The p-kernel: A framework for mobile and wireless network protocol implementation on personal digital assistants

-

Steven Robinson and Michel Barbeau
School of Computer Science
Carleton University
1125 Colonel By Drive
Ottawa, ON K1S 5B6 Canada
Steve.Robinson@psti.com, barbeau@scs.carleton.ca

Abstract

-

This paper describes a software called p-kernel providing a configurable architecture for experimenting implementations of mobile and wireless network protocols on personal digital assistants.

1. Introduction

A network software is aiming at managing the network adapters of a computer, making the resources of the computer available to other network nodes, and providing the data and control abstractions for passing information from process to process over the network.

Most network software are structured into layers. Each layer contains a certain number of protocols. A network software is normally a part of the kernel of an OS (e.g. the Linux kernel [Satc 00]) but not always and may be embedded within a user level application (e.g. x-kernel [Hutc 91]).

A number of protocols for supporting mobile and wireless communications has emerged such as Bluetooth [Haar 00], Mobile IP (MIP) [Perk 96], Service Location Protocol (SLP) [Gutt 99], and Wireless Application Protocol (WAP) [Wire 98]. They are, however, not always widely deployed and available on laptops, handheld devices, and Personal Digital Assistants (PDAs) [Vars 00].

The problem addressed in this paper is the support of mobile and wireless network protocols in PDAs.

Palm OS is the operating system of a widely spread PDA architecture [Palm 00]. There is a need for an environment for doing research on mobile and wireless network protocols on the Palm OS, i.e. experimentation of new protocols at the physical, link, and network layers. The Net Library available for the Palm OS does only support a fixed set of protocols and does not allow integration of

new protocols because it is not open source. It not usable as a vehicle for experimenting with the new mobile and wireless protocols such as MIP and SLP.

In this paper, we present and discus a framework we have been developing for implementing mobile an wireless network protocols. The framework is configurable and allows integration of a protocol graph which executable version runs as a component of a communication application on the Palm OS.

In Section 2, we discuss the integration of an instance of the p-kernel framework within an application on the Palm OS. Some design issues of our framework are discussed in Section 3. In Section 4, we discuss the experimentation being conducted with the p-kernel. We conclude with Section 5.

2. The p-kernel model and Palm OS

The p-kernel is a framework. It means that it provides a set of data structures, subroutines, a control loop, and slots in which a developer can hook its own data structures and subroutines in order to actualise the framework according to its needs. The needs correspond to a graph in which specific protocols such as MIP, UDP, and SLP appear.

The p-kernel is inspired of the x-kernel [Hutc 91] of which is adopted the object based model. There are three types of abstract objects: protocol, session, and message.

A protocol is a static entity, created when a protocol graph is instantiated, modelling a given protocol such as MIP.

Protocols are composed of sessions which are created dynamically and modelling end-points of communication channels. A session maintains the state of the channel end-point, e.g. values of sequence numbers.

Messages are created dynamically and flow from one protocol/session to another. A message is normally made of a header part and a user data part.

An important concept within the model of x-kernel is the Uniform Protocol Interface (UPI). The communication primitives supported by protocols and sessions are uniformed and generic across all the layers. They are actualised by the protocol developer, using a function pointer mechanism. The UPI contributes to achieve a uniform structure from one layer to another which helps considerably to the clarity of a protocol stack.

An interesting feature in the implementation of this model is the message-per-process model. Indeed, each message, starting from its reception at the hardware level or its creation at the application level, is taken in charge by a thread. A protocol is implemented by a set of procedures and a thread calls successively the procedures implementing tasks of the protocols and sessions the associated message has to go through.

Concurrent threads may have to synchronise together. For instance is may be necessary to synchronise a thread handling an acknowledgement message with threads handling data messages and waiting for empty slots in a transmission window. Inter threads synchronisation is achieved using semaphores.

The x-kernel supports the aforementioned model on full scale computers. The challenge of p-kernel is to support that model on PDAs which have low memory, are slow, and have few communication resources. In addition, the Palm OS does not provide all the system programming concepts that we found on a full scale OS. For instance, there are no notion of thread and semaphore. It makes support of the x-kernel model on the Palm OS a little harder to achieve.

The p-kernel is implemented as a library which is embedded into an application that needs communication. A protocol graph is defined programmatically. When the application is entered, the protocol graph is instantiated, the p-kernel OS is created, and every protocol of the graph is initialised.

A typical application on the Palm OS is driven by a event processing loop. Execution of the application is triggered by events, e.g. a user interaction, arrival of a message. The association between the application and a p-kernel instance is made within the event loop of the application, as illustrated by the following pseudo code:

```
App_Event_Loop()
do {
     EvtGetEvent(event);
     <Give control to the p-kernel OS>
     <Usual event processing of a Palm OS application>
} while (event!=StopEvent);
```

When an event occurs, function `EvtGetEvent()` returns with an element of data representing the event. The control is first given to the p-kernel OS which either handles the event, if it has to do with telecommunications, or returns, otherwise. In the latter case, the event is handled according to usual event handling model of a Palm OS application.

