

A Decision-Theoretic Approach to File Consistency in Constrained Peer-to-Peer Device Networks

David L. Roberts Sooraj Bhat Charles L. Isbell Jr.
Brian F. Cooper Jeffrey S. Pierce
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0706
{robertsd,sooraj,isbell,cooperb,jpierce}@cc.gatech.edu

ABSTRACT

As users interact with an increasing array of personal computing devices, maintaining data consistency across those devices becomes more difficult. Typical solutions assume access to centralized servers, continual connectivity, or unbounded storage and CPU capacity. In practice, users' devices vary widely in capabilities, use intermittent or sparsely-connected networks and incur (asymmetric) transfer costs. We present a multi-agent system built upon a decision-theoretic approach to constructing and executing multiple plans to achieve consistency in a peer-to-peer, partially observable, non-deterministic environment. We analyze the performance in comparison to a standard epidemic replication algorithm.

Categories & Subject Descriptors

I.2.8 Plan execution, formation, and generation

Keywords

multiagent, decision theory, device networks

1. INTRODUCTION AND MOTIVATION

We consider the problem of transparently ensuring that a user's data remains consistent across all her devices. We construct a *Personal Information Environment* (PIE), a virtual data space not tied to any one physical device. This provides the user the illusion that every file is available on every device all the time.

Users could try to use a single storage unit (such as a USB key or iPod) to store their files, mounting it on any device they are currently using. There are several problems with this approach. First, it requires the user to carry the storage unit and plug it in to every device she uses. If the user forgets to plug it in she will be unable to access her files. Second, these devices often have less than 2GB of storage, so only a subset of the user's files may fit.

Alternatively the user could synchronize with a central file server; however, many devices have only intermittent connectivity. Even users with broadband connections can be behind a firewall, making it difficult for the home (or work) device to be a file server.

When we move toward a decentralized automated solution, several difficult problems arise. We must know the files the user will

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...\$5.00.

need on any particular device. The topology of the device network is not known and may change as devices come and go. Further, devices may not have a fixed IP address or may reside behind firewalls. In this situation, the role of limited capacity devices such as PDAs and cell phones frequently on a user's person becomes that of a crucial vessel to transfer data. This turns out to be a problem of propagating information in a resource-constrained, distributed and peer-to-peer non-stationary network that is only locally observable.

Described this way, the problem seems hopeless; however, we have several advantages. First, users are often at least locally predictable. They cannot be in many places at once, allowing time to move data between devices. Second, a limited number of files are touched in a given day and the number of devices is quite small so the problem admits practical computational tractability.

In our approach each device identifies important data it might possess, other devices that need that data, and constructs *plans* to deliver that data (using any intermediaries). Each device adapts and updates the plans of other devices as they become aware of them.

2. RELATED WORK

One of the best-known systems for managing information on mobile devices is Coda [5]. Coda uses manually constructed or heuristically derived “hoard profiles” that determine which files should be replicated to which devices. Coda does not support multi-hop transfers over resource-constrained devices.

There has also been a large amount of other work on update consistency in disconnected or weakly connected networks. Examples include FICUS [8] and Thor [7] (amongst others). The goal of most of these systems is to manage updates, propagating them to a primary copy or to various clients. Our goal is somewhat different: to place cached copies of files where a user will need them as proactively as possible. Once the replicas are placed, standard techniques can be used or extended to manage consistency.

In the *joint intentions* framework [6], the implied contract between agents requires that each agent, upon obtaining knowledge of another agent's goal, assumes the goal as its own until it is achieved or it becomes irrelevant. Goal existence information and goal status information is communicated via peer-to-peer updates.

We do not explicitly represent goals like some multi-agent systems, but our agents operate in unison to achieve a globally useful—if not globally consistent—state with an implied contract to make their best effort to complete all multi-plans they receive. In our system, goals are represented as “multi-plans”. These multi-plans are sent from agent to agent along with whatever data they were constructed for. A multi-plan can become invalidated when a device receives information that there is a newer version of the data

associated with that multi-plan. This is identified during a *merge* process that can trigger replanning.

