A Group Communication Approach for Mobile Computing*

Kenjiro Cho†

Media Technology Laboratory
Canon, Inc.
Kawasaki, Japan 211

Kenneth P. Birman‡

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

Abstract

This paper describes the design and implementation of a set of tools, called MobileChannel, for use with the Isis system. A simple scheme to support user mobility—switching a control point between replicated servers—provides a uniform mechanism to handle both client migrations and server failures. The hand-off mechanism is simplified by integrating a FIFO channel implementation into the server replication mechanism. Our scheme provides a simple abstraction of migration, practically eliminates hand-off protocols, provides fault-tolerance and is implemented within the existing group communication mechanisms of Isis.

1 Introduction

This paper examines group communication as an infrastructure to support mobility of users. Group communication systems offer primitives in support of *distributed groups of cooperating processes*. Their technologies are based on multicasting and membership service [1, 3, 7, 10, 13]. Though group communication has been studied mainly for fault-tolerance, experience has demonstrated the approach can considerably simplify any distributed program which needs coordination among multiple processes.

Group communication is also intrinsically appealing for handling mobility. In group communication, multicast messages are sent to abstract groups and senders do not need to know about the other members of the group. Dynamic group membership management allows one to dynamically join or leave groups. Hence, one can leave a group, then move to another place, re-join the same group and continue working with the other members. In this sense, group communication already provides location independence and is able to adapt to a dynamically reconfiguring network topology. In addition, reliable ordered multicasting—all group members observe the same set of messages in a guaranteed order—simplifies synchronizing multiple processes or coordinating cooperative actions for handling mobility of users. Moreover, fault-tolerance—a system can continue to work in the presence of failures—is part of the group communication design and is also essential to everyday use of mobile devices.

Prior work related to host mobility has focused on schemes to implement transparent mobile support in the network layer [12, 21]. Some of them address multicasting as an effective way to improve performance and reduce the cost [1, 12, 15]. On the other hand, hand-off schemes and protocols from a higher level layer are presented in [2] by means of multicasting to a group of mobile hosts. However, these approaches require complicated protocols for hand-offs, and fault-tolerance is not often addressed.

An important benefit of group communication resides in its ability to support *consistent replicas*, a feature that can also be useful for handling mobility of users. This paper presents a simple scheme to support user mobility, which is based on an abstraction of replicated servers provided by group communication. In our scheme, hand-off is realized by switching a control point between replicated servers. The hand-off mechanism is simplified by integrating a FIFO channel implementation into the server replication mechanism. Our scheme provides a simple abstraction of migration, practically eliminates hand-off protocols and hand-off time, provides fault-tolerance and is implemented within the existing group communication mechanism.

To demonstrate our approach, we have developed a system called MobileChannel. Our goal is to identify and implement a suitable tool to support building mobile services and to integrate those services into the existing distributed environments based on group communication.

2 Design of the MobileChannel tool

In this section, we present the design of Mobilechannel. The MobileChannel tool provides continuous services to mobile clients and tolerate failures of servers. Services are available as long as one of the servers survives. For geographically-defined wireless cells, physi-

*This work was performed at Cornell University, and was supported under ARPA/ONR grant N00014-92-J-1866, and by a grant from Canon, Inc. The source code of the prototype is available via anonymous ftp at `ftp.cs.cornell.edu:/pub/kjc`.

†was visiting the Dept. of Computer Science at Cornell University during '92–'94. e-mail: `kjc@cis.canon.co.jp`.

‡Professor in the Dept. of Computer Science at Cornell University, and Chief Scientist of Isis Distributed Systems, a division of Stratus Computer, Inc. e-mail: `ken@cs.cornell.edu`.

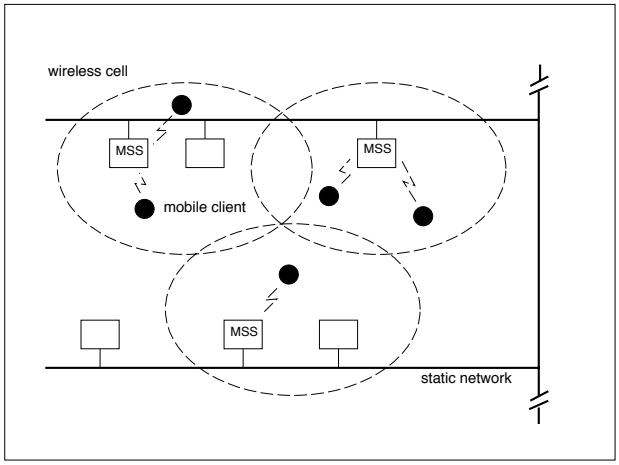

Figure 1: System Model

cal layout should provide fault-tolerance by means of overlapping cells or backup stations so that a client can reach the service as long as one of the reachable servers survives. MobileChannel provides a reliable but cheap communication means for mobile clients. A communication channel looks like a one-to-many channel carrying Isis messages.

