varying stick probabilities. We first analyze Fig. 4.5(a) with 20MB message sizes. In the SnW protocol, a limited number of copies of the message are created and hence the number of transfer requests (and aborts) is smaller than those from the other two protocols.

For smaller message sizes of 100KB sticky transfer was rarely used and thus the reductions in the number of aborts were small (Fig. 4.5(b)). However, for 20MB message sizes the number of aborts were large without sticky transfers. An improvement of (decrease in the number of aborts) 9% to 19% and 36% to 38% were observed at 54Mbps and 11Mbps transmission rates respectively in different protocols when the SP>0.

4.3.1.6 Average Number of Contact Opportunities per Hour



(a) Message size = 20MB.

(b) Message size = 100KB.

Figure 4.6: Average Number of Contact Opportunities per Hour.

Fig. 4.6 shows the number of contacts per hour as a function of stick probability.

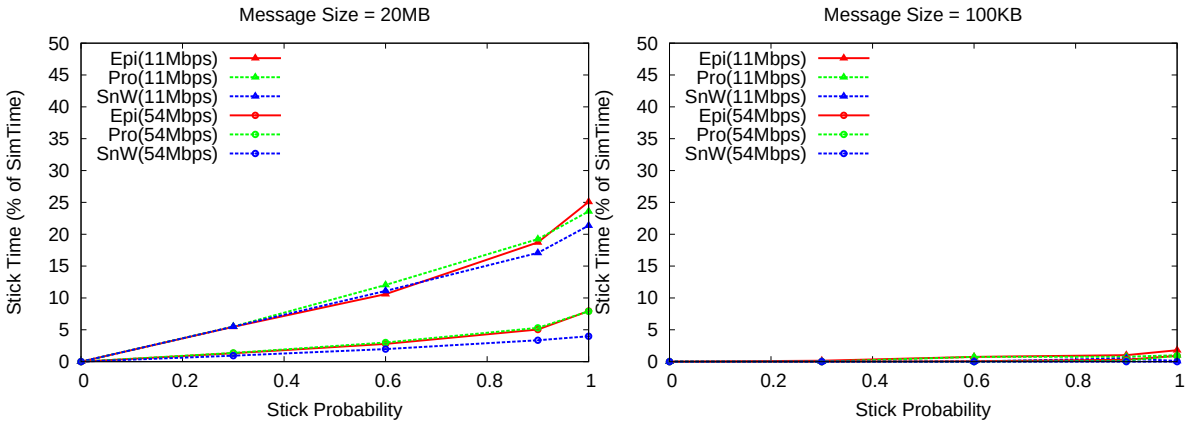
We observe for 100KB-sized message (Fig. 4.6(b)), the number of contacts remained almost linear across all stick probabilities due to the fact that as the message sizes were small, sticky transfers were not required for long durations. We expect that the average stick time in these cases was negligible, as messages were being transmitted almost instantly upon mutual encounters.

For 20MB-sized messages (Fig. 4.6(a)), the stick time is expected to be longer. Hence, as the SP is increased, the number of contacts per hour decrease, as pair-wise contacts stick longer to successfully transfer message and thus meet fewer new contact within the same amount of time (i.e., in comparison to smaller message sizes). This is even more

observable in the case of the flooding-based Epidemic routing protocol, as gradually every pair-wise encounter will lead to a sticky transfer.

In Fig. 4.6(a), in the case of flooding based protocols the number of contacts decreased by 13% when SP=1.0. This is because the movement of the nodes were restricted during sticky transfers and thus the number of contacts decreased. However, compared to the benefit of sticky transfers (via improving delivery performance, reducing message transfer aborts etc.), this negative impact on node mobility is insignificant. Also, instead of the "Stop" mode if other modes, like "Follow" or "Slow down" are used, then the above discussed impeding effect of sticky transfers on the network performance will be very small.

4.3.1.7 Average Stick Time



(a) Message size = 20MB.

(b) Message size = 100KB.

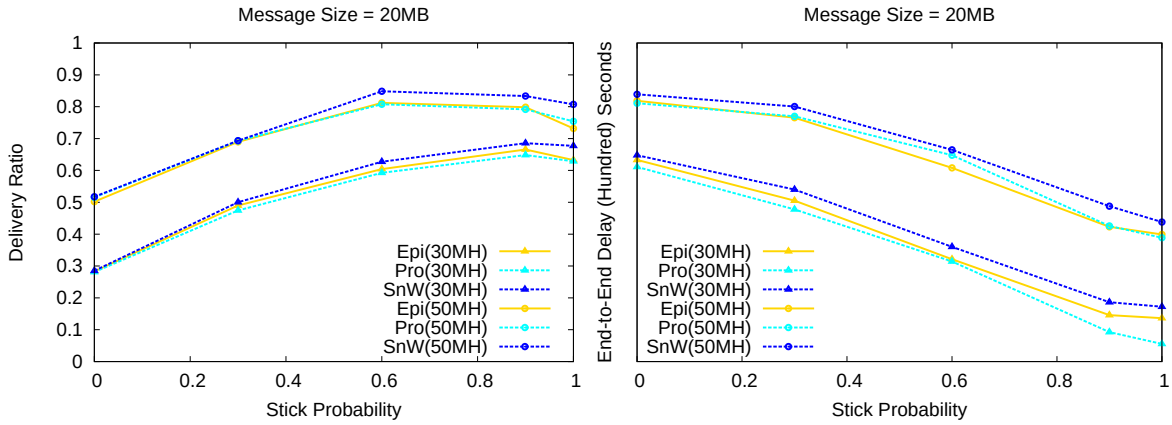
Figure 4.7: Average Stick Time as a function of Stick Probability.

