Split-Proxy Concept for Application Layer Handover in Mobile Communication Systems

Jens Kammann, Tim Blachnitzky
German Aerospace Center (DLR), Institute of Communications and Navigation
P.O. Box 1116, D-82230 Wessling, Germany
Jens.Kammann@dlr.de, Fax: +49 8153 281871

Abstract— **This paper presents an HTTP based approach for handover within and between heterogeneous communication systems. Therefore a Split-Proxy-Concept has been developed, consisting of client- and server side proxies. Communication between proxies takes place using TCP connections, thus benefiting from existing IP mobility approaches and virtual serial connections as provided by Bluetooth or circuit switched access. The latter allows to forgo a full TCP/IP stack allowing very light-weight implementations on resource limited client devices such as smart phones. Furthermore, the protocol allows payload data caching and retransmission in case of gaps of network coverage. Throughput measurements indicate that the signaling overhead for the inter-proxy communication can be very well compensated by HTTP header and optional payload compression.**

I. Introduction

A growing number of mobile devices offer various types of wireless network access. The range spans from mobile phones with circuit switched or packet based network access over Personal Digital Assistants (PDA) with Bluetooth[1] to small notebooks with Wireless LAN (WLAN), Bluetooth and other options (see Fig. 1).

Today, the user must configure each access method separately and decide on his own when to use which device. This decision is based on a multitude of factors from obvious ones such as network coverage to application specific decisions such as quality of service and throughput requirements in consideration of the cost of usage. Independent on how the initial access method is selected, it is highly desired to maintain network access despite varying network availability with only minimum user interaction.

This paper examines the trade-offs in existing handover approaches of the typical protocol stack and proposes as a new option to perform all connection and handover related tasks on the application layer of a mobile device without modification to the well established lower layers. To keep the overview concise, we limit ourselves to Bluetooth and Wireless LAN as short range communication systems and GSM (GPRS) respectively UMTS in its current ongoing implementation (”Release 99”) as public mobile communication systems. Sample implementations will be on smart phones with JAVA Mobile Information Device Profile (MIDP)[2] support and PDAs, although the protocol itself requires only operating system support for HTTP[3] and access to the available network modules. Although HTTP will be used for handover management, there is no need for a full-blown TCP/IP protocol stack (important for resource limited devices).

II. Mobility and Handover

A typical protocol stack seen at mobile device (see Fig. 2) consists of RF, baseband and network layer for each radio interface and the TCP/IP stack as a basis for applications ranging from mobile Web/WAP access to E-Mail clients and Personal Information Management (PIM).

The intra-system or horizontal handover usually happens transparent to the TCP/IP stack and works well within the existing public mobile networks such as GSM or GPRS providing administrative restrictions do not prevent it (e.g. missing roaming agreements). However, there is no current specification for horizontal handover in Bluetooth nor Wireless LAN, barring a few proprietary implementations from hardware manufacturers. Inter-system or vertical handover is even more critical. Although there has been made efforts in several standardization bodies to specify handover between networks, few are likely to become implemented (why would a mobile operator deploying UMTS want users to perform handover to ISM-band operated Wireless LAN?).

One way out of this is to perform handover at the TCP/IP level. Mobile IP provides a working solution using two IP addresses and a home agent. However, full functionality requires IPv6 which result in significant increase of complexity of the protocol stack. One layer up, TCP is known for poor performance on wireless links and was therefore often subject for suggestions of improvement[4]. M-TCP, I TCP and TCP* - all exhibit improved performance in wireless networks, some also provide partial support for vertical handover. However, as with Mobile IP, significant changes to an existing TCP/IP stack is required.

On the other hand, many mobile applications rely on HTTP for data exchange at the application layer. Applications such as Web-Browser, HTTP-based file transfer, mobile agents exchanging XML encoded messages[5] or even some peer-to-peer file sharing programs may benefit from the proposed Split-Proxy Concept.

