

ON EVALUATION OF A NOVEL METHOD FOR ADAPTIVE MANAGEMENT OF HETEROGENEOUS WIRELESS NETWORKS IN MOBILE CLIENT-SERVER COMMUNICATIONS

Jun-Zhao Sun, Jukka Riekk, Marko Jurmu, and Jaakko Sauvola

Department of Electrical and Information Engineering,
P.O.Box 4500, FIN-90014 University of Oulu, Finland

Abstract - A HTTP input channel can adapt its behavior according to the changing network status in a context-aware fashion. The adaptation includes both selecting and dynamically switching the underlying connection used, as well as coping with disconnection situation. This paper describes the architecture and design of HTTP input channel for multi-access wireless network. A series of experiments were performed in a GPRS-WLAN integrated environment. Performance metrics were measured and analyzed. In particular, the time delay of channel creation and connection switching was investigated. For channel creation, the majority of time delay comes from the normal operation of HTTP URL connection. More time is needed for best connection evaluation and default gateway changing when more than one connection is available. For passive switching delay both event detection and connection evaluation contribute to the most besides the normal HTTP connection opening.

I. INTRODUCTION

Next generation mobile communication system, based on the pervasive and ubiquitous computing paradigm, is emerging with heterogeneous wireless access networks as the leading feature. The requirement of persistent connectivity in multiple wireless accesses leads to the problem of effectively managing and synthetically using multiple wireless resources. Nowadays it is common that one mobile device is equipped with multiple network interfaces. New methods are needed to manage multiple access systems so that diverse communication resources can form wireless connections in a simultaneous, collaborative, and complementary way.

HTTP is the key protocol in enabling client-server services. In our previous paper [1] the design and implementation of Java-based HTTP input channel architecture was described. HTTP input channel is an application layer solution to manage multi-access wireless system for mobile devices. It is a unidirectional HTTP URL connection between client and server, in which the network interface under use can be automatically selected during the creation and seamlessly switched during the input operation.

In our architecture a Channel Management Agent (CMA) provides simple APIs to applications for the dynamic management of network connectivity. The application simply gets an HTTP input channel from CMA and uses it to download a remote file from e.g. a public web server. If mobile device is equipped with multiple wireless connections, then the best connection available will be selected for the created channel. During input operations on the channel, the mobile device may move into or out of the overlapping area of multiple accesses. In this case the connection used by the

channel can be seamlessly switched without being perceived by the application. When the connection is temporarily unavailable, the input request to the channel can be suspended and resumed later.

This paper gives a brief introduction on the architecture and design, and focuses on the series of experiments conducted for the quantitative evaluation of the proposed method. The comparison of our method with existing works is also presented in detail. A prototype has been implemented based on Java. A PDA with integrated WLAN and GPRS connections was used as the mobile device. A series of experiments were performed in a GPRS-WLAN integrated environment to demonstrate the operational correctness of our solution. Performance was evaluated according to the collected data. In particular, the time delay of channel creation and connection switching was investigated.

II. ARCHITECTURE AND DESIGN

The proposed architecture is illustrated in Fig. 1. Two enabling components, Connection Monitor (CMor) and Channel Management Agent (CMA), are added into the original TCP/IP protocol stack to enable new HTTP input channel based network applications. In particular, CMor is responsible for the collection of local transient network information. Simple APIs are provided so that network information can be explicitly queried by applications as well as by the CMA. CMor also provides interfaces for the CMA and the applications to subscribe to a particular type of connection event and get notification as it occurs. CMA is the core module for adaptive network connectivity management. The functionality is realized through adaptively maintaining a channel's life cycle.

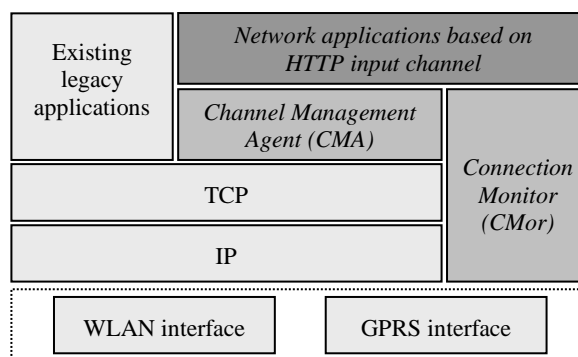


Fig. 1. Overview architecture.

