Mobility support for Collaborative Real-time Applications

Johan Kristiansson and Peter Parnes

Department of Computer Science & Electrical Engineering, Media Technology
Luleå University of Technology,
971 87 Luleå, Sweden
{Johan.Kristiansson, Peter.Parnes}@sm.luth.se

Abstract. *Today we are approaching a truly connected and wireless world where users can be always connected by switching between different carriers and providers. This paper presents an architecture which enables mobile media applications to operate across a wide range of wireless networks, gracefully handle network failures and share environmental information, such as packet loss-rate or available bandwidth with remote peers. The architecture is evaluated using a commercially available E-meeting application, Marratech Pro by seamlessly switching between three carriers; LAN, WLAN and GRPS.*

Keywords: Seamless roaming, robustness, overlay networks, mobile e-meetings

1 Introduction

The future wireless landscape will be rich in carriers and in competing technologies. Cheap high-speed wireless connectivity will be available with limited range in hot-spot areas and will be complemented by more traditional connectivity offering reduced performance over a much wider area.

Providing the best available connection from these networks, will help users to stay connected regardless of where they are. Switching between networks is however extremely difficult today since users are required to make active choices and remember configuration parameters like passwords or IP addresses. The situation is even more frustrating when applications must be restarted after a handover or a short period of disconnectivity. Additional problems are caused by low-performance networks like GPRS, which become congested when utilizing resource intensive applications.

This paper proposes an architecture, Resilient Mobile Socket (RMS), which can be used as a building block when implementing mobile media applications. Applications using the RMS architecture will be able to handle mobility and disconnectivity as normal program states. They will also be able to share environmental information, which can be used by applications to react on changes in the wireless environment and handle variations in bandwidth or latency.

The rest of the paper is organized as follows. Section 2 gives an introduction to the research problems. In section 3, a discussion about previous work is given. Section 4 outlines the architecture. An evaluation is done in section in 5 and in section 6 the paper is concluded with a discussion and future work.

2 Research issues

The goal of this paper is to describe and evaluate methods that can be used by applications to efficiently utilize and navigate in a rich wireless landscape. The vision is that *everything should just work* with a minimum of user effort and interaction.

Below is a description of the main issues that must be addressed in order to realize this goal.

How can applications survive an IP address update? There are several fundamental problems caused by mobility. The Internet routing model forces mobile hosts to acquire new IP addresses when roaming between different networks. This makes it not only difficult to preserve already established sessions, but also makes it cumbersome to locate mobile hosts visiting foreign networks. Additional problems are caused by Internet-working applications and protocols which are using IP addresses as none-volatile identifiers in data structures, assuming that they never change. As a result, many applications cannot survive an IP address update and must be restarted in order to function properly on new networks.

How can applications survive longer periods of disconnectivity? Many applications assume that some sort of network connectivity always exists. Disconnectivity is normally treated as an unexpected state or a failure state. Alive and time-stamp protocols are for example commonly used to detect crashed clients and to decide when to de-allocate used resources. These protocols will cause serious problems when supporting mobile clients which are very likely to hibernate in order to save power.

How can applications be always best connected? Deciding the most beneficial carrier or provider becomes a complicated problem when considering every possible attribute associated with each network. Correct in-data to handover decision algorithms must be acquired in order to make fair handover decisions. Dynamic information about reachable networks such as bandwidth or cost can be used in this process, but it is also important to consider applications and user requirements like QoS or IP multicast when deciding the best network.

How can applications perform well in a dynamic environment? Media streams suitable for high-bandwidth networks will very likely become exhausted when switching to a network with lower performance. Clients may not be able to show high quality video streams simple because there is not enough bandwidth or the cost of using the network is too high. If applications should be able to perform well then they need to obtain information about the environment so that media streams and other communication primitives can be adapted to suitable levels.

How can NAT/firewalls be supported? Many existing Internet protocols are incapable of supporting Network Address Translator (NAT) [1] firewalls. Protocols which are sending IP addresses and other address derivate in data streams for integrity checks will immediately fail since private addresses are only valid inside one NAT network and not routable on the Internet. This inconsistency will also cause problems for protocols using multiple channels and expecting some relationship between the local port and the actual port used by the NAT firewall. A complete description of these problems and design guidelines are given in [2], [3] and [4].

3 Related work

Mobility management can roughly be divided into two categories; network-layer and higher-layer mobility. Network-layer solutions try to add mobility support to the network layer in the Internet protocol stack, thus transparently providing mobility to all higher layers. Higher-layer mobility schemes on the other hand try to add mobility support to layers above the network layer and are often more flexible and deployable, but tend to be less generic.

