LIGHTWEIGHT CLIENT-PULL PROTOCOL FOR MOBILE COMMUNICATION

Stefano Sanna, Emanuela De Vita, Andrea Piras
CRS4 - Center for Advanced Studies, Research and Development in Sardinia
Parco Scientifico Polaris - 09010 Pula(CA) - Italy
Email: gerda@crs4.it, emy@crs4.it, piras@crs4.it

Christian Melchiorre
Softeco Sismat S.p.A.
Via De Marini 1 - WTC Tower - 16149 Genova - Italy
Email: christian.melchiorre@softeco.it

Keywords: Priority management, communication protocol, mobile device.

Abstract: Consumer mobile devices, such as cellular phones and PDAs, rely on TCP/IP as main communication protocol. However, cellular networks are not reliable as wired and wireless LAN, due to both users mobility and geographical obstacles. Moreover, limited bandwidth outside urban areas requires an application level data priority management, in order to improve user experience and avoid communication stack deadlocks. This paper presents early specification and first prototype of the LCPP (Lightweight Client-Pull Protocol), a UDP-based communication protocol specially designed to provide better performance, fast responsiveness and save processing power on mobile devices. Using some concepts adopted in the field of P2P file sharing, LCPP provides data priority management approach, which enables application to negotiate concurrent access to communication channel and to be notified about delaying, network congestion or remote device inability to process data.

1 INTRODUCTION

Mobile devices communication use wireless channel that, to date, cannot be considered as reliable as their wired counterpart. TCP/IP, which is the basis for well-established application-level protocols such as HTTP, has not been designed for such unreliable channels. Using TCP/IP on wireless connection can be extremely boring for users, which continuously get ”connection lost” error messages. On the other hand, cellular network data access is still expensive and users tend to avoid using Internet services on mobile devices, especially if they have experienced poor performances and high connection rates.

Peer-to-peer paradigm (P2P) has emerged in the last years as innovative way to create scalable systems over wired networks. P2P communities have proposed two important concepts: data should be exchanged according to some classification provided by the user and hosts have to find themselves preferably without any central registration facility. Priority management refers to the ability of communication layer to suspend data transfer if any other important data needs to be transferred using the all available bandwidth.

In this paper we describe benefits of priority management in mobile communication and propose a protocol suitable for mobile devices, such as PDA and programmable cellular phones. We refer to mobile communication as network access by means of a GPRS or UMTS terminal (either mobile phone or PCMCIA data card). WLAN access raises other interesting issues, although large bandwidth and restricted operation area give, for most applications, more reliability than cellular network in a mobile intensive scenario.

2 PRIORITY MANAGEMENT IN MOBILE COMMUNICATION

Desktop users are used to perform multiple access to network resources: they read and write emails, while sharing large files and downloading some interesting software. At the same time, background processes update system libraries, check for new virus signatures and gather information about weather, stock quotes and so on. This class of users rely on a wideband connection, either with office LAN or home DSL subscription. With such connections, overall responsiveness of applications is pretty comfortable, although very point-to-point connections may cause

network access locked for a while: it is common, for instance, to lose connection to Instant Messaging (IM) server or to have some delays in sending emails if data transfer has been previously opened to a very fast peer. While this scenario mainly applies to desktop users, mobile device users become to experience similar problems related to concurrent network access. Flat rate per-month subscriptions offered by cellular network operators encourage customers to use PDAs and high-end cellular phones to access web site, mailboxes and adapted content (audio and video). Business oriented applications (remote office and file sharing, email with attachments, simplified editor for text documents, spreadsheets and presentations) and hardware tailored for mobile workers (extended keyboards, large displays, enhanced connectivity) are bundled and offered to selected customers.

Network-oriented mobile applications refer to three main classes: user interactive applications, user browsing applications, system applications. User interactive class identifies applications which response time is critical for user experience. It comprises email clients, IM systems, navigation systems. These applications are expected to have very short delays and give best responsiveness. For instance, IM clients enable people to talk nearly in real-time and most of time is spent reading and typing, while data transfer involves a few data; moreover, since it consist of synchronous communication between humans, user expects continuous dialog with remote peer (response time half a second or less). User browsing class identifies applications which response time is not critical, so that some delay is being tolerated. Video and audio downloading, web browsing and software installation may require some time to be completed and experienced users are aware about it. Finally, system class identifies applications executed in background by the operating system. It includes system updates (bug fixes, security updates), network synchronization and similar.

In an ideal scenario, any system application should suspend any data transfer if a user interactive application requires to send or receive data. This way, the user perceives that communication channel is devoted to his conversation or important web browsing. To simplify, it is better to give the user best performance of device and network connection, while a delay of a few seconds will not affect system operations (such as those previously described). Such an approach can be found in modern operating system design, where process scheduler tries to provide good responsiveness to graphical interface and delaying system processes (such as writing on filesystem).

