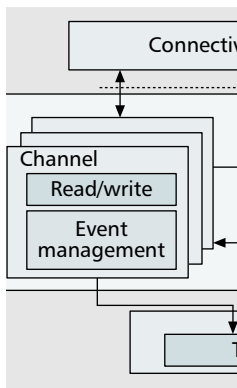


ADAPTIVE CONNECTIVITY MANAGEMENT MIDDLEWARE FOR HETEROGENEOUS WIRELESS NETWORKS

JUN-ZHAO SUN, JUKKA RIEKKI, MARKO JURMU, AND JAAKKO SAUVOLA,
UNIVERSITY OF OULU, FINLAND



The trends of network convergence and mobile accessibility in the Internet are bringing new challenges to the connectivity management of end hosts. Concerning network convergence, the configuration of heterogeneous access networks should be taken into consideration.

ABSTRACT

The trends of network convergence and mobile accessibility in the Internet are bringing new challenges to the connectivity management of end hosts. Concerning network convergence, the configuration of heterogeneous access networks should be taken into consideration. As for mobile accessibility, seamless handoff between diverse access points is a challenging issue. This article presents the design and implementation of connectivity management middleware (CMM), a channel-based architecture for context-aware connectivity management. This architecture can both provide network awareness to applications and manage network resources in an adaptive fashion. In the case of network awareness, the platform provides interfaces for applications to query network QoS and availability status, as well as subscribe connection events. As for adaptive resource management, channel-based transport services for seamless access switching and disconnection treatment are provided based on a policy mechanism. A prototype is implemented with which experiments were performed in a GPRS-WLAN integrated environment in order to demonstrate the operational correctness of the architecture. Performance metrics are measured and analyzed.

INTRODUCTION

New advances in both wireless networks and portable devices are making mobile Internet a reality. A wide range of wireless network technologies can be used for mobile Internet access including, for example, Bluetooth, WLAN, 2G (GSM), 2.5G (GPRS), 3G (UMTS), and so forth. On the other hand, nowadays it is very common to have one mobile device equipped with multiple network interfaces. For example, a laptop may be equipped with LAN, WLAN, Bluetooth, and IrDA, and a mobile phone may have GSM, GPRS, Bluetooth, and IrDA. In this way, a mobile device can be attached to multiple heterogeneous wireless networks.

Wireless access networks vary greatly by nature, for example, with regard to data rate, coverage, subscriber volume, supported mobile velocity, anti-interference, and suitable transmitting environment. Usually networks with bigger coverage are of lower data rate, and vice versa. It has been well recognized that no single access technique can fulfill all the requirements. Moreover, the QoS parameters of each single wireless network vary dynamically over time as well, in terms of reliability and availability, bandwidth, delay, jitter, response time, and packet-loss rate. Finally, user mobility leads to continuous changing of location and environment, network operator and service provider, and access networks. These highly dynamic factors in mobile devices, wireless networks, and end users have led to the demand for adaptive-connectivity management, so that diverse networking resources can be utilized in a simultaneous, collaborative, and complementary way. Among others, host mobility in heterogeneous wireless networks is the core problem.

In dealing with host mobility in the Internet, there are three main issues. The first issue is how to acquire a new valid IP address when the mobile host enters a new area. The second issue is how to make the mobile host reachable to any other potential peer hosts anytime and anywhere. The third issue concerns how to maintain the ongoing communication session during the connection change (i.e., handoff). With the current Internet architecture, the first issue can be easily solved by the widely deployed Dynamic Host Configuration Protocol (DHCP) [1] servers, in which IP address can be automatically configured. With regard to the second issue, the extension-dynamic DNS [2] can be used for the purpose of mapping permanent domain names to the dynamic IP address of the mobile host. In particular, this article focuses on the third issue: handoff.

In the environment of heterogeneous multi-access networks, one of the major challenges is the so-called *vertical handoff*. Here, by vertical we mean that the handoff occurs between differ-

ent networking technologies, for example, WLAN and GPRS. In this case, we face the following challenges:

- *Network interface selection.* This is to ensure that the best connection is used by the application at any time. Some rules must exist for the determination of “the best” connection, in which information about user, device, networks, and applications has to be examined.
- *Connection switching.* When the application is operating on data transmission, network status is subject to change. At some point a better connection may be available to the application or the used one is lost. Then the connection should be automatically and seamlessly switched to the new one while maintaining the connection’s continuity.
- *Disconnection treatment.* During the period that no valid network connection is available, I/O requests of the application should be suspended. The operations have to be resumed later when one qualified interface becomes available.
- *Perception of application.* Network-aware applications need to be aware of connectivity contextual information (such as chosen interface, connection status, available bandwidth, delay, etc.), so that application may adjust its activities accordingly.

Some work on host mobility and vertical handoff has been reported in the literature. Various solutions have been derived according to the different protocol layers at which the mechanisms are implemented. Link-layer host mobility solutions (e.g. UMTS, GPRS, and IEEE 802.11) are standardized solutions. In the 3rd Generation Partnership Project (3GPP), standardization work is being carried out on the architecture for WLAN and 3G system interworking [3]. Link-layer solutions are low overhead and suitable for movement within an administration domain. However, they are not feasible for larger movement across domains.