A channel is a logical link between HTTP client and server applications. Each channel uses a specific type of network interface to transfer data from the server to the client. CMA provides a mechanism to re-evaluate the best connection for each channel at any moment. The connection for one channel can then be dynamically switched, while leaving the channel unchanged and so making applications imperceptible. When no network interface can fulfill the requirements, CMA can suspend the operation of the channel and wake it up later when a new connection is available. The application can also get notification when a channel event occurs.

It is worth noting that in the proposed architecture the adaptive management of multiple network accesses is done at channel level instead of device level. Hence, at a moment, any application thread in one user terminal can utilize its own channel relying on a particular network access. In this way simultaneous usage of multi-access networks to one user device is possible. For example while another application is downloading content from the web using WLAN connection, a message application can start to use GPRS simultaneously.

We adopt a pure client-side solution, which means no special change is needed at HTTP server side. This greatly eases the deployment and reduces the changes needed to existing applications. To utilize the provided functions, extra software components need only be installed in the mobile device. Moreover, there is no change needed on existing protocols, e.g. HTTP, TCP/UDP, and IP. There is no influence to existing legacy applications. The APIs provided by CMA are simple and operations to HTTP input channel are very similar with HTTP URL connection interfaces. This makes it very easy to modify existing applications to utilize adaptive management functions.

Fig. 2 illustrates the overall class diagram. Detailed information about the implementation of CMor and CMA can be found in [1]. In our implementation, both CMor and CMA also provide other functions. For example a remote event can be subscribed to CMor and an internal signal channel is maintained, whereas CMA supports the creation and maintenance of other channel types like TCP channel [2]. These functions are out of the scope of this paper.

III. EXPERIMENTS

A prototype has been implemented according to the proposed architecture. The implementation contains a set of Java classes and a native library. The current Java implementation is based on Sun JDK 1.1.8. This is to ensure the best compatibility because most Java Virtual Machines (JVMs) for mobile devices are still at their early versions. The native library is used to access native IP Helper APIs and it interacts with CMor Java class by Java Native Interface (JNI). We have implemented and tested the Dynamic-Link Library (DLL) files for both Pocket 2002 and Window 2000. Experiments were performed with the implementation to demonstrate the operational correctness of the architecture as well as to measure and analyze performance metrics.

A. Experimental setup

The test environment consists of a user device (PDA) and a public web server interconnected through two access networks and the Internet, as illustrated in Fig. 3. The user device is COMPAQ iPAQ 3970 running Windows Pocket PC 2002. IBM J9 is used as the JVM. A Lucent Orinoco Gold IEEE 802.11b PCMCIA card was equipped for Wi-Fi wireless access. We used panOulu [3] as the WLAN network. A NOKIA 7650 mobile phone serves as a GPRS modem connected to the PDA through Bluetooth ad-hoc connection. TeliaSonera commercial GPRS connection was used.

The experiment is performed so that the user downloads a MP3 file (372 Kbytes) from the public web server to PDA. The URL of the music file is at:

<http://www.mwscmp.com/sounds/mp3/finland.mp3>.

Buffer size is set to be 20KBytes to fit the high speed of the WLAN connection. Seven scenarios were used in the experiment, including

0. Plain implementation based on HTTP URL connection without any channel functions supported. This scenario is just for reference.
1. Both of the two access networks are available and no connection event occurs, to simulate the situation of download at the hotspot.
2. Only WLAN access is available to the end, to