Because the devices that comprise a PIE have varied characteristics, it is important for agents to have some notion of the ability of their peers. [1] presents a set of criteria metrics that indicate characteristics of an agent’s abilities and intentions in heterogeneous, complex planning architectures. These metrics consist are *commitment*, *responsibility*, *authority*, and *independence*. Agents in our system are all committed to joint goals; however, different types of devices in a PIE have varying levels of responsibility, authority, and independence. We describe devices using profiles that allow more powerful devices to aid in the operations for less powerful devices.

Our approach exploits the fact that the state space is limited by the number of devices in a typical PIE which allows for a brute force greedy approach. We improve over the limitations suggested by [2] of early decision-theoretic planning systems such as [9]) by defining innovative utility functions to encode more complex information about the environment.

3. DEFINITIONS

Our goal is to place data (or files) on different devices. A *placement* is an ordered list of *unique* device IDs on which a file is to be located. For example, $l(1, 2, 3, 4)$ is the placement corresponding to a file being located on devices 1, 2, 3, and 4. A *plan*, P_i , is an ordered list of placements: $\{l_{i,1}, l_{i,2}, \dots, l_{i,k}\}$ where $l_{i,j}$ denotes the j -th placement of plan P_i . We constrain each plan so that each of its placements is a prefix of the placement that follows. A plan can be abbreviated by its final placement (e.g. $P_i \equiv l_{i,k}$). Each placement has an associated *utility*; that is, there is some value to having a particular file on a particular device or set of devices. We define the utility of a plan from those placement utilities: $U(P_i, j)$ is the utility of the plan P_i at position j ($U(P_i, j) = U(l(i_1, i_2, \dots, i_j))$)

Let $p_{i,j}$ and $c_{i,j}$ be the probability and cost of transferring some bits successfully from device i to device j respectively. Then we can calculate the *expected utility* of a plan in a straightforward way. For notational convenience, let $L(P_i, j) = \prod_{x=1}^{j-1} p_{i_x, i_{x+1}}$ be the likelihood of data reaching device i_j from device i_1 following plan P_i ; let $C(P_i, j) = \sum_{x=1}^{j-1} c_{i_x, i_{x+1}}$ be the cost associated with data reaching device i_j from device i_1 following plan P_i ; let $NL(P_i, j)$ be the net loss¹ of the plan P_i at its placement $l_{i,j}$; and let $NV(P_i, j) = U(P_i, j-1) - C(P_i, j-1) - NL(P_i, j)$ be the net value associated with data reaching device i_{j-1} from device i_1 but not reaching device i_j following plan P_i . The expected utility of a plan is:

$$\begin{aligned} EU(P_k) &\equiv EU(l(k_1, k_2, \dots, k_n)) \\ &= (1 - p_{k_1, k_2}) * -NL(P_k, 2) & (1) \\ &+ \sum_{i=3}^n L(P_k, i-1)(1 - p_{k_{i-1}, k_i}) * NV(P_k, i) & (2) \\ &+ L(P_k, n) * (U(P_k, n) - C(P_k, n)) & (3) \end{aligned}$$

Expression 1 is the cost of failure of the entire plan scaled by the probability of it failing during the first step. Expression 3 offsets the utility of the success of the plan with the cost of its execution and scales it by the probability of success. Expression 2 is a similar computation to Expression 3 but it considers the effect of the plan failing at each intermediate step.

¹In our experiments, it is the utility not realized by failing to complete the plan: $NL(P_i, j) = U(\{l_{i,j}, \dots, l_{i,n}\}) - U(\{l_{i,1}, \dots, l_{i,j-1}\})$, where n is the number of placements in a plan.

A *multi-plan* is a (ordered) set of plans: $M_i = \{P_{i,1}, \dots, P_{i,n}\}$. Making an assumption about independence, we define the expected utility of a multi-plan to be the sum of the expected utilities of the plans it contains: $EU(M_i) = \sum_{P_k \in M_i} EU(P_k)$.