As the most widely studied approach to host mobility handling, Mobile IP [4, 5] manages terminal roaming at network layer. Based on the extension to IPv4 or IPv6, Mobile IP provides transparent mobility support to application by tunneling all the traffic of the mobile host through a home agent at its home network. Mobile-IP-based vertical handoff [6, 7] can be used for heterogeneous connections as in WLAN and GPRS integrated networks. By sending new filter information to the home agent, Mobile IP can be used further to support multiple simultaneous interfaces and data flow control as well [8]. All the network layer solutions, including Mobile IP, significantly rely on newly introduced network infrastructures such as the home agent, which in practice makes the deployment difficult.

As for the transport-layer solution, an end-to-end host mobility solution named Migrate was proposed in [9]. From the application point of view, a transport-layer solution means that the opened sockets remain the same during the host movement. Migrate is based on the extensions of TCP and DNS, which means some changes have to be made to the primary protocols of the Internet. There are some proprietary solutions at transport layer as well, including, for example,

MSOCKS+[10] and IBM Connection Manager [11]. These solutions are dependant on the introduction of a special-purpose proxy/gateway. The intermediate proxy and gateway also break the semantic of the end-to-end application connection of transport layer.

There are also some upper-layer solutions targeting to host mobility. Host Identity Payload (HIP) [12] is a solution at the mid-network-transport layer (3.5 layer), in which a new name space is introduced for host identification. This extra layer means a significant change to the existing Internet protocol stack. Application-layer solutions focus mainly on SIP with mobility support [13], which can be used to support real-time multimedia applications. Moreover, a mobility solution at the application level can also address the problem of personal and service mobility. However, an SIP-based solution is too protocol-specific, with potentially higher performance overhead.

This article presents connectivity management middleware (CMM), a software platform for adaptive network applications based on connectivity context awareness. Compared with the research described above, our CMM-based solution manages dynamic network status with middleware architecture, through the creation and maintenance of communication channels. In the next section, the overall architecture of CMM is presented. We then describe the methods for connectivity context awareness and channel-based adaptive connectivity management. The prototype implementation and experimental results are also presented in detail.

ARCHITECTURE OVERVIEW

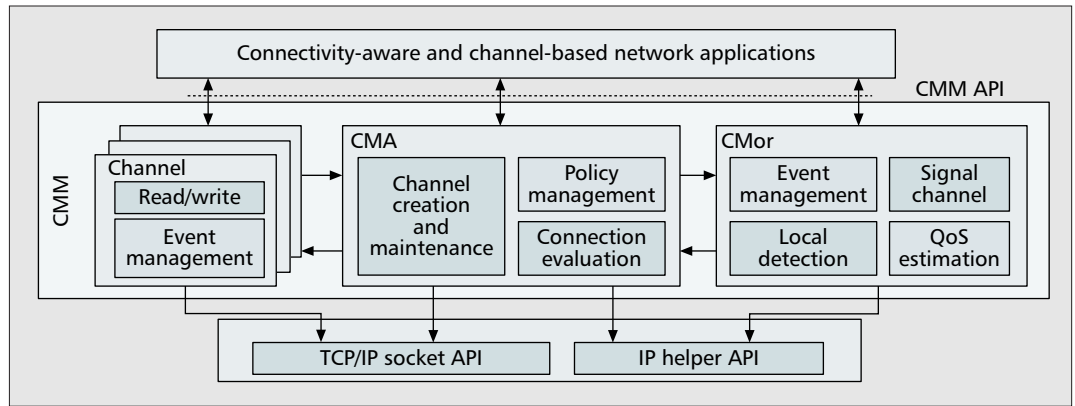
CMM is a middleware architecture built upon network and transport layers, as illustrated in Fig. 1. The execution environment includes adaptive applications as the consumers of the CMM’s networking services, and an underlying infrastructure that serves as a necessary support module. Consumers are mainly end-service applications, but can also be other system-level platforms at the middleware layer, for example, file-access systems. The infrastructure consists of the components performing monitoring, control, and query, and provides physical communication services to CMM.

CMM provides flexible support to adaptive network applications with a set of functions and interfaces, including both connectivity context awareness and adaptive connectivity management. In the case of connectivity context awareness, CMM provides interfaces for applications through a connection monitor (CMor). The CMor is responsible for the collection of local and remote transient-network information. Simple APIs are provided so that network information can be explicitly queried by applications. CMor also provides interfaces for applications to subscribe connection events and receive notifications on their occurrence.

Adaptivity in connectivity management is achieved by the channel management agent (CMA). The functionality is realized by the creation and maintenance of channels. A channel is a logical end-to-end link between physical appli-

CMM provides flexible support to adaptive network applications with a set of functions and interfaces, including both connectivity context awareness and adaptive connectivity management. In the case of connectivity context awareness, CMM provides interfaces for applications through a connection monitor.

