

CSE 552 HW2

Jeffrey Hightower

March 11, 2002

1 Predicting the Future

Taking the web as the model of the most successful distributed system and projecting out the trends of mobility and consumer computing, I see two major trends and one potential revolution that could shape the future of distributed computing. Each has implications for fault tolerance, authentication, security, and consensus.

- Very small devices.
- Rapid service composition and deployment.
- Data-Centric Networking and Mobile Code

A possible architecture for future distributed computing is shown in Figure 1. A set of tightly administered servers replicated across the wide area provide a powerful core. Many of the canonical solutions to consensus and fault tolerance will still be very relevant in the core. Although these servers may be maintained by a single administrative entity, the software services running in the core is provided from a variety of independent sources. A set of clients interact with the core but may also interact with each other in a limited manner.

2 Small Devices with Multiple Communication Capabilities

Heterogeneous, small, and mobile devices are becoming pervasive. Cell phones, PDAs, and other types of “personal servers” have limited but ever increasing computation power and storage capacity and are becoming important participants in certain distributed operations.

Traditional clients will interact directly with the core just like the way current web clients request data from replicated sources such as Google’s cache or the Akamai servers. However, clients increasingly have the ability to interact with each other in a much more limited manner through alternative communication channels. For example, PDAs can communicate over direct short-range wireless links as opposed to connecting through a cellular infrastructure.

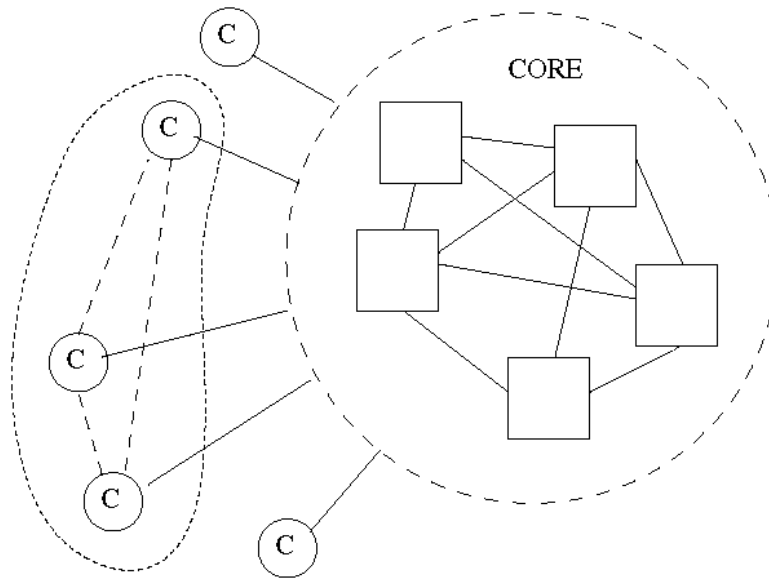


Figure 1: An emerging architecture for distributed computing.

This new connectivity creates an interesting distributed systems problem because it introduces new challenges for consensus and fault tolerance. In this world, the most common “failure” may be a lack of wide area connectivity requiring groups of clients to self-organize into a temporary distributed system to continue operating and then later resynchronizing as a group with the core. For example, a group of users’ PDAs cut off from a core file-sharing service can form limited peer-to-peer wireless links instead to continue operation. Another example is a laptop cut off from its email server yet still able to send mail to another user if the recipients laptop can be reached by ad hoc 802.11b.

This is somewhat different from existing approaches to providing availability through lazy replication. In traditional lazy replication schemes, each client simply interacts with any available server to perform updates. This server is then eventually resynchronized without the client’s involvement when connectivity is reestablished. Enabling this sort of alternate communication channel failure mode, however, requires the ability to dynamically alter the consensus protocols to meet the severely limited resources of the small devices and then to provide a mechanism to resynchronize groups of devices with the core when connectivity is reestablished. The obvious solution of electing one of the clients to act as a temporary server is probably not feasible due to resource constraints. The protocol may be able to take advantage of the fact that wireless communication is inherently multicast to reduce the number of messages that must be exchanged. Authentication and security do not seem significantly more difficult in this model as clients can still verify authenticity of messages and secure their

communications as they do now.

3 Service Composition and Deployment

Turning our attention to the core of the architecture, new wide area distributed services, most of them targeted to small communities of users, will need to be rapidly deployed by independent providers. Moreover, we should be able to assemble each distributed service based on a high-level description of the properties desired. For example, Alice's replicated storage system needs provide one-copy equivalence and she will trade availability for consistency, whereas Bob's service (possibly running on some of the same hardware) needs the inverse.

The Virtual Machine Monitor (VMM) work here at UW provides a hint as to what will be needed at the base level in the future for deploying such services. VMMs take advantage of the large amounts of unused cycles on a single machine and asynchrony of execution requests to individual programs to enable large numbers of untrusted programs to execute on a single machine. To deploy independent *distributed* services means pushing to the next level to create a Virtual Cluster Monitor (VCM). A VCM virtualizes a distributed cluster of machines as if each service were running on its own securely isolated cluster. Each virtual cluster provides the proper set of fundamental distribution operations such as casually, totally, and atomic virtual multicast. Virtual group membership need not correspond to actual group membership.

The power of a VCM is not in providing a clustering abstraction. Indeed, forcing developers to use distributed operations such ordered multicast and other ISIS-like interactions could significantly complicated their lives. The leverage gained with a VCM is the potential to specify and control distributed services using higher-level constructs while allowing many different services to securely coexist. Practically speaking, these higher level services could then be provided as commodities.

In a hypothetical implementation, suppose each service has a contract. The contract places burdens on both the VCM system and the service itself. The contract specifies at a high level what the function of the service is, the replication and consistency requirements, and quotas as to the amount of resources the service can use (e.g. the number of virtual nodes, the resiliency to fail-stop or Byzantine failure, or more mundane quantities such as the amount of network traffic or storage). Much like constructing a programming language to abstract away assembly instructions, this contract fully specifies the service and allows it to execute in a way that is secure from interference of other services and possessing exactly the qualities needed.

4 Data-Centric Networking and Mobile Code

Data-centric networking (DCN) and mobile code is a potential revolution that could alter the way we approach some distributed systems problems. DCN takes

the application specific conflict resolution ideas from Bayou and similar systems and pushes it to the limit. Whereas Bayou-like systems use application specific handlers only to resolve inconsistent updates, a data-centric approach uses them continually for normal operation. Data items injected into the network are self-describing, may contain mobile code, and can automatically marshal resources as needed.

Mobile code fundamentally changes the assumptions about distributed systems. Whereas most systems we have seen assume parallel deterministic state machines, mobile code breaks this assumption and allows some nodes to perform a different set of operations based on application specific tasks and may introduce some nondeterminism. I believe DCN's value for consensus and fault tolerance is uncertain because we do not (yet) have the proper abstractions to reason about these issues.