We seek a locally optimal (in expected utility) set of multi-plans to approximate a globally optimal set. Because multi-plans are constructed in a partially observable environment, it would be infeasible to consider all plan interaction. Assuming independence allows concise and computationally tractable plan construction.

We are in the process of integrating multi-plans into the Accord [3] middleware system we are developing which will enable us to derive values and costs from actual user data. Specifically, the utility of a placement for a file is based on the “affinity” of that file for each of the devices in the placement, typically implemented as a linear combination of reads and writes for that file. Transfer probability is the empirical probability of the success of transferring bits. Costs are related to the actual bandwidth consumption during a transfer and the amount of a device’s resources that would be consumed receiving, storing, and/or sending a file. Regardless of the specific representation of probabilities, utilities, and costs, our algorithm will select a multi-plan to maximize expected utility.

While each placement must be considered during multi-plan construction, possible placements change very slowly and are enumerated before any connection. Similarly, utility calculation can occur offline and incrementally. Since the values of the costs and utilities are based on information gathered from other devices, some recalculation must occur in response to new information.

Information obtained during a connection about the location of a file in the system can have two effects. First, it can indicate that the file has arrived at a destination for which there is a pending multi-plan, invalidating the multi-plan. Second, it can indicate that a device previously believed to have a file no longer does, triggering the creation of a new multi-plan. Fortunately, the number of devices tends to be small, so new multi-plans are constructed efficiently. Additionally, we allow for short circuit evaluation of multi-plans.

Each agent in the system uses the following strategy which can be contracted to a peer device if its processing power is limited: Identify every file it contains that it believes is the most recent version and that is out of sync with other devices.² For each such file, construct a multi-plan to transfer the file to any device needing it. When a connection is established with another device, exchange and synchronize file location information and all multi-plans (re-planning/merging if necessary). The underlying data is then transferred in decreasing order of associated multi-plan utility.

This ordering allows the agent to make a decision about which files are most important to move to a connected device. As we shall see below, the most benefit is realized when either connectivity between devices or the capacity of any crucial device is limited. When device capacity is limited, a device must decide which files and associated multi-plans to accept or reject during transfers. In our case, a limited capacity device always accepts newer versions of files it already contains, but when faced with a choice, accepts new files only if the multi-plans associated with the new file have higher utility than the lowest utility multi-plans for files already on the device. We later refer to this as the *smart eject/reject policy*.

4. BOOTSTRAPPING

We ran bootstrap simulations using five devices with 200 and 400 files in total. Initially, files were distributed uniformly across the devices. Connections between devices intermittent not available and were randomly terminated after 3 to 10 seconds. We ran

²The Accord middleware provides this information.

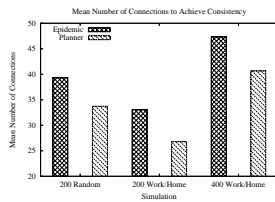


Figure 1: Performance during bootstrapping.

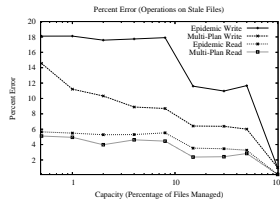


Figure 2: Performance as a function of “go between” capacity.

three sets of trials to determine performance in comparison to the epidemic replication algorithm [4], a popular mechanism for distributing data. In the epidemic model, each device simply transfers any files it has that are missing from any other device it comes in contact with. For each set of trials, the simulations were started from the same initial state and were considered complete when bootstrapping completed. For our experiments, one set of 15 trials was run with 200 files and a randomly generated connection scenario. The second set of 15 trials was run using 200 files and connection probabilities that encode a typical five device scenario: there are two machines in the work place behind a firewall, two machines at home behind a firewall, and one device that is carried between work and home that is the only means of connection between the two sets of machines. We refer to this as the “bow tie” scenario. The last set of 15 trials was run with 400 files and the bow tie connection scenario.

