A Comparative Evaluation of Mobile Computing Systems

Damla Turgut, Nevin Aydin and Ramez Elmasri
Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX 76019, USA
Email:{turgut, aydin, elmasri}@cse.uta.edu

ABSTRACT

As mobile computing systems and applications become more common, we see many differences and many similarities among various applications. In this paper, we study three existing mobile computing systems: Bayou, Odyssey, Rover and then propose a methodology for describing mobile systems. We illustrate how current systems fit within our taxonomy and postulate new systems and describe their characteristics. We describe how mobile applications differ from traditional distributed systems and identify common features of mobile computing systems: mobile-awareness, network flexibility, data integrity, update semantics, and consistency guarantees. We compare and contrast these systems based on these identified common features. We also evaluate them with respect to two detailed mobile scenarios. Calendar management scenario involves mobile and fixed clients, which update textual data as separate workgroups. Multimedia presentation scenario involves a high-resolution presentation, where adjustments are made over a low-bandwidth connection. Bayou handles the first scenario well; Odyssey is more suited to the second scenario. Rover, in contrast, provides infrastructure support for both scenarios but require additional object-oriented programming.

I. INTRODUCTION

The growth of the internet has led to the emergence of mobile computing. Many applications that had been unforeseeable a few months ago have become reasonable or will be in the near future. Multimedia in mobile applications has increased the size and complexity of data that such applications need to process.

The researchers have designed information systems to address the needs of mobile computing users. We examine three such systems: Bayou [3], Odyssey [2], and Rover [5]. Previous research [4] has compared Bayou, Rover, and Coda (the predecessor of Odyssey). That comparison focused on the ability of each system to handle weakly connected hosts. In this paper, however, we compare the systems based on criteria common to any mobile computing environment.

First, we describe the differences between mobile computing and more traditional distributing systems. Section 2 introduces the common requirements of mobile computing systems. Section 3 gives an overview of the three mobile computing systems. In section 4, we qualitatively rank the ability of the mobile computing systems to meet the requirements given in section 2. Section 5 discusses two mobile computing scenarios in detail and describes how the three systems support the scenarios. We conclude by contrasting the abilities of the three systems considered. For the remainder of the introduction, we outline general application areas for mobile computing.

A common office activity is reserving conference rooms for presentations and meetings, known as the group calendar application [3]. We would like managers to be able to reserve rooms even while away on a business trip. Traditional calendar programs assume that everyone updating the (shared) calendar has reliable network connections and thus the software cannot handle disconnected mobile hosts.

Another mobile application is email using the POP protocol which allows a server to hold a user mail and allows the user to retrieve the email from any system. With only slight additional coding, a POP client can support mobile computing by pre-fetching all new messages during a brief network connection and queuing the user replies until the user reconnects [3].

Emergency response is also well suited application area. Often emergency response personnel need specific information, such as a copy of an emergency call report for a medical or police emergency. Fixed networks are unavailable for an emergency response team, and often the only wireless communications available are satellite communications with very high latency. Ideally, a mobile system supporting emergency response teams would allow two way communication of both voice and graphical data.

II. COMMON REQUIREMENTS

Much research literature has focused on systems designed to support distributed computing. This section contrasts mobile computing systems with traditional distributed systems and briefly describe the common requirements of the mobile computing systems.

Mobile-awareness: Traditional distributed systems use a variety of data migration algorithms to move data to the places where it is needed. In mobile computing environments, however, a mobile client might change some data and then disconnect from the network. The data migration algorithm must not use the mobile host as sole owner of the data. Mobile-transparent systems modify the server to allow applications to run on mobile host without any modifications. Mobile-aware systems in contrast modify both the server and the mobile host applications.

Network Flexibility: Traditional distributed systems distinguish between client/server architectures and peer-to-peer architectures. In mobile computing, the distinction between clients and servers is not as crucial as the distinction between mobile hosts and fixed hosts. Some mobile computing systems use a fixed client/server architecture and some are more flexible, but all systems must handle network failures more robustly than traditional distributed systems. Often, a mobile host will be voluntarily or involuntarily disconnected from the network for an extended period of time, and while connected, the mobile host network connection may be less reliable than the network connections for the fixed hosts.