3. Some design issues

Key internal data structures of the p-kernel are an event queue, two message queues, and a list of semaphores.

The queue of events maintained by the p-kernel is orthogonal to any event structure maintained by the Palm OS. The queue of p-kernel contains scheduled events related to telecommunications, as a delta list. Each element stores a value representing the time remaining before the event can occur, a

function that is executed when the event occurs, and arguments of the functions. Typical events stored in the event queue of p-kernel are retransmissions triggered by a timeout or periodic broadcasts of beacon messages.

Processing of events stored in the p-kernel are triggered by the Palm OS timeouts (a type of Palm OS event). When an event is put in the event queue of p-kernel, a timeout is set to the time value associated to the event. When the timeout occurs, the application event loop processes the event at the head of the queue.

Two queues of messages are maintained by the p-kernel, namely, an incoming message queue and an outgoing message queue. Priority is given to the processing of incoming messages to avoid losing received data. Messages are, indirectly, put in the outgoing queue by applications while messages in the incoming queue are put by the low level protocols. Conceptually, the unit of execution interpreted by p-kernel is a thread handling a message from the beginning to the end of its existence. The concept of thread is not available to Palm OS programmers. They are simulated by the p-kernel.

The semaphore is a very convenient tool for the implementation of protocols. For instance, within a session implementing a flow control algorithm it may be useful to synchronise threads handling outgoing messages waiting for empty slots in a send window with threads handling incoming acknowledgements that create empty slots in the send window. The Palm OS does not make available to the programmer the concept of semaphore. So we implemented our own.

Semaphore APIs are often implemented using a system call that saves the context of a thread when it gets block on a semaphore and another system call that restores the context when the semaphore is such that the thread may be unblocked. Palm OS does not offer these system calls to the programmer.

To implement our semaphore API without context saving and restoring functions, we adopted the following model. Our wait function on a semaphore as a function parameter representing the continuation of the thread when it becomes unblocked. Saving and restoring of context are not required. Hence, the call to wait function on a semaphore is always a tail statement in a sequence because logically the continuation is defined by its function parameter.

4. Experimentation

We are developing a stack of mobile and wireless protocols within the p-kernel. Currently, one wireless link level protocol has been implemented within the p-kernel, namely, the KISS protocol. It is a protocol for computer to data radio communication [Chep 87]. The data radio itself handles the medium access control (using CSMA/CA) and the data link control (using a variation of HDLC). This type of data radio is available from several manufactures including Kenwood [Kenw 00] and Symek [Syme 00]. Data rates go from 9.6 Kbps to 614 Kbps (with Symek data radios). These are comparable with the data rates promised by GPRS [Bett 99], i.e. 115 Kbps. We operate in 144-148 MHz and 440-450 MHz segments of the radio spectrum.

We are currently working on a support of Mobile IPv6 within the p-kernel.

5. Conclusion

We have presented the p-kernel framework which aim is to provide a framework in which mobile and wireless network protocols can be experimented on the Palm OS. A data link level protocol is currently supported, i.e. KISS. A network level protocol is being implemented, i.e. Mobile IPv6. Experiments are being conducted with data radios.

References

[Bett 99] C. Bettstetter, H.-J. Vogel, and J. Eberspacher, GSM Phase 2+ General Packet Radio Service GRPS: Architecture, Protocols, and Air Interface, IEEE Communication Surveys, Third Quarter, Vol. 2, No. 3, 1999.

[Chep 87] M. Chepponis and P. Karn, The KISS TNC: A simple host-to-TNC communications protocol, in: Proceedings of 6th Computer Networking Conference, Redondo Beach, California, August, 1987, pp.38-43.

[Gutt 99] E. Guttman, C. Perkins, and J. Kempf, Service Location Protocol, Version 2, Request for Comments: 2608, June 1999.

[Haar 00] J.C. Haartsen, The Bluetooth Radio System, IEEE Personal Communications, February 2000, pp. 28-36.

[Hutch 91] N.C. Hutchinson and L.L. Peterson, The x-Kernel: An Architecture for Implementing Network Protocols, IEEE Transaction on Software Engineering, Vol. 17, No. 1, January 1991, pp.64-76.

[Kenw 00] Kenwood, http://www.kenwood.net/products, 2000.

[Palm 00] Palm, Inc., Palm OS Platform, http://www.palmos.com, 2000.

[Perk 96] C.E. Perkins (Editor), IP Mobility Support, IETF Request for Comments: 2002, October 1996.

[Satc 00] S.T. Satchell and H.B.J. Clifford, Linux IP Stacks Commentary, CoriolisOpen Press, 2000.

[Syme 00] Symek GmbH Homepage, http://www.symek.com, 2000.

[Vars 00] U. Varshney and R. Vetter, Emerging Mobile and Wireless Networks, Communications of the ACM, June 2000, Vol 43, No. 6, pp. 73-81.

[Wire 98] Wireless Application Protocol Forum, Wireless Application Protocol Architecture Specification, Version 20-Apr-1998, 1998.