3.1 Network-layer mobility

Mobile IP [5] is the current Internet Engineering Task Force (IETF) standard for host mobility. Using a home agent that is responsible for intercepting and redirecting packets destined to mobile hosts located on foreign networks, Mobile IP enables users to keep their home IP address wherever they are. This is particularly useful for mobile servers, like mobile web and file servers.

There are unfortunately some serious drawbacks with Mobile IP. To achieve mobility all packets must be tunneled from a home agent to a foreign agent which requires foreign agents to be deployed all around the Internet. Triangular and reverse tunneling will cause routing anomalies and consequently add extra overhead to the routing system.

Static address configuration can also be debatable. Today a lot of ISPs are using DHCP for address assignment. Configuring permanent home addresses and additionally distributing authentication keys adds extra administration overhead.

Several of these limitations are however addressed in Mobile IPv6 [6]. Route optimization [7] will for example eliminates triangular routing and enables packets to be tunneled directly to mobile hosts instead of using home and foreign agents. Mobility support will however be restricted to native IPv6 networks simple because IPv6 is not widely deployed yet. On IPv4 networks users will be left with no other option then using complex transition mechanisms or other mobility schemes.

It should also be pointed out that Mobile IP in general does not provide support for all the problems discussed in section 2. In fact, it completely lacks support for disconnections, handover decisions and adaptation interfaces.

3.2 Higher-layer mobility

Extensive work has been done to add mobility support to layers above the network layer. Most of the previous work in this area focuses on building a session layer, which is inserted between the application and the transport layer.

Communication with legacy systems is normally supported by placing a proxy server in the communication path, very similar to home agents in Mobile IP. In the MSOCKS [8] proposal a split-connection architecture for transport layer mobility is suggested. Communication is preserved by letting hosts re-connect to a dedicated proxy server when changing point of attachment or loosing connectivity.

Proxy servers are often considered attractive because they are in comparison easy to deploy. A serious drawback is however triangular routing, which can be completely avoided if mobility support is implemented directly into applications or as middlewares as suggested in [9]. A complete end-to-end architecture that uses TCP migration to restart previously established connections is proposed by Snoeren and Balakrishnan [10]. Similar end-to-end schemes are discussed in [11] [12] and [13]. Work has also been done to augment communication services in operating system in order to enhance performance and to provide system support for load-balancing and redundancy management [14].

The work presented in this paper differs from previous work in several ways. Most importantly, it uses a novel data stream in-band protocol for mobility and connection management. This protocol is also used for detecting network failures and distributing control information to remote sockets. RMS is also designed to operate on any IP network, including NAT networks.

4 The Resilient Mobile Socket architecture

This section describes the proposed architecture. The main component, the RMS datagram socket is an improved UDP socket with dynamic connection management and data stream facility. It provides the same set of methods as a standard UDP socket and additionally three methods, *suspend*, *resume* and *migrate*, which can either be called by users or automatically by the system.

The *suspend* method is used to hibernate communication and will automatically be called by the system when network connectivity is lost. The *resume* provides the opposite operation and is used to restore suspended communication. The *migrate* method takes a bind-address as argument and offers functionality to migrate communication between different network interfaces. An adaptation manager is involved in all processes and provides a subscription-oriented interface, which can be used by applications to send and receive notifications about resource constraints and connection status.

An important component in the architecture is the Resilient Mobile Socket Control Protocol (RMSCP), which provides an inband-stream communication facility allowing sockets to exchange control and environmental related information.

4.1 Mobility management

The main idea with the RMS architecture is to provide virtual connections by translating destination and source fields in packets according to a location table, which contains the remote IP and UDP port of all connected end-points. The location table is updated when receiving socket location update packets that are sent by remote end-points changing their point of attachment. This scheme is principally similar to binding updates in Mobile IP [7], but operates on the application layer instead of the network layer.

Socket encapsulation is a key mechanism used by the architecture to ensure robustness and mobility. By encapsulating one or many internal sockets, current communication can transparently be migrated to different internal sockets without disturbing running applications. A new encapsulated socket will for example automatically be created when a new operatable network interface is detected.

Two important steps are involved in the migration process. First of all, outgoing packets must be sent using the new internal socket and remote peers must be informed where to send packets. Outgoing packets are correctly routed by using the new internal socket as the default local socket and adding source specific routes to the routing table in the operating system to make sure that correct default gateway is used. Incoming packets are correctly routed by informing remote peers about the new socket address. Note, that the architecture assumes that at least one socket end-point is stable during a handover.