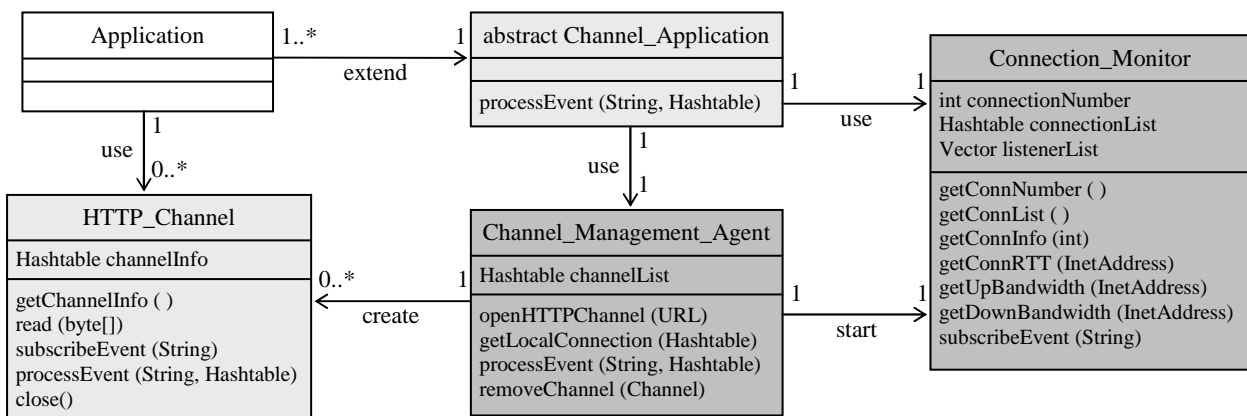


Fig. 2. Overall class diagram.

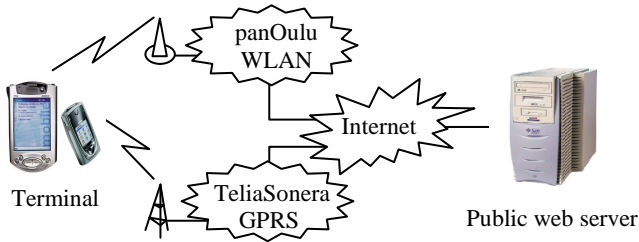


Fig. 3. WLAN-GPRS integrated environment.

3. Only GPRS access is available to the end, to simulate the scenario of download with no WLAN.
4. Both of the two access networks are available at the beginning and WLAN is lost a moment later, to simulate the case of moving out of the hotspot.
5. Only GPRS access is available at the beginning and WLAN access is available a moment later, to simulate the scenario of moving into the hotspot area.
6. Only one connection is available at the beginning, then down when download is on progress. After a while one connection (a new one or still the old one) is up. This is for testing the disconnection process.

A timer is started after the channel is opened. Then for each data input, the time and data size are recorded. For each scenario, the experiment is run 10-20 times in order to get 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.

B. Experimental results

Table 1 summarizes the results from reference Scenario 0. The rest of the tables will be explained later in this section. As a basic HTTP connection with no channel functions is used, it is clear that during the data transmission both “*connection up*” and “*connection down*” events of another connection have no effect to the ongoing download, while “*connection down*” event occurring to the currently used connection leads to download failure. If both accesses are available at the moment when HTTP connection is created, the first available one will

be used for download. This is because the system automatically selects it as the default one. Obviously there is no adaptation in the plain implementation.

Table 2 summarizes the result values for HTTP input channel creation, with plain HTTP URL connection as reference. Small overhead (0.4 to 0.6 s) occurs due to the operations of connection evaluation and default gateway changing. When both WLAN and GPRS are available, the WLAN connection will be used for the channel with time overhead a bit more than in the case where only WLAN is available. This is because more time is needed for the evaluation and gateway operation when more than one connection is available.

Fig. 4(a) indicates the results of Scenarios 1 to 3 in HTTP channel experiments. In Scenario 1, WLAN connection was selected with a total download time of 8.72s. In Scenario 2, WLAN was used with a total time download of 8.34s. From the figure it can be noticed that there is no obvious difference between the two scenarios. In Scenario 3, GPRS was used with total time of 90.81s.

In Fig. 4 (b) the results of Scenarios 4 and 5 are shown. The numerical values are summarized in Table 3. For Scenario 4, the figure shows the switching from WLAN to GPRS. Packet delay time (time between last packet through old connection and first packet through new connection) is 5.41s. In Scenario 5, connection switching from GPRS to WLAN occurred. Packet delay was 0.83s.