2.1 System Model

Figure 1 shows our system model which consists of two distinct sets of computers: static servers and mobile clients. A group of servers provide services to mobile clients. Servers act as a proxy for a mobile client. Mobile clients talk to these servers usually through a wireless link. The wireless network is organized by geographically-defined cells. Mobile clients can cross the cell boundaries maintaining communication connectivity (hand-off). The static network and wireless network are connected by gateway servers, called Mobile Support Stations (MSS). A MSS serves as an access point of mobile clients.

We assume that a mobile client has a less powerful CPU with the constraints on power consumption, and the wireless link has lower bandwidth.

In our system, only servers require the group communication capability. If mobile clients could have the group communication capability, things would be much easier. We, however, assume that the group communication mechanism is too much for most mobile clients with the current technology. Thus, our design shifts the workload to servers and makes clients light-weight. Still, we found that a hand-off mechanism requires the properties well known to the group communication community such as atomicity, ordering and failure handling.

Our group communication platform is the Isis system. Isis is a toolkit for building applications consisting of cooperating processes in a distributed system [4, 5, 6, 7]. Group management and group communication are two basic building blocks provided by Isis.

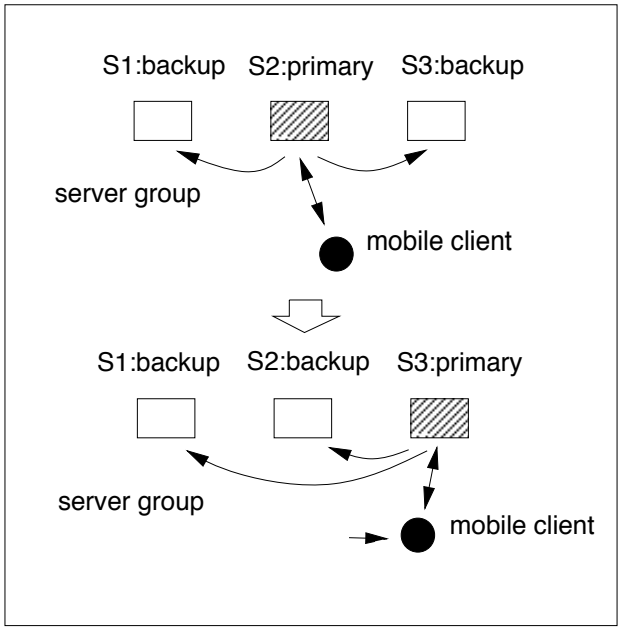

Figure 2: Intentional Primary Switch

2.2 Replication Model

MobileChannel employs a combination of the two replication schemes: the primary-backup approach [9] between a client and servers and the state machine approach [19, 20] among servers.

The primary-backup approach is a centralized mechanism in which a client makes a request by sending a message only to the primary server. If the primary fails then a *failover* occurs and one of the backup servers takes over.

On the other hand, the state machine approach has no centralized control where a client makes a request by multicasting to all servers. According to the request, all servers change their states identically in lock step. A server failure is invisible to clients and does not introduce any response delay.

MobileChannel implements the primary-backup approach on top of the state machine approach. A mobile client talks only to a primary but the primary forwards incoming messages to the backups in order to provide an illusion that the mobile client has a multicast capability. From the programmer's point of view, mobile clients see a single reliable server, and migrations or primary failures are not noticeable. Server-side programming is based on the state machine approach similar to the Isis model and the primary-backup mechanism is hidden in MobileChannel.

2.3 Migration as an Intentional Primary Switch

The most unique feature of MobileChannel is that migrations of mobile clients are handled as an *intentional primary switch.* In the primary-backup approach, a primary switch usually occurs only at a primary failure. This same mechanism, however, can be used for a migration control in order to move the

primary to a nearby backup. Especially in a cellular wireless network, this mechanism allows the system to place the primary within a desired communication range. In short, the primary status can be handed over from one server to another to geographically follow the user. In the primary-backup approach, backups are designed to take over a primary at any time in case of a primary failure; there is no timing constraints. Thus, the design allows one to trigger an intentional primary switch at any time; the only difference is that the primary switch is triggered not by a primary failure but by a migration of a mobile client. Figure 2 shows a primary switch from a server *S2* to another server *S3*. This approach provides a simple uniform abstraction to handle both client migrations and server failures.

