DiTON: Towards Facilitating Distributed Transactions in Opportunistic Networks

Chance Eary, Mohan Kumar, and Gergely Zaruba
Dept. of Computer Science and Engineering
University of Texas at Arlington
chance.eary@mavs.uta.edu, mkumar@uta.edu, zaruba@uta.edu

***Abstract*—Opportunistic networks (ONs) exploit mobility of wireless devices to route messages and distribute content independent of fixed network infrastructure. An inherent property of ONs is that the unpredictable mobility of devices results in brief and intermittent connections between networked nodes. This presents a challenge to implementing useful distributed system paradigms, such as distributed transactions. Transactions are a sequence of read and write operations that must be executed in a reliable and coherent fashion, even in the presence of multiple concurrent operations or process failures. Distributed transactions require the participation of multiple processes utilizing data available locally to collaborate with other processes.**

In this work in progress paper, we outline the requirements behind a novel cache-based mechanism for distributed transactions in ONs. The requirements are defined with the expectation that the proposed mechanism will be robust and energy efficient.

I. Introduction

In opportunistic networks, wireless mobile devices interact with each other in a peer-to-peer fashion when they are within communication range [1] [2]. Using onboard radios, mobile devices can create temporary, point-to-point connections between themselves without the use of pre-existing networking infrastructure (i.e., WiFi hotspots, cell phone towers, etc). These connections present the opportunity for devices with shared goals and interests to collaborate and exchange data [3] [4].

Transactions define a sequence of actions that must be completed as specified by their program, or aborted completely with no changes to memory [5]. First introduced in [6], transactions adhere to the ACID properties [7] 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.

These properties ensure that writes to memory result in the intended outcome, even while multiple concurrent operations are being performed 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) are important techniques to allow multiple processes, called participants, to collaborate operations on a shared set of data over a network [8] [9]. In order for ONs to enhance their utility beyond exchanges of content or routing data, a variant of distributed transactions should be supported. Such a variant could be used to support mobile commerce, mobile auctions or e-medicine for example.

ONs provide a challenging environment for deploying transactions. As devices within ONs are mobile, connections among 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. Functioning exclusively under the strict ACID properties may be unfeasible 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 due to complications resulting from the completely erratic status of the network. For transactions to be viable in ONs, additional capabilities are needed.

II. Related Work

Transactions on distributed data have been an area of investigation in computer science since the 1980's [8] [9] [10]. While methods used in traditional distributed systems on wired networks have very limited applicability here, recent 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), devices that move around their environment, and fixed hosts (FHs), stationary devices often operating on high-throughput wired networks. MANET transactions can be broadly categorized as follows [11]:

1) Complete execution on FH - Here, MHs simply submit their transactions to fixed hosts, which complete the transaction and return the results;
2) Complete execution on MH - 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 to offload operations to, leaving the mobile devices to perform transaction operations exclusively between themselves; and,
5) Distributed execution among MHs and FHs - This case could be considered the "fully distributed" scenario, where all available resources of the MANET are cooperating to complete MTs. This scenario is an extension of Category 3.

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 the most closely related to the opportunistic environment. Opportunistic networks increase the difficulty of Category 4 by utilizing unpredictable peer-to-peer connections created when other nodes are present in their immediate vicinity. Connections are assumed to be short-lived and nodes will have minimal ability to self-organize.

Extant schemes for transactions in MANETs are unsuitable to ONs [12]. While existing schemes have the ability to recover from node faults and link faults [12] [13] , loss of connectivity among nodes (e.g., a partitioned network) is treated as a failure [5]. In ONs, lack of end-to-end connectivity is expected behavior and thus the challenges of Category 4 become more significant.

The primary goal of our proposed work is to develop improved transaction mechanisms that either tolerate delay or mitigate effects of delay at additional cost. To the best of our knowledge, we are the first to propose a mechanism for distributed transactions in opportunistic networks. The scheme presented in §III uses caching to mask disconnection and delays. Our system of caching and resuming operations after releasing mutual exclusion to shared objects is a novel transaction methodology.

III. Architecture

Our proposed architecture is specifically tailored to work with the erratic connectivity inherent in opportunistic networks. Some details have been omitted due to space requirements.

In our system, each node assumes two roles:

1) *Initiator* - The node that initiated the transaction for consumption by a local process. Any node in the network can initiate a transaction; and,
2) *Participant* - Any node in the network that is participating in the sequence of read / write operations that compose the transaction.

