Ad Hoc Networks 25 (2015) 480–494

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Survey Paper

A survey on congestion control for delay and disruption tolerant networks

Aloizio P. Silva [a],*, Scott Burleigh [c], Celso M. Hirata [a], Katia Obraczka [b]

[a] Instituto Tecnológico de Aeronáutica, Department of Electronic and Computer Engineering, Brazil
[b] University of California Santa Cruz, Computer Engineering Department, United States
[c] Jet Propulsion Laboratory, California Institute of Technology, United States

ARTICLE INFO

Article history:
Received 24 February 2014
Received in revised form 22 July 2014
Accepted 28 July 2014
Available online 20 August 2014

Keywords:
Delay and disruption tolerant networks
Interplanetary networks
Congestion control
Intermittent connectivity

ABSTRACT

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

1. Introduction

Delay and Disruption Tolerant Networks (DTNs) were initially motivated by the idea of deploying an Interplanetary Internet (IPN) [1] for deep space communication. As a result, a framework for an IPN which aims to use an interplanetary backbone to connect internetworks in space was developed. Over time, a diverse set of other DTN applications for "extreme" environments on Earth have emerged including vehicular networks, emergency response and military operations, surveillance, tracking and monitoring applications, and bridging the digital divide. In these applications, long delays are a consequence of the long distances and/or episodic connectivity which are characteristic of "extreme" environments.

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

The DTN architecture described in [2] uses the so-called *store-carry-and-forward* paradigm, as opposed to the Internet's *store-and-forward*, to deliver messages from source to destination. In *store-carry-and-forward*, nodes store incoming messages and forward them when transmission opportunities arise. Note that in traditional networks, nodes also store messages before forwarding them;

---

* Corresponding author.

E-mail addresses: aloizio@ita.br (A.P. Silva), scott.c.burleigh@jpl.nasa.gov (S. Burleigh), hirata@ita.br (C.M. Hirata), katia@soe.ucsc.edu (K. Obraczka).

http://dx.doi.org/10.1016/j.adhoc.2014.07.032
1570-8705/

however, the time scales at which data is stored locally while waiting to be forwarded are typically orders of magnitude smaller when compared to DTNs. Therefore, storage in *store-carry-and-forward* typically uses persistent storage which implies that DTN nodes need to be equipped accordingly.

According to the *store–carry-and-forward* paradigm, when a DTN node "encounters" another DTN node, it decides whether to forward messages it is carrying to the other node. Therefore, the concept of *links* in traditional networks (wired or wireless) is replaced with the notion of *contacts*. In scenarios where these encounters are random, *store–carry-and-forward* is also referred to as *opportunistic forwarding*. On the other hand, when contacts are known a priori (e.g., in deep space communication applications), *store–carry-and-forward* is known as *scheduled forwarding*. Finally, there are scenarios where node encounters follow a probability distribution based on past history; in these cases, *store–carry-and-forward* is based on *probabilistic forwarding* [3]. Note that, since *contact times* are finite and may be arbitrarily short, a node may need to choose which messages to forward based on some priority; a node may also decide whether the new neighbor is a "good" candidate to carry its messages. A node's "fitness" as a relay for a particular message depends on several factors that can be dependent on the message's ultimate destination (e.g., how often that potential relay encounters the destination, etc.); there are also factors that are destination–independent, for example, the relay's mobility patterns, its capabilities (e.g., storage, energy, etc.) [4,5].

The simplest DTN forwarding technique is called *epidemic forwarding* [3,6–9], which is to DTNs what *flooding* is to traditional networks. To address issues such as limited contact times and limited network and node resources, several variants of "pure" *epidemic forwarding* [7,10,11] have been proposed. For instance, before a node forwards its messages to another node upon contact, the two nodes perform an initial "handshake" in which they exchange a summary of the messages each one has; then they only exchange messages that the other does not already carry. There are also a number of "controlled" epidemic variants that try to, implicitly or explicitly, limit the number of copies of the same message in the network.

The fact that in DTNs the existence of an end-to-end path between any pair of nodes at all times cannot be guaranteed raises fundamental challenges in end-to-end reliable data delivery. In DTNs, the Internet model of end-to-end reliability (as implemented by TCP) is not applicable. The DTN architecture proposed in [2] replaces TCP's end-to-end reliability with *custody transfer*, which uses hop-by-hop acknowledgements to confirm the correct receipt of messages between two directly connected nodes. Additionally, due to the inability to guarantee end-to-end connectivity at all times, functions based on the TCP/IP model such as congestion and flow control will not always work in DTNs. Instead, hop-by-hop control can be employed.

In this paper, we survey the state-of-the-art on DTN congestion control mechanisms. To this end, we propose a taxonomy to help (1) map the DTN congestion control design space and (2) compare existing DTN congestion control mechanisms. The survey presented in [12] considers reliability and congestion control proposals focusing on opportunistic networks. Note that opportunistic networks are a special case of DTNs where contacts between nodes are not known a priori. In our survey, we consider DTNs as more broadly defined: in addition to opportunistic networks, i.e., networks where contacts are random, we also explore networks in which contacts are scheduled well as networks in which contacts are probabilistic (based on some probability function derived from past contacts). The tutorial presented in [13] discusses the prospects of using DTN in future satellite networks, in particular LEO/GEO satellite constellations. Studies such as [14,15] confirm that congestion control is a fundamental issue in DTNs and note that it has not received much attention from the DTN research community. Our work goes a step further and provides a deeper analysis of existing DTN congestion control mechanisms.

The remainder of this paper is organized as follows. Section 2 provides an overview of the DTN architecture and discusses DTN congestion control comparing it against traditional Internet congestion control. In Section 3, we present a taxonomy for DTN congestion control and in Section 4, we describe existing DTN congestion control mechanisms in light of the proposed taxonomy. Section 5 provides design recommendations for future DTN congestion control mechanisms based on insights gained from our discussion of the current DTN congestion control state-of-the art. Finally, Section 6 concludes the paper.

2. Background

2.1. The DTN architecture

The DTN architecture, which was originally proposed in [16], aims at providing implementations for reliable message delivery in intermittently-connected networks. It introduces the *store–carry-and-forwarding* paradigm under which messages may remain stored for relatively long periods of time in persistent storage at intermediate nodes while in transit from source to destination. The DTN architecture was designed to operate as an intermediate layer, called the *bundle layer*, between the application and the transport layers of the networks it interconnects (see Fig. 1). It provides services such as in-network data storage and retransmission, interoperable naming, authenticated forwarding, and coarse-grained classes of service.

The DTN architecture also specifies the *bundle protocol* [16–19] which controls the exchange of *bundles*, i.e., application-layer messages. The Bundle Protocol can operate either atop transport protocols (e.g., TCP, UDP, etc), or atop lower layer protocols (e.g., Bluetooth, Ethernet, etc). The term "bundle" was chosen to connote the self-sufficiency of the messages: application-layer messages are expected to contain sufficient metadata to enable processing by the recipient without negotiation, as if all relevant metadata query and response messages have been anticipated by the sender and "bundled" into a single application data unit. When operating atop the transport layer, the bundle protocol receives messages from the application layer, encapsulates them into bundles, whose format is depicted