2.4 Sliding-Window Replication

Our scheme simplifies the hand-off procedure yet achieves high performance by replicating the server states at the sliding-window level. MobileChannel implements a sliding-window protocol for communication between a mobile client and servers; the sliding-window state on the server side is replicated using the group communication mechanism. A sliding-window protocol is used to implement a FIFO channel and to control message flow in a communication channel. The protocol can make full use of network bandwidth if carefully tuned, which is important especially for wireless communication media because of their relatively low bandwidth.

Replication at the sliding-window level eliminates the problem of handshaking and buffering that are sources of the difficulties of hand-off protocols [2, 15]. In general, hand-off procedures require handshaking among three parties: a new server, an old server and a mobile client. When multiple processes need to synchronize, a handshake of two processes introduces a handshake time, and thus, buffering is necessary for messages from the other process during this period. Our scheme, however, does not need explicit handshaking at all but handshaking is automatically achieved by the underlying structures.

A handshake between servers can be achieved in the Isis *virtual synchrony* model [7] for the following reasons. Virtual synchrony is an illusion that every event happens synchronously in a distributed system. Events such as receiving messages are synchronous in terms of logical time [16], though each process receives a same message at a different physical time. Virtual synchrony can be used to synchronize multiple processes and to release programmers from problems of handshaking and buffering. MobileChannel converts messages from mobile clients, generated outside Isis, to virtually synchronous Isis events by forwarding messages from the primary to all servers. Thus, all servers observe the same set of incoming messages in the same order, which allows servers to act as a replicated state machine. When a hand-off of a mobile client is requested by a multicast, all servers can take appropriate actions without any handshaking. The need for a handshake between servers can thus be eliminated by exploiting virtual synchrony.

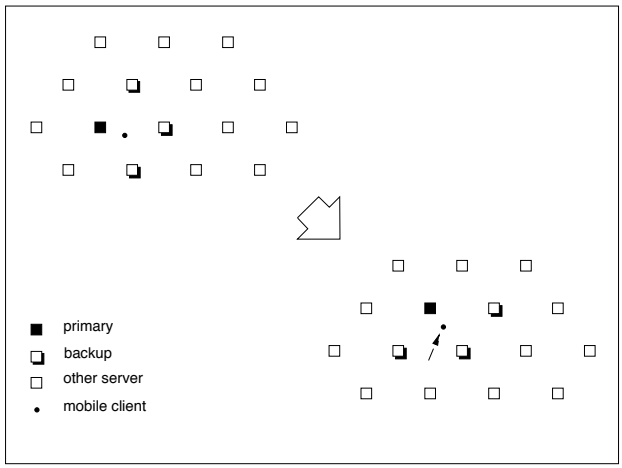

Figure 3: User Subgroup

A handshake between a mobile client and servers is confined to a low-level transport mechanism. Prior hand-off protocols assume a FIFO channel between a server and a client [2, 15]. However, the problem of a FIFO channel is that a hand-off procedure needs to flush and terminate the old connection and then establish a new connection. These two procedures require handshaking between both ends and introduce delays and buffering. Furthermore, a channel takeover of a FIFO channel is difficult to make fault-tolerant—if a server fails, another server should take over the channel. Because a FIFO channel does not provide a way to know the state of the channel or the state of the other party, additional protocols are necessary to restore the consistent states if a failure occurs. A server failure in the middle of a hand-off procedure will further complicate protocols.

However, if a lower level transport mechanism is replicated, such as the packet level, the communication state can be replicated on a per packet basis. If a new primary knows the packets last transferred and acknowledged, the new primary can take over and continue communication without any handshaking. Moreover, protocols of this level already assume that packets may get lost or duplicated so that the protocol is inherently robust to packet loss, duplicates or timing delays, which might be introduced by a takeover. Thus, no special protocol is necessary for a hand-off, and explicit handshaking between a mobile client and servers can be eliminated.

2.5 Scalability

An additional mechanism is necessary to scale the MobileChannel system. Obviously, the flat structure of replicating the user states everywhere does not scale much.