CMM provides the flexibility for existing legacy applications and any other applications to bypass the CMM functions without losing any features. In other words, CMM is provided on demand instead of being mandatory. Therefore, CMM is more flexible and efficient than a Mobile-IP-based scheme.



■ **Figure 1.** CMM architecture and execution environment.

network components that are located in different network devices, for example, terminal or server. The end points of a channel are the application processes instead of socket ports. The channel can always use the best currently available local connection to transfer data. CMA provides a policy-based mechanism to evaluate the best connection for each channel at a given moment.

When creating a channel, if a mobile device is equipped with multiple wireless connections, then the best available connection will be selected for the creation. During I/O operations, the mobile device may move into or out of the overlapping area of multiple accesses, or to another access subnet. In this case, the connection used by the channel can be seamlessly switched without necessarily being perceived by the application, so as to ensure that the best connection is always used at any time. When a connection is temporarily unavailable, CMA can suspend the channel operations and wake it up later when a new connection is available.

The application may explicitly control each channel if necessary, but in most cases the control is made implicitly by the policy mechanism. CMA's policy management module provides interfaces for applications to express their connectivity requirements. The policy mechanism allows rich context information to be taken into consideration, including user preference, device capacities, network condition, and application features. Thus, the application is able to focus on its functionalities while the detailed processing of the adaptation demands are left to CMA.

The CMM architecture partitions all the adaptive functionalities into different levels, with each level taking care of the functions that are most suitable to be concerned with. The adaptation of the end application is separated into an application layer and a middleware layer. All the network adaptation mechanisms are abstracted and placed onto the middleware level, since the monitoring and control of network resources are mostly convenient to be implemented at this level. Semantically oriented adaptation mechanisms are left to the application level. This is because the application knows best the content and the media that it consumes and processes.

It is worth noting that in the proposed architecture the adaptive management of multiple network accesses is done at channel level instead

of the device level. Hence, at a moment, any application thread in one user terminal can utilize its own channel by relying on a particular network access. In this way, simultaneous usage of multi-access networks to one user device is possible. For example, while an application is downloading content from the Web using a WLAN, a message application can start to use GPRS simultaneously.

In practice, CMM can be installed in any end device such as a mobile user device or fixed server. CMA is always presented to any host, while CMor is optional and has to be installed only in multi-access enabled mobile devices. No change is needed to any existing protocols (e.g., TCP, UDP, IP, etc.). Moreover, CMM is a total end-to-end solution. There is no modification to the existing network infrastructure and no newly introduced network equipment. Furthermore, CMM provides the flexibility for existing legacy applications and any other applications to bypass the CMM functions without losing any features. In other words, CMM is provided on demand instead of being mandatory. Therefore, CMM is more flexible and efficient than a Mobile-IP-based scheme where the generality of transparent mobility support becomes too expensive.

CONNECTION MONITOR

From the viewpoint of applications, two different methods can be used to obtain corresponding network status and condition information: *explicit query* and *event subscription*.

CONNECTION INFORMATION QUERY

CMA as well as the applications can explicitly inquire network information with APIs provided by CMor. There are both host connectivity context and end-to-end network condition context information. Host network information describes all available network interfaces. CMor supports the retrieval of both local and remote host information. In CMor's implementation, network connectivity information is obtained by invoking underlying IP Helper APIs. CMor can report:

- The amount of network interfaces. A modern mobile device may be equipped with multiple network interfaces, e.g. Bluetooth, IrDA, modem, Ethernet, WLAN, GSM, GPRS, etc. Local loop interface is not concerned.

- Connection list, consisting of connection indexes and corresponding connection names. The connection index serves as the unique identifier of each connection. This index is then used to access detailed connection information, if needed.
- Configuration information of each network interface, including name, type (e.g. PPP, Ethernet, Token Ring, etc.), IP address, gateway IP, and speed. This is obtained with a connection index as a parameter.
- Operation status of each network interface, for example, available, operable, connecting, connected, idle, sleeping, transmitting, receiving, unconnected, unreachable, disabled, and so forth.
- Packet statistics information, that is, the number of received, sent, discarded, and dropped IP packets on each network interface.

End-to-end network context information includes round-trip time (RTT), upload bandwidth, and download bandwidth. In CMor, a UDP ping echo procedure is used to estimate network delay. Cooperation between peer CMors is necessary for bandwidth estimation, which is achieved through a signal channel. Each local CMor opens an UDP socket at the beginning of its execution, and then uses it to exchange signals and data. Apart from the messages and data used for end-to-end bandwidth estimation, other signals and data include remote-connection information query and response messages, and remote-connection event subscription and notification messages. The signal channel is open to be used to exchange any further information between peer CMAs including, for example, application-specific signals or channel policy-negotiation messages.

CONNECTION EVENT SUBSCRIPTION

CMa and the applications use CMor's event-driven method to subscribe network events in order to be informed of their occurrences. This mechanism can be accessed simply by invoking a method with the event-type string as an input parameter. Event notification is made through a call-back method with event type and related event information as output parameters.