Fig. 4.7 illustrates a measurement of the total time nodes stick in the network as a percentage of the total simulation time. Since we implement sticky transfers through the "STOP" mode, the stick time represents the total amount of time that a node stops (is stationary) and thus is the cumulative delay on its journey to all of its destinations. We notice the same trend for this parameter for both larger and smaller sized messages, shown in Fig. 4.7(a) and Fig. 4.7(b), respectively. The stick time was non-existent (zero value) for all routing protocols at SP=0, since the probability of sticky transfers is zero.

In Fig. 4.7(b), since the message sizes was smaller, nodes needed not engage much in sticky transfers and the stick time was less that 2% across all protocols. In general, for both graphs, the stick time was higher for flooding protocols (up to 25%), particularly at SP=1, since nodes remained in contact upon mutual encounters until all messages were forwarded. For both 11Mbps and 54Mbps, SnW incurs less cumulative stick time in general as nodes have message copies to forward upon encounters. Too much stick time can reduce node mobility in the DTN, which can lead to fewer contact opportunities over time.

4.3.2 Different Node Densities

Fig. 4.8 shows the performance improvement achieved with sticky transfers at different node densities for large message sizes (MS=20MB). The delivery ratio is shown in Fig. 4.8(a). At the lower node density (30MH), the performance increased by 40%. At the higher node density (50MH), the performance increased by 33%. Better performance



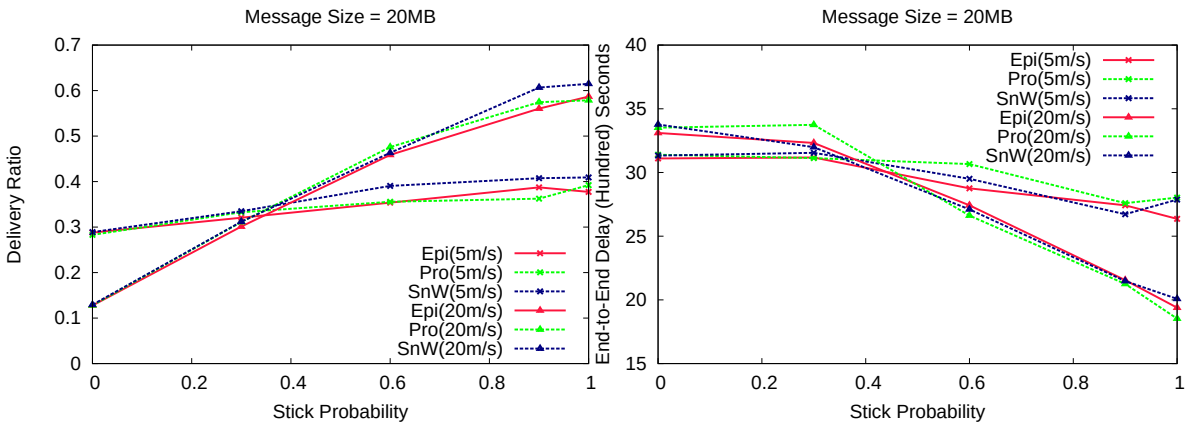
(a) Average Delivery Ratio as a function of SP.

(b) Average End-to-End Delay as a function of SP.

Figure 4.8: Performance of sticky transfers at different node densities.

at lower node densities compared to higher node densities results from more contact opportunities in the same area and an increase in stick time compared to movement time, specially when SP=1.0. The performance at higher node densities improved from SP=0 until up to SP=0.9, then fell significantly at SP=1.0 due to an always sticking tendency among nodes. The end-to-end delay (Fig. 4.8(b)) was reduced by as much as 43% at lower node densities and by as much as 40% at higher node densities as the SP increased.

In summary, using sticky transfers provides an improvement in the delivery ratio compared to not using sticky transfers. However, at higher node densities the sticky parameter should be adjusted to consider the mobility factor as well.



(a) Average Delivery Ratio as a function of SP.

(b) Average End-to-End Delay as a function of SP.

Figure 4.9: Performance of sticky transfers at different node speeds.