While schemes proposed thus far have not fully solved the problems associated with distributed transactions exclusively among MHs, existing techniques that relax ACID properties but ensure high degrees of consistency can still be leveraged.

A. Relaxing ACID

The ACID properties were intended to work with potentially faulty processes on relatively stable wired networks [7]. Mobile environments, especially ONs are not conducive to satisfying the strict ACID properties.

While the goal of our system, as well as a number of transaction systems developed for MANETs [11], is to attempt to provide strongly consistent mobile transactions, in some execution cases the consistency requirements must be relaxed in order to provide any functionality at all. In order to ensure transaction sustenance in such dynamic environment, the initiator specifies the acceptable level of consistency prior to starting the transaction. Based on the specifications, our system allows the following outcomes:

- *Abort* - participant nodes will be informed of the acceptable level of consistency required by the initiator. The participants can determine if they can meet the required specifications based on techniques discussed in §III-B. If acceptable consistency cannot be met, the participant can abort its portion of the transaction;
- *Graceful degradation* - if the initiator is prepared to accept a lower degree of consistency for the transaction, other participants can attempt to meet the decreased standard or abort;
- *Cost renegotiation* - if the participants cannot achieve the desired level of consistency specified, the participants and the initiator renegotiate to meet consistency requirements at additional cost. Cost is discussed in §III-C; or,
- *Commit* - if the strongest form of consistency is met by participants of the transaction, the participants commit the changes to permanent storage.

A dynamic system to accommodate varying levels of consistency produced by a distributed transaction permits the system to be flexible. The consumer's needs will either be met to within specification, or aborted with no changes to memory at any participant.

B. Dependency Graphs

Nodes are assumed to know which pieces of distributed data they will require when initiating a MT. Based on this knowledge, a graph of dependencies is created to determine which elements of distributed data need to be read and written to complete the MT. This dependency graph is then shared with participants, so that each participant knows its place in the graph.

A node is assumed to have no a priori knowledge of future connections. As a result, it cannot hold locks it initiated for extended periods due to the following reasons:

- In the worst case, the departed node will never reappear and the participant nodes will be deadlocked permanently; or,
- In the preferred case, the node will reappear after a brief period of time and resume operations. However, any time spent waiting for the node to reconnect reduces the potential for concurrency on the network and requires the consumer to tolerate more delay.

In a concurrent environment, multiple transactions may be in execution with new MTs being constantly initiated. New