Data Integrity: Mobile systems often differ from distributed systems on how they handle data integrity issues in that a *conflict* means two or more hosts have issued incompatible updates to the same piece of data. Conflict detection and resolution refers to a system's ability to detect conflicts and manage the conflicts either with or without user intervention.

Update Semantics: Update protocols for distributed systems assume a particular host is the manager of a piece of data and the host is known or can quickly be determined. In contrast, in a mobile computing environment, a host may be disconnected from the network for an extended time, and thus cannot communicate with the manager of a piece of data.

Consistency Guarantees: A distributed system with fast and reliable network communication can offer a stronger consistency guarantee than a mobile system. Often, mobile computing systems distinguish between tentative and committed updates. A committed update is final; a tentative update is subject to being rolled-back if it conflicts with other updates.

III. OVERVIEW OF THE SYSTEMS

In this section, we briefly describe three mobile computing systems and consider how they address each of the requirements we identified above.

A. Bayou

Bayou, developed by Xerox PARC [3], uses a flexible client/server architecture in which any host can become a server. Each server has a complete copy of the database and can handle tentative updates. Only the primary server can commit transactions. Bayou does not directly distinguish between mobile hosts and fixed hosts, but its algorithms can handle low-bandwidth connections, disconnected hosts, and partitioned networks [7]. An application-specific merge procedure identifies and resolves conflicts. For example, requests to reserve two different rooms at 3 o'clock will not be a conflict unless the meeting involves the same participants. Bayou uses a novel anti-entropy algorithm which over time guarantees lazy-release consistency. The anti-entropy algorithm [3] is designed so that it makes progress towards consistency even if a network connection is unexpectedly broken. The anti-entropy protocol can also accommodate an unconnected host through floppy disks or other removable media. Figure 1 shows Bayou's architecture. Clients can connect to any reachable server; servers can be created or retired as needed.

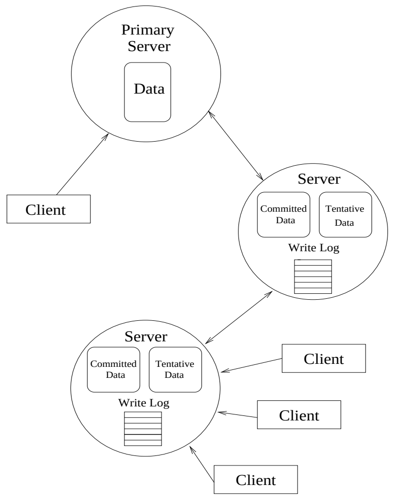

Figure 1: Bayou System Architecture

B. Odyssey

Odyssey, developed at Carnegie Mellon University [2], uses a client/server architecture in which a single server can send data to many different clients. Instead of a file-based server, Odyssey uses a series of wardens, each of which has specific knowledge for specific data. When a client requests a data item, it sends a list of its capabilities, which are forwarded to the appropriate warden. The warden prepares and transmits the data in a format suited to the capabilities of the client. For example, if a fast client with a high-bandwidth connection requests a movie file, the warden can send a full-screen color version of the movie. If another client on a low-bandwidth connection makes a request for the same movie, the warden will send a black and white, 1/4-screen version of the movie. Odyssey does not directly support disconnected hosts. One unique feature of Odyssey is that it can respond to dynamic changes in bandwidth availability. Odyssey stores data only in a single server and thus, the server can easily arbitrate between conflicting updates. Odyssey's update protocol resembles distributing computing rather than mobile computing. Figure 2 shows Odyssey's system architecture (from [1]).

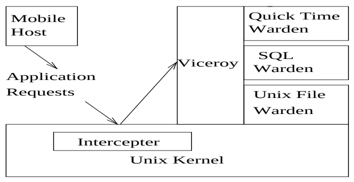

Figure 2: Odyssey System Architecture

C. Rover

Rover, developed at MIT [5, 6], is a client/server architecture which uses Relocatable Dynamic Objects (RDOs). Each object has a host server that manages consistency issues related to that object. Clients can retrieve objects from the server or use remote procedure calls (RPCs) instead. If a client is disconnected from a server, it can queue the remote procedure calls (QRPCs); these will be transmitted to the server when the client reconnects. Rover supports disconnected hosts by allowing RPC commands to be queued and by the use of split-phase transactions, meaning that the client can disconnect between initiating the transaction and receiving the results of the transaction. Rover does not directly detect conflicts; a server instead will use a custom conflict detector based on the application, data types, and operations involved. Once a server commits a transaction, it sends a response to the client allowing the client to maintain data consistent with the server. Figure 3 shows Rover's system architecture.