Figure 1 shows the results. There is a clear advantage to using our method in all cases. The multi-plans completed bootstrapping with an average 14.25% fewer connections for the randomly generated connection scenario, in an average of 14.08% fewer connections with 400 files, and an average of 18.95% fewer connections with 200 files and the bow tie scenario.

5. PERFORMANCE UNDER LOAD

Our simulator allows us to specify reads and writes on specific files over different periods of time. To capture meaningful *error*, we measured *stale* accesses: the number of reads and writes on files that are not the most recently modified version across all devices. In each scenario, a user travels between home and work in the bow tie. As before, the go-between is the only way to transfer data between home and work. Here, scenarios differ in the connectivity and storage characteristics of the go-between: a Bluetooth phone (low capacity, low connectivity); a BlueTooth PDA (low capacity, high connectivity when docked); and a laptop (high capacity, mixed connectivity depending on connection mode). All devices are capable of executing plans and using the smart eject/reject policy. When the epidemic is used, there is no utility data available so the limited capacity device randomly ejects files to accept new incoming files.

Scenario data was gathered over 20 trials using 1000 files. Only 450 of the files were eligible for reads and writes by the user. The capacity of the go-between device was varied from 0.05% to 100% of the number of files in the simulation. For simulations with capacity under 8%, the go-between was modeled as a cell phone; for

simulations between 8% and 50%, as a PDA; for 100% capacity, as a laptop. For each test, we simulated eight days worth of user activity with a workload in which the working set of files changed after the first four days.

With the exception of the 100% capacity simulations, the multi-plans (in most cases significantly) outperformed the epidemic algorithm. With severely limited capacity (0.05%), the multi-plans did not reduce the error by as large a margin as with other capacities but still provided some benefit. In the best case (with the go-between cell phone having 8% capacity), the error rate on write operations was less than half of the error rate for the epidemic model.

6. CONCLUSION AND FUTURE WORK

We have described the problem of managing data across an array of resource-constrained devices. Because it is difficult for users to deal with this problem directly, we have argued for building a distributed support system to allow multiple agents to act on the user’s behalf. We have presented motivation for the use of decision-theoretic multi-plans for file consistency management in a distributed peer-to-peer device network. We have shown that there is a distinct advantage to intelligently ordering file transfers in the presence of constrained device networks over current algorithms for distributed data replication.

Acknowledgments

We wish to thank Thomas M. Parks and Christopher Nevison for their donation of CPU time to gather data.

7. REFERENCES

- [1] K. Barber and D. Han. Multi-agent planning under dynamic adaptive autonomy. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 1998.
- [2] J. Blythe. Decision theoretic planning. *The AI Magazine*, 20(2):37–54, 1999.
- [3] B. F. Cooper, C. L. Isbell, J. S. Pierce, D. L. Roberts, and S. Bhat. Accord: Middleware support for contextual, ubiquitous data management on user devices. Technical Report GIT-CC-06-06, College of Computing, Georgia Institute of Technology, 2006.
- [4] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, August 1987.
- [5] J. J. Kistler and M. Satyanarayanan. Disconnected operation in the Coda file system. *ACM Transactions on Computer Systems*, 10(1):3–25, Feb. 1992.
- [6] H. J. Levesque, P. R. Cohen, and J. H. T. Nunes. On acting together. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 94–99, 1990.
- [7] B. Liskov, P. Johnson, R. Gruber, and L. Shriram. A highly available object repository for use in a heterogeneous distributed system. In *Proceedings of the 4th International Workshop on Persistent Objects*, pages 255–266, 1990.
- [8] P. L. Reiher, J. S. Heidemann, D. Ratner, G. Skinner, and G. J. Popek. Resolving file conflicts in the ficus file system. In *USENIX Summer*, pages 183–195, 1994.
- [9] M. Williamson and S. Hanks. Optimal planning with a goal-directed utility model. In K. J. Hammond, editor, *Proceedings of the Second International Conference on AI Planning Systems*, pages 176–181. American Association for Artificial Intelligence, 1994.