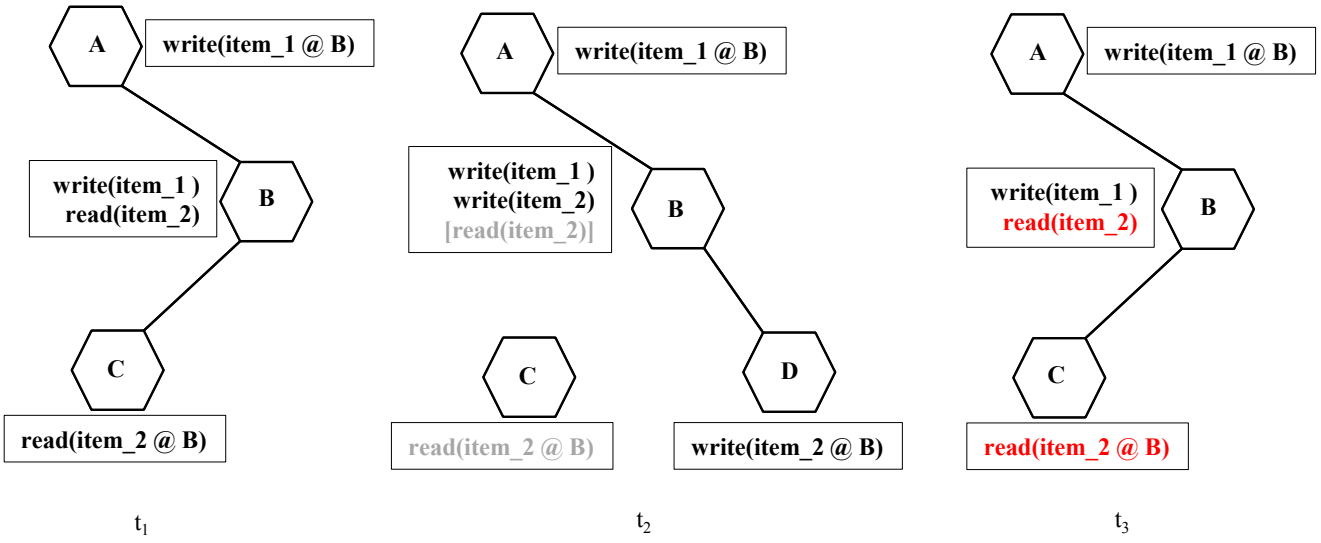

Fig. 1: Example Dependency Graph

transactions must be allowed to proceed. We investigate three issues related to transactions in ONs.

1) Conflicting Operations: A node may determine that it needs to unlock an object to allow another transaction to proceed before the waiting transaction has completed. In this case, its portion of the dependency graph for the waiting MT is cached. The dependency graph of the incoming transaction is compared to the dependency graph of the waiting MT. Should the incoming transaction require reads or writes to objects locked by the waiting transaction, those objects can be unlocked to permit the incoming transaction to proceed.

Once the incoming transaction has completed, the process compares the dependency graphs of the completed transaction and the waiting transaction. The node then determines what, if any, operations can still be completed for the waiting transaction, if and when the node carrying the objects required for it reconnects to the network.

If conflicting operations are found (e.g., the completed transaction wrote to an object locked by the waiting transaction), the node will know what relaxation in the ACID requirements it can still adhere to. If the reduced ACID requirements meet the specifications provided by the initiator, the transaction can complete with the lowered requirements. Otherwise, the node can abort.

Figure 1 illustrates the following scenario with participating nodes A, B, C and D:

1) At time t_1, A writes item_1 at B, and C reads item_2 at B.
2) At t_2, C disconnects from B before the working transaction is committed, and D connects to B. B unlocks item_2 to allow Ds transaction to complete and caches node Cs read operation. Cs previous read of item_2 is now inconsistent.
3) At time t_3, D completes its transaction and disconnects from the network. C reconnects with B and attempts to complete its piece of the transaction. B notes that the cached operation for C can no longer be completed as specified.

2) Heterogenous Consistency Requirements: As multiple participants can initialize transactions concurrently, conflicting consistency requirements may become an issue. For example, one participant may require an object to be written to with as high a consistency as possible, whereas another process can tolerate weaker consistency on that object and elect to proceed with loosened requirements. This would effectively result in one participant starving as its consistency requirements are repeatedly undercut by other processes operating on the data.

Dependency graphs can find these conflicting interests and can thus provide a mechanism for participants to negotiate an acceptable plan to share data. The following illustrate a subset of acceptable results:

- Participants requiring stronger consistency incur additional cost (§III-C) to cover the expense necessary for all participants operating on that data to adhere to stronger consistency;
- Participants permitting weaker consistency incur additional cost to assist the participant requiring stronger consistency to meet its consistency demands;
- Participants permitting weaker consistency proceed to read and write to tentative versions (a copy of the most recently committed version), while nodes requiring stronger consistency can read from their committed versions only; or,
- The participant requiring stronger consistency can partition the data necessary for its strong transaction and only work to arrive at network-wide consistency at determined intervals.

In order to reduce delay to consumers, dynamic methods for managing different consistency requirements would be an important aspect of making distributed transactions in

opportunistic networks useful.

C. Managing Cost

Any MT management scheme in ONs has to be tailored to work without fixed infrastructure. However, if an initiator's requirements in consistency or time cannot be met by the ON, it can elect to incur additional cost by attempting to access pre-existing networks. This section discusses how costs are managed.

1) Time: Opportunistic networks are only suitable to delay-tolerant applications but systems should still attempt to complete operations as quickly as possible. If the initiator has not completed the transaction, or has not obtained the desired level of consistency, the transaction can be resubmitted to the network and tried again. Developing an algorithm to precisely specify how much additional delay a consumer is prepared to tolerate will be addressed in our oncoming work.

2) Monetary: While our system is tailored to work without reliance on pre-existing network infrastructure, fixed wireless networks are effectively omnipresent in the developed world. These fixed networks often require monetary payment for their use (e.g., paying for access to WiFi or utilizing a cell phone network provided by a wireless data subscription).