By default, CMa subscribes all the connection events from CMor on behalf of the channels and notifies all channels when the events occur. Channel can then decide its behavior accordingly. CMor allows the subscription of the following three connection events:

- A "connection up" event occurs when a new connection is available (i.e., a new network interface is available and assigned with an IP address), for example, due to plugging in a PCMCIA card or moving into a particular area that the new wireless access covers. Connection switching due to a "connection up" event is called "upgrade switching."
- A "connection down" event happens when an existing connection is not available any more, for example, due to pulling out an adapter card or moving out of a coverage area. Connection switching due to a "connection down" event is called "downgrade switching."

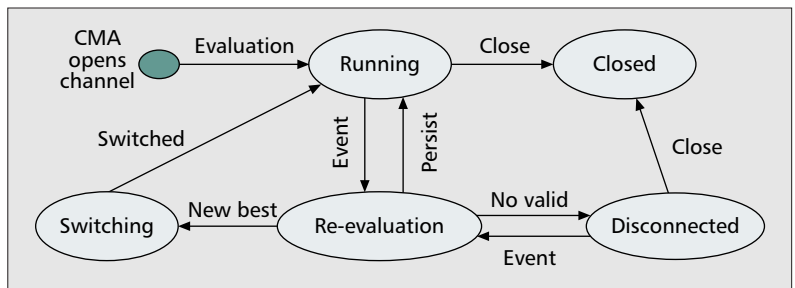


Figure 2. Channel execution states and transitions.

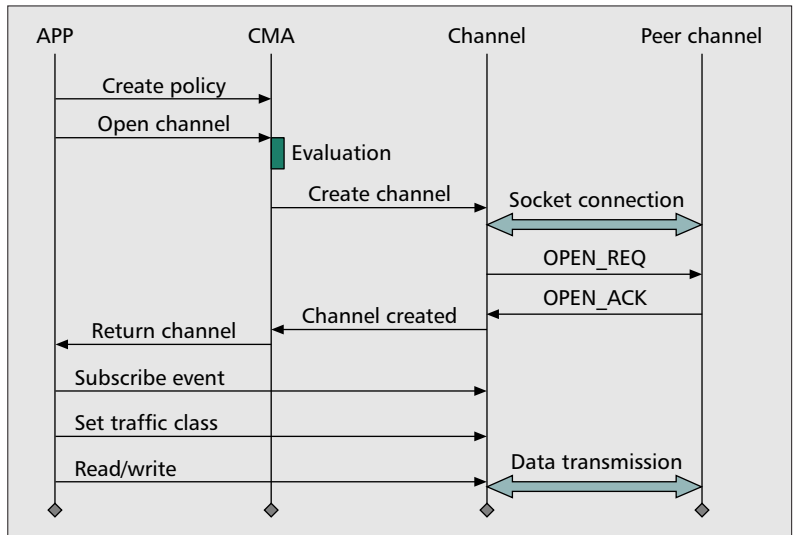


Figure 3. Channel creation diagram.

- An "address changed" event happens when the IP address assigned to a network interface has been changed, for example, due to an intradomain handoff between access points.

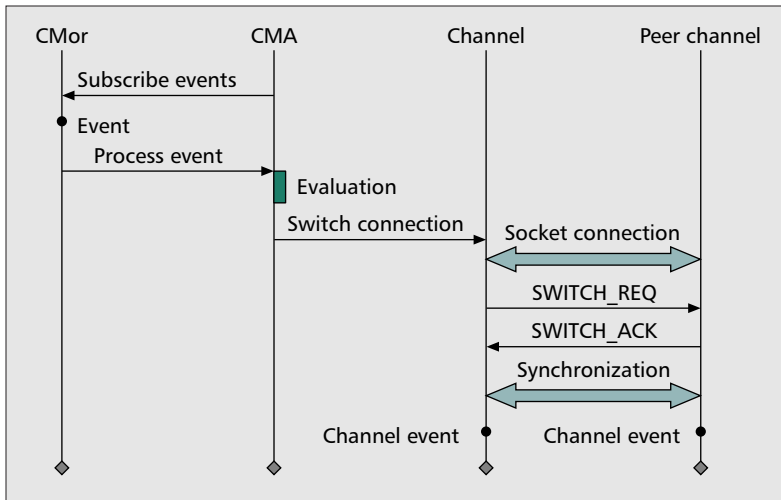
For remote event subscription and notification, the remote host name needs to be provided as one parameter. Request messages, event parameters, and event notifications are transmitted between peer hosts through the signal channel.

CHANNEL MANAGEMENT AGENT

CHANNEL OPERATION

The core adaptation mechanisms for connectivity management are mainly realized by CMa, through adaptively maintaining a channel's life cycle. Figure 2 illustrates the execution states of a channel as well as the transitions between states. Applications first request CMa to open a communication channel. An evaluation process is then launched to select the best currently available local connection to be used for this channel. The channel is then created using the chosen connection, and the state is set at "running." Now the channel can be used for normal data transmission. Figure 3 shows a diagram of the related operations for channel creation.