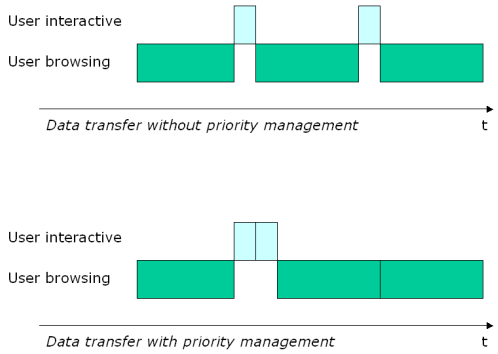

Figure 1: Priority management in mobile communication: user interactive and user browsing applications require different responsiveness.

3 LCPP

Lightweight Client-Pull Protocol (LCPP) has been designed to provide mobile application developers with a priority- and receiver- driven protocol. It has been designed as a lightweight protocol, easy to implement even on resource constrained devices. As said before, priority management is key element for mobile communication; in order to manage data coming from multiple source, client is responsible for managing data packets transmission. We identify peer that sends data as Transmitter (TX), while the one that receives is called Receiver (RX). Main LCPP concepts can be summarized as follows:

- RX drives communication progress: data sender (TX) cannot arbitrarly send any data if it has not be requested by the RX
- all data packets moving from point to point contain a priority information: RX should make requests for higher priority data fragments first
- communication stacks of peers share information about priority management and network congestion.

LCPP relies on UDP, because it is lighter than TCP and requires less resources to be implemented on low-end devices. While TCP/IP offers good infrastructure for continuous connections, it is time and resource consuming for intermittent connections. LCPP does not introduce any custom addressing or routing schema; therefore LCPP headers do not contain any information about source nor destination of data: it is implicitly assumed that such an information will be accessible through the underlying protocol. LCPP uses raw UDP packets and it is responsible for data synchronization, transport and connection control. LCPP defines two type of data packets: Command and Data. Command packets are exchanged

Command ID	Description
REQUEST TO SEND DATA	Sent by a TX that requires to send data to a RX
ACCEPT TO RECEIVE DATA	Sent by a RX that accept to receive data from a TX
REFUSE TO RECEIVE DATA	Sent by a RX that does not accept to receive data
SEND FRAGMENT	Sent by a RX that is ready to receive a new data fragment
GET FRAGMENT	Sent by a TX after a SEND FRAGMENT has been received
ABORT	Sent by a peer to interrupt data transfer (sessions will be reset)
CONFIRM DATA RECEIVED	Sent by the RX to notify that all data have been received
WAIT FOR HIGHEST PRIORITY	Current session is being suspended to process higher priority data
WAIT FOR SYSTEM BUSY	Communication delays caused by local time-consuming processes
WAIT FOR MEMORY BUSY	Communication delays caused by memory-consuming processes
RESUME RECEIVE DATA	A previously suspended data transfer can be recovered

Table 1: LCPP command list.

between peers to manage data transfer setup and request, error reporting, communication closing; they are 18 bytes long and contain: Command ID, priority level, service ID, message, session and reference. The first two bytes contain the Command ID, which identifies what action is being notified to the target (e.g., start transfer, abort transfer, get/send some data fragments...) and the priority level. Remaining header fields contain parameters of current command (such as data size), service ID (that refers to a specific data processor on the RX side), session and session referrer information (used to keep track of correlation between sequence of request/response). Table 1 shows the LCPP command list. Commands are always transmitted and received regardless their priority. In fact, since Commands are very small data fragments and the information they contain may influence overall data transfer policy, they are expected to be processed as soon as possible by peers.

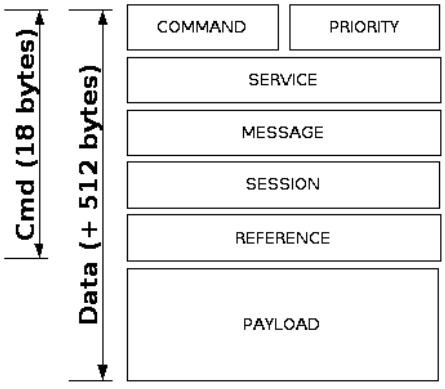

Figure 2: Structure of Command and Data packets.

Data packets are extended Command packets with a 512 bytes payload. The Client-Pull approach refers to the RX side control against data fragments being sent; while TCP allows the server to send several packets within a data window, potentially fulfilling available bandwidth, LCPP server waits for client request (pull) before sending any data packet. Moreover, since every data stream has its own priority, client will ask for higher priority packets first, delaying lower priority ones, assuring that data over channel is always that with highest priority.

