Developing communication-oriented applications for smart devices

Jörg Roth

University of Hagen, Department for Computer Science, 58084 Hagen, Germany
Joerg.Roth@Fernuni-hagen.de

Developing software for smart devices is very cost intensive. A huge amount of different software and hardware platforms (e.g., digital cameras, handhelds, electronic pens, mobile phones) has to be combined with many communication technologies (e.g., IrDA, BlueTooth, FireWire, GSM). Sometimes, such devices operate in co-operation with desktop computers, thus communication links to traditional infrastructures have to be considered. Currently the market for devices and communication technologies is rapidly changing. To reuse software, even when the underlying communication infrastructure is modified or exchanged, applications should not be developed 'from scratch' but with help of platforms. An ideal communication platform for ubiquitous computing should

- have a lean runtime system, e.g. should even run on embedded systems;
- support different operating systems, programming languages and communication techniques;
- support different communication abstractions, e.g. unreliable datagrams (e.g. for voice), reliable data streams, multicast (unreliable and reliable);
- support service lookup;
- negotiate communication parameters;
- offer encrypted channels and data compression;
- support debugging and testing, e.g. offer simulation, monitoring and logging facilities;
- translate simple data formats (e.g. strings, integers) as well as multimedia formats (e.g. images) between different platforms.

(Note: this list of requirements does not include remote method invocations or remote procedure calls).

We strongly believe that concepts which are well-known from desktop computing can hardly be adapted to ubiquitous computing devices. To give an example: desktop computing heavily benefits from Java, since Java bridges the gaps between different software and hardware platforms. However, in our opinion Java will not be suitable for small devices in the nearer future. Current operating systems for small devices e.g. PalmOS, ARIPOS, WindowsCE, mainly support C since compilers for C are easy to realize for different systems. In contrast, virtual machines for Java are difficult to port and require a lot of processing power and memory. In addition, Java (and thus RMI and Jini) needs TCP/IP as communication protocol. A communication link has to provide IP support (at least an IP emulation), even if the native communication technique is often more suitable.
To meet the list of requirements above, we developed our own solution, the *network kernel framework* (NKF). It is available for Palm OS, currently we are working on an NKF implementation for the C-Pen. NKF was used in the groupware platforms DreamTeam and QuickStep, the latter platform was designed to support groupware on handheld devices. With NKF, the underlying network can be exchanged without rebuilding the platform or applications.