If a MH desires a higher level of consistency and elects not to incur additional delay, our system will attempt to accommodate it by utilizing existing networks with a technique termed "consistency on demand." The following important questions will be answered by our future work when addressing issues with regards to using pay-for-service FHs:

- Which networks are available to route data between MHs and how can these networks be used while minimizing cost?
- Which participant pays for the use of FHs?
  - The node requesting stronger consistency pays for everything; or,
  - The additional cost is shared, either evenly or based on a ratio negotiated by the participants.
- How are monetary costs balanced with time costs, if the user is also prepared to wait in order to obtain the highest consistency possible?

Our system will also consider the heterogeneous cost associated with different participants utilizing different connections. For example, node A may be connected to node B with a free Bluetooth connection, while node B is attached to node C with a for-pay cellular data connection. Our system will balance the applicable monetary considerations for all participants and thus previous work with heterogeneous connections in transaction systems for MANETs can be leveraged [11].

Attempting to dynamically utilize accessible connections to the Internet will add flexibility in cases when users are prepared to tolerate delays within an ON, but occasionally desire more immediate results.

3) Energy: Energy is an important consideration in our algorithms as mobile hosts are assumed to be battery powered. An initiator electing to spend additional time reattempting an aborted transaction results in the additional drain on the battery of both the initiator and participants. An initiator attempting to use fixed network infrastructure will also incur additional cost on the battery, as using WiFi or cellular data networks requires more energy than shorter-range radios such as Bluetooth.

Our algorithms for managing cost will balance energy requirements with both time and monetary cost parameters. Finding balance in cost will give the user the most optimal results available and allow our system to suit as wide an audience of end-users as possible.

IV. Conclusion

Opportunistic networks show interesting potential to use mobile wireless devices to their maximum potential. While ONs present a challenging networking environment, important distributed system paradigms can still be employed. This work proposes a system to facilitate distributed transactions in opportunistic environments.

Acknowledgments

This work was supported in part by the US National Science Foundation under grant CNS-0834493. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the author and do not necessarily reflect the views of the National Science Foundation.

References

[1] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," *SIGCOMM Comput. Commun. Rev.*, vol. 34, pp. 145 – 158, Aug. 2004.

[2] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic networking: Data forwarding in disconnected mobile ad hoc networks," vol. 44, no. 11, pp. 134 – 141, Nov. 2006.

[3] M. Conti and M. Kumar, "Opportunities in opportunistic computing," vol. 43, no. 1, pp. 42 – 50, Jan. 2010.

[4] C. Eary and M. Kumar, "Delay tolerant lazy release consistency for distributed shared memory in opportunistic networks," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, Jun. 2012, pp. 1 –6.

[5] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design, 5 ed.* Boston, MA, USA: Addison-Wesley, 2012.

[6] B. Lampson, "Atomic transactions," in *Distributed systems: Architecture and Implementation. Vol 105 of Lecture Notes in Computer Science*, vol. 2. Springer-Verlag, 1981, pp. 254 – 259.

[7] T. Harder and A. Reuter, "Principles of transaction-oriented database recovery," *ACM Computer Surveys*, vol. 15, no. 4, 1983.

[8] B. Liskov, "Distributed programming in argus," *Communications of the ACM*, vol. 31, no. 3, pp. 300–312, 1988.

[9] S. Shrivistava, G. Dixon, and G. Parrington, "An overview of the arjuna distributed programming system," *IEEE Software*, vol. 8, no. 1, pp. 66–73, 1991.

[10] J. Mitchell and Dion, "A comparison of two network-based file servers," vol. 25, no. 4, pp. 233–245, 1982.

[11] P. Serrano-Alvarado, C. Roncancio, and M. Adiba, "A survey of mobile transactions," *Distrib. Parallel Databases*, vol. 16, no. 2, pp. 193 –230, Sep. 2004.

[12] B. Ayari, A. Khelil, and N. Suri, "On the design of perturbation-resilient atomic commit protocols for mobile transactions," *ACM Trans. Comput. Syst.*, vol. 29, no. 3, pp. 7:1–7:36, Aug. 2011.

[13] ——, "Partac: A partition-tolerant atomic commit protocol for manets," in *Mobile Data Management (MDM), 2010 Eleventh International Conference on*, May 2010, pp. 135 –144.