It is easy to understand that to HTTP channel active connection switch (switch due to “*connection up*” event) is quicker than passive connection switch (switch forced by “*connection down*” event), because detecting a connection event at network level takes a relatively long time. During the period the original connection can still be used if available. Another reason for this is that in our experiment WLAN access network has shorter setup time than GPRS. Results show that to HTTP channel connection switching, time delay is on the order of access setup time.

Finally, in Scenario 6, a series of experiments was run. Results showed that no matter whether the up connection was still the old one or a new one, the download always resumed.

C. Time delay

Since channel creation and packet delays are the major overhead of our design, and they greatly affect real-time

Table 1 Reference scenario results

Experiment case	Result
GPRS in use when WLAN up	GPRS in use
GPRS in use when WLAN down	GPRS in use
GPRS is first available	GPRS is used
GPRS is down when used	Download fails
WLAN in use when GPRS up	WLAN in use
WLAN in use when GPRS down	WLAN in use
WLAN is first available	WLAN is used
WLAN is down when used	Download fails

Table 2 Channel creation time

Solution	WLAN	GPRS	Both
Plain HTTP	0.74 s	4.83 s	N/A
Channel	1.34 s	5.17 s	1.73 s

Table 3 Channel connection switching time

Switching	Packet delay
WLAN to GPRS	5.41 s
GPRS to WLAN	0.83 s

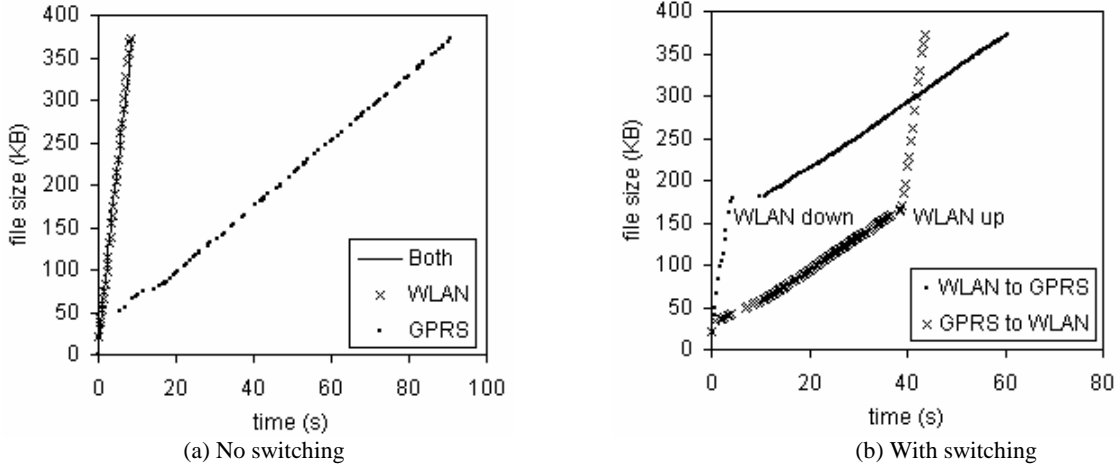


Fig. 4. HTTP channel experiment results.

applications, more data is needed to understand what really happens during the channel creation and connection switching. In order to investigate the behavior behind channel operations in more detail, probes were put into the implementation to obtain the time durations of time delay.

The time for channel creation, t_{CC} , can be expressed as:

$$t_{CC} = \text{SUM}(t_{ET}, t_{GW}, t_{CO}), \quad (1)$$

where t_{ET} is the time for evaluation, t_{GW} is the time for default gateway changing, and t_{CO} is the time for normal HTTP connection opening. Similarly the time for passive channel connection switching, t_{PCS} , can be represented as:

$$t_{PCS} = \text{SUM}(t_{ED}, t_{ET}, t_{GW}, t_{BR}, t_{CO}), \quad (2)$$

where t_{ED} is the time for “connection down” event detection and t_{BR} is the time for buffer reading. The time for active channel switching, t_{ACS} , can be represented as:

$$t_{ACS} = \text{SUM}(t_{BR}, t_{CO}), \quad (3)$$

Detailed time durations for each delay time are listed in Table 4. For channel opening, the majority of time delay comes from the normal operation of HTTP URL connection. More time is needed for best connection evaluation and default gateway changing when more than one connection is

available. For passive switching delay both event detection and connection evaluation contribute to the most besides the normal HTTP connection opening. For active switching, only a little amount of time is needed for old buffer reading.