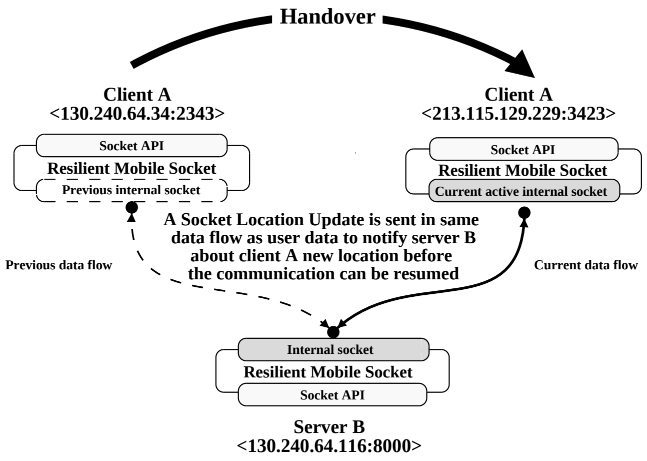


Fig. 1. A data stream migration process.

In figure 1 a data stream migration process is illustrated. Client A reconnects to a new network, which causes the Resilient Mobile Socket to create a new internal socket bound to the new socket address, 213.115.129.229:3423 and send a socket location update to Server B. The RMS on Server B will then redirect packets with destination 130.240.64.34:2343 to 213.115.129.229:3423 and re-stamp all traffic received from 213.115.239.229:3423 to 130.240.64.34:2343.

This scheme is completely transparent and RMS-enabled applications running on Server B will always perceive Client A as connected to 130.240.64.34:2343, where the connection was first initialized.

4.2 In-band communication

Exchanging control information can principally be done in two different ways. One solution is to use separate control channels similar to FTP or RTCP, but then it becomes extremely difficult to support NAT firewalls.

A more deployable solution, which is utilized by RMSCP protocol is to segment the traffic and send control packets in the same stream as normal data traffic as illustrated in

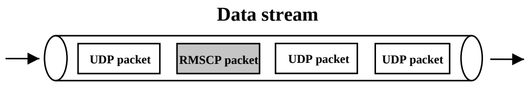


Fig. 2. In-band transmission of socket control packets.

figure 2. This functionality is achieved by using a magic header to identify and separate RMSCP control packets from regular UDP packets.

Below is a description of some problems that the RMSCP protocol is designed to solve.

Connection management Since RMSCP packets are sent in the same stream as user data, it is possible to map previous and new connections by comparing an unique socket id obtained from location update packets. Once a connection has been identified, remote IP addresses and UDP port numbers can easily be determined by examining the source address and the source port obtained from the received RMSCP packet.

Message passing The RMSCP protocol is also used as a message passing system, which allows applications to transparently notify remote peers about environmental changes, like packet loss, preferred data rate and connection status. The RMSCP protocol is also used to explicitly notify remote RMS about expected time-outs such as hibernation, thus allowing scarce resources like UDP ports to be deallocated.

Network failure detection Alive packets are regularly sent in data streams to detect network failures. It is assumed that network connectivity has been lost if a remote RMS has not reported back for some time. This will cause all communication to the peer to be suspended, which also means that all outgoing packets will be dropped. A drawback of active probing is however low performance networks, like GPRS where alive packets can consume a significant part of available bandwidth. To prevent this from happening, network failures is also detected by reacting to operating system events, such as ”ICMP: host unreachable”.

NAT firewall support The RMS architecture will never exploit IP address derivates to remote peers since private IP addresses are only valid inside NAT networks. Remote IP addresses and port numbers are instead implicitly determined by examine RMSCP control packets as previously described. RMSCP alive packets will also prevent NAT firewalls from timing-out, which will happen if communication has been idle too long.

5 Evaluation

The architecture has been evaluated by measuring handover latency and maximum performance on a prototype that we have developed. Tests have also been done with real users in real scenarios by extending Marratech Pro[1], a commercially available collaborative workspace providing tools for group communication and e-meetings.

[1] Marratech AB `http://www.marratech.com`

5.1 Implementation

We have implemented a prototype in Java JDK 1.4. A handover decision scheme has also been implemented to ensure that traffic is sent and received using the most beneficial network or internal socket. Since the focus has been to develop a prototype that works on LAN, WLAN and GPRS, the choice of selecting the best network is quite obvious. However, since handover decisions and the actual handover mechanism are completely separated, more advanced handover decision algorithms can easily be plugged-in to the architecture in the future.