CMa provides an interface for connection evaluation. The evaluation is the process of finding the best local connection at a given moment by referring a set of context information and



■ Figure 4. Channel connection switching diagram.

input parameters. In our design, a wide set of context information can be considered, including user preferences, application requirements, network condition, and device status. As illustrated in Fig. 2, connection evaluation is made both at channel creation and at run time when connection events occur.

Channel policy can be specified by the application at the moment of channel creation, as a key parameter for evaluation. The policy is then used as context information during the selection of the best currently available local connection to be used. Channel policy is stored for the possible future use during connection switching. Policy can also be changed by the application during the execution. Another important parameter for evaluation is traffic class. The channel traffic class can be set by the application to characterize the feature of packet flow.

Basically, the channel is the layer above transport (socket) layer. So after the socket connection between peer hosts are established, a channel layer signal pair, channel open request (OPEN_REQ) and channel open acknowledgment (OPEN_ACK), are exchanged between peers. If the creation succeeds, CMA adds the new channel into its local maintained channel list (i.e., a Hashtable named *Channel-List*). The state for the channel is then set at *running*. A channel can later be removed from the channel list when it is closed by the application. Each channel contains a private data object (a Hashtable named *ChannelInfo*) to store channel information including channel ID and type, connection index, peer name and IP address, traffic class, channel policy, and so forth.

An application can subscribe to channel events (note: not connection event as in previous section) and receive notification with a call-back function when corresponding events occur. The application can then adjust its behavior accordingly. The two types of channel events are:

- “*Disconnection*” event, which occurs when re-evaluation indicates that there is no valid connection. In this case, the channel enters *disconnected* state.

- *Local switched* and *peer switched* events, which occur when the channel changes local or remote connection, respectively.

CMA subscribes all connection events to CMor by default. When an event occurs, CMA is notified through a call-back method, together with the related event parameters. During the event processing if, by evaluation, no valid connection is available, the channel will enter the *disconnected* state. In this state I/O operation is blocked and the channel will not be woken up for re-evaluation until a new event occurs. If the best connection is still the one under usage, the channel then persists with the connection and keeps running. On the other hand, if a new best connection is found, the channel will initiate a connection switching session in order to seamlessly switch the connection to the new one. These state transitions are illustrated in Fig. 2.

In particular, Fig. 4 illustrates the message sequence for channel connection switching. First, a new socket connection is established above the newly chosen connection. Then a channel-layer signal pair, switch request (SWITCH_REQ) and switch acknowledgment (SWITCH_ACK), are exchanged during the switching operation. Each channel maintains a local buffer and R/W pointers for the purpose of data synchronization during connection switching. During the exchange of switching signal pair, R/W pointers are included into the signal packets. Data are then synchronized by data retransmission before the following I/O operation is performed on the new connection. If the switching succeeds, a related channel event is generated.

POLICY MANAGEMENT

The connection evaluation is based on a policy mechanism. A policy represents the criteria for the selection of the best network interface. CMA maintains the state transitions of each channel on the basis the corresponding policies. Three policy scopes can be defined. A *device level policy* is usually set by the user to express a personal preference on the usage of the network connections of the whole device. At the same time, each individual application can set its own connectivity management rules through *application-level policy*. The policy is shared by all the channels owned by the application. Finally, *channel-level policy* can be set by the application during creation. CMA maintains a policy lists for the user and the applications, and each channel policy is stored by the individual channel.

A policy is a four-tuple:

(*port, traffic class, requirement expression, policy items*)

Policy can be used with a channel that has a concrete port number and/or a specified traffic class. Requirements are set using a logic expression. The expression consists of a series of comparison expressions connected by logic operators. Each comparison expression sets a limitation boundary for a specific connection feature. These features include cost, speed, RTT, up/down bandwidth, power consumption, and memory footprint.

Policy can be either static or adaptive. There are three policy items for static policy:

(use/default, type/index, value),

where a concrete connection type or index to be used is explicitly specified or set as default. One example static policy could be given by

(-, *STREAM*, (speed > 200000), (default, type, *ETHERNET*)).

This static policy specifies that the connection speed for *STREAM* traffic should be above 200 kb/s and *ETHERNET* should be used by default.

Adaptive policy items are a set of (*factor*, *weight*) pairs, in which factors are exactly the same as those features used for the requirements expression. Adaptive policy enables the selection of connection at run-time. Each connection has a normalized value for comparison. One example of an adaptive policy could be

(80, *THROUGHPUT*, (down BW > 100 kb/s) AND (rtt < 5 s), (cost, 0.2)(power, 0.2)(bandwidth, 0.6))

This policy specifies that each connection's value is calculated by multiplying the connectivity's cost by 0.2 and power consumption by 0.2, and the reciprocal of bandwidth by 0.6. The value is the sum of these three terms, and the connection having the lowest value is selected.

The connection evaluation procedure takes the channel policy and information as input parameters. The evaluation is made on each connection according to all user and application policies, as well as the channel policy. When the network condition (e.g., RTT) needs to be estimated with CMor, relative address information can be obtained from the channel information. The requirements expressions are used to filter all the available connections, which leads to a qualified connection subset. Finally, static and adaptive policy items are used to make the real determination. The evaluation is made at the channel basis, that is, each channel provides its own information and policy and makes an evaluation for itself. In this way, simultaneous usage of multiple heterogeneous access wireless networks is achieved.

