

Position Summary: The Lana Approach to Wireless Computing

Chrislain Razafimahefa Ciarán Bryce Michel Pawlak*
Object Systems Group – University of Geneva
Switzerland

Wireless communication is revolutionizing computer systems. On the one hand, long-distance technologies like GSM, UMTS, or satellite facilitate access to existing information services. It is expected that wireless Internet access in Europe will exceed fixed line access by the year 2005 [4]. Further, short-distance wireless (SDW), e.g., Bluetooth or wireless LAN, permit ad hoc or *spontaneous* systems composed of people carrying PDAs. These networks can offer new kinds of services, e.g., micro-payments, localization of offers at a market place. The hardware infrastructure for these systems exists, and it is expected that there might be 700 billion Bluetooth enabled devices by the year 2004 [4].

There are three major issues that SDW application programmers have to be aware of: *disconnected operation*, *security* and *coordination*. The composition of SDW networks can be very dynamic, so disconnections have to be planned for. This requires mechanisms that enable a node application to continue running despite changes in its network configuration. Coordination covers activities such as service announcement and lookup in a network. Thus, when a node joins a network, its programs can locate other programs and services. Security is required so that sensitive exchanges between two devices are not attacked.

The goal of the **Lana project** is to develop system support for SDW applications¹. We chose a *language-based* approach: mechanisms like scoping and typing are used to enforce system properties. The Lana language is strongly influenced by Java [1] – it contains interfaces, packages, single inheritance etc. – though is designed with support for disconnected operation, coordination and security.

Lana supports concurrent *programs*. The language semantics states that all memory locations transitively reachable from a program object must be moved along with the program. Further, the set of programs is organized into a hierarchy. When a program moves between nodes then all of its sibling programs are moved along with it. This feature is used by applications to specify hoarding policies: all related programs and objects are grouped under a common

umbrella program which is moved. Method calls between programs are asynchronous; thus, a caller is never blocked awaiting a reply that might never come. Each method call generates a unique *key* object that is used by a program to locate the reply message or exception if ever the program momentarily leaves the network. Return messages – or security or mobility exceptions provoked by the call – have this key value bound to them. Any program that is delegated the key by the caller may therefore service the reply message. Thus a node may leave a network yet safely delegate its pending jobs to other nodes.

Concerning coordination, each environment contains a Linda like message board [3] that is used by devices that meet to exchange an initial set of program or object references. The board is also used to store orphan communication replies (if the caller node disappears during a call).

Concerning security, the language prohibits a program from gaining access to objects stored outside of its scope. Keys are another security mechanism. As seen, keys are used to identify method returns; this also prevents a rogue program from intercepting replies destined at other programs. Entries in a message board are also locked with keys. An entry can only be read if the requesting program furnishes the matching key.

The language approach to wireless is useful because the application programmer has control over the security and hoarding policies. For hoarding, mechanisms implemented in an OS kernel can be inefficient since the lack of application behavior information can lead to the wrong data being hoarded [2]. Security also requires application knowledge so that meaningful security constraints can be enforced.

References

- [1] K. Arnold and J. Gosling. *The Java Programming Language*. The Java Series. Addison-Wesley, Reading, MA, 1998.
- [2] B. D. N. et al. Agile application-aware adaptation for mobility. In *ACM SOS*, pages 276–287, Oct. 1997.
- [3] D. Gelernter. Generative Communication in Linda. *ACM Trans. Prog. Lang. Syst.*, 7(1), Jan. 1985.
- [4] The-Wireless-World-Research-Forum. The Book Of Visions. RFC draft, The Wireless World Research Forum., Jan. 2001.

* {razafima,bryce,pawlak}@cui.unige.ch

¹This work is supported by the Swiss National Science Foundation (FNRS 2100-061405.00).