III. SplitProxyConcept

Usually an application running on a mobile device communicates directly with an server in the fixed network via

Message	Data	Source	→	Destination
HTTP_REQ	✓	Client	→	Server
HTTP_RESPONSE	✓	Server	→	Client
HTTP_ACK	-	Client	→	Server
STORE	-	Access Point	→	Server
STORE_ACK	-	Server	→	Access Point
RETRIEVE	-	Client	→	Server
RETRIEVE_NACK	-	Server	→	Client
UPDATE	✓	Client	→	Server
UPDATE_ACK	-	Server	→	Client
GET_NEIGHBOR	-	Client	→	Server
NEIGHBOR_LIST	✓	Server	→	Client
ARE_YOU_THERE	-	Client	→	Access Point
YES	-	Access Point	→	Client

TABLE I
MESSAGES BETWEEN PROXIES

the wireless infrastructure. The split-proxy concept as shown in Fig. 4 introduces three proxies in between this link: The first one resides at the mobile devices itself to intercept all outgoing HTTP traffic. In case of an operating system environment without support for server sockets (as in the current implementation of the MIDP found on several smart phones), modifications of existing applications may be required, otherwise application can easily be instructed to use 'localhost' as proxy.

The second proxy is located at the access point, accepting connections from the first one and forwarding it to the next proxy within the backbone, which finally forwards the request to the destination server.

IV. OPTIMIZATIONS

At a first glance this may sound like a huge overhead, but performance measurements with respect to overall throughput and delay have shown only minimal performance degradation because of several options for improvement:

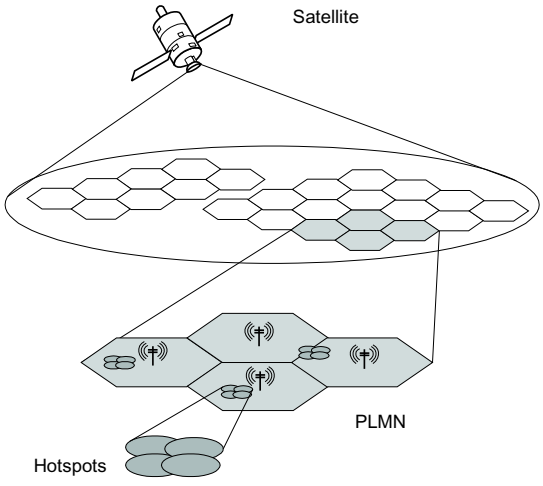


Fig. 1. Network Overlay

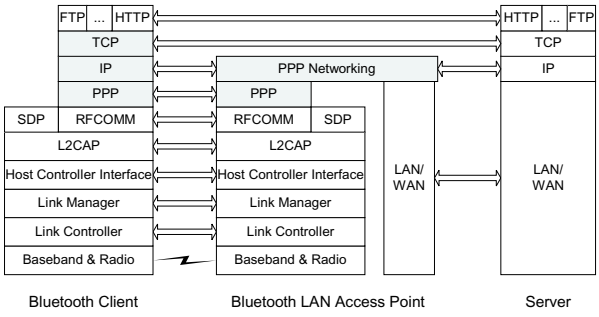


Fig. 2. Bluetooth Protocol Stack

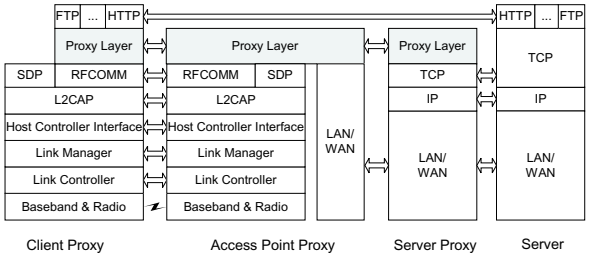


Fig. 3. Protocol Stack with Proxy Layer

- Client- and access point proxy need not to communicate in plain text (HTTP), a binary protocol shows performance gains especially on wireless links with low bandwidth, such as Bluetooth or circuit switched GSM links.
- HTTP Header caching at the client avoids repeated transmissions of the same, often very lengthy header information over the wireless link.
- Mobile devices often still use HTTP/1.0 as transfer protocol. The access point proxy can easily rewrite a request into HTTP/1.1, thus benefiting from several performance enhancing options from persistent connection over chunk transfer to data compression.
- The server proxy could resize and re-encode images. Due to the complexity and potential system load this hasn't been implemented yet.
- The server proxy could prioritize text based content (HTML, Scripting Code) over graphics and multimedia. This allows faster access to the actual content while design and navigation loads in the background. Also within all graphics (e.g. GIF, JPEG, PNG), an intelligent ordering could cause navigation buttons (usually linking within the server) to load before banner advertising (typically point to external servers).
- There is no need for a full TCP/IP stack at the mobile device anymore (see Fig. 3): HTTP is a text-based protocol and therefore only requires a transparent serial link as it is