PROTOTYPE IMPLEMENTATION

Based on the proposed CMM architecture, a prototype has been implemented. The implementation consists of a set of Java classes and a native library. The current Java implementation is based on Sun JDK 1.1.8. The native library is used to access native IP Helper APIs provided by underlying OS, and it interacts with CMor Java class by Java Native Interface (JNI). We have implemented the dynamic-link library (DLL) files for both Pocket PC 2002 and Windows 2K. In treating the case when no policy is specified, a default evaluation procedure was implemented by taking only connection speed into consideration, that is, the network interface with the highest speed was always selected as currently the best one. For WLANs and cellular integrated networks, this default policy means that WLANs will be used as long as possible. Although the default policy is simple, it represents the most typical use case.

According to our design, during channel con-

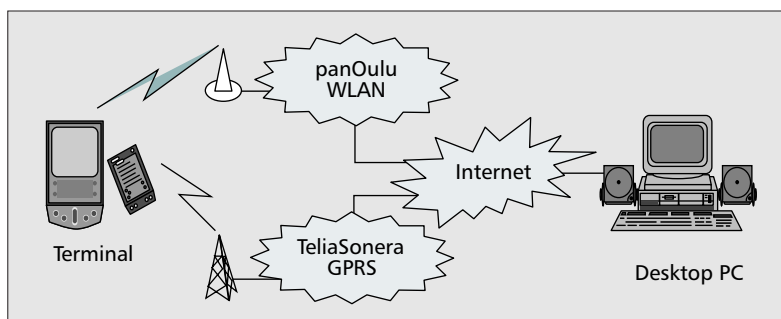


Figure 5. WLAN-GPRS integrated environment.

nection switching, packet loss is avoided via data synchronization. Thus, packet delay is the major performance overhead accompanied by light signal traffic in both channel creation and channel connection switching (i.e., channel-layer signal-pair exchanging). As illustrated in Fig. 3, channel creation time delay, T_{cc} , can be given by

$$T_{cc} = t_{ev} + t_{sk} + t_{sg},$$

where t_{ev} is the time for evaluation, t_{sk} is the time for socket setup, and t_{sg} is the signaling time. Similarly as illustrated in Fig. 4, the upgrade connection switching time delay of client side, T_{usc} , is given by

$$T_{usc} = t_{sg} + t_{sc},$$

where t_{sc} is the time for data synchronization. Upgrade switching delay of server side, T_{uss} , is given by

$$T_{uss} = t_{sg} + t_{sc},$$

where, in this case, t_{sg} is the time for sending channel signal SWITCH_ACK only. Downgrade switching delay for client side, T_{dsc} , is given by

$$T_{dsc} = t_{et} + t_{ev} + t_{sk} + t_{sg} + t_{sc},$$

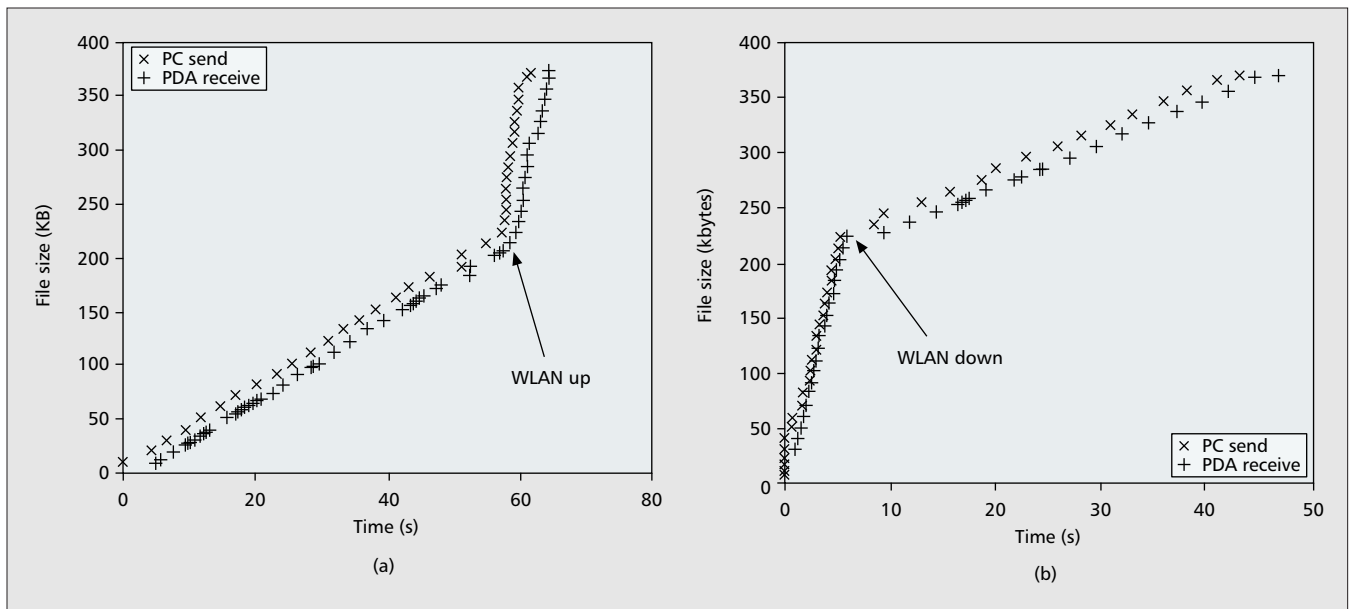
where t_{et} is the time for connection down event detection. At server side, the downgrade switching delay, T_{dss} , is given by

