MCK: Wireless Communication with Mobile Agents.

Authors: ***Leon Hurst, Trinity College Dublin, Dublin 2.***

Fergal Somers, Broadcom Éireann Research Ltd., Dublin 2.

Pádraig Cunningham, Trinity College Dublin, Dublin 2.

In mobile computing, a computer which in 1970 filled several large rooms, has been reduced in size so that it is now slips neatly into your pocket or is bound to your wrist. It has taken the last decade for this feat of engineering to come to fruition. Meanwhile, we have also experienced many evolutions and revolutions in the software systems and communication fields. The most dramatic of these is the unprecedented rise of the Internet and large Intranets. Key to this revolution is the great need which people have for ubiquitous and instant communication.

Hence, a fundamental question which Computer Science is asking itself is: "How do we support complex communication-based applications on our now sophisticated, yet limited, mobile computing devices?" Initial answers to this question range from the infrastructure-level (e.g. GSM which offers ~1Kbps maximum bandwidth [3]), to the system level (e.g. PPP and TCP/IP), up to the application level (e.g. Lotus cc:Mail and Lotus Notes). In order to tackle part of this question we concentrated on the mismatch between conventional communication protocols (e.g. TCP/IP) and the hostile characteristics of cellular wireless communication networks (e.g. GSM). To this end we have developed the Mobile Communication Kernel (MCK).

The cellular wireless environment is deemed hostile as it suffers from device mobility, frequent and prolonged disconnections, very low variable bandwidth and high latencies.

A Mismatch of two Technologies

Communication-based applications (e.g. email, WWW, Lotus Notes) were developed when communication systems were based on fixed networks and powerful desktop computers. Little thought was given to their performance in a mobile computing context. Thus, we find ourselves today with distributed applications running over fixed network-based protocols in a mobile environment. These protocols are based on a set of assumptions related to wired networks which are false in the wireless case (see sidebar: Limitations of Conventional Protocols). The assumptions are presented in the second column of Table 1.

Table 1. Comparison of attributes of a fixed network with a celluler mobile network.

Network Attribute	**fixed network-Internet**	**cellular network-GSM**
Location (logical/geographical)	Both fixed	variable/mobile
Connection life-time	long-lived (years)	short-lived (minutes)
Bandwidth	~64Kbps-Gbps	maximum of ~10Kbps
Latency	mostly <300 msec	>1 second
Reliability	very high	very low
Rate of change of topology	very slow (over months)	very fast (over seconds)
Power of end-point devices	Powerful desktop PCs and workstations	Limited PDAs or laptops, physically vulnerable

However, a mobile computing and communication system is characterised by very different characteristics [5, 6] as presented in the third column of Table 1. The important characteristics of the mobile communication environment (third column) are essentially the opposite of the characteristics assumed in the development of conventional communication protocols (second column). The characteristics of a cellular wireless network fluctuates wildly while those of a fixed network changes slowly over time. Therefore, protocols such as TCP, RPC and CORBA method invocation (essentially RPC) suffer a variety of limitations in a mobile environment. Some projects including IPv6, I-TCP and M-RPC attempt to address some of these problems (see section on Related Work).

In summation, the key problems of using conventional protocols over cellular wireless networks are:

- *Bandwidth limitations*. Such a scarce resource should be used wisely.
- *Routing*. point-to-point routing is inadequate when a user has no knowledge of the geographically local network.
- *Robustness*. Network topology and service availability change frequently in a mobile environment.
- *Disconnections*. Upon disconnection, any messages sent by a mobile device which fail, fail silently, as no error can be returned to the mobile.

In response to these problems we have developed a list of requirements for a new, *active*, transport protocol aimed specifically at cellular mobile networks:

1. *Autonomous message filtering* so as to increase bandwidth efficiency.
2. *Dynamic re-prioritisation of messages* so as to increase bandwidth efficiency.
3. Add *hierarchical routing* so that messages can be routed on service or content types .
4. Support for *dynamic routing* so that disconnected messages can independently change routing in the face of network or service failure.
5. *Message cloning, creation and persistence* in order to support increased robustness and enable dynamic multicasting.
6. *Active (code carrying) messages* capable of autonomous operation when the end terminal is disconnected.

The remaining question is: “With what technology can we be build such an *active* protocol?”