IV. MOBILE COMPUTING SYSTEMS

In this section, we describe the characteristics common to Bayou, Odyssey and Rover: transparency, data replication, consistency guarantees, network flexibility and network robustness.

Transparency: Transparency refers to the ability of the system to interface with ordinary network applications. Existing mobile computing systems require

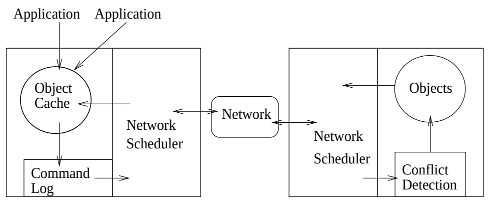

Figure 3: Rover System Architecture

changes to the server software. Mobile-transparent systems use existing client software, but mobile-aware systems require changes to the client software. Another transparency issue is how involved the user must be in data management. A system with no transparency would require extensive modifications to both server and client software as well as significant user intervention. Perfectly transparent system would require no changes to either client or server.

Data Replication: Data replication measures not just the number of copies of the data but also the degree to which the system keeps track of replicated data. A system with no data replication stores data only at a single central server. A system with medium amount of data replication allows data to be stored in multiple locations, but keeps track of which hosts have copies of the replicated data. A system with high data replication allows any host to have any piece of data and may not keep track of which hosts have which piece of data.

Consistency Guarantees: The systems are ranked by the strengths of their consistency guarantees using the following types: inconsistent data, application-specific consistency, lazy-release consistency, weak consistency, pipelined-PRAM, sequential consistency, and strong consistency.

Network Flexibility: Network flexibility of a system indicates whether a system allows servers to be dynamically created or retired. A system with no network flexibility uses a fixed client/server architecture with a single server. An extremely high value for network flexibility indicates a maximally flexible system in which servers can be dynamically created or retired. Intermediate values indicate systems which can support more than one server, but either have a designated primary server or do not support dynamic retirement of a server.

Network Robustness: Network robustness indicates whether a system can maintain its consistency guarantee in the face of unreliable network communication. A system with no robustness assumes the network topology is stable and the communication links remain functioning. A system with very high network robustness can continue operation even when the host disconnects and reconnects to the network. Intermediate values indicate a system that is flexible in some areas, but has strong requirements in other areas.

A. Rankings of The Systems

In this section, we rank the systems discussed in section 3 based on the criteria described above. Table 1, in the conclusions, summarizes our evaluation.

A.1 Bayou

Bayou has very low transparency. The implementer not only must the use custom software for both the client and the server, but also must write application-specific merge procedures. Bayou has very high data replication; any host can acquire any copies of data items without informing the global system. Over time, Bayou will achieve lazy-release consistency. Bayou's *merge-proc* allows the programmer to define an application-specific consistency guarantee. Bayou has high network flexibility in that any host can become a server and any client can retrieve or update data from any server. Bayou's only area of inflexibility lies in having a primary server which commits changes. Bayou has extremely high network robustness and accommodates changing topologies and unreliable communications. Even if the network link fails in the middle of an anti-entropy update message, Bayou still makes progress toward data consistency. Bayou's algorithm is flexible enough to allow the update messages to be transported on removable media such as floppy disks. The only drawback to unreliable or slow communication is that it lengthens the time required to achieve lazy-release consistency.

A.2 Odyssey

Odyssey has medium amounts of transparency. It requires extensive changes to server software and also requires that the data to be stored (or at least converted) in several forms. However, it requires only minimal changes to client software. Odyssey's operations are completely transparent to the end user. Odyssey has no data replication; because data is stored in a central server, Odyssey can offer sequential consistency. Odyssey has very low network flexibility; it uses a traditional client/server model. Odyssey's sole source of flexibility is allowing unknown entities to use internet connections and become clients. Odyssey also has low network robustness; if the server or the server's network link goes down, then Odyssey will be unable to function. Odyssey does, however, take into account the quality of the client's network connection.