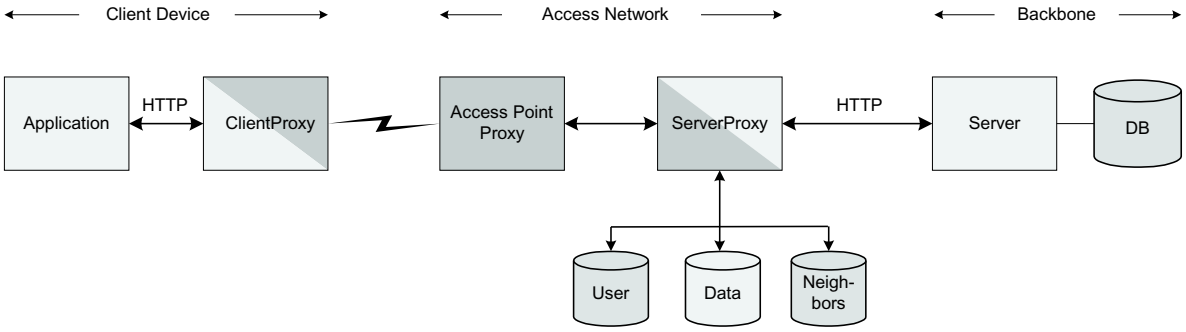


Fig. 4. Split-Proxy Concept

provided by many communication systems. Especially if the usual PPP link can be avoided (e.g. because the underlying baseband and data link layers provide sufficient error detection and correction), overall system performance with respect to data throughput and processor load increases. However, one should note that TCP provides many other features, such as multiple sessions on the same link. If they are required, an implementation on a usually high level language such as JAVA is more inefficient than using the native TCP/IP functions provided by the operating system. Still, sidestepping PPP and TCP/IP altogether remains a promising option, especially on smart phones where simultaneous usage of multiple programs or multiple windows of a web browser are unlikely due to small screen sizes or other device limitations.

V. Implementation and Measurements

A sample implemetation of the proxies was written in JAVA to allow easy porting to various platforms. Bluetooth connectivity is still problematic as there exists a Bluetooth JAVA API[6] for MIDP, but no devices which would support this specification are available on the market presently. Therefore, the test setup consisted of two Laptops with Bluetooth interface cards: One acted as the client running client proxy and user applications, the other one simulated the access point running the access point proxy. The server proxy and the databases were running on different machines in the intranet.

Measurements took place in three steps with small, mixed and large request sizes. First, all proxies were bypassed to evaluate the maximum throughput of the link, then the throughput was measured with all proxies turned on with links established via Bluetooth LAN Access Profile using Point-to-Point Protocol (PPP). Finally, these measurements were compared with the throughput that can be archieved if the link is established using the serial port profile without PPP.

Measurements conducted at our lab show up to 50 % degradation in the worst case scenario (small binary requests with the mobile agent already using HTTP/1.1) to doubled performance in case of large plain text requests. See table II for details. The overall user experience for mobile web-browsing remained roughly the same, as long the user had only one active browser window open at a time, which will be the case on our target devices with small displays.

VI. Handover

The Split-Proxy Concept enables another important feature for mobile data access: Handover.

A mobile device may continuously monitor available network connections and may establish provident links to alternate access points. This is, where the user and content database at the server proxy (see Fig. 4) comes into play: Each client proxy registers with a server proxy its network parameters (IP address, gateway etc.) and capabilities every time a new link becomes available. As long the link remains up, all requests from and to the mobile user passes the server proxy without storing them. If the existing link fails for any reason, the access point proxy (see Fig. 5) instructs the server proxy (see Fig. 6) to store any outstanding data to its content database (usually the responses to previous GET/POST requests). This issue has proven most difficult in the actual implementation as most Bluetooth and WLAN access point provide no information about the current link quality. Often a link just appears to be ”dead” with no data coming across. To mitigate this problem, a time-out mechanism has been implements. However, tuning the time-out value was important to limit its impact on the handover performance in case of small cell sizes.