Overview

The simple answer to the pervious question is Mobile Agents. In satisfying the requirements for a new protocol the MCK was developed. In essence this is a messaging system where the messages themselves are active (encoded as small efficient Mobile Agents capable of self determination, ~1.5K in size). We call these messages *Smart Messages*. Thus, instead of the routers controlling passive messages, as in IP (see Figure 1a), MCK routers provides the necessary *mechanisms* for a smart message to operate (see Figure 1b). The MCK provides information (in the form of events) regarding changes in the local network. It is then up to the smart messages themselves to implement the necessary *policies* such as routing, lifetime control and delivery. This typifies the principle of separating policy from mechanism. By using active messages, it is hoped that the MCK can support *dynamically adaptable*, *robust* and *efficient* transmission of data in a cellular wireless network.

The difference between Figure 1a and 1b is that the smart message in 1b is operated by a metaphorical courier (the smiley man) who uses the facilities offered by an open MCK router. However, the IP packets in Figure 1a are entirely controlled by the IP routers which act as black boxes. We therefore call the MCK routers ‘white box routers’.

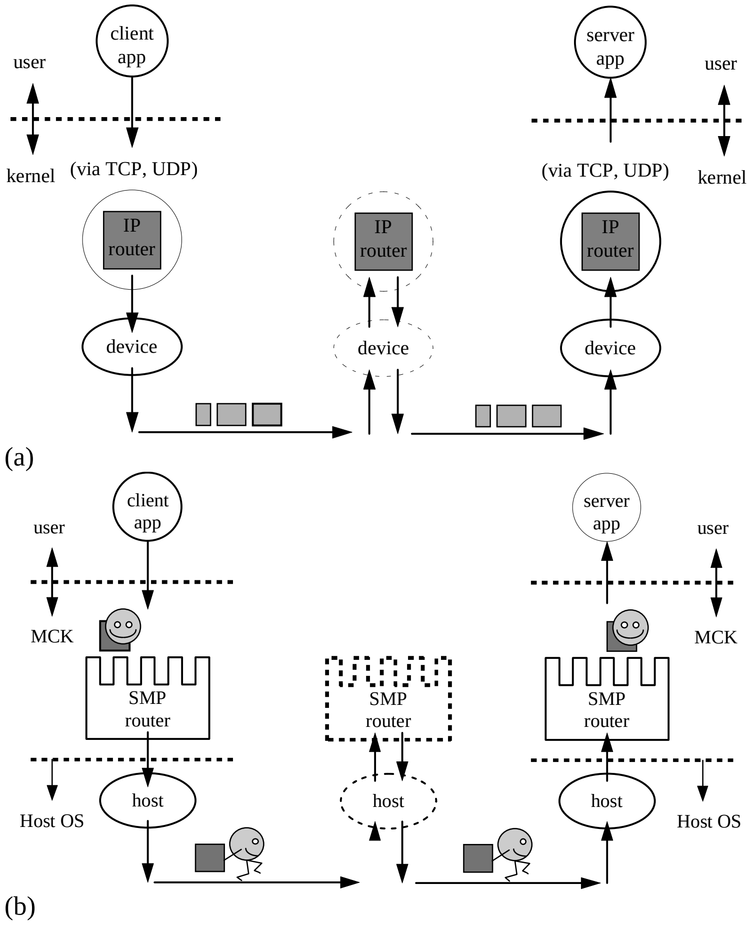

Figure 1. Comparison of a conventional IP network and an MCK network. (a) The passive IP network is populated by black box routers. (b) The MCK network is populated with open white box routers, smart messages are managed by couriers (smiley men).

Architecture

The Mobile Communication Kernel architecture is illustrated in Figure 2. It can be layered on top of nearly all conventional transport protocols from UDP to X.25. The current incarnation of the MCK framework supports both UDP and TCP underlying transport protocols. The MCK itself has been implemented in Smalltalk. This unusual choice of development system was precipitated by the fact that Java could not easily support a generic and dynamic event system until the recent introduction of JavaBeans, by which time we had an operational prototype.