VI. DISCUSSION

Our solution on adaptive HTTP input channel is closely related to the research of mobility management for mobile Internet. Lots of research has been carried out on this area. Various solutions are derived according to different protocol layers at which the mechanisms are implemented. Compared with the existing research, our solution focuses on HTTP protocol and manages dynamic network status with middleware architecture by the creation and maintenance of communication channels. Most of the related works focused on IP [4, 5], TCP [6, 7], and SIP [8, 9] protocols. To our knowledge, this is the only work that enables mobility for HTTP protocol. In the architecture, network situations taken into account include mobility among both homogeneous and heterogeneous accesses as well as temporary disconnection.

Most of the existing solutions require the existence of some intermediate network equipment in order to support location updating and packet routing. In Mobile IP [4, 5] based solution home agent and foreign agent are needed for this purpose. A redirect server is needed by SIP based solution [8, 9] to register the current location of a mobile user. Solutions of [6, 10] requires a gateway to be installed in the user’s home domain. In [7] an end-to-end solution was proposed, while dynamic DNS should be employed in the user’s ISP. Nearly all the related works need some extensions to the original protocols. In the proposed architecture a pure client-side solution is adopted for HTTP input channel. There is no change needed for the original HTTP protocols and no patch needed for HTTP server side. Thus deployment is easy by only installing a simple software component to the user device. There is no effect to the existing system and applications. Moreover, because the provided APIs are very

Table 4 Time duration results

Delay	Time pieces	Value (ms)
t_{CC} (WLAN/GPRS/Both)	t_{ET}	278 / 271 / 494
	t_{GW}	258 / 242 / 397
	t_{CO}	808 / 4664 / 843
t_{PCS} (WLAN to GPRS)	t_{ED} (down)	425
	t_{ET}	564
	t_{GW}	343
	t_{BR}	173
	t_{CO}	3908
t_{ACS} (GPRS to WLAN)	t_{BR}	177
	t_{CO}	601

simple and similar to those of HTTP URL connection interfaces, it is straightforward to enhance the existing legacy applications to utilize channel-based adaptive functions.

In coping with a multi-access issue most of the solutions manage mobility at device level by vertical handoff [11, 12] mechanism, which means that only one network interface can be used at a time. In case of simultaneous multi-access, there are solutions at network and transport layers. Basically network layer solutions [13, 14] need some additional binding update packets sent to the home agent, and then the home agent serves as the filter to traffic stream. Transport layer solutions [6, 15, 16] lead to the development of new protocols that support multi-home and flow partitioning. The proposed architecture enables the adaptive management of multi-access system on the granularity of channel instead of device, which means that multiple (most possibly heterogeneous) connections can be simultaneously used by one user device for different channels. From the viewpoint of applications, channel is the object under operation. Each individual channel may select to utilize the favourite connection of its own based on a set of context information.

The architecture is designed as a platform to easily involve other channel types. This can be done simply by adding a new *openChannel()* method into the CMA. All the other functions, e.g. connection monitoring and evaluation, can be shared. Actually we have already implemented TCP channel on the basis of the functions provided by CMor and CMA.

Java was chosen as the implementation language due to its platform independent feature. This greatly enhances the portability of the architecture. Current implementation of the CMor depended on a native DLL file that was implemented in C++. The DLL file was loaded by CMor's Java code through JNI and used to access native IP Helper APIs. Two versions of native library were available for Pocket 2002 and Window 2K. The implementation was not pure Java mostly because of the early version of JVM for mobile devices. Sun's J2SE 1.4.0 and later versions have already provided a class named *java.net.NetworkInterface* by which the name and address of each network interface can be obtained. With the evolution of JVM for mobile device, it is reasonable to expect that the proposed architecture can be implemented by pure Java.