One way to scale up the system is to utilize the dynamic membership and hierarchical group structures. When a service needs to be available to hundreds of mobile users at tens of sites, it is desirable to replicate the service at each site but replicate the state of a client only at a small subset of sites. Such a group

will be structured in the way that the group of the service, which consists of the whole sites, has subgroups; each subgroup corresponding to a mobile user. Each user subgroup consists of neighbor sites of the current location of the user. Dynamic group membership can support user mobility. The idea is that a subgroup moves along a user as a flock. As a user travels, new sites located in the direction of travel join the subgroup. Then, the member sites located behind the user leave the subgroup in order to keep the subgroup size (shown in Figure 3).

When a disconnected user establishes a new connection at a remote site from the previous location, a new subgroup sprouts at the new location, and then, the old subgroup vanishes.

These group management is easily done by keeping track of the user location.

To scale the system up to thousands or millions of sites, further hierarchy would be necessary. Though the current group communication infrastructures do not scale up to this level, research efforts into this question is underway [11].

3 Implementation

MobileChannel is implemented as libraries. The channel mechanism assumes a datagram service underneath and currently uses UDP. The server library requires Isis that runs on most Unix[1] based platforms. The client library runs on the X Window Toolkit or Microsoft Windows as well as the console mode. The client library consumes 70K bytes and the Isis message library consumes another 60K bytes on Microsoft Windows. Note that in MobileChannel, the Isis library available to the client is a very limited one, with only the functionality related to message handling.

The current development environment consists of Sun SPARCstations and subnote PCs. The subnote PCs are equipped with NCR WaveLAN PCMCIA wireless Ethernet cards and connected to the Ethernet via a bridge. However, we do not have a device to locate mobile clients nor a wireless device capable of cell hand-off so that migrations are triggered by an emulator and the wireless clients remain physically in a single cell.

3.1 Mobile Client Management

In our model, a migration is equivalent to a primary switch. If a client has an active channel at a migration, a hand-off takes place. A hand-off is a primary switch with an active channel; the channel is dynamically switched to the new primary. If a client does not have an active channel at a migration, a primary switch takes place without a channel switch. When a disconnected client establishes a new connection at a different site, a primary switch with a new channel takes place. A failover is also handled as a migration. When a primary server fails, one of the backups is elected as the new primary. Depending on the channel state at the failure, the primary switch can be with or without a channel switch.

[1] All trademarks appearing in this paper are recognized registered trademarks of their respective companies.

To manage clients, each server maintains a consistent (identical) state list of mobile clients called *mobile view*. When a client establishes a connection for the first time, a new entry for the client is created and the channel is attached to this entry. When a client disconnects from the system, the channel is closed and detached from the entry but the entry remains in the list to allow the server to act on behalf of the disconnected client. When a migration occurs, the corresponding entry is updated.

Migration can be triggered by requesting a migration to a current primary; the easiest way is multicasting a request to the server group. Upon receiving a migration request, the primary multicasts a primary switch command to the group. When the new primary receives this primary switch command, it takes over the primary status. By virtue of Isis virtual synchrony, all the servers observe the command in the same order with regard to other messages. As a result, a mobile client has a single primary at one "virtual synchrony" time.

Note that a migration is informed by a single multicast message to all servers. This means that a migration looks like an atomic (indivisible) action to the observers. This atomicity makes it easy to write a fault-tolerant program since there is no need to keep intermediate states and restore them for a failure recovery. More details are discussed in Section 3.2.

Also note that a primary switch command creates a *causal chain*[16] from a previous primary to a new primary. This guarantees the order of forwarded messages to the backups at a hand-off.

The migration decision mechanism—which decides which server should take over which client—is outside the system. A decision is made by the location information of a mobile client, usually by detecting the signal level. The decision mechanism varies from system to system. For example, decisions can be made by a current primary or a new primary, or by a client.

3.2 Channel Mechanism

The channel mechanism provides an abstraction of a communication channel which acts like a reliable one-to-many FIFO channel. The mechanism is based on a point-to-point sliding-window protocol. One-to-many connection is realized by replicating the sliding-window state of the primary to the backups.

When a server joins a group, the entire sliding-window state including buffered messages is transferred to this server by means of the Isis state transfer mechanism.

When the primary receives a message from a mobile client, the primary forwards this incoming message to the backups by a totally-ordered multicast. Thus, all servers see the same set of incoming messages in the same order and the input window state of each server is kept identical. According to an incoming message, all servers take identical deterministic actions as a state machine. Hence, the output window state of each server is also kept identical. When the servers send a message to a mobile client, all the servers put the message in their own output buffer but only the primary actually sends out the message. Each channel

entry on the server side has a flag which shows if the server is primary or not. The low-level output routine checks this flag and sends the message only when the flag is set. The client management layer is responsible to maintain at most one primary per client at one time.