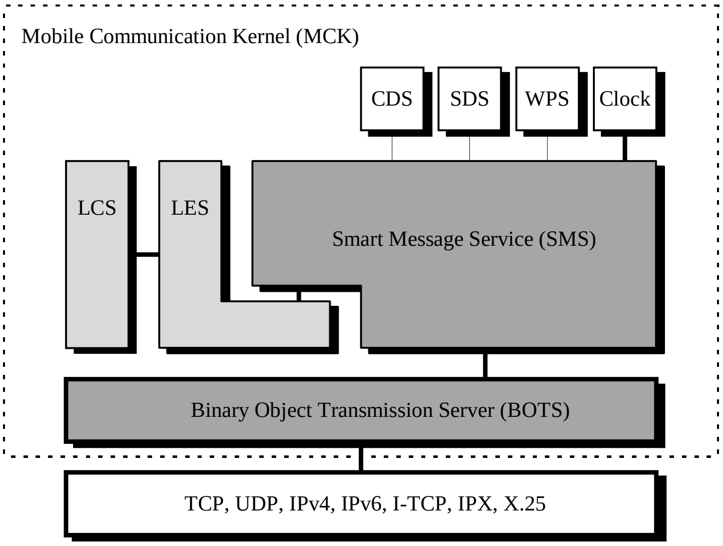
Mobile Communication Kernel (MCK)
CDS
SDS
WPS
Clock
LCS
LES
Smart Message Service (SMS)
Binary Object Transmission Server (BOTS)
TCP, UDP, IPv4, IPv6, I-TCP, IPX, X.25

Figure 2. The structure of the MCK router is split into core services and support services. The CDS, SDS, and WPS are directory services which supports a 3-level routing hierarchy. The LES and LCS support an event system which delivers events to smart messages. These constitute the support services. The core services are the SMS and the BOTS servers. Note: our SMS server has no relationship to the Short Messaging Service in GSM. These directly implement the routing functions for smart messages.

Core Communication Services

The communication services are layered hierarchically as shown in Figure 3. The host communication service depends on the underlying OS and simply provides transport level facilities. The BOTS server supports transparent serialization and point-to-point transmission of smart message objects. Finally the SMS server is the heart of the MCK as it acts as the smart message router. Note: our SMS server has no relationship to the Short Messaging Service in GSM. The SMS server has an external API through which applications can access the service. In addition it has an internal API through which hosted smart messages gain access to SMS functions as illustrated in Figure 1b.

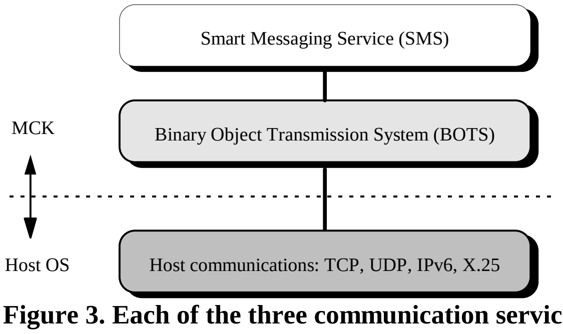
Smart Messaging Service (SMS)
MCK
Binary Object Transmission System (BOTS)
Host OS
Host communications: TCP, UDP, IPv6, X.25

Figure 3. Each of the three communication services are layer on top of each other.

Applications use the *SMS external API* in the following way:

1. First, an empty smart message is allocated from the SMS server. It will have been initialized with some system information.
2. It is then initialized by the application. As we will see, it is not just initialized with data, but also with functions which implement the different routing, filtering and recovery policies of a given application. It is predominantly this aspect of a smart message which makes it a Mobile Agent.
3. The application then dispatches the message to the SMS server for transportation to its destination (possibly unspecified at dispatch time). There is a default routing function in the SMS server. However,

as with almost all aspects of a smart message, it can be customized by initializing the message with the appropriate functions.

4. Assuming successful routing, the smart message is received at the destination server and is delivered up to the destination application.
5. If the message encounters a warning or failure event, it is temporarily withdrawn to a limbo area. There it has 30 seconds in which to resolve the event and proceed on its way. In doing this it has full access to the *SMS internal API*.

As a smart message routes itself to its destination it may encounter innumerable problems such as unavailable destination nodes or services, disconnections, blockages due to overused bandwidth. These situations are notified to smart messages by sending out events. From there the message can access SMS functions through the *SMS internal API*, open only to hosted smart messages. The internal API supports the following classes of operation: creation, navigation, modification and communication (see Table 2). Under limited circumstances a smart message can generate its own events and therefore communicated with other messages. However, the conventional form of inter-message communication is via the Message Board which is based on a *put and get* model. A smart message can not gain direct access to another message unless the other message invites it through an event communication.

Table 2. Important SMS internal API operations.