5.2 Marratech Pro and Mobile E-meetings

Initial experiments to test media adaptation has been done, but providing a scalable solution that benefits all e-meeting participants is a complicated problem that needs more attention. The RMS architecture has also been used to inform e-meeting participants about other members connection status. Participants will for example be notified when users become disconnected or switch connection type.

An interesting feature, not provided by the RMS architecture, is a distributed synchronization protocol based upon SRM [15] and used in Marratech Pro to ensure consistency between shared whiteboard pages and chat conversation. This was particular useful and made it possible for users to work with whiteboard pages even if connectivity was temporary lost.

5.3 Handover latency

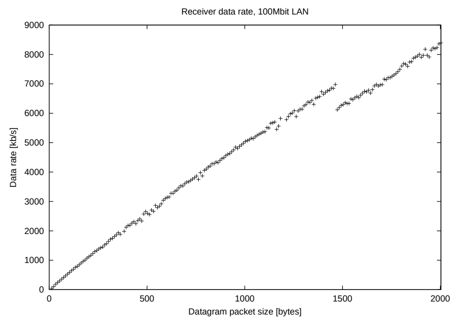
Handover latencies were measured by sending a constant bit-rate stream and calculating the time the Resilient Mobile Sockets were disconnected when switching between two active network interfaces. Handovers were initiated by de-activating and activating the network interfaces with highest priority, thus forcing communication to be migrated to the other network interface.

Table 1. Handover latency

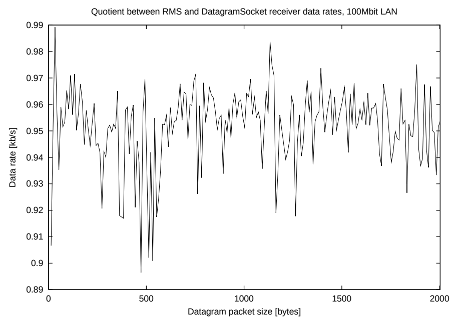
Handover scenario	Typical handover latencies
LAN $\rightarrow$ WLAN	300 ms
WLAN $\rightarrow$ LAN	180 ms
GPRS $\rightarrow$ LAN	180 ms
LAN $\rightarrow$ GPRS	6600 ms
WLAN $\rightarrow$ GPRS	5700 ms
GPRS $\rightarrow$ WLAN	300 ms

Table 1 shows some typical handover latencies for WLAN, LAN and GPRS. Note that a handover to GPRS takes very long time. The reasons for this behavior is the limited capacity offered by the GPRS technology and GPRS providers.

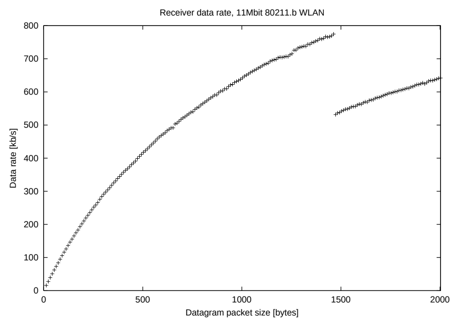
5.4 Performance and overhead



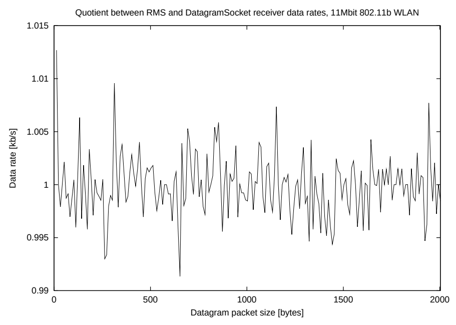
(a) Maximum receiver data rates on a 100 MBit LAN network



(b) Overhead on a 100 MBit LAN network



(c) Maximum receiver data rates on a 11 MBit 802.11b network



(d) Overhead on a 11 MBit 802.11b network

Fig. 3. Maximum receiver data rates and performance overhead on LAN and 802.11b WLAN networks.

The performance was measured by sending as many packets as possible to a receiver application. The overhead was calculated by comparing data rates between a standard Java DatagramSocket and a RMS datagram socket. The sender application was run on a Pentium 2, 800 MHz processor and the receiver application was run a Pentium 3, 1.2 GHz processor.

The performance and overhead tests in figure 3 shows that a RMS datagram socket is approximately 5-6% slower than a standard Java DatagramSocket on a 100Mbit ethernet LAN. The tests also show that there were no significant overhead on 802.11b wireless networks, which is probably because the network becomes a bottleneck instead of the CPU. No significant difference in overhead between sender and receiver data rates was either found.

Note that Maximum Transfer Unit (MTU) is set to 1500 which will cause UDP packets larger than 1500 bytes to be fragmented into two ethernet segments thus drastically decreasing the data rates, which can be seen in figure 3.