$$T_{dss} = t_{sc} + t_{sg} + t_{sc}.$$

In the next section, numerical results for the metrics are shown.

EXPERIMENTS

A series of experiments were performed using the prototype. The experimental environment consists of a set of user devices interconnected with a desktop PC through two access networks and the Internet, as illustrated in Fig. 5. The major user device was a COMPAQ iPAQ 3970 PDA running Windows Pocket PC 2002. IBM J9 was used as the JVM. A Lucent Orinoco Gold IEEE 802.11b PCMCIA card was equipped for Wi-Fi WLAN wireless access. The panOulu [14] network was used as the WLAN access network. A NOKIA 7650 mobile phone served as a GPRS dial-up modem connected to the PDA through a Bluetooth ad hoc network connection. A Telia-Sonera commercial GPRS connection was used as the GPRS access network. The desktop PC, on a University LAN network (100 Mb/s Ethernet) with a public IP address, was running Win2K and being used as a channel server.



■ **Figure 6.** Delay performance of channel connection switching in live data transmission: a) upgrade switching in download; b) downgrade switching in download.

The experiment was performed so that a channel was opened from the PDA to the desktop PC. Then a MP3 file (363 KBytes) was transferred between the PDA and the PC, both upload and download. A continuous upload and download application was chosen for the experiment so as to stabilize the network utilization. During data transmission, the connectivity situation was manually changed to generate connection events “WLAN up” and “WLAN down.” When a “WLAN up” event occurs, an upgrade switching of the channel connection from GPRS to WLAN is performed, whereas a “WLAN down” event triggers the downgrade switching of the channel connection from WLAN to GPRS. So there were a total of four scenarios concerned in the experiments, that is, both upgrade and downgrade switching in both upload and download, respectively.

The size of the channel buffer is set to be 10 KBytes to fit the high speed of the WLAN connection. A timer was started exactly prior to data transmission. The time and data size were recorded to each data input and output at both the sender and the receiver sides. For each scenario, experiment was run 10–20 times in order to obtain stable numerical values. No physical movement was performed during the experiments. Rather, the network connection was changed manually by plugging/pulling the PCMCIA card in/out of the iPAQ to simulate the movement into/out of WLAN covered area.

Figure 6 shows the download data transmission during connection switching. In Fig. 6a, a GPRS connection was used for file transfer first, and then the connection was switched to the WLAN when the WLAN connection was available at a given moment. In Fig. 6b, when channel creation occurred and both connections were available, the WLAN connection was selected as the connection to be used for the channel. The connection was then switched to GPRS during

the file transfer, at the moment when the WLAN connection was down.

The detailed numerical results are summarized in Table 1. It is easy to understand that the time delay in upgrade connection switching (GPRS to WLAN) is much shorter than in downgrade connection switching (WLAN to GPRS). This is because, in downgrade switching, time is needed for event detection, connection evaluation, and socket re-establishment, in addition to the time needed for two-way signal handshaking and data synchronization. In the duration while the preparation occurs, I/O operation is blocked. While upgrade switching also needs to perform these preparations, real connection switching is only started after all the preparations have been finished.

In our experiment, the numerical results also show that the time for connection evaluation (t_{ev}) and data synchronization (t_{sc}) is very short and thus can be neglected. Therefore, channel creation delay is on the order of socket setup time (t_{sk}) plus RTT time (t_{sg}). Upgrade switching time delay is on the order of RTT time (t_{sg}). Downgrade switching time delay is on the order of socket setup time (t_{sk}) plus RTT time (t_{sg}) and some relatively short time for event detection (t_{et}).

CONCLUSIONS

The architecture of CMM aims to provide adaptive connectivity management to network-aware applications. Mechanisms for the enhancement of both connectivity context awareness and adaptive connectivity management have been discussed in detail in the article. By employing the channel, both disconnection and connection switching were enabled. A policy mechanism was used for the context-aware evaluation of the best connection to be used. Experimental results based on the prototype implementation have shown that the architecture can provide expect-

Experiment case	Numerical result
Channel creation delay	
WLAN	237 ms
GPRS	2532 ms
Upgrade switching (GPRS to WLAN) delay	
PDA sending/PC receiving	95 ms/0 ms
PC sending/PDA receiving	0 ms/98 ms
Downgrade switching (WLAN to GPRS) delay	
PDA sending/PC receiving	3285 ms/ 3315 ms
PC sending/PDA receiving	2473 ms/2579 ms

■ **Table 1.** Numerical results.

ed adaptive management functionalities and the performance overhead is on the order of RTT time for both channel creation and connection switching.