A.3 Rover

Rover Toolkit requires additional programming to support particular mobile computing applications. Rover can either support mobile-aware or mobile-transparent systems. In [5, 6], Rover authors describe how Rover can be used with an unmodified Netscape client to support mobile operations. Thus, transparency in Rover can vary from none to high. Rover programmers can choose between using Remote Procedure Calls (RPCs) and using Relocatable Dynamic Objects (RDOs). Rover supports moderate amount of data replication although programmers can use a fixed client/server architecture. Rover directly supports primary-copy, tentative-update, optimistic consistency. If the user wishes to have a stronger consistency guarantee, the user must add additional objects to Rover to support the consistency guarantee. Thus, we rank Rover having an application-specific consistency guarantee. In Rover, each object has a home server. The objects are transported to clients as needed and different applications can use the objects on a Rover client. Thus, Rover has very low network flexibility. Once the object is initially retrieved, Rover allows for disconnected operations. Thus, Rover has high network robustness.

V. MOBILE COMPUTING SCENARIOS

In this section, we develop two typical scenarios and then describe how each of the three systems can address specific needs of mobile computing.

Calendar Management: Consider a calendar management program used in a conference. Originally, the organizers are connected to a high-bandwidth corporate network in Dallas. However, the conference is in San Francisco. The organizers will be making changes to the conference schedule on the plane while traveling to San Francisco. While the organizers are en route, the conference center, in conjunction with organizers who have remained in Dallas, may need to adjust the schedule as well. Ideally, when the conference organizers land in San Francisco, they will be able to quickly merge their changes with the changes made in Dallas. A mobile system supporting this application should resolve conflicting updates and allow a workgroup to make changes even while they are disconnected from the main network. This application does not require transparency and if we assume that the mobile hosts have reasonably sized disk drives then data replication is unimportant. This application requires a network robustness and a flexible network architecture.

Multimedia Presentation: Consider a corporate executive working in Santa Cruz, California. The executive is preparing a multimedia presentation for display to some investors in New York. Both New York and Santa Cruz have high-capacity networks; however, while on the plane, the only network connection available to the executive is an expensive, low-speed modem connection. A mobile system supporting this application should allow our executive to modify his presentation over a low-bandwidth connection on a plane while still being visually impressive when presented over a high-bandwidth connection in New York. Since the multimedia application may exceed the capacity of the mobile host, it is essential that a system supporting this application have medium data replication. The system does not need to be able to handle flexible network architecture, but it needs to efficiently handle low-bandwidth network connections.

A. Bayou Support for the Scenarios

Bayou can directly support our calendar management scenario. Bayou allows any system to become a server and allows disconnected workgroups to make changes to the content even if the primary server is unreachable. Our conference organizers should allow one of their mobile computers become a server while still connected to the corporate network. The organizers can make changes to the schedule independent of any changes made to the copy on the primary server. If desired, the workgroup can split and the resulting two mobile servers can achieve local consistency using the anti-entropy algorithm even if the primary server is unreachable. When the mobile computers are again able to reach to the corporate network, the mobile server(s) can run the anti-entropy protocol with the primary server to achieve global consistency. An application-specific merge procedure provides a way to automatically detect false conflicts. Figure 4 shows this scenario.

Bayou cannot handle the multimedia presentation scenario as well as the first scenario. For the executive to be able to make changes, the executive's computer must become a server, meaning the computer must hold a copy of the entire presentation which could easily exceed the disk space available on the mobile host. Even if the executive has enough storage space for the entire presentation, update messages sent while on the flight would likely take an inordinately long time to transmit over the low-bandwidth modem connection. Bayou does allow updates made while on the flight to be queued until a high-bandwidth connection is available, but Bayou does not have a method of reducing the storage or bandwidth costs of read requests.

B. Odyssey Support for the Scenarios

Odyssey provides only limited support for the first scenario. Odyssey allows only a single server. Our conference organizers must decide whether the server will be on a mobile host on their plane or the fixed host connected to the corporate network. While the conference organizers are disconnected from the corporate network, the network has been partitioned and some clients will be unable to make changes to the data on the server.