Creation	Navigation	Modification	Communication

`new`
`clone`
	`route`
	`sendto_IP`
	`dispatch`
	`lookup_ServiceInstance`
	`search_ServiceType`
	`search_ContentType`
		`dequeue`
		`transfer`
		`new_Priority`
		`delete`
			`announce_Event`
			`MB_put`
			`MB_get`

Support Services

The directory services are used to register content types, service types and service instances (servers and applications). These services are heavily used in the routing of smart messages above the IP level. In Figure 2 these are represented by the Content Directory Service (CDS), the Services Directory Service (SDS) and the White Pages Service (WPS). The hierarchical routing system is shown in Figure 4. If a message has a Content Type as a destination, the automatic routing system will resolve it down to a Service Type which can handle the content, and then further down to a Service Instance, representing an application or server. Thus, the ‘Currency Transfer’ content is resolved to the ‘Bureau de Change’ service type, which is further resolved to the BdCServer at Nybank.com. Service instances are directly mapped to IP names which are then resolved using DNS. This type of hierarchical routing is very useful on environments where one has little or no knowledge of local services.

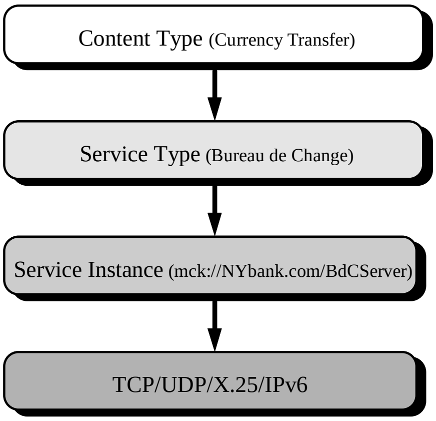
Content Type (Currency Transfer)
Service Type (Bureau de Change)
Service Instance (mck://NYbank.com/BdCServer)
TCP/UDP/X.25/IPv6

Figure 4. Routing hierarchy of destination types.

The event system provides a service to all components that wish to communicate through events, that is to announce and receive events. This system is dynamic so that new event types can be introduced to the system *as it is running*. Events are announced to anonymous groups/clusters of subscribers. Thus, there are two structural components to this system, the Local Event Service (LES) which handles the events, and the Local Cluster Service (LCS) which handles event destinations. It is envisaged that this event system could be extended to support efficient distributed events. Events are a general communication service which the SMS router (and other services) use to announce incidents, which in turn trigger functions in a smart message. For example a disconnection event may cause a message to destroy itself. Events are classified into a hierarchy (see Figure 5).

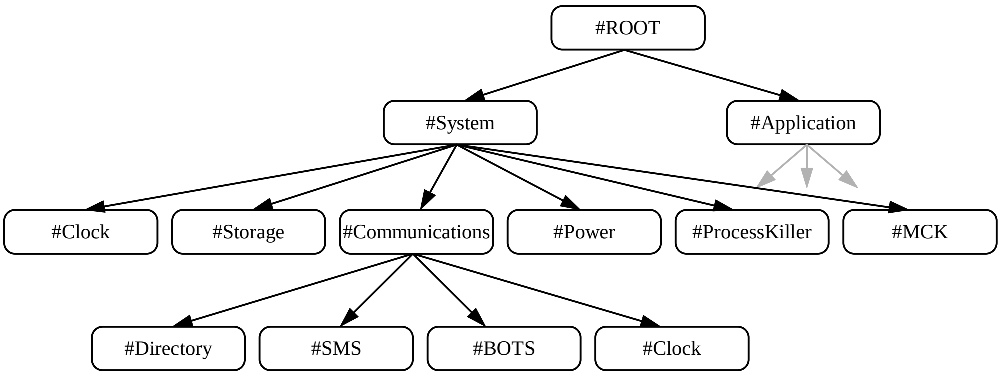

Figure 5. Event classification in current implementation of the MCK.

Smart Messages

Having described the nature of the new protocol and the structure of the SMS router implemented in the MCK framework, it is time to examine the smart messages themselves. As has been mentioned they are a specialized form of a Mobile Agent. Direct comparisons can be drawn between the structure of a smart message (see Figure 6) and that of the model view of Mobile Agents described the sidebar: A Model View of Mobile Agents.