We discern three points in time when a handover can takes place (see Fig. 7):

- *Proactive:* The handover to the next cell is done before the user leaves the coverage area of the first cell, i.e. both cells overlap.
- *Ideal:* The cells do not necessarily overlap, but the signaling for the handover is done in time so that the handover can be completed without loosing connectivity
- *Reactive:* If the coverage of the cells do not overlap and the requirement for a handover could not be detected while

	Size of Requests					
	large		small		Mix	
	Throughput [kbit/s]	**Overhead** [%]	**Throughput** [kbit/s]	**Overhead** [%]	**Throughput** [kbit/s]	**Overhead** [%]
TCP over PPP (direct)	178.0	—	127.7	—	142.1	—
TCP over PPP (proxy)	177.9	0.07	62.6	5.90	78.1	4.06
virtual serial (proxy)	284.0	0.08	77.2	7.46	97.9	5.14

TABLE II
BLUETOOTH THROUGHPUT (MEASUREMENTS)

being still in the first cell, connectivity is lost. After entering the coverage area of a new cell, lost data need to be retransmitted.

If the handover was done proactively, the server proxy forwards the response through another access proxy to the client. Especially in Hot-spots areas, where Wireless LAN, or Bluetooth is deployed without prior sumptuous cell planning, it is more likely the handover is reactive, i.e. there is some dead time leaving the user without network access. In this case the server proxy stores the content in its database until the same user is able to establish a new link to another access point.

The client proxy can only delay its output for some limited time until a web browser times out with an error message to the user. Warning messages generated by the client proxy to the user usually distorts a graphical web-page, so a user most likely has to reload the entire page which in case of local data caching does not result in much overhead on the wireless link. However, it is expected that the possibilities of client side programming (e.g. MIDP) will result in new applications replacing web browsers and being able to deal smoothly with temporary network failures.

Best performance can be archived, if a short range link such as Bluetooth uses a reliable back up, e.g. using GPRS. Large amount of data may be transferred cost effective via Bluetooth where as the packet based GRPS link is used in case of missing Bluetooth coverage. This scenario usually applies to the new smart phones with combined GPRS and Bluetooth radios.

VII. CONCLUSIONS AND OUTLOOK TO ONGOING RESEARCH

This paper advocates to handle mobility and handover at the application level. Performance degradation can either be compensated or outweigh the flexibility which is gained by supporting multiple networks at a time without having the networks know about each other. One outcome of the current research is that overall performance mainly depends on when and to which network a handover is performed. Therefore, a cost function for each network need to be formulated which allow the handover task to optimize the decision. Data Prefetching[7] is a promising approach to bridge over coverage breaches. Finally, a decentral picture of the network structure is required which allows to manage a neighbor list at each client. This will extend the functionality of the proposed proxies from data caching and forwarding to network management elements, one of which is maintaining the network neighborhood list. Whereas public mobile phone networks usually have a central management, Hot-Spot infrastructure usually does not, except for access points belonging to the same administration domain. In future enhancements, the server proxy could solicit access points for their ”view of the current network”, therefore coping with a dynamic network environment. In fact, every user of a mobile device equipped with the here proposed architecture could help to create a map of the total wireless network by posting it to the server proxy. This way, also the vertical handover between systems becomes more efficient: A client proxy currently using a public mobile network may get instructed to change to a more cost effective local network, even if the satisfactory coverage of the public network would not necessarily call for a handover.

This work has been done as part of the ”Heywow”-Project [8], a research project on developing a m-commerce and travel information platform for users of mobile and resource limited devices.