6 Discussion and conclusion

This paper describes a powerful mobility scheme that besides mobility management also offers network failure protection and support for sharing environmental information.

Two key mechanisms are utilized to enable this functionality. Socket encapsulation allows internal sockets to fail and be replaced without disturbing higher-level applications. In-band transmission of control packets in data streams allows information to be transparently distributed to remote peers without redesigning existing protocols or using extra control channels.

In section 2, a number of problems were introduced.

– *How can applications survive an IP address update?* This problem is solved by migrating on-going communication to a new encapsulated internal socket bound to the new IP address. The RMSCP protocol is a key component in this process and provides support for identifying connections, allowing packets to be correctly redirected.
– *How can applications survive longer periods of disconnectivity?* The RMS architecture will automatically suspend on-going communication and notify higher-level applications when connectivity is lost.
– *How can applications be always best connected?* A handover decision scheme for proactivily initiating handovers to better network interfaces was quickly mentioned in the paper. In our current prototype we are only considering connection types in the decision process, but we are planning to develop and test more advanced handover decision algorithms capable of dealing with more parameters.
– *How can applications perform well in a dynamic environment?* The RMS architecture provides an inband-stream communication facility which can be used by applications to send and receive environmental information. This information can then be used by applications to adapt media.
– *How can NAT/firewalls be supported?* NAT firewalls are supported by implicitly determining IP addresses and port numbers from RMSCP control packets rather than sending topological information as pay-load in packets.

6.1 Future work

An important problem left out in this paper is security, which can easily can be compromised without any preventative actions. Connections can for example easily be hijacked without an authentication header in the RMSCP protocol.

Besides improving security, our main focus will be to develop better handover decision algorithms and user profiles, which we are planing to test on Borderland [16], a pervasive architecture for wearable computing. This will also give us an opportunities to conduct more users tests and evaluations.

7 Acknowledgement

We thank Mikael Drugge, Fredrik Johansson, Marcus Nilsson, Dick Schefström, Jeremiah Scholl, Kåre Synnes and Göran Öberg and for their help with this work and paper. The work was sponsored by the Centre for Distance-spanning Technology (CDT) and Mäkitalo Research Centre (MRC) under VITAL and VINNOVA RadioSphere.

References

1. Egevang, K., Francis, P.: The IP Network Address Translator (NAT) (1994) IETF RFC1631.
2. Srisures, P., Holdreg, M.: IP Network Address Translator (NAT) Terminology and Considerations (1999) IETF RFC2663.
3. Senie, D.: Network Address Translator (NAT)-Friendly Application Design Guidelines (2002) IETF RFC3235.
4. Hain, T.: Architectural Implications of NAT (2000) IETF RFC2993.
5. Perkins, C.: IP Mobility Support (1996) IETF RFC2002.
6. Perkins, C.E., Johnson, D.B.: Mobility support in ipv6. In: Mobile Computing and Networking. (1996) 27–37
7. Perkins, C.: Route optimization in Mobile IP (2001) Internet Draft, IETF, draft-ietf-mobileip-optim-11.txt.
8. Maltz, D.A., Bhagwat, P.: MSOCKS: An architecture for transport layer mobility. In: INFOCOM (3). (1998) 1037–1045
9. Snoeren, A.C., Balakrishnan, H., Kaashoek, M.F.: Reconsidering internet mobility. In: Proc. 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII). (2001)
10. Snoeren, A.C., Balakrishnan, H.: An end-to-end approach to host mobility. In: Proc. 6th International Conference on Mobile Computing and Networking (MobiCom). (2000)
11. Qu, X., Yu, J.X., Brent, R.P.: A mobile TCP socket. In: International Conference on Software Engineering (SE '97). (1997)
12. Yau, D.K.Y., Lam, S.S.: Migrating sockets - end system support for networking with quality of service guarantees. IEEE/ACM Transactions on Networking **6** (1998) 700–716
13. Zandy, V.C., Miller, B.P.: Reliable network connections. In: ACM MobiCom. (2002)
14. Haungs, M., Pandey, R., Barr, E., Barnes, J.F.: Migrating sockets: Bridging the os primitive/internet application gap (2002)
15. Floyd, S., Jacobson, V., Liu, C.G., McCanne, S., Zhang, L.: A reliable multicast framework for light-weight sessions and application level framing. IEEE/ACM Transactions on Networking **5** (1997) 784–803
16. Nilsson, M., Drugge, M., Parnes, P.: In the borderland between wearable computers and pervasive computing. Research report, Lulea University of Technology (2003)