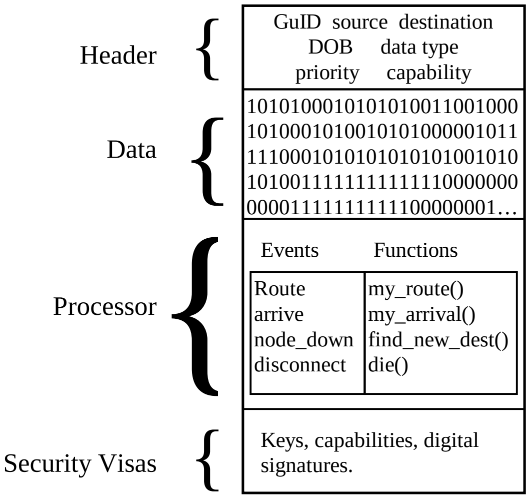

Figure 6. The structure of a smart message. Note the addition of a Processor and a dedicated security segment which do not appear in conventional packets such as IP.

The Header segment contains typical packet information such as an ID, source and destination. Notable among these fields is the datatype field which could be a mime datatype specification, and the capability field which controls the Processor segment's access to the message (itself). The capability is initialized by the SMS server at allocation time. The Data segment is a variable length field containing a bit sequence.

The heart of a smart message is the Processor segment. It contains a set of functions (some default and some user defined). The Processor also contains an event mapping table. The table maps event types (e.g. #SystemGoingDown and #NetworkConnectionLost) to functions which are to be invoked when such events are raised. Events are usually raised by the SMS server or system-level servers such as a Network Topology server. Finally, any additional security tokens are stored in the Visa segment where they can be revalidated by other SMS servers. For security reasons, the SMS server only activates a messages Processor after it has been validated.

Sample Message

Figure 7 shows a very simple example of a PDA device dispatching a smart message onto the fixed network. The message carries a Currency Transfer as its data.

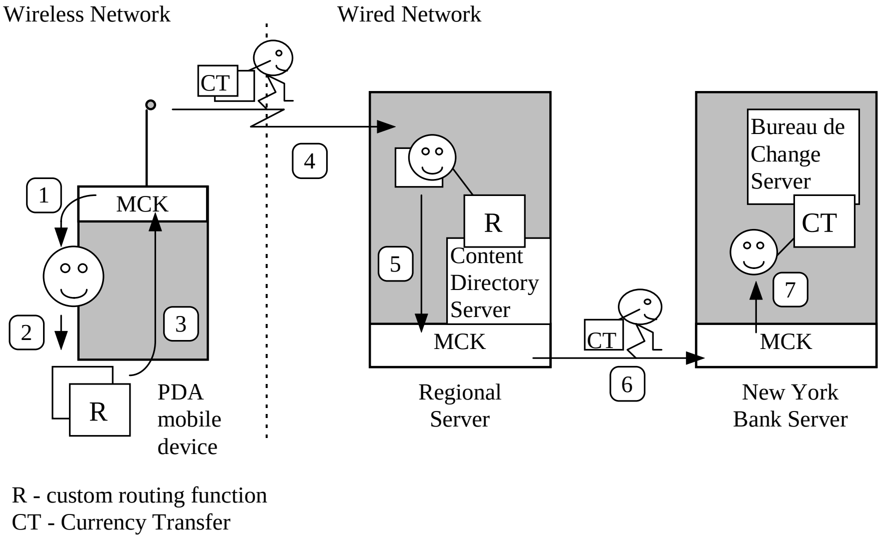

Figure 7. A mobile PDA device dispatches a smart message onto an unfamiliar wired network. It is the job of the message to find a suitable service willing to process the Currency Transfer it carries.

1. At the PDA a smart message is allocated from the MCK using the `new` operation.
2. The message is initialized with data which is a Currency Transfer (CT in Figure 7). In addition, a custom routing function (R in Figure 7) is loaded into the Processor segment. Smart messages have an external API which allows application to make these initializations.
3. The initialized message is dispatched to the MCK with the `dispatch` operation.
4. If the PDA is disconnected the message is queued until a connection is established. Eventually, the message gets delivered to the nearest Mobile Support Station. At this point the PDA is free to disconnect as the message is independent of the sending application.
5. The smart message is received by the MCK at the Regional Server. It then queries the Content Directory Server for any services willing to process a 'Currency Transfer' type. Once it has resolved a destination (e.g. mck://NYbank.com/BdCServer) it discards the old routing function and proceeds to the new destination.
6. The message with a new destination proceeds to the New York Bank Server and is accepted by the local MCK.
7. Finally the message is delivered to the Foreign Currency Server where the transaction can be processed.