Outgoing messages in the output buffer of each server are discarded when the corresponding acknowledgment is forwarded from the primary. Those delayed acknowledgments are piggybacked in incoming messages.

To keep the output windows consistent, it is essential that all servers take identical actions. For a same set of incoming messages, a deterministic action suffices. Two other event sources should be mentioned which may affect deterministic actions. One is a timer. A timer produces events local to the machine and the sliding-window protocol requires timeouts for delayed acknowledgments and retransmission. However, sending these packets does not change the window state, and thus, does not affect the window state consistency. The other source is scheduling; even when a server's action blocks, ordering of actions should not be changed at all servers. Nevertheless, the Isis scheduler employs a strict FIFO order scheduling, and unblocking is triggered only by ordered events. As a result, blocked actions resume in the same order at all servers and the scheduling does not affect the window state consistency either.

The connection management is based on the TCP model that is defined as a finite state machine [18]. The connection establishment and termination mechanisms are the same as TCP. To support a channel switch between servers, a special flag "SWITCH" is defined in a packet header. Figure 4 shows the primary switch mechanism. When a primary receives a migration request, it clears the primary flag and then multicasts a primary switch command to the server group specifying a new primary. Upon receiving this command, one of the backups recognizes itself as the new primary and sets the primary flag in the channel entry then sends a SWITCH segment to the client specifying the address and port number of the new primary. Upon receiving this SWITCH segment, the client updates the address and port number of the peer. The acknowledgment of the SWITCH segment is processed in the same manner as ordinary packets.

Note that the main part of the channel switch procedure is executed by the new primary so that this mechanism works even in case of a failover. In the Isis virtual synchrony model, a failure report is totally ordered so that the backups can locally update their mobile view and select a new primary by some deterministic rule at a failure report. In short, a failure report works instead of the primary switch command in Figure 4. This atomic takeover mechanism makes it possible to deal with cascaded failures.

In a channel switch procedure, the sliding-window states, including sequence numbers and buffered messages, of both ends are not re-initialized and remain intact. Thus, both ends can continue communication from the same state just before the channel switch. The only difference is the primary flag on the previous and current primaries and the peer address on the client. Though the clients have their permanent addresses in the prototype, a slight modification will allow us to switch addresses or frequency bands.

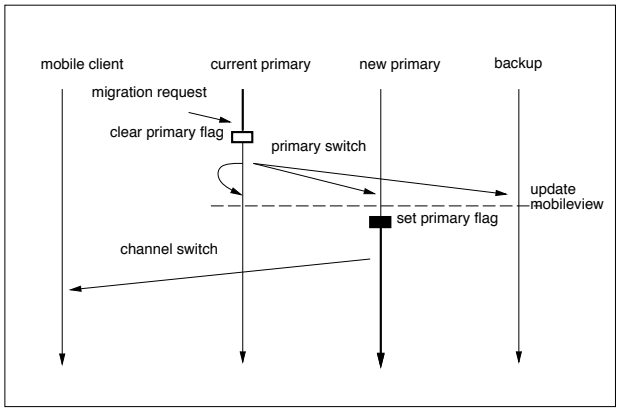

Figure 4: Primary Switch Mechanism

There is a small timing gap between clearing the primary flag at the current primary and setting the flag at the new primary, where no server has a primary flag set. Packet loss or duplicates may occur during this period, but the quick switching mechanism narrows the possibility and errors are suppressed by the sliding-window mechanism.

There is a restriction with regard to the state transfer of the sliding-window: no thread should be waiting to write to the channel at a state transfer. The state transfer sends out the sliding-window state but cannot send the states of blocked threads. To avoid this situation, MobileChannel inhibits joining a group while some thread is blocked. A joining server will be blocked during this period.

3.3 Communication Mechanism

A typical MobileChannel application keeps a consistent application state on servers as a state machine. For those applications in which clients too need to share the application state, the state can be transferred at a connection time. In this sense, connecting to a server in MobileChannel is similar to joining a group in Isis.

Although the channel layer provides a reliable channel, MobileChannel also provides a simplified interface: the RPC style and the diffusion style of communication. In the RPC style, clients interact with servers in a request/reply style. A client blocks until the corresponding reply comes back. In the diffusion style communication, servers multicast messages to the full set of clients. Clients are passive and simply receive messages. These two types of communication styles are common for many applications and most applications will use a mixture of the above two styles. For example, a trading system will use the diffusion style to disseminate stock quotes and the RPC style to make stock transactions.