Once TX needs to transmit data to RX, it sends a `REQUEST_TO_SEND_DATA` command; by reading the priority field and data size, RX can decide if it is able to process data (that is, since it knows how much data will be transferred it can decide if thereś enough memory - or credit on the SIM card - to get data) and it responds with `ACCEPT_TO_RECEIVE_DATA` command which informs TX that RX has accepted the transfer. However, no data transfer actually starts, until the RX sends a sequence of `SEND_FRAGMENT` commands. How packets are requested depends on RX implementation: TX has only be ready to split and send data as required. The central idea consists on moving intelligence from the protocol itself to the hosts implementing it. Protocol metadata does not provide enough data to understand communication status (while TCP/IP, where such an information can be showed by windows and acknowledgments).

4 IMPLEMENTATION AND TEST

First prototype of LCPP stack has been implemented as a communication library PersonalJava-compliant and has been deployed on iPAQ PDA running Insignia Design Jeode and IBM J9 embedded VMs. Tests have been performed using GPRS cellular network. Although LCPP packet size can be varied arbitrarily, it has been choosen to fit at least a GSM (**?**) time slot (current implementation uses 512B packets, while early alpha releases were using 1KB packets). Preliminary tests have shown very good performance

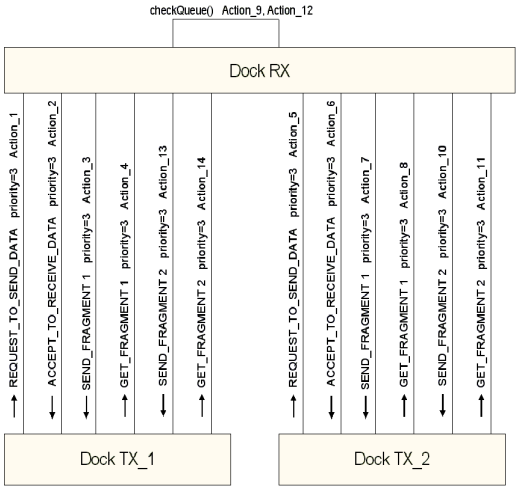

Figure 3: Sequence diagrams of two concurrent data trasfers with different priority.

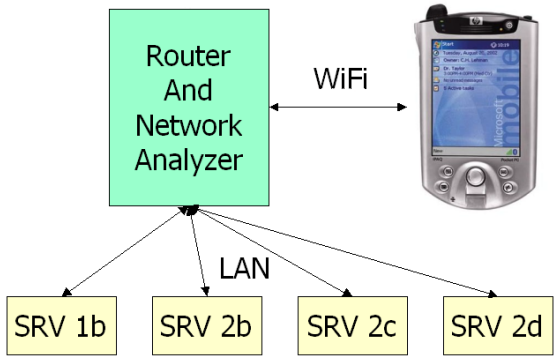

Figure 4: Network configuration used during extensive tests.

for small, spot data transfer (25KB-50KB); performances decrease due to JVM limits on Windows CE device. We plan to rewrite LCPP in C in order to minimize latency caused by VM and Garbage Collector.

GSM network compatibility tests have been performed using PCMCIA and Bluetooth-enabled terminals. These tests have shown that LCPP runs with networks without Network Address Translation (NAT) between operator's network and the public network. Since complete tests on GPRS network is very expensive, analysis LCPP has been tested using a wireless LAN connection. Moreover, in order to avoid to slow down iPAQ performances, we inserted a router node between PDA and servers, equipped with the network analyzer (Ethereal).

We have written a clone of well-known FTP file transfer utility using LCPP protocol stack and compared with standard TCP-based client. Results show that two simultaneous 1MB data transfers over TCP connection share communication channel and bandwidth and the user cannot specify which connection has to take control of communication channel (Figure 5). The same application written using LCPP suspends a running data transfer when the user requires to transfer data with higher priority (Figure 6). Overall performances can be improved with customized RX implementations of more intelligent algorithms for priority management.

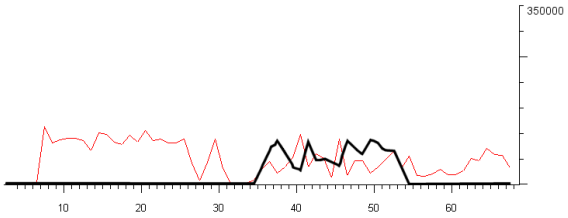

Figure 5: Two concurrent data transfer without priority management (TCP over GPRS).

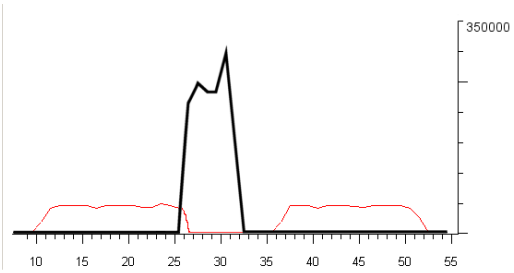

Figure 6: Two concurrent data transfer with priority management enabled: low priority connection (thin line) is suspended until high priority data has been transferred. Priority management affects data transfers only: Command packets are expected to be processed though their priority is lower (LCPP over GPRS).