The above example is simple and only illustrates the general operation of smart messages. In order to evaluate the MCK we developed a highly distributed application called TNET [6]. TNET is a nationwide tourist information system aimed specifically at mobile tourists. By using TNET as a driver we have been able to test the MCK. Specifically, the test scenarios investigated the following aspects of smart messages and the MCK: automatic routing, custom routing, routing failure recovery, system failure recovery, message filtering, custom multicasting and message specialization through inheritance.

Limitations

The MCK has some limitations, both in its implementation and its design. These are discussed below.

Although the MCK supports a capability based security system, this system relies heavily on specific functionality provided by Smalltalk. This hinders its portability to other environments such as Java. In addition, the security system is aimed at supporting a closed environment. When used in an open system the MCK is subject to a wider range of attacks. This requires a reevaluation of security in the MCK as a whole.

Currently data-less message sizes range from 2.5K upwards depending on the complexity of the initialized functions. It should be possible to further reduce the size of messages by using compression and alternative function encodings (e.g. Java or TCL).

The MCK implementation supports both TCP and UDP underlying transport protocols. In order to overcome the stateful nature of TCP (both ends being left inconsistent after a disconnection), additional data must be communicated during a message transfer. This reduces bandwidth efficiency. However, TCP provides reliability which UDP does not. In order to make UDP (a stateless protocol) reliable we added sequence numbering which also effected bandwidth efficiency. Usually such extra information has little effect on efficiency except that with a cellular wireless network the maximum bandwidth is pitifully small at ~1Kbps. The extra information constitutes an sizable proportion of the data bandwidth. It could be worthwhile testing other transport protocols such as I-TCP and X.25.

During the MCK's testing phase, it was noted that application supplied functions often followed identifiable patterns. Rather than have each application write its own functions, it should be possible to exploit the patterns so that applications could use an MCK supported library of common functions.

Ultimately, the question must be asked: "Does the MCK make mobile communication over a cellular network more efficient?" The answer to this question is currently unavailable. In order to make a very limited judgment we have to re-implement the TNET system using conventional protocols, and then make a functional and performance comparison. Regardless of the outcome, our work has highlighted the basic problem which is the mismatch between conventional protocols and the hostile characteristics of a cellular wireless network.

Related Work

Due to user pull and industrial push, wireless communication is becoming popular. Thus, it is not surprising that it receiving attention in the research community.

The new IPv6 [4] protocol supports transparent routing of IP datagrams to mobile hosts on the Internet. Each mobile host is identified by a home address, regardless of its current point of attachment to the Internet. While situated away from its home, a mobile host is also associated with a care-of address, which provides information about its current point of attachment to the Internet. IPv6 packets which are addressed to the home address are transparently routed to the care-of address where the destination is currently attached to the network. Nodes can cache the care-of-addresses rather than continually routing traffic through the home address. When a mobile host moves, packets sent to the care-of address will fail causing them to be re-routed through the home-address. The home agent will be informed of any movement and will know the new care-of address for the mobile host.

With this information, datagrams are effectively 'tunnelled' between the mobile hosts and other hosts, regardless of the mobile host's location. Using this tunnelling of datagrams, a mobile host can attach to the network from any location, yet functionally operate as though it were attached from a home location. In logical terms, the mobile host is attached to the network from its home location.

DATAMAN is a large, ongoing project at Rutgers University. Much of their work concentrates on communication protocols. This has resulted in new versions of TCP and RPC. Indirect TCP (I-TCP) [1] is an enhancement of conventional TCP which improves end-to-end throughput by a factor of two to three. While this is a significant success it should be noted that three times something very small is still something very small. Hence this is not a total solution to the bandwidth problem, but a significant step in the right direction.

Mobile RPC (M-RPC) [2] is significantly different from the QRPC of the Rover toolkit [7]. Rather than changing the semantics of RPC, M-RPC will only store the results of RPC calls which cannot be delivered to a disconnected mobile host. However, M-RPC will *not* permit a mobile host to make non-blocking (asynchronous) RPC calls during disconnection. Calls made during disconnection can either return an error, or automatically call an application-supplied handler to manage the disconnected operation.