V. CONCLUSION

The proposed architecture aims to provide adaptive connectivity management to HTTP-based network applications. By employing HTTP input channel, both disconnection and connection switching are enabled. Experiments show that there is no packet loss during connection switching and packet delay is on the order of access setup time for HTTP channel. Currently CMA supports the creation of HTTP input channel and TCP related channels. While our design provides an open platform for extension, we are extending the CMA so that more channel types will be supported later, including e.g. connected UDP channel as well as virtual port based TCP related channels.

Moreover, rich context information will be taken into consideration to achieve more intelligent evaluation of the best connection to be used for a channel.

ACKNOWLEDGMENT

Financial support by National Technology Agency of Finland is gratefully acknowledged.

REFERENCES

- [1] J.Z.Sun, J.Riekk, M.Jurmu and J.Sauvola, Design and implementation of Java-based HTTP input channel for integrated WLAN and GPRS networks. *12th IEEE International Conference on Networks*, Singapore, 2004, 258-262
- [2] J.Z. Sun, J. Riekk, M. Jurmu and J. Sauvola, Channel-based connectivity management middleware for seamless integration of heterogeneous wireless networks, *Proc. 2005 International Symposium on Applications and the Internet*, Trento, Italy, 2005, 213-219
- [3] panOulu public access network, www.panoulu.net
- [4] C. Perkins, IP mobility support, IETF RFC2002, Oct. 1996.
- [5] D. Johnson, C. Perkins, J. Arkko, Mobility support in IPv6, IETF Internet draft, draft-ietf-mobileip-ipv6-24.txt, June 2003.
- [6] P. Bhagwat, D.A. Maltz, A. Segall, MSOCKS+: an architecture for transport layer mobility, *Computer Networks* 39 (4) (2002) 385-403.
- [7] Alex C. Snoeren, Hari Balakrishnan, An end-to-end approach to host mobility, *Proc. 6th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Boston, Massachusetts, United States August 2000: 155-166.
- [8] H. Schulzrinne, E. Wedlund, Application layer mobility support using SIP, *ACM Mobile Computing and Communications Review* 4(3) (2000) 47-57.
- [9] N. Banerjee, W. Wu, K. Basu, and S.K. Das, Analysis of SIP-based mobility management in 4G wireless networks, *Elsevier Journal of Computer Communications*, 27 (2004) 697-707.
- [10] Nikos A. Nikolaou, Konstantinos G. Vaxevanakis, Sotirios I. Maniatis, Iakovos S. Venieris, Nicholas A. Zervos, Wireless convergence architecture: a case study using GSM and wireless LAN, *ACM/Baltzer Mobile Networks and Applications (MONET)* 7 (4) (2002) 259-267.
- [11] M. Stemm, R.H. Katz, Vertical handoffs in wireless overlay networks, *ACM Mobile Networks and Applications (MONET)* 3 (4) (1998) 335-350.
- [12] J.W. Floriu, R. Ruppelt, D. Sisalem, J.V. Stephanopoli, Seamless handover in terrestrial radio access networks: a case study, *IEEE Communications* 41 (11) (2003) 110-116.
- [13] H. Soliman, K.E. Malki, C. Castelluccia, Per-flow movement in MIPv6, IETF Internet draft, draft-soliman-mobileip-flow-move-01.txt, November 2001.
- [14] X. Zhao, C. Castelluccia, M. Baker, Flexible network support for mobile hosts, *ACM/Balzer Mobile Networks and Applications (MONET)* 6(2) (2001) 137-149.
- [15] P. Nikander, J. Lundberg, C. Candolin, T. Aura, Homeless Mobile IPv6, IETF Internet draft, draft-nikander-mobileip-homelessv6-01.txt, February 2001.
- [16] M. Riegel, M. Tuexen, Mobile SCTP, IETF Internet draft, draft-riegel-tuexen-mobile-sctp-01.txt, August 2002.