Towards an organic mobile terminal by utilising agent-based monitoring in a reconfigurable protocol stack

Thorsten Schöler, Moez Mnif, Vladimir Kossovoi, Christian Müller-Schloer

Institute of Systems Engineering, System- and Computer Architecture,
University of Hannover
Appelstraße 4
30167 Hannover
Germany
{schoeler, mnif, vkossovoi, cms}@sra.uni-hannover.de

Abstract: This paper introduces a flexible and reconfigurable protocol stack framework containing agent-based self-optimisation-capabilities for mobile terminals. The framework architecture and implemented self-optimisation and reconfiguration concepts are described. The feasibility of the approach is demonstrated by integrating a TCP / IP protocol stack implementation on the proposed framework architecture.

1 Introduction

Traditionally, protocol stack software (in stationary terminals, such as personal computers, as well as in mobile terminals) is implemented as a highly optimised and more or less monolithic piece of software. The monolithic approach does not satisfy the raising flexibility which is demanded from nowadays mobile terminals. The raise in flexibility stems from the increasing number of wireless standards to be supported by terminals as well as from higher software complexity and its inherent failure probability, which needs to be mastered. Furthermore it's the enormous configuration complexity which calls for methods of reducing the user's burden to configure his mobile terminal properly. We intend to introduce principles of organic computing, adopted from nature, into protocol stack software.

Organic computing is a fast developing area in computer science. Systems based on organic computing ideas will change and adapt their components dynamically to follow the challenges of the environment [VDE04]. Software agents are one of the possible ways to answer the necessary intelligence requirements for organic computing.

The first section of the paper describes a framework architecture for creating reconfigurable protocol stacks as a foundation for organic self-x abilities. Followed by a description of the implemented TCP / IP protocol stack, the organic monitoring abilities will be described and introduced to the framework by an agent-based monitoring system.

2 Framework architecture and mechanisms

The following section describes a protocol stack framework for building modular, reconfigurable protocol stacks for mobile terminals. It is used as a foundation for the implementation of real networking protocol stacks (as described in the next section of this paper).

As seen in Figure 1, the proposed reconfigurable protocol stack architecture consists of three major components. A *framework,* which implements the main functionalities like configuration management, validation, and monitoring (see also [SM04]); a *library* of generic software components (such as timers, checksum generators, message handling, encryption / decryption, etc.), from which the actual protocol stack layers will be constructed from and the *actual protocol stack* instances. The protocol stack implementation consists of a number of protocol stack layers which are composed from generic library and protocol-stack-specific components (downloaded over the air from external sources or from terminal framework library respectively).

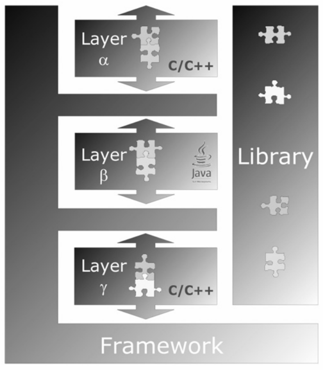

Figure 1: Reconfigurable protocol stack framework architecture

The protocol stack framework uses a *thread-per-message approach* to pass data messages between the protocol stack layers (user plane). Compared to the thread-per-layer approach, this model yields performance advantages at to-be-crossed layer-boundaries. The chosen model does not require a context switch for crossing layer-to-layer boundaries, thus although Java is considered as the primary implementation language, sufficient performance will be achieved (see also [HP91] [OP92]).

Additionally to this mechanism, a mailbox model is available to handle internal protocol stack control messages, exchanged directly between adjacent layers (control plane).

An exemplary protocol stack implementation of the TCP / IP protocol stack, based on the proposed architecture and its concepts will be described in the next chapters to validate the feasibility of the proposed software architecture.

3 TCP / IP Realisation

While the TCP / IP protocol is not designed especially for wireless communication, with the increasing success of the Internet, it has become the de-facto industry standard. There is evidence that in the future all devices, especially mobile ones, will communicate using TCP / IP. This was the principle reason to choose TCP / IP for validation of the framework and its functionalities.

The TCP / IP protocol suite is a collective term applied to several protocols: The IP protocol itself and various protocols related to IP such as ARP, UDP, TCP, ICMP, SMTP, etc. Our proposed implementation consