Pervasive and Mobile Computing 29 (2016) 57–72

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

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

Delay tolerant lazy release consistency for distributed shared memory in opportunistic networks

Chance Eary [a,*], Mohan Kumar [b], Gergely Zaruba [a]

[a] *Department of Computer Science and Engineering, The University of Texas at Arlington, Box 19015, 500 UTA Boulevard, Room ERB 559, Arlington, TX 76019, United States*

[b] *Department of Computer Science, The Rochester Institute of Technology, 102 Lomb Memorial Dr., Rochester, NY 14623, United States*

ARTICLE INFO

Article history:
Received 14 February 2014
Received in revised form 27 February 2015
Accepted 10 March 2015
Available online 23 March 2015

Keywords:
Opportunistic networks
Delay tolerant networks
Distributed systems
Distributed shared memory
Lazy release consistency

ABSTRACT

Opportunistic networks (ONs) allow mobile wireless devices to interact with one another through a series of opportunistic contacts. While ONs exploit mobility of devices to route messages and distribute information, the intermittent connections among devices make many traditional computer collaboration paradigms, such as distributed shared memory (DSM), very difficult to realize. DSM systems, developed for traditional networks, rely on relatively stable, consistent connections among participating nodes to function properly.

We propose a novel delay tolerant lazy release consistency (DTLRC) mechanism for implementing distributed shared memory in opportunistic networks. DTLRC permits mobile devices to remain independently productive while separated, and provides a mechanism for nodes to regain coherence of shared memory if and when they meet again. DTLRC allows applications to utilize the most coherent data available, even in the challenged environments typical to opportunistic networks. Simulations demonstrate that DTLRC is a viable concept for enhancing cooperation among mobile wireless devices in opportunistic networking environment.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

The opportunistic networking paradigm allows mobile wireless devices to interact when they are within communication range of one another [1,2]. As devices move about their environment and enter one another's transmission range, they can create a temporary, point-to-point connection between themselves. This connection presents an opportunity for processes to exchange data and collaborate with no dependence on pre-existing networking infrastructure, or foreknowledge of where a device will be and when.

Because wireless devices in an opportunistic network are mobile, connections among them are erratic and susceptible to being dropped without warning [3]. Devices are assumed to have no ability to determine when, or if, they will again meet to resume collaboration.

Many technologies developed to facilitate collaboration among processes over network connections, such as distributed shared memory (DSM), were intended for use with consistent and stable links between nodes [4]. DSM has been a research topic within computer science for several decades, with the initial implementation being proposed in [5]. Central to all models proposed in traditional distributed computing is that each process participating in the system will have consistent access

* Corresponding author.
E-mail addresses: chance.eary@mavs.uta.edu (C. Eary), mjk@cs.rit.edu (M. Kumar), zaruba@uta.edu (G. Zaruba).

http://dx.doi.org/10.1016/j.pmcj.2015.03.001
1574-1192/© 2015 Elsevier B.V. All rights reserved.

to shared memory, be it located on the same physical machine or available across a network connection. As DSM systems evolved, different methods to ensure the consistency of shared data were proposed [6]. These schemes began to incorporate mechanisms to increase a system's tolerance to network delay, but still assumed participating nodes and their data would ultimately be accessible when necessary. Within an ON, neither of the aforementioned assumptions are applicable.

To facilitate the collaboration of mobile wireless nodes in the presence of unpredictable and intermittent network connections, we propose delay tolerant lazy release consistency (DTLRC). DTLRC has two goals:

- Allow two or more processes to share content in an ON; and,
- Ensure that a node can continue working on shared data even if it is separated from its peers for extended periods of time.

Using DTLRC as a foundation, we propose Social Cache (SC). Social Cache allows frequently encountering nodes to have increased shared memory.

We carry out extensive simulation studies to evaluate DTLRC. Through various scenarios, we demonstrate that maintaining consistency of shared memory among nodes utilizing brief opportunistic connections is possible.

Section 2 gives an introduction to opportunistic networking and a discussion of existing DSM systems. Section 3 provides an overview of the theory and algorithms used in DTLRC. Section 4 describes the operation of DTLRC. Section 5 discusses the theory and algorithms used in Social Cache. Sections 6 and 7 discuss simulations and analysis of DTLRC. Sections 8 and 9 discuss simulation and analysis of Social Cache. Section 10 includes closing remarks.

2. Background

This section discusses opportunistic networks as well as extant consistency schemes for distributed shared memory in traditional networks.

2.1. Opportunistic networks

An ON is created by a series of pair-wise opportunistic contacts between devices, distributed in space and time. ONs show great potential in exploiting the mobility of wireless devices by allowing communication with one another when they are in range. ONs require neither infrastructure support nor prior planning [1–3,7].

When attempting to apply computing techniques intended for traditional wired networks over opportunistic networks, a number of key challenges are encountered. Amongst the most difficult to address is the issue of devices not knowing with any certainty how long other nodes within the ON will be available to participate in the network. Uncertainties about a node's availability, join and leave patterns, and the length of time for which it remains connected, are major challenges of ONs [1,2]. This, combined with the intermittent delays and data losses that are typical of the wireless medium [8], makes ONs a highly challenged networking paradigm.

2.2. Distributed shared memory

One method for executing processes to collaborate with one another over networks is distributed shared memory (DSM). Distributed shared memory allows multiple processes, either on the same physical system or connected via a network, to concurrently operate on a set of data [4]. In order to facilitate content sharing among mobile devices connected opportunistically, an implementation of DSM for use in ONs should be formulated.

Traditional DSM systems are designed for use over networks with relatively high reliability links among highly available nodes [9]. While traditional DSM systems are tolerant to network delays, they were not designed for repeated loss of connections a DSM implementation in ONs would entail. The separated nodes will remain fully functional while disconnected from one another, but their processing power cannot be utilized until the resumption of connectivity. Such disconnections are expected to be commonplace in opportunistic networks.

In order for distributed shared memory to work correctly, all processes participating in the system must have a consistent view of shared data. Consistency schemes are methods that allow processes to decide which value in the shared data is appropriate to use, and which value should be overwritten to reflect more relevant writes. Simply using the "most recent" write at all processes is not feasible, as it is assumed nodes do not share a common clock due to complications from inconsistent network conditions [10]. Many consistency schemes have been proposed over the years [5,6,11]. One scheme focusing specifically on memory consistency in potentially high-latency networking environments is lazy release consistency (LRC) [12,13].

2.3. Lazy release consistency

LRC utilizes a variety of techniques to avoid synchronizing memory between two processes until absolutely necessary, and then only by exchanging a minimum of data. One such technique is only exchanging consistency information between

Download English Version:

https://daneshyari.com/en/article/465840

Download Persian Version:

https://daneshyari.com/article/465840

Daneshyari.com