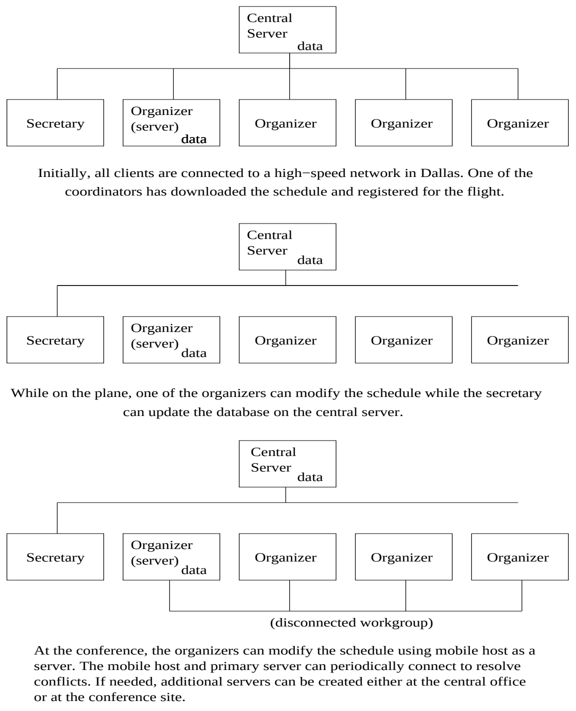
Central Server data
Secretary
Organizer (server) data
Organizer
Organizer
Organizer
Initially, all clients are connected to a high−speed network in Dallas. One of the coordinators has downloaded the schedule and registered for the flight.
Central Server data
Secretary
Organizer (server) data
Organizer
Organizer
Organizer
While on the plane, one of the organizers can modify the schedule while the secretary can update the database on the central server.
Central Server data
Secretary
Organizer (server) data
Organizer
Organizer
Organizer
(disconnected workgroup)
At the conference, the organizers can modify the schedule using mobile host as a server. The mobile host and primary server can periodically connect to resolve conflicts. If needed, additional servers can be created either at the central office or at the conference site.

Figure 4: Bayou Support for Mobile Calendar Application

Odyssey does however, provides extensive support for the second scenario. Odyssey allows several representations of the same data optimized for different bandwidths. Thus, the executive can re-sequence a low-resolution black and white movie while on the plane; when he returns to a high-bandwidth network, Odyssey will reflect the changes in the high-resolution, high-bandwidth color presentation. Figure 5 shows this scenario.

C. Rover Support for the Scenarios

Rover partially addresses the requirements needed for the first scenario. Our conference organizers can make changes even when disconnected from corporate network. However in Rover, the organizers cannot synchronize their changes locally; instead they must wait until they reconnect to the main network and synchronize their changes with the fixed server. Any conflicts arising from different conference organizers will not be noticed until the organizers reconnect to the corporate network.

Rover also provides some support for the second scenario. Rover allows the executive to store a cache of objects. The corporate executive can freely modify any objects in his/her cache. The changes made

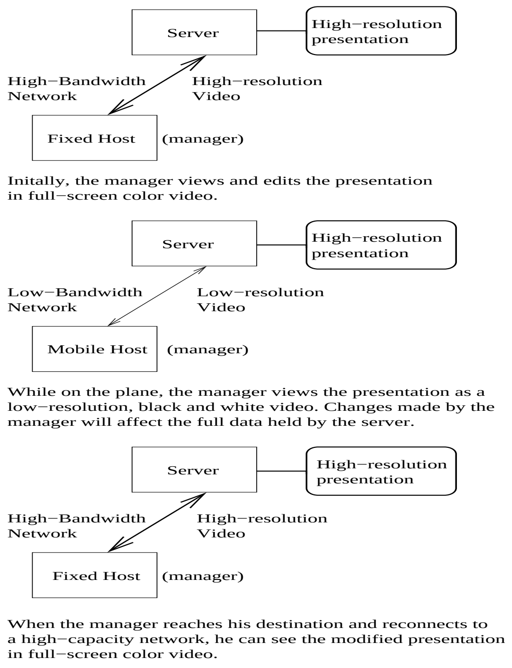
Server
High−resolution presentation
High−Bandwidth Network
High−resolution Video
Fixed Host
(manager)
Initally, the manager views and edits the presentation in full−screen color video.
Server
High−resolution presentation
Low−Bandwidth Network
Low−resolution Video
Mobile Host
(manager)
While on the plane, the manager views the presentation as a low−resolution, black and white video. Changes made by the manager will affect the full data held by the server.
Server
High−resolution presentation
High−Bandwidth Network
High−resolution Video
Fixed Host
(manager)
When the manager reaches his destination and reconnects to a high−capacity network, he can see the modified presentation in full−screen color video.

Figure 5: Odyssey Support for Multimedia Presentation

will be stored in an operations log, which can be sent to the server over a relatively low-bandwidth connection. The executive can also queue changes to objects, which are not in his/her cache; however, to retrieve information from an object, the executive must have a local copy of the object. The programmer of the Rover objects has the opportunity to make tradeoffs among operations performed on the server, operations performed on the client, and data transmitted from the server to the client. When the client changes the local object, an operations log records the transactions made depending on the size of the operations log and the size of the data. The client can either send the complete object to the server or it can send the operations log which the server executes on the master object. Thus, our programmer can design the presentation object to only download a low-bandwidth version of the presentation and yet apply any changes to the high-bandwidth version on the server side. With this additional programming, Rover supports the multimedia scenario.

VI. CONCLUSIONS

In this paper, we have identified and contrasted characteristics common to mobile computing systems: mobile-awareness, network flexibility, data integrity, update semantics, and consistency guarantees. We evaluated Bayou, Odyssey and Rover mobile computing systems according to the support they provided for calendar management and multimedia presentation scenarios. Table 1 shows our evaluation of these systems. Bayou directly supports disconnected workgroups and has flexible server architecture to react to partitioned networks. Bayou's anti-entropy protocol allows servers to progress towards consistency even while on a intermittent or unreliable network connection. Bayou, however, offers only a weak consistency guarantee. Odyssey in contrast requires continuous (but possibly low-bandwidth) connection to a central server, but Odyssey can offer a stronger consistency guarantee. Rover requires additional programming, but can offer to choose between consistent data and disconnected operations.

Table 1: Comparison of Mobile Computing Systems

Characteristics	Bayou	Odyssey	Rover
Transparency	very low	medium	none-high
Data replication	very high	none	medium
Consistency	lazy-release	sequential	varies
Network flexibility	high	very low	very low
Network robustness	very high	low	high

One of the major tradeoffs in mobile computing systems is between offering strong consistency guarantees and robustly handling disconnected workgroups or unreliable network communication. Another issue is how much transparency to support. We expect that as the systems become more flexible, they will support operations that are both transparent and flexible.

References

[1] Noble, B., Price, M., and Satyanarayanan, M. "A Programming Interface for Application-Aware Adaptation in Mobile Computing." Proceedings of the Second USENIX Symposium on Mobile and Location-Independent Computing, Ann Arbor, MI, April 1995.

[2] Noble, B. Mobile Data Access. School of Computer Science, CMU, CMU-CS-98-118, May 1998.

[3] Peterson, K., Spreitzer, M.J., Terry, D.B., Theimer, M.M., and Demers, A.J. "Flexible Update Propagation for Weakly Consistent Replication." Proceedings Sixteenth ACM Symposium on Operating Systems Principles (SOSP), Saint-Malo, France, October 1997.

[4] Pitoura, E. and Samaras, G. Data Management for Mobile Computing, Kluwer Academic Publisher, 1998.

[5] Tauber, J.A. Issues in Building Mobile-Aware Applications with the Rover Toolkit. MS Thesis, MIT, May 1996.

[6] Tauber, J.A. and Kaashoek, M.F. "Mobile Computing with the Rover Toolkit." IEEE Transactions on Computers: Special issue on Computing, 46(3), March 1997.

[7] Terry, D.B., Demers, A.J., Peterson, K., Spreitzer, M.J., Theimer, M.M., and Welch, B. "Session Guarantees for Weakly Consistent Replicated Data." Proceedings of the International Conference on Parallel and Distributed Information Systems (PDIS), Austin, Texas, September 1994, pp. 140-149.