The two styles behave differently when the output buffer is full. Mobile clients tend to be temporarily

unreachable because of power-saving or obstacles on the wireless link, which often leads to filling up the buffers. In the sliding-window mechanism, a send operation blocks when the output buffer is full until a slot becomes available. However, the diffusion style needs a special data flow control. Since messages are usually generated unrelated to the states of clients, data flow does not stop even when a client is temporarily unreachable. As a result, it is possible that blocked write operations pile up during this period. Though applications can specify the output buffer size according to their needs, dynamic message traffic and unreachable duration are unpredictable. In addition, a blocked send operation prevents a new server from joining the group as discussed in Section 3.2. To avoid this situation, the diffusion mechanism implements the following scheme: when the output buffer is full, a diffusion send just discards the message and marks the message overflow, and when a slot becomes available the system notifies the application about the overflow. It is application's responsibility to take an appropriate action. For those applications in which clients require the consistent state, the state can be reinitialized by transferring the latest state from the primary as done at a new connection. Even when diffusion messages are being discarded, however, RPC replies are never lost. An alternative approach is to implement an infinite backlog as Isis News does [5]. This scheme, however, may cause an unfavorable flood of messages when the communication is restored.

The design of MobileChannel assumes applications for operational clients. MobileChannel does not directly support the delivery guarantee—everyone receives a message even when it is disconnected for a long time. If an application requires such a guarantee, it can use the Isis News facility that implements publish/subscribe schemes by means of virtually infinite backlogs.

We have built demo applications with a graphical user-interface. The "reserve" application implements a train ticket reservation system and uses the diffusion style communication to monitor the reservation status and the RPC style to make reservations. The "grid" application presents consistent distributed grids and can be used to indicate the system performance.

3.4 Consistency Issues

Although migrations and failovers are handled in the same abstraction, their requirements of consistency are different in terms of the message delivery model. The failover mechanism requires a stronger property of delivery atomicity—known as *uniformity* [14, 17]—than the migration mechanism. Hence, MobileChannel provides two different forwarding methods: the non-uniform forwarding and the uniform forwarding.

Isis guarantees only that if a correct process p delivers a message m, then all correct processes eventually deliver m. On the other hand, the uniform delivery holds if a process (whether correct or faulty) delivers a message m, then all correct processes eventually deliver m.

Accordingly, Isis does not guarantee actions done by a failed process and it is possible that even if a primary p accepts and delivers a message m, the failure of p could lead to a situation in which m is not delivered to the other servers [6].

This design allows Isis to take advantage of an asynchronous delivery mechanism to achieve high performance; while the uniform delivery requires to wait for acknowledgments from the others before delivering the message, the non-uniform delivery can be made without blocking.

At a failover, however, the following scenario could happen. A client requests an update and its primary forwards the request but all the forwarded messages are lost. Then the primary does update locally and reports the successful update to the client but fails before no other backups receives the original update request. When the new primary takes over the client, the new primary does not know about the previous successful update.

In practice, the possibility of the non-uniform delivery is quite small. The problem arises only when a primary fails at some very critical condition and timing. Because there is a trade-off between performance and consistency, MobileChannel offers both methods and the user can specify which protocol to use on a per message basis.

The non-uniform forwarding is a direct implementation by the Isis totally ordered multicast primitive (a.k.a. ABCAST). On the other hand, there are several ways to implement the uniform delivery on top of Isis, though it would perform better if implemented inside Isis. MobileChannel implements the two-phase protocol that satisfies the uniformity property. The protocol works in the following way. A primary forwards an incoming message by an ABCAST but marks it undeliverable. At receiving this first phase message, the backups keep the message in their pending queue and reply to the primary. Subsequent messages are also added to the pending queue and not delivered until the precedent two-phase messages are delivered. After sending the first-phase message, the primary waits for replies from the majority of the group, and then, sends the second-phase message by ABCAST indicating that the previous message can be delivered.

If the primary fails in the middle of the two-phase protocol, the backups can safely deliver the pending messages from the failed primary relying on the Isis delivery atomicity and failure report properties. The first-phase ABCAST guarantees the delivery is delayed until the majority of the group have a copy and the message is delivered to all members in the total order. The second-phase ABCAST guarantees the delivery timing with regard to other events.

4 Performance

One might be concerned about performance degradation due to replication, but our experience suggests that replication need not be costly.