4.3.3 Different Node Mobility Speeds

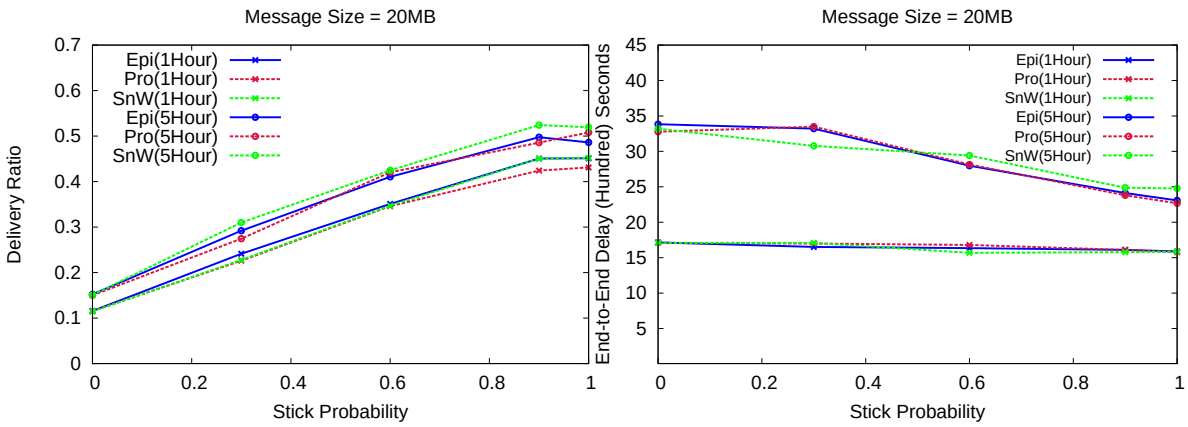
Delivery ratio improved by 12% (for node speeds of 5m/s) and by 49% (for node speeds of 20m/s) across different DTN protocols (Fig. 4.9(a)). Also end-to-end delay decreased by 15% and 45% respectively for 5m/s and 20m/s speeds (Fig. 4.9(b)). When sticky transfers were not used in the network (SP=0), delivery performance at lower speeds (5m/s) were much better than at higher speeds (20m/s), because at faster node speeds the natural contact duration of the nodes were not enough to successfully complete the message transfers of larger message sizes (20MB). However, as we increased the SP, the improvement in the delivery performance increased significantly at higher node speeds. Nodes were able to extend the contact duration to the required duration by using the sticky transfer protocol. Thus with the combination of meeting encounters faster (faster

movement speed) and being able to transfer more messages (longer contact durations), the delivery and latency performance at 20m/s speeds surpassed that of the lower speed scenario. Therefore, sticky transfers give higher benefits in higher mobility conditions where message transfer disruptions are more likely.

4.3.4 Different Time-to-Live(TTL) Values

As we know from our literature review, DTN technology can be applied in the case of a wide range of applications, starting from underwater to outer space. Messages may have an expiration timeout (TTL) set by applications. Generally there is a mission deadline, to be met by nodes/participants in the network. Messages may have TTL values set to meet these deadline requirement. The messages may be useless for the mission after this and therefore, once the TTL expires, are dropped. For space applications the TTL of messages may be very large, maybe upto several months. For military applications the TTL could be in the range of days or hours. For emergency situations the TTL could be in the range of minutes. To keep the simulation time feasible and to shed insight into how the sticky transfer mechanism can assist applications with different TTL values, the results in In Fig. 4.10 show the average delivery ratio and average end-to-end delay with TTL values for 1 and 5 simulation hours, respectively.

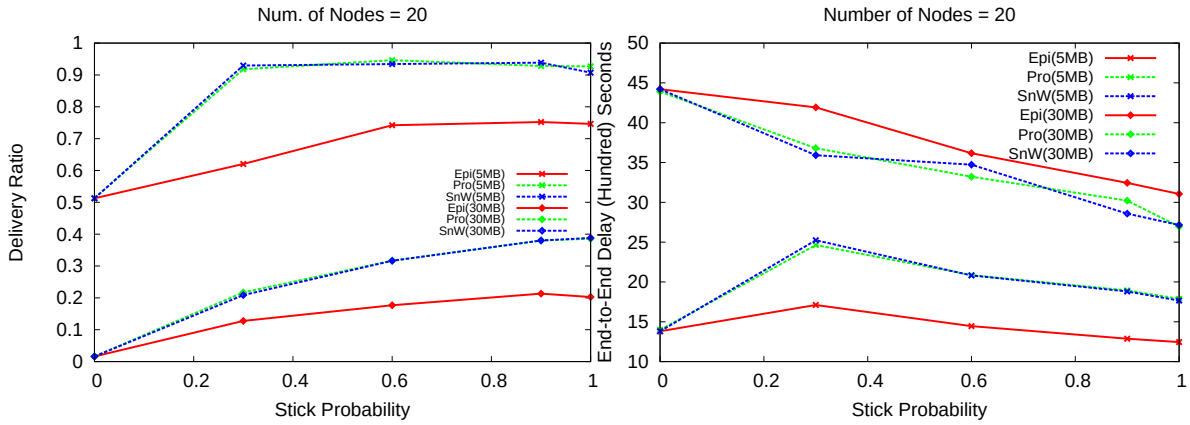
Delivery ratio improved by 34% (for TTL of 1 hour) and by 37% (for TTL of 5 hours) across different DTN protocols (Fig. 4.10(a)). Also end-to-end delay decreased by 8% and 32% respectively for 1 hour and 5 hour TTL values (Fig. 4.10(b)). We notice in



(a) Average Delivery Ratio as a function of SP. (b) Average End-to-End Delay as a function of SP.

Figure 4.10: Performance of sticky transfers at different message time-to-live values.