REFERENCES

[1] Bluetooth Specification (Bluetooth SIG). http://www.bluetooth.com.
[2] JAVA MIDP Specification (Sun Microsystems Inc.). http://java.sun.com/products/midp.
[3] R. Fielding, “Hypertext transfer protocol – http/1.1,” *RFC 2616*, June 1999.
[4] G. Xylomenos, G. Polyzos, P. Mähönen, M. Saaranen, “Tcp performance issues over wireless links,” *IEEE Communications Magazine, Vol. 39 No. 4*, 2001.
[5] T. Strang and M. Meyer, “Agent-environment for small mobile devices,” in *Proceedings of the 9th HP OpenView University Workshop (HPOVUA), June, 2002*, HP, 2002.
[6] “Java apis for bluetooth.” http://jcp.org/jsr/detail/082.jsp.
[7] M. Angermann, “Analysis of speculative prefetching,” *ACM Mobile Computing and Communications Review*, vol. 6, April 2002.
[8] A. Steingass, M. Angermann, and P. Robertson, “Integration of navigation and communication services for personal travel assistance using a jini and java based architecture,” in *Proc. GNSS ’99*, (Genova, Italy), October 1999.

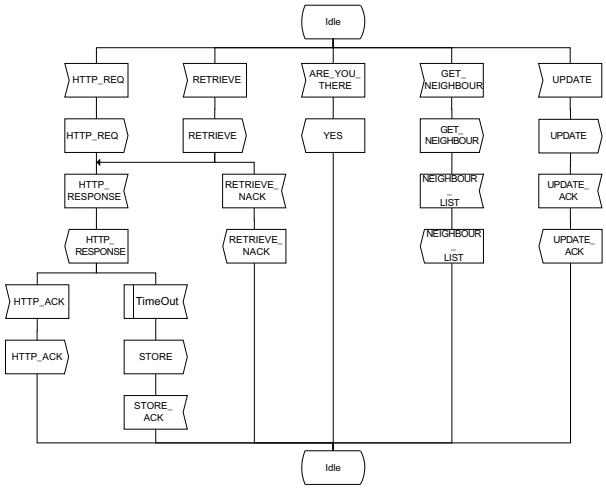


Fig. 5. Access Point State Diagram

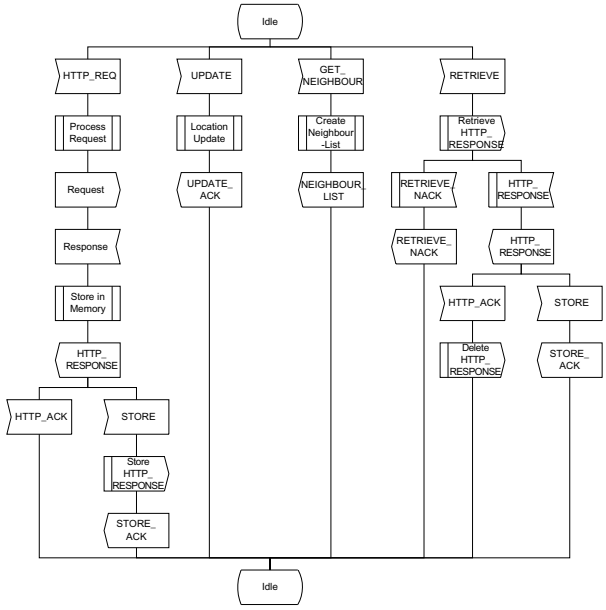
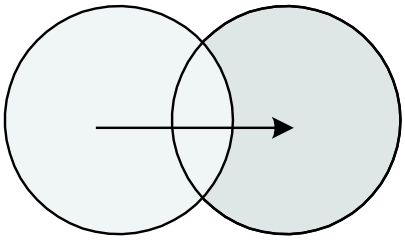
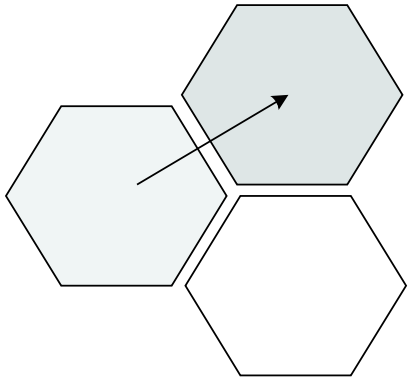


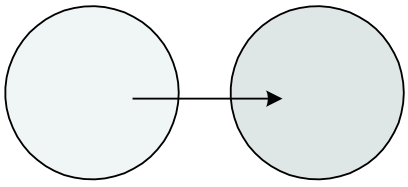
Fig. 6. Server Proxy State Diagram



(a) Proactive



(b) Ideal



(c) Reactive

Fig. 7. Handover - Point in time