Performances have been evaluated for two, four and eight concurrent connections (1Mb data transfer each), with different assigned priority. Figure 7 and 8 show results for 4 concurrent connections, with and without priority management support: with priority management enabled . Table 2 shows numeric results: priority management connections require more time to be completed, but highest priority connection data transfer is expected to be completed before any other.

5 RELATED WORKS

Priority management approach has been implemented in the fields of mobile communication, sen-

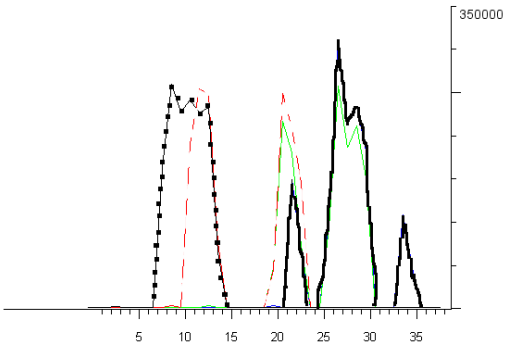

Figure 7: Four concurrent 1MB data transfers without priority management enabled (wireless LAN).

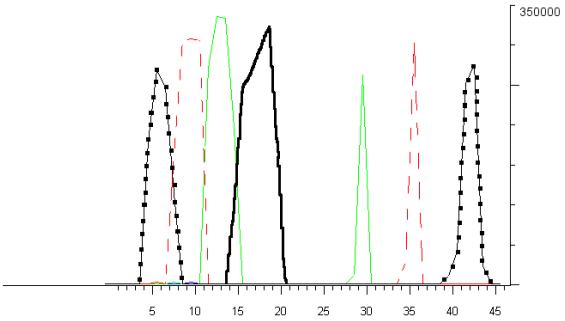

Figure 8: Four concurrent 1MB data transfers with priority management enabled: bold line refers to highest priority connection. This starts late, but it is the first to be completed (wireless LAN).

sor networks (**?**) and embedded systems. Prefetching mechanism (**?**) may improve user experience with mobile device; at the same time they can take advantage of priority management enabled communication stacks. LCPP can be easily integrated in existing network-oriented applications and used to test algorithms for prefetching. Priority management for mixed voice and data services is addressed by other interesting works (**?**), although proposed approaches are not so easy to be implemented in a web-oriented prototype. Other works focused on application-level approach, which is out of the scope of our work.

6 CONCLUSION

Priority managed communication approach could improve user experience on mobile devices, giving fastest access to user interactive applications. Current LCPP prototype has been implemented in a Java environment; therefore, to take advantages of priority management, applications have to share the same runtime environment, that is responsible for incoming data fragments. Therefore, to date applications have to use the custom LCPP protocol library and related interfaces. Such an approach is suitable for fully integrated applications, where several modules access concurrently to the same network stack (the runtime environment stack). On the other hand, native applications still use the operating system layer, which cannot be controlled by the LCPP library. We plan to implement LCPP concepts in the kernel IP stack, so that applications will be able to use standard libraries and, if needed, to assign and negotiate data fragments priorities to incoming data with remote peers. We also plan to improve protocol by adding more commands for detailed automatic error reporting. LCPP is being used as communication protocol for the EU IST EurEauWeb project.

Connections	Priority	No priority	THP
2	24s	20s	6s
4	44s	36s	7s
8	92s	70s	7s

Table 2: Results for four concurrent data transfers. Priority management delays overall data transfer, but Time of Highest Priority (THP, the time needed to complete data transfer with highest priority) does not depend on the number of actual concurrent connections. LCPP assures that highest priority data will be transferred in the shortest time.

REFERENCES

C. Hunt (2002). *TCP/IP Network Administration*. O'Reilly

N. Minar et al. (2001). *Peer-toPeer*. O'Reilly

E. Rusty Harold (2000). *Java Network Programming*. O'Reilly

P. Buonadonna, J. Hill, and D. Culler (2001). Active message communication for tiny networked sensors. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'01)*.

D. Makrakis, R. S. Mander, L. Orozco-Barbosa, P. Papantoni-Kazakos (1998). A spread-slotted random-access protocol with multi-priority for personal and mobile communication networks carrying integrated traffic In *Mob. Netw. Appl.*.

H .Kirchner, R. Krummenacher, T. Risse, D. Edwards-May (2004). *A Location-aware Prefetching Mechanism*. Fourth International Network Conference (INC 2004)

Sun Microsystems. *PersonalJava Specification*. http://java.sun.com/products/personaljava

European Telecommunications Standards Institute. *GSM Specification*. http://www.etsi.org