ACKNOWLEDGMENT

Financial support from the National Technology Agency of Finland is gratefully acknowledged.

REFERENCES

- [1] R. Droms, "Dynamic Host Configuration Protocol (DHCP)," IETF RFC 2131, Mar. 1997.
- [2] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)," IETF RFC 2136, Apr. 1997.
- [3] 3GPP, "Feasibility Study on 3GPP System to WLAN Interworking," Tech. rep. 3GPP TR 22.934 v6.2.0, Sept. 2003.
- [4] C. Perkins, "IP Mobility Support for IPv4," revised IETF Internet Draft, available at draft-ietf-mip4-rfc3344bis-01.txt, June 2004 (work in progress).
- [5] D. Johnson, C. Perkins, and J. Arkko, "Mobility Support in IPv6," IETF RFC 3775, June 2004.
- [6] M. Stemm and R. H. Katz, "Vertical Handoffs in Wireless Overlay Networks," *ACM Mobile Networks and Applications (MONET)*, vol. 3, no. 4, 1998, pp. 335–50.
- [7] J. W. Floriu *et al.*, "Seamless Handover in Terrestrial Radio Access Networks: A Case Study," *IEEE Commun. Mag.*, vol. 41, no. 11, 2003, pp. 110–16.
- [8] X. Zhao, C. Castelluccia, and M. Baker, "Flexible Network Support for Mobile Hosts," *ACM/Balzer Mobile Networks and Applications (MONET)*, vol. 6, no. 2, 2001, pp. 137–49.

- [9] A. C. Snoeren and H. Balakrishnan, "An End-to-End Approach to Host Mobility," *Proc. 6th ACM/IEEE Int'l. Conf. Mobile Comp. and Net. (MobiCom)*, Boston, MA, Aug. 2000, pp. 155–66.
- [10] P. Bhagwat, D. A. Maltz, and A. Segall, "MSOCKS+: an Architecture for Transport Layer Mobility," *Comp. Net.*, vol. 39, no. 4, 2002, pp. 385–403.
- [11] IBM WebSphere Everyplace Connection Manager, available at http://www-306.ibm.com/software/pervasive/ws_everyplace_connection_manager/, Nov. 2004.
- [12] R. Moskowitz, Host Identity Payload And Protocol, IETF Internet draft, Nov. 2001, draft-ietf-moskowitz-hip-05.txt (expired); available at <http://homebase.htt-consult.com/~hip/>
- [13] H. Schulzrinne and E. Wedlund, "Application Layer Mobility Support Using SIP," *ACM Mobile Comp. and Commun. Review*, vol. 4, no. 3, 2000, 47–57.
- [14] panOulu public access network, available at <http://www.panoulu.net/>

BIOGRAPHIES

JUN-ZHAO SUN (junzhao.sun@ee.oulu.fi) received his Dr.Eng. in computer science in 1999 from the Harbin Institute of Technology, China. He is a senior researcher at the MediaTeam Oulu Group, University of Oulu, Finland. His research interests are in mobile and pervasive computing, context awareness, middleware, and mobility management.

JUKKA RIEKKI [M] (jukka.riekki@ee.oulu.fi) received his Doctor of Technology degree in 1999, Licentiate of Technology degree in 1993, and M.S.E.E. degree in engineering in 1989, all from the University of Oulu, Finland. Since 1986 he has been a faculty member at the University of Oulu, where he is currently professor of software architectures for embedded systems in the Department of Electrical and Information Engineering. From 1994 until 1996 he was a visiting research scientist in the Intelligent Systems Division of the Electrotechnical Laboratory, Tsukuba, Japan. His main research interests are software architecture and components of reactive (i.e., situated, context-aware) systems.

MARKO JURMU (marko.jurmu@ee.oulu.fi) is an undergraduate student in the Department of Electrical and Information Engineering at the University of Oulu, Finland. He is also working as a researcher at MediaTeam Oulu Group, University of Oulu. His research interests are mobility and connectivity management for heterogeneous wireless networks.

JAAKKO SAUVOLA (jaakko.sauvola@ee.oulu.fi) received his M.Sc. (computer science) and Dr.Tech. degrees in 1994 and 1997, respectively. He is a member of many scientific organizations and is currently a professor and director of the MediaTeam Oulu (<http://www.mediateam.oulu.fi>) research group and the Information Networks master-level academic degree program of the University of Oulu. He is Board Chair of the Mobile Forum (<http://www.mobileforum.org>), Digital Media Service Innovations (DIMES), for multichannel and mobile technology research and related business development. He is also a senior director at Nokia Strategic Architecture, for new technology, architecture, and business development (<http://www.nokia.com>). His interests cover mobility, multimedia applications, advanced software-based communication, and system architectures.

Experimental results based on the prototype implementation have shown that the architecture can provide expected adaptive management functionalities and the performance overhead is on the order of RTT time for both channel creation and connection switching.