ROVER [7] is a toolkit intended to support the development of roving mobile applications. The core of this work is based upon a combination of Relocatable Dynamic Objects (RDO) and Queued Remote Procedure Calls (QRPC). An RDO is an object with a well defined public interface. These objects can be dynamically exchanged between clients and servers, usually across a mobile communication environment. QRPC is a communication protocol which builds on the original RPC protocol of 1984. An RPC call made with the original RPC protocol will cause the calling application to block until the remote computation is completed and returns. This presents significant problems in a mobile environment where disconnections are frequent. With QRPC one can make *non-blocking* RPC calls. This is essentially an *asynchronous* form of RPC where the calls are queued in a log until they can be serviced. Although *asynchronous RPC* can be useful, it is not entirely clear that changing such a fundamental principle of a long established protocol is *the right thing* to do.

In summary, IPv6 tackles the problems of device mobility and connection maintenance at the network-level. Both DATAMAN and ROVER tackle the problems of low bandwidth frequent and disconnections at the protocol-level, with the I-TCP, M-RPC and QRPC protocols.

Future Work

The MCK in its current incarnation is limited by its implementation in Smalltalk. With the recent advent of JavaBeans the framework should be easily ported to Java. It is worth considering supporting a greater variety of underlying transport protocols such as X.25 and IPv6 directly. Although the current MCK supports capability-based security, it needs further development which should include authentication services. The system should be extended to support distributed events. However, it would be important to limit event propagation in a mobile system so as not to waste bandwidth. Finally, it has been observed that the highly dynamic and adaptable nature of smart messages could be used in the design of a real-time messaging system.

Acknowledgments

This research was supported under the IAF framework between Broadcom Éireann Ltd. and Trinity College Dublin. Thanks to them for their funding.

References

1. Brake B. and B. R. Badrinath, “I-TCP: Indirect TCP for mobile hosts”. Proceedings of the 15th International Conference on Distributed Computing Systems, 1995.
2. Brake B. and B. R. Badrinath, “M-RPC: A Remote Procedure Call Service for Mobile Clients”. Technical Report, Rutgers WINLAB TR-98, 1995.
3. Déchaux C. and R. Scheller, “What are GSM and DCS?”. Electrical Communication, 1993.
4. Eklund T., “IPv6 - The next Generation Internet Protocol”. Ericsson ERA Technical Report, 1996.

5. Green S., L. Hurst, et al., “Software Agents: A review”. Tech. Report TCD-CS-97-04, Trinity College Dublin, 1997.
6. Hurst L., P. Cunningham and F. Somers, “Mobile agents−Smart messages”. Lecture Notes in Computer Science No. 1219, Springer Verlag, 1997.
7. Joseph D., A. deLespinasse, et al., “Rover: A toolkit for mobile information access”. Proceedings of the 15th ACM Symp. On Operating Systems Principles, 1995.

Sidebar: Limitations of Conventional Protocols

As has been illustrated in Table 1, the assumptions on which fixed network protocols (e.g. TCP and RPC) are based do not hold in the cellular communication environment.

Transmission Control Protocol (TCP) [3] is a reliable, end-to-end, connection oriented protocol. Operating TCP over a cellular wireless network results in bad performance and lack of robustness. This is due to the following design problems: 1) The TCP header contains redundant information. 2) TCP operates as an end-to-end protocol. Thus, retransmissions are propagated across the wireless link. Many of these retransmissions are due to long wireless propagation delays and are redundant. 3) Reasonable packet sizes on the fixed network are not the same on a wireless network. 4) Finally, the Congestion Control Policy of TCP is based on timeouts suitable for fixed networks. In a wireless network it causes the sender to back off prematurely and reduces throughput.

Furthermore, TCP was intended for long-lived communication sessions. However, wireless communication is dogged by frequent and unpredictable disconnections. Most applications based on TCP will fail upon disconnection.

Tanenbaum said it best [4] page 543: *"In theory, transport protocols should be independent of the technology of the underlying network layer. In particular TCP should not care whether IP is running over fiber or over radio. In practice, it does matter because most TCP implementations have been carefully optimized based on assumptions that are true for wired networks but which fail for wireless networks. Ignoring the properties of wireless transmission can lead to a TCP implementation that is logically correct but has horrendous performance."*

The Remote Procedure Call (RPC) [1] mechanism extends the ubiquitous procedure call mechanism as found in Pascal and Modula-2 so that it may operate across a network. An RPC invocation causes a local program to suspend execution, the RPC invocation is transmitted to the destination, where the associated procedure is executed and the results are then returned.

