Computer Communications 109 (2017) 176–183

Contents lists available at ScienceDirect

Computer Communications

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

DiTON: Distributed transactions in opportunistic networks

Chance Eary[a,*], Mohan Kumar[a,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 22 May 2016
Revised 4 April 2017
Accepted 17 June 2017
Available online 19 June 2017

Keywords:
Opportunistic networks
Delay tolerant networks
Distributed systems
Distributed transactions

ABSTRACT

Opportunistic networks (ONs) allow wireless devices to communicate and collaborate in the absence of dedicated networking infrastructure. Recent research work exploits mobility of nodes to create delayed paths in opportunistic networks. Execution of distributed transactions that comprise a sequence of atomic operations across multiple nodes continues to be a challenge for opportunistic networks. Implementing distributed transactions poses new challenges as participation of multiple processes, both local and remote, is required.

In this paper, we propose and implement a novel transaction system called DiTON, specifically tailored to operate in opportunistic networks for delay-tolerant applications. To the best of our knowledge, our work is the first attempt to implement a transaction paradigm specifically intended for opportunistic networks. DiTON is a novel transaction scheme that allows multiple nodes to collaborate on shared sets of data while providing global coherency despite network interruptions. Through experimental results, we demonstrate that DiTON can be successfully implemented in opportunistic networks.

1. Introduction

In opportunistic networks, wireless mobile devices interact with one another by forming peer-to-peer connections when they are within communication range [1,2]. Using onboard radios, mobile devices can create temporary, ad hoc connections between themselves without the use of pre-existing networking infrastructure (i.e., WiFi hotspots, cell phone towers, etc). These connections present opportunities for a set of devices with shared goals and common interests to collaborate and exchange data [3–5].

Transactions are defined as sequences of actions that must be completed as specified by their program, or aborted completely with no changes to memory [6]. Originally introduced in [7], transactions adhere to the ACID properties [8], outlined below, to maintain the most rigorous consistency of operations attainable:

- *Atomic* – a transaction must complete in an appropriate way, or all effects of the transaction must be discarded;
- *Consistent* – a transaction takes the collective system from one consistent state to another consistent state;
- *Isolated* – operations performed for transactions are free from interference by operations being performed on behalf of other concurrent clients; and,
- *Durable* – once a transaction has completed successfully, all its effects are saved in permanent storage.

Transactions are important tools in the context of shard memory, as they can be used to ensure a consistent view for all actors. The ACID properties ensure that writes to shared memory result in the program's intended outcome, even when multiple concurrent operations are being performed on the shared set of data, or when one of the participant processes becomes unavailable either through a process crash, network disconnection, or other undesirable event.

Distributed transactions (simply called *transactions* in this paper for brevity) allow multiple processes, called participants, to perform operations on a shared set of data over a network [9,10]. While numerous variants of transactions exist, a typical client-server example might proceed as follows [11]:

1. As a transaction begins to execute, it must first obtain locks on relevant data. In general, all necessary locks must be obtained prior to executing operations. This helps to ensure both the *atomic* and *isolated* requirements;
2. The transaction reads and writes all required data. This sequence of operations completes a single, logical function (i.e., updating the balance of a bank account);
3. When the transaction has finished its operations, the server and client agree on the data to be saved.

* 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.comcom.2017.06.005
0140-3664/

- If an agreement cannot be reached, due to a process crash, network disconnection, or other disruption, all changes to data must be discarded. This is called an 'abort'; or,
- If an agreement is reached, both the client, and the server, write the data to non-volatile storage. This is called a 'commit'.

These steps ensure the *consistent* and *durable* requirements; and,

4. The transaction releases all of its held locks, and the client terminates its connection to the server.

Transactions are critical to ensuring reliable functionality in distributed computing. They facilitate multi-stepped functionality across various interconnected processes which will dependably terminate in a coherent, and expected, fashion. In order for ONs to enhance their utility beyond exchanges of content or routing data, a variant of transactions should be supported. As the Internet of Things (IoT) gradually becomes a reality, such a variant could be used to support mobile commerce, mobile auctions, or e-medicine, for example.

For instance, an individual attending a swap meet with the intention of selling an item could use an app to advertise that item, and accept potential offers from interested parties. As all of the participants at the meet are proximally related, this could be done without needing to setup an offsite server, and instead, use peer-to-peer connections. After setting the minimum acceptable price, the user proceeds to move across the fair grounds to browse the wares from other vendors.

Upon receiving a notification that an item is available, an interested buyer could use her app to make a bid on the item. However, prior to a full commit of the bid, the seller moves out of range and the devices lose their connection, leaving the data in an indeterminate state. In an environment where disconnections will occur frequently, a mechanism to ensure coherent data in this scenario is necessary for any meaningful progress to occur.

ONs provide a challenging environment for deploying transactions. As most devices within an ON are mobile, connections between them are erratic and susceptible to being dropped with no warning. Devices are assumed to have no knowledge about when, or if, they will meet again to resume collaboration. This is expected behavior in ONs which extant transaction schemes would treat as a failure, almost certainly resulting in aborting operations with little or no useful work accomplished.

Functioning exclusively under the strict ACID properties may not produce satisfactory results in such an environment. Relaxing the ACID properties, similar to previous work on transactions in mobile ad hoc networks (MANETs), is still insufficient to support distributed transactions in ONs. For transactions to be viable in ONs, additional capabilities that operate with frequent and unpredictable network disruptions are needed.

In this work, we develop a novel transaction system, called DiTON (distributed transactions in opportunistic networks), specifically tailored to ONs, where incessant network interruptions present major obstacles to the successful completion of complex operations. DiTON attempts to provide useful progress towards finishing transactions, even when the participating processes have been temporarily, or permanently, disconnected from each other. We implement DiTON, and provide experimental, and analytical, results demonstrating its behavior in operation.

2. Related work

Distributed transactions have been an area of investigation in computer science since the 1980s [9,10,12]. Andrychowicz et al. [13] demonstrated that Bitcoin transactions can be implemented for secure multiparty computations in traditional wired networks. While methods used with distributed systems on wired networks have limited applicability in opportunistic networks, work on transactions in mobile ad hoc networks is worth considering in the context of opportunistic networks.

Many of the mobile transaction (MT) models proposed for deployment in MANETs assume the presence of both mobile hosts (MHs) that move around their environment, and fixed hosts (FHs) often operating on high-throughput wired networks. MANET transactions can be broadly categorized as follows [14]:

1. Transaction execution on FH only – Here, MHs simply submit their transactions to fixed hosts, which complete the transaction and return the results;
2. Transaction execution on MH only – In this case, the transaction is entirely executed on a single mobile host. The MH is assumed to have all the necessary data to complete the operation independent of other devices;
3. Distributed execution between a MH and FHs – This model allows for some operations to be performed on the MH, with other resource-intensive operations performed by available FHs;
4. Distributed execution among MHs – This scenario assumes no availability of FHs, leaving the mobile devices to perform transaction operations exclusively between themselves; and,
5. Distributed execution among MHs and FHs – This case could be considered as a "fully distributed" scenario, where all available resources of the MANET are cooperating to complete MTs. This scenario is an extension of Category 3, with the distinction being the participation of multiple MHs.

Category 4 proves the most challenging. In this scenario there are no fixed, dependable hosts or networks available for mobile nodes to utilize, and is thus the most closely related to the opportunistic environment. Unlike transactions in MANETs where network disruptions are treated as a challenge, DiTON exploits short-lived opportunistic peer-to-peer connections to make transactions happen.

Extant schemes for transactions in MANETs are unsuitable to ONs [15]. While existing schemes have the ability to recover from node faults and link faults [15,16], loss of connectivity among nodes (e.g., a partitioned network) is treated as a failure in MANETs [6]. Conversely, in ONs, lack of end-to-end connectivity is expected behavior and thus the challenges of Category 4 become more significant.

Costea et al. have proven that causal ordering can be efficiently established in ONs, and propose a Commutative Replicated Data Type / Conflict-Free Replicated Data Type (CRDT) that will achieve total ordering within opportunistic networks [17]. Eary and Kumar [4] proposed a novel delay tolerant lazy release consistency (DTLRC) mechanism for implementing distributed shared memory in opportunistic networks. This article did not investigate transactions in distributed opportunistic networks.

Eary et al. [5] outlined the requirements of a cache-based mechanism for distributed transactions in ONs. In contrast, in this current paper, we propose and implement a novel transaction system tailored to operate in opportunistic networks for delay-tolerant applications. The architecture, operating methodology, implementation, and experimental results presented in this paper are all new.

The primary goal of our proposed work is to develop robust transaction mechanisms that either tolerate disruptions or mitigate effects of disruptions at additional cost. To the best of our knowledge, we are the first to propose a mechanism for distributed transactions in opportunistic networks.