Although Isis is a complex system, its communication primitives are well-engineered and optimized to typical communication patterns. The throughput of the current Isis multicast with a small group is comparable to TCP [7, 8]. The cost of joining a small

group is several tens of milliseconds.

Moreover, we assume wireless links are slower than the static network and/or a mobile client has a less powerful CPU than a server. If this is the case, it is reasonable to shift the workload to the servers and the overhead of replication will have a reduced impact on performance.

Our prototype system shows encouraging results; the increase of message traffic due to the sliding-window replication is surprisingly small, though performance ultimately depends on communication structures of applications.

One reason is that MobileChannel supports a "diffusion" style communication that can reduce the read traffic from clients. For typical event-driven applications, most messages are event notifications that can be supported by the diffusion style instead of the RPC style. Since only incoming messages from clients are multicasted, the diffusion style is useful to reduce the traffic for replication.

When compared with other replication models, the overhead of replicating the sliding-window is almost negligible for typical applications. Again, the diffusion style support will lower the read ratio in the traffic. If we anyhow need data replication for fault-tolerance, it is necessary to multicast write operations even without the sliding-window replication. Another increase of the traffic comes from ack packets. The mechanism of delaying and piggybacking acknowledgments in the sliding-window protocol works well under heavy traffic so that ack packets are much fewer than data packets. In addition, ack packets can be forwarded by cheaper causally ordered multicast because the window state is local to the client and ack packets do not interfere with shared resources.

Another possible concern would be the state transfer of the sliding-window at joining a group. The entire window state including the buffered messages is transferred at the state transfer. Nevertheless, the sliding-window normally holds zero or less than a few messages, and Isis tries to pack multiple messages into one for efficiency. As a whole the increase of message traffic is relatively small.

The throughput of the prototype was measured with Sun SPARCstation1s on an ordinarily loaded 10Mbps Ethernet using the non-uniform protocol. The current uniform two-phase protocol is about three times slower than the non-uniform case. The data in parentheses were measured with the subnote PC clients via the 2Mbps wireless network. The wireless case is about twice slower than the static case due to the lower bandwidth, the presence of the bridge, the difference of byte order and the slower CPU. Because the bandwidth of the wireless network is 1/5 of that of the static network, the bridge cannot forward all messages to the wireless network, and thus, frequent packet loss was observed under heavy traffic. The server machines ran Isis v3.1 but hardware multicast was not used.

Table 1 presents the channel throughput in which one-way messages are continuously sent from a server to a client. Table 2 presents performance of the RPC style communication in which a request message from a client is forwarded to replicated servers and then the client waits for a null reply from the primary. The cost of the RPC case is governed by the cost of Isis multicast that grows roughly linearly with the size of the group when used without hardware multicast [6, 8], hence experiments with the recently introduced IP multicast feature of Isis are clearly needed. The throughput includes creation/deletion of Isis style messages that support marshalling/unmarshalling complex data structures.

user data size [bytes]	4	64	1024
throughput	429	408	300
[msgs/sec]	(220)	(204)	(127)

Table 1: Channel Throughput

user data size [bytes]	4	64	1024
1 server	123	122	105
	(56)	(58)	(40)
2 replicated servers	85	95	77
	(54)	(53)	(37)
4 replicated servers	75	73	60
[rpcs/sec]	(45)	(41)	(31)

Table 2: RPC Throughput

The estimated hand-off time is typically less than 20 ms. The hand-off time can be defined as the time between receiving a migration request at an old primary and switching the primary at the client. One Isis ABCAST message for a primary switch command and one channel message for a channel switch segment are required. ABCAST typically costs less than 10 ms for four servers and a channel switch segment typically costs less than 5 ms. If a packet loss occurs during this period, it is handled by the retransmission mechanism of the sliding-window so that the impact is same as an ordinary packet loss.

Although the prototype has limited capabilities, we believe that the overall performance of the prototype is acceptable to most applications and, at least, appealing to some class of applications given that the system provides a fault-tolerant mechanism and the hand-off time is excellent.

5 Conclusion

We presented a mobile support facility, MobileChannel, for the Isis system. MobileChannel is an example how group communication can be used in support of small hand-held devices. MobileChannel employs a unique scheme of a combination of replication and intentional primary switch to support mobility of users. This scheme provides a simple abstraction of migration, practically eliminates hand-off protocols yet provides fault-tolerance. The scheme fits well into the current static network environment and programming model, and is easily built on the current technologies.

Looking into the future, the issues on performance

and scalability will be able to benefit from efforts of the group communication community. Our experience suggests that server replication will be a simple but powerful abstraction in mobile computing for development of robust and sophisticated applications.