RPC by its very nature is synchronous. Thus, on disconnection, the invoker is left waiting for the completion of outstanding calls. This is best appreciated by users of the NFS file system, which runs over RPC. If the link between a remote host and its NFS server is lost, all applications with open files on the NFS server freeze. This usually results in the whole workstation freezing. As disconnections occur frequently in the mobile environment this proves to be a significant problem. The bandwidth consumed by RPC calls is high [2] because the functions parameters must be marshalled and sent to the destination. This results in several communication flows. However, secure RPC requires even more communication overhead as it is based on encryption using public keys administered by authenticated servers. This often results in ten times the amount of transmitted data [2].

Whereas the CORBA standard represents a very comprehensive distributed computing system, its basic remote method invocation mechanism can be thought of as RPC for Object-oriented systems. Hence, CORBA suffers similar problems to RPC.

References

1. Birrell A. D. and B. J. Nelson, "Implementing Remote Procedure Calls". ACM Transactions on Computing Systems, Vol. 2, No. 1, 1984.
2. Chess D. et al., "Mobile Agents: Are they a good idea?" Technical Report, IBM T.J. Watson Research Center, NY, 1995.
3. Postel J., "Transmission Control Protocol". Request for Comments 769, 1981.
4. Tanenbaum A., "Computer Networks". Third Edition, Prentice Hall, 1996.

Sidebar: A Model View of Mobile Agents

A mobile agent can be viewed as a collection of models which are highly integrated and interdependent [3]. This is illustrated by the following diagram, Figure X:

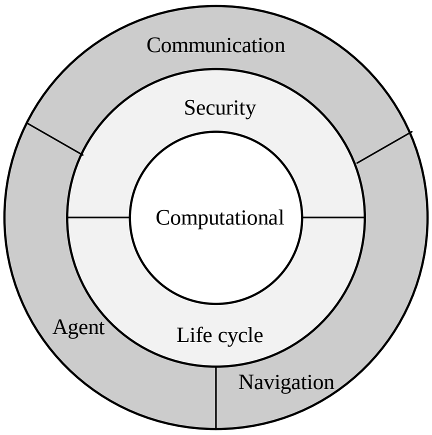

Figure X. Simple view of the structure of a mobile agent.

The core is based on the computational model. This has significant impact on the other models. It defines how we address other agents, hosts and resources. This is important for the security and communication models. The type of life cycle model adopted is dependant on the facilities of the computational model. Therefore, almost all current mobile agent implementations based on the Java computational model [2] have been forced to adopt the task-based model. The exception is Ara [5] which transmits the guts of the supporting Java VM when the resident agent migrates. On the other hand, the computational models of AgentTcl [4] and Telescript [6] are such that they support a persistent process life cycle model.

The second layer consists of the security and life cycle models. They are both highly dependant on the core computational model. Security issues permeate every aspect of a mobile agent and therefore must be provided for at the most basic level. In Telescript, primitive security features are implemented as authenticated identities, protected (read only) references, permits (a form of capability), secure channels and engine-mediated protocols for receiving new agents and for agent meeting. The life cycle model defines the valid states of an agent. Depending on the facilities of the computation model, the life cycle model can be as flexible as the persistent process model, or as simple as the task-based model.

The outer layer contains the communication, navigation and agent models. The agent model defines the ‘intelligent agent’ aspects of a mobile agent, such as learning and collaboration functions. The communication model is heavily dependant on the security and life cycle models, so that agents are not corrupted when communicating with other agents or hosts. It also defines the types of communication (i.e. streams, messages, RPCs and events). Finally, the navigation model uses the security model when it hands itself over to the host to be transported to another node. The transmission process often moves the agent into a different life cycle state, as in Aglets [1].

References

1. Chang D. and D. Lange, “Mobile Agents: A New Paradigm for Distributed Object Computing on the WWW”. OOPSLA'96 Workshop, 1996.
2. Gosling J. and H. McGilton, “The Java Language Environment: A White Paper”. Sun Microsystems, 1995.
3. Green S., L. Hurst, et al., “Software Agents: A review”. Tech. Report TCD-CS-97-04, Trinity College Dublin, 1997.

4. Kotz D., R. Gray and D. Rus, “Transportable Agents Support Worldwide Applications”. In Proceedings of the 7th ACM SIGOPS European Workshop, 1996.
5. Peine H. and T. Stolpmann, “The Architecture of the Ara Platform for Mobile Agents”. Lecture Notes in Computer Science No. 1219, Springer Verlag, 1997.
6. White J., “Telescript technology: The foundation of the electronic market place”. General Magic White Paper, 1995.