Fig. 4.10(b), the end-to-end delay was almost constant across all SP values for messages having TTL value of 1 hour, meaning using sticky transfers do not have much impact on the delay performance (i.e. were not able to reduce the end-to-end delay) due to the already shortened lifetime of the messages. However, the delay was significantly reduced at larger TTL values, meaning when applications have a large message (20MB), and TTL values ranging in the hours - our sticky transfer mechanism can help to meet application critical deadlines by delivering the messages faster in the network. Furthermore, the delivery ratio for 20MB messages was greatly improved (by up to 37%) by using sticky transfers. Therefore, the sticky transfer mechanism can help with guaranteed delivery for time critical applications.



(a) Average Delivery Ratio as a function of SP.

(b) Average End-to-End Delay as a function of SP.

Figure 4.11: Performance of sticky transfers at different message sizes.

4.3.5 Different Message Sizes

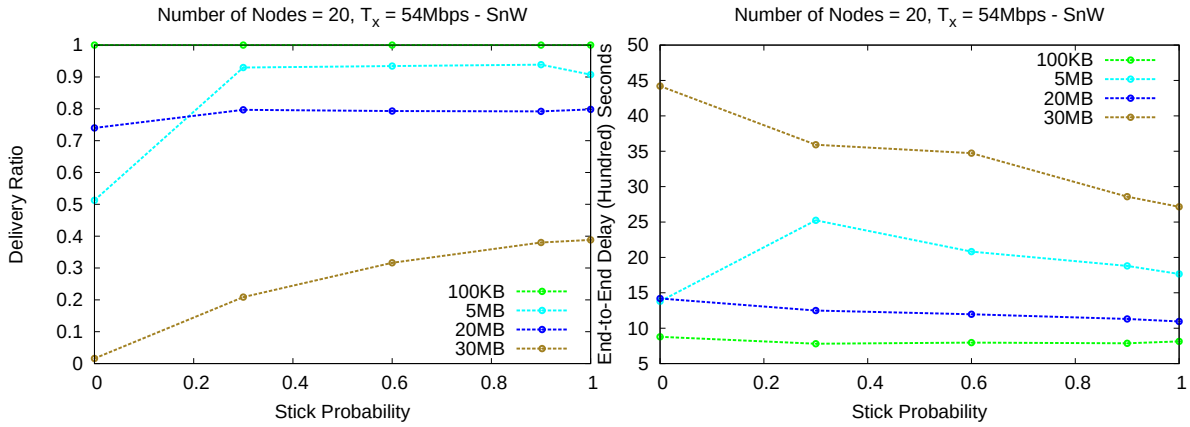
In (Fig. 4.11), the delivery ratio increased by 24% and 20% in Epidemic protocol while it increased by 43% and 37% in the other protocols for 5MB and 30MB messages respectively (Fig. 4.11(a)). End-to-end delay (Fig. 4.11(b)) decreased gradually from 30% to 39% with increasing SP for 30MB-sized messages. This shows the benefit of using higher SP values when message sizes are large. The overall end-to-end delay for 5MB message sizes is much lower compared to 30MB sizes in general. However, an increase in delay was observed around SP=0.3 for all protocols when transmitting 5MB-sized messages (see Fig. 4.11(b)). We observe in the corresponding delivery ratio graph (Fig. 4.11(a)) that at this point the delivery ratio increased as well from a ratio of 0.5 to 0.9. At SP=0 sticky transfers were not being used and we anticipate only a few packets were being

delivered, and the delivery was mostly within a single hop. However, at SP=0.3 as more packets were delivered and since $SP \neq 0$, multiple hops were required contributing to the higher end-to-end delay. The delay from this point (SP=0.3) gradually dwindled and the increase became smaller towards SP=1.0, as expected with increasing stick probability. Thus, sticky transfer improves delivery ratio for all message sizes, but may increase end-to-end delay in some cases if more packets have to travel longer distances.

4.3.6 Supporting Message Sizes: 100kB, 5MB, 20MB, 30MB

DTN Protocol Data Units (PDU) are called bundles (a.k.a 'messages'). A key DTN functionality is that each bundle is kept in memory in its entirety, and is deleted upon receipt of an acknowledgment for its successful delivery to the next node on the path to the destination. However, the DTN Bundle Protocol specification [33] does not limit bundle size or specify content of bundles. A bundle may be of any size depending on the application. As discussed in [34], the bundle protocol uses bundle as a protocol data unit which may be of various lengths. Ivancic et al. have used bundle sizes of 160 MB to transfer images from orbit to a ground station [35]. On the other hand in [36], Scott Burleigh suggests use of small bundles, less than 64KB long, to enable partial data delivery at application-appropriate granularity. It is obvious that the scientific community has not yet defined a commonly accepted formula of encapsulating application data into specifically sized bundles.

This lack of a fixed bundle size for DTNs suggests that a mechanism to improve



(a) Average Delivery Ratio as a function of SP. (b) Average End-to-End Delay as a function of SP.

Figure 4.12: Performance of sticky transfers at different message sizes: Spray-and-Wait

the performance of the DTN across various message sizes would be beneficial and would address an area which requires attention. We believe that the sticky transfer protocol is able to achieve this goal, which we show by Fig. 4.12. These graphs are composed by integrating results from 4.3.5, 4.3.1.1 and 4.3.1.2. It shows the performance improvement achieved by the sticky transfer protocol across a range of message sizes: 100kB, 5MB, 20MB, and 30MB. The number of nodes is 20 and the transmission rate (T_x) is 54 Mbps. We perform a brief analysis of the graphs as the results have already been covered in previous sections. The graph shows results for the SnW routing protocol. The trend is similar for Epidemic and PRoPHET. Fig. 4.12(a) shows the average delivery ratio across message sizes. The delivery ratio was improved in each case by using sticky transfers. Fig. 4.12(b) shows the average end-to-end delay across the different message sizes. Using the sticky transfer protocol, end-to-end delay was reduced across all message sizes (with

the exception of 5MB, which has been explained in Section 4.3.5).

4.4 Chapter Summary

Based on our results and analysis of the sticky transfer mechanism, we can conclude that using the sticky transfer protocol can improve several performance parameters such as the delivery ratio, end-to-end delay, overhead ratio, and so on. It can be well adopted for challenged scenarios where larger messages sizes need to be delivered with application deadline constraints. Performance of the DTN improved (upto 43%) at higher node densities (30 nodes) with sticky transfers. Message transfer disruptions are more likely in increased mobility conditions, and by using sticky transfers the performance of the DTN improved (up to 49%) at higher movement speeds (20 m/s). Furthermore, sticky transfers improve the delivery ratio at all message sizes (5MB-30MB), but may increase end-to-end delay for larger message sizes (30MB) at certain stick probabilities depending on the routing protocol. This is due to the combined effects of restricting mobility and reducing message aborts simultaneously, while sticking to transfer larger messages. Specifically, sticky transfers improve the performance of the DTN, specially under high mobility speeds or high node densities.

We implemented the "STOP" mode as the mode for sticky agreements from the possible sticky transfer modes described in section 3.3.1. As we mentioned at the beginning of this chapter, we chose this mode as it will have the most inhibiting effect on the natural mobility of nodes and thus will be the most effective for observing the lower-

bound of sticky transfer performance. We expect the other sticky trasfer modes, namely "FOLLOW" and "SLOW DOWN", to perform at least as well as the "STOP" mode or better.

Our current assumption is that nodes have preset SP values; e.g. set by a network administrator. Algorithms to predict optimal SP values for nodes on-the-fly, which will be calculated based on the network environment and relevant node parameters, is part of our future work.

Chapter 5

Conclusions and Future Work

Delay-and disruption-tolerant networks(DTNs) implement a store-carry-forward network technology to facilitate data forwarding in infrastructure-less networks with lack of instantaneous end-to-end paths between communicating end points. Application areas of such networks include disaster recovery, emergency operations, ecological monitoring, underwater communications, vehicular communications, and virtual social networks.

The terms "opportunistic networks" and "delay-tolerant networks" are sometimes used interchangeably, in scenarios where the network comprises mostly of wireless mobile nodes/devices with network constraints, such as intermittent connectivity, due to mobility of nodes, power cycle of nodes, or geographical sparsity. In opportunistic networks, no assumption is made with regard to the existence of a complete path between two nodes wishing to communicate. Source and destination nodes might never be connected to the same network, at the same time. Furthermore, nodes may not possess or acquire any knowledge about the network topology. Routes are built dynamically, while messages are

en route between the sender and the destination(s), and any possible node can opportunistically (i.e. when the chance arises) be used as a next hop, provided it brings the message closer to the destination than the current node.

An *encounter* refers to two neighbor nodes coming within the communication range of each other, presenting an opportunity to exchange data. When a encounter occurs in an opportunistic network, it is usually of limited duration. To be able to analyze situations that include limited bandwidth, we discussed the properties of *inter-contact time* (i.e. how frequently nodes encounter other nodes in the network on average) and *contact duration* (i.e. the average time two nodes have to exchange data during an encounter) and established that they directly impact the throughput of the network.

Several routing protocols have been proposed and implemented for opportunistic networks which use strategies such as flooding messages, or using historical encounters to predict future encounter probabilities. Irrespective of the forwarding technique used by routing algorithms, the actual time to transfer messages between two nodes is limited by the *contact duration* of the nodes, which depends inherently on node mobility.

If the expected contact duration is not sufficient for the entire message to be transmitted, which can change at any moment due to node mobility or other factors, messages tend to fail to be forwarded to the next hop. This can cause the routing protocol to retransmit the message, thus wasting valuable bandwidth, and wasting the resource consumed by the failed transfer. In addition, many other messages which have been processed and ready for transmission cannot be forwarded. These messages will stay longer in buffers

of limited sizes, which may eventually be discarded due to buffer overflow, wasting node resources. The end result is low message delivery ratio and long end-to-end delay. The above problems are exacerbated in highly mobile DTNs that must handle large messages such as vehicular networks.

The objective of our thesis was to:

- prolong the natural contact duration of mobile nodes in a DTN to improve the network performance, measured by the delivery ratio, end-to-end delay, and average buffer time.
- propose a solution that can be integrated into the existing DTN architecture without requiring additional infrastructure or storage.

We addressed the above objectives of our thesis by proposing a novel framework called the *sticky transfer framework* that enables nodes to prolong their contact durations for message transfers in DTNs. In this framework, nodes send out periodic beacons for neighbor discovery. Once a neighbor is detected with which a node can perform sticky transfers, the nodes exchange information such as mobility speed and direction, current location, transmission range, available buffer size, the amount of data to be sent and the corresponding destination, using our proposed *sticky transfer protocol* within the framework.

5.1 Summary of Contributions

The main contributions of the thesis include:

1. a novel framework called the sticky transfer framework that enables mobile nodes in a DTN to modify their mobility and prolong contact durations to enhance successful message transfers. The framework allows users of mobile devices to cooperatively adjust their natural movement behaviors and ensures users flexibility in the agreement to 'stick' using the following three components: *user preference*, *sticky modes*, and *compatibility list*.

2. a sticky transfer protocol within the framework that governs how two mobiles nodes will stick to each for a negotiated period of time in order to complete the transmissions of messages.

3. seamless integration of the sticky transfer framework with the existing DTN network management modules.

4. evaluation of the proposed framework with extensive simulations showing the performance of the sticky transfer protocol under various network setting and measuring several performance parameters. To evaluate the effectiveness of our framework, we ran tests on three DTN routing protocols: Epidemic, PRoPHET, and Spray-and-Wait; with and without sticky transfers in a realistic mobile environment. Our analysis showed that the sticky transfer protocol improved the performance parameters we considered, and that our framework is especially beneficial for large message sizes and/or high mobility situations, where contact times allow for few successful transfers.

5.2 Future Work

The sticky transfer framework and protocol can be enhanced and extended in many ways. Currently, we use a constant stick probability in our simulations for all nodes in the network. A more intelligent, adaptive algorithm can be developed which allows nodes to dynamically decide whether to stick or not using available information such as its mobility speed relative to its neighbors' speeds and local node density.

We also plan to consider the overhead of sticky negotiations and time required for resolving medium contention prior to sticky data transfers. Other future research issues include:

- developing new algorithms using Bayesian networks and Markov models that consider past network performance to predict future optimal stick decisions without requiring intervention from the network administrator.
- developing algorithms to select the best neighbors to perform sticky transfers with, which will take into account several factors such as neighbors' residual energy, transmission rates, and amounts of data to be exchanged.
- evaluating the effectiveness of sticky transfers in multi-rate networks.
- implementing sticky message transfers in a realistic network using Mindstorm® NXT robots.

Bibliography

[1] L. Pelusi, A. Passarella, M. Conti, *Opportunistic Networking: data forwarding in disconnected mobile ad hoc networks*, IEEE Communications Magazine, Vol. 44, No. 11, Nov. 2006.

[2] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, J. Scott, *Pocket switched networks: Real-world Mobility and its Consequences for Opportunistic Forwarding*, Technical Report UCAM-CL-TR-617, University of Cambridge, Feb. 2005.

[3] S. Jain, K. Fall, R. Patra, *Routing in a Delay Tolerant Network*, ACM SIGCOMM, 2004.

[4] E. Royer, C-K Toh, *A Review of Current Routing Protocols for Ad-hoc Mobile Wireless Networks*, IEEE Personal Communications Magazine, 1999.

[5] Charles E. Perkins , Elizabeth M. Royer, *Ad-hoc On-Demand Distance Vector Routing*, IEEE Workshop on Mobile Computer Systems and Applications, Feb. 1999.

[6] Charles E. Perkins , Pravin Bhagwat, *Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers*, ACM SIGCOMM, Aug. 1994.

[7] J. Broch, D.B. Johnson, D.A. Maltz; *The Dynamic Source Routing Protocol for Mobile Ad hoc Networks*, IETF MANET Working Group, Internet-Draft, Oct. 1999.

[8] A. Vahdat, D. Becker, *Epidemic routing for partially connected ad hoc networks*, Technical Report CS-200006, Duke University, Apr. 2000.

[9] A. Lindgren, A. Doria, and O. Scheln, *Probabilistic Routing in Intermittently Connected Networks*, ACM MobiHoc, 2003.

[10] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, *Spray and wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks*, ACM SIGCOMM, 2005.

[11] B. Burns, O. Brock, B. N. Levine, *MV routing and capacity building in disruption tolerant networks*, IEEE INFOCOM, 2005.

[12] D. Nain et al., *Integrated Routing and Storage for Messaging Applications in Mobile Ad Hoc Networks*, WiOpt: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, 2003.

[13] F. Tchakountio, R. Ramanathan, *Tracking Highly Mobile Endpoints*, IEEE WoWMoM, 2001.

[14] M. Musolesi, S. Hailes, C. Mascolo, *Adaptive Routing for Intermittently Connected Mobile Ad Hoc Networks*, IEEE WoWMoM, 2005.

[15] J. Leguay, T. Friedman, V. Conan, *Evaluating Mobility Pattern Space Routing for DTNs*, IEEE INFOCOM, 2006.

[16] T. Spyropoulos, K. Psounis, C. S. Raghavendra, *Single-copy routing in intermittently connected mobile networks*; IEEE SECON (Sensor and Ad Hoc Communications and Networks), 2004.

[17] J. Ott, D. Kutscher, *A Disconnection-Tolerant Transport for Drive-thru Internet Environments*, IEEE INFOCOM, 2005.

[18] J. Burgess, B. Gallagher, D. Jensen, B. N. Levine, *MaxProp: Routing for Vehicle-Based Disruption- Tolerant Networks*, IEEE INFOCOM, 2006.

[19] M.M.B. Tariq, M.H. Ammar, E.W. Zegura, *Message ferry route design for sparse ad hoc networks with mobile nodes*, ACM MobiHoc, 2006.

[20] R. C. Shah, W. Brunette, *Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks*, Intel Research Tech Report IRS-TR-03-001, 2003.

[21] T. Spyropoulos, A. Jindal, K. Psounis, *An Analytical Study of Fundamental Mobility Properties for Encounter-Based Protocols*, Technical Report CENG-2007-8, University of Southern California, 2007.

[22] X.Zhuo, Q.Li, W.Gao, G.Cao, Y.Dai, *Contact Duration Aware Data Replication in Delay Tolerant Networks*, IEEE ICNP, 2011.

[23] T. Spyropoulos, K. Psounis, C. S. Raghavendra, *Spray and focus: Ef?cient mobility-assisted routing for heterogeneous and correlated mobility*, IEEE PERCOM, 2007.

[24] P.Hui, J. Crowcroft, E. Yoneki, *Bubble rap: Social-based forwarding in delay-tolerant networks*, IEEE Transactions on Mobile Computing, Vol. 10, No. 11, 2011.

[25] A. Balasubramanian, B. N. Levine, A. Venkataramani, *DTN Routing as a Resource Allocation Problem*, SIGCOMM, 2007.

[26] T. Small, Z. Haas, *Resource and Performance Tradeo?s in Delay-Tolerant Wireless Networks*, ACM WDTN, 2005.

[27] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, D. Rubenstein, *Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet*, ACM SIGPLAN Notices, Vol. 37, 2002.

[28] K. Fall, S. Farrell, *DTN: An Architectural Retrospective*, IEEE Journal on Selected Areas in Communications, Vol. 26, Issue. 5, Jun. 2008.

[29] M. Demmer, E. Brewer, K. Fall, S. Jain, M. Ho, R. Patra, *Implementing Delay Tolerant Networking*, IRB-TR-04-020, Dec. 2004.

[30] *The ONE simulator*, URL: `http://www.netlab.tkk.fi/tutkimus/dtn/theone`

[31] A.Keranen, J.Ott, T.Karkkainen, *The ONE Simulator for DTN Protocol Evaluation*, SIMUTools, 2009.

[32] C.V. Samaras, V. Tsaoussidis, *Adjusting Transport Segmentation Policy of DTN Bundle Protocol under Synergy with Lower Layers*, Elsevier Journal of Systems and Software, Vol. 84, Issue 2, 2011.

[33] K. Scott, S. Burleigh, *Bundle Protocol Specification*, RFC 5050, The MITRE Corporation, NASA Jet Propulsion Laboratory, Nov. 2007.

[34] N. Bezirgiannidis, V. Tsaoussidis, *Packet size and DTN transport service: Evaluation on a DTN Testbed**, ICUMT, Oct. 2010.

[35] W. Ivancic, W. M. Eddy, D. Stewart, L. Wood, J. Northam, C. Jackson, *Experience with delay-tolerant networking from orbit*, 4th Advanced Satellite Mobile Systems, Aug. 2008.

[36] *Interplanetary Overlay Network (ION) Design and Operation*, V1.11, Jet Propulsion Laboratory, California Institute of Technology, Dec. 2009. URL: `https://ion.ocp.ohiou.edu`

[37] A.Keranen, *Opportunistic Network Environment Simulator*, Special Assignment Report, 2008.

[38] URL: `http://tier.cs.berkeley.edu/drupal/`

[39] A Kernen, T Krkkinen, J Ott, *Simulating Mobility and DTNs with the ONE*, Journal of Communications, 2010.

[40] Y. Cao, H. Cruickshank, Z. Sun, *A Routing Framework for Delay Tolerant Networks Based on Encounter Angle*, IWCMC, 2011.

[41] K. Fall, *A Message-Switched Architecture for Challenged Internets*, IRB-TR-02-010, Intel Research, Berkeley, Jul. 2002.

[42] Evan P. C. Jones, *Practical Routing in Delay-Tolerant Networks*, MASc Thesis, 2006.

[43] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, H. Weiss, *Delay-Tolerant Networking Architecture*, RFC 4838, Google, NASA/Jet Propulsion Laboratory, The MITRE Corporation, Intel Corporation, SPARTA, Inc., Apr. 2007.

[44] S. Symington, *Delay-Tolerant Networking Metadata Extension Block*, RFC 6258, The MITRE Corporation, May 2011.

[45] M. Demmer, J. Ott, S. Perreault, *Delay-Tolerant Networking TCP Convergence-Layer Protocol*, RFC 7242, UC Berkeley, Jun. 2014.

[46] K. Fall, *A Delay-Tolerant Network Architecture for Challenged Internets*, SIGCOMM, Aug. 2003.

[47] N.L. Clarke, B. V. Ghita, S. M. Furnell, *Chapter 3: Delay-Tolerant Networks (DTNs) for Deep-Space Communications*, Book: Advances in Delay-tolerant Networks (DTNs), ISBN: 978-0-85709-840-5, Nov. 2014.

[48] *Transmission Control Protocol*, RFC 793, University of Southern California, Sept. 1981.

[49] URL: `www.cse.buffalo.edu/~qiao/cse620/fall04/ipn.ppt`

[50] S. Farrell, V. Cahill, D. Geraghty, I. Humphreys, P. McDonald, *When TCP Breaks: Delay-and Disruption-Tolerant Networking*, Internet Computing, IEEE Vol.10, Issue 4, Sept. 2006.

[51] F. Warthman, *Delay-Tolerant Networks (DTNs): A Tutorial*, 2003.

[52] H. Eriksson, P. Jönsson, *Implementation and Analysis of the Bundling Protocols for Delay-Tolerant Network Architectures*, M.Sc. Theis, Luleå University of Technology, 2005.

[53] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, E. Travis, and H. Weiss, *Interplanetary Internet (IPN): Architectural Definition*, IPN Research Group, Internet Draft, May 2001.

[54] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, H. Weiss, *Delay-Tolerant Networking: An Approach to Interplanetary Internet*, IEEE Communications Magazine, Jun. 2003.

[55] I. F. Akyildiz, O. Akan, C. Chen, J. Fang, W. Su, *Interplanetary internet: state-of-the-art and research challenges*, Computer Networks, 43(2):75-112, 2003.

[56] *Beyond MANETs: Dissertation on Opportunistic Networking*, found at URL:`http://citeseerx.ist.psu.edu`

[57] C. Gifford, *Eyewitness : Media and Communication*, Book, DK Publishing, 1999.

[58] *Internet Protocol*, RFC 791, University of Southern California, Sept. 1981.

[59] C. Caini, H. Cruickshank, S. Farrell, M. Marchese, *Delay- and Disruption-Tolerant Networking (DTN): An Alternative Solution for Future Satellite Networking Applications*, IEEE Communications Journal, Vol. 99, No. 11, Nov. 2011.

[60] S. Lin, D. Costello, M. Miller, *Automatic-repeat-request error-control schemes*, IEEE Communications Magazine, Vol. 22, No. 12, Dec. 1984.

[61] H. Kruse, *UDP Convergence Layers for the DTN Bundle and LTP Protocols*, Internet-Draft, Nov. 2008.

[62] S. Burleigh, *Delay Tolerant Networking LTP Convergence Layer (LTPCL) Adapter*, Internet-Draft, Aug. 2010.

[63] M. Ramadas, S. Burleigh, S. Farrell, *Licklider Transmission Protocol Specification*, Internet RFC 5326, Sept. 2008.

[64] B. Adamson, C. Bormann, M. Handley, J. Macker, *NACK-Oriented Reliable Multicast (NORM) Transport Protocol*, Internet RFC 5740, Nov. 2009.

[65] E. Kohler, M. Handley, S. Floyd, *Datagram Congestion Control Protocol (DCCP)*, Internet RFC 4340, Mar. 2006.

[66] *DTN2 Reference Implementation*,
URL: `https://sites.google.com/site/dtnresgroup/home/code`

[67] T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifier (URI): Generic Syntax*, RFC 3986, STD 66, Jan. 2005.

[68] *Message Switching*, Wikipedia,
URL: `http://en.wikipedia.org/wiki/Message_switching`

[69] K. Fall, W. Hong, S. Madden, *Custody Transfer for Reliable Delivery in Delay Tolerant Networks*, 2003.

[70] S. Symington, S. Farrell, H. Weiss, P. Lovell, *Bundle Security Protocol Specification*, Internet RFC 6257, Mar. 2011.

[71] M. Seligman, K. Fall, and P. Mundur, *Alternative custodians for congestion control in delay tolerant networks*, ACM SIGCOMM Workshop on Challenged Networks, 2006.

[72] T. Small, Z. Haas, *Resource and Performance Tradeoffs in Delay-Tolerant Wireless Networks*, ACM WDTN, Aug. 2005

[73] URL:`https://en.wikipedia.org/wiki/Routing_in_delay-tolerant_networking`

[74] S. C. Nelson, A. F. Harris, R. Kravets, *Event-driven, role-based mobility in disaster recovery networks*, Workshop on Challenged Networks (CHANTS), 2007.

[75] A. Pentland, R. Fletcher, A. Hasson, *DakNet: Rethinking Connectivity in Developing Nations*, IEEE Computer, Vol. 37, No. 1, Jan. 2004.

[76] URL: `http://www.haggleproject.org/`

[77] V. Bharghavan et. al., *MACAW: A Medium Access Protocol for Wireless LAN's*, ACM SIGCOMM, Aug. 1994.

[78] URL: `http://ns1758.ca/winch/winchest.html`

[79] M. Grossglauser, D. N. C. Tse, *Mobility Increases the Capacity of Ad Hoc Wireless Networks*, IEEE/ACM Transactions on Networking (TON), Vol. 10 Issue.4, Aug. 2002.

[80] URL: `https://en.wikipedia.org/wiki/Beacon_frame`

[81] L. Wood, W. Eddy, W. Ivancic, C. Jackson, *Saratoga: A scalable Data Transfer Protocol*, Internet Draft, Sept. 2011. URL: `https://tools.ietf.org/html/draft-wood-tsvwg-saratoga-10`

[82] P. Matzakos, C. Bonnet, *Transport Layer Protocols and Strategies for Delay and Disruption Tolerant Networks*, EUROCOM Research Report RR-13-285, Jul. 2013.