Acknowledgments

The authors gratefully thank the members of the Isis group at Cornell University for their helpful comments and suggestions.

References

[1] Arup Acharya and B. R. Badrinath. Delivering Multicast Messages in Networks with Mobile Hosts. *Proceedings of 13th International Conference on Distributed Computing Systems,* IEEE, May 1993, 292-299.

[2] Arup Acharya and B. R. Badrinath. A framework for delivering multicast messages in networks with mobile hosts. *Technical Report,* Department of Computer Science, Rutgers University, 1994.

[3] Yair Amir, Danny Dolev, Shlomo Kramer and Dalia Malki. Transis: A Communication Sub-System for High Availability. *Technical Report,* CS91-13, The Hebrew University of Jerusalem, Israel, November, 1991.

[4] Kenneth P. Birman. Replication and fault tolerance in the Isis system. *Proceedings of the Tenth Symposium on Operating Systems Principles,* Orcas Island, WA, 1985, 79-86.

[5] Kenneth P. Birman, Thomas Joseph and Frank Schmuck. *Isis—A Distributed Programming Environment, Version 2.1–User's Guide and Reference Manual,* Department of Computer Science, Cornell University, July, 1987.

[6] Kenneth P. Birman, Andre Schiper and Pat Stephenson. Lightweight Causal and Atomic Group Multicast. *ACM Transactions on Computer Systems,* Vol. 9, No 3, August, 1991, 272-314.

[7] Kenneth P. Birman. The Process Group Approach to Reliable Distributed Computing. *Communications of the ACM,* December, 1993.

[8] Kenneth P. Birman and Timothy Clark. Performance of the Isis Distributed Computing Toolkit. *Technical Report,* TR94-1432, Dept. of Computer Science, Cornell University, June 1994.

[9] Navin Budhiraja, Keith Marzullo, Fred B. Schneider and Sam Toueg. The Primary-Backup Approach. *Distributed Systems second edition,* Addison-Wesley, 1993, 199-216.

[10] David R. Cheriton and Willy Zwaenepoel. Distributed Process Groups in the V Kernel. *ACM Transactions on Computer Systems*, Vol. 3, No. 2, ACM, May 1985, 77-107.

[11] Bradford B. Glade, Kenneth P. Birman, Robert C. B. Cooper and Robbert van Renesse. Light-Weight Process Groups. *Proceedings of the Open-Forum '92 Technical Conference,* November 1992, 323-336.

[12] John Ioannidis, Dan Duchamp and Gerheld Q. Marguire Jr. IP-based Protocols for Mobile Internetworking. *Proceedings of SIGCOMM'91,* ACM, September, 1991, 235-245.

[13] M. Frans Kaashoek and Andrew S. Tanenbaum. Group Communication in the Amoeba Distributed Operating System. *Proceedings of the IEEE International Conference on Distributed Computing,* IEEE, May 1991, 222-230.

[14] Vassos Hadzilacos and Sam Toueg. Fault-Tolerant Broadcasts and Related Problems. *Distributed Systems second edition,* Addison-Wesley, 1993, 97-145.

[15] Kimberly Keeton, Bruce A. Mah, Srinivasan Seshan, Randy H. Katz and Domenico Ferrari. Providing Connection-Oriented Network Services to Mobile Hosts. *Proceedings of USENIX Symposium on Mobile & Location-Independent Computing,* USENIX, August, 1993, 83-102.

[16] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM,* 21(7):558-565, July 1978.

[17] Dalia Malki, Ken Birman, Aleta Ricciardi and Andre Schiper. Uniform Actions in Asynchronous Distributed Systems. *Technical Report,* TR94-1447, Dept. of Computer Science, Cornell University, September, 1994.

[18] J. B. Postel. Transmission Control Protocol. *RFC793,* SRI Network Information Center, Menlo Park, CA, September, 1981.

[19] Fred B. Schneider. Paradigms for distributed programs, in Distributed Systems—Methods and Tools for Specification, *Lecture Notes in Computer Science,* Vol 190, Springer-Verlag, New York, NY, 1985, 343-430.

[20] Fred B. Schneider. Replication Management using the State-Machine Approach. *Distributed Systems second edition,* Addison-Wesley, 1993, 169-197.

[21] F. Teraoka, Y. Yokote and M. Tokoro. A Network Architecture Providing Host Migration Transparency. *Proceedings of SIGCOMM'91,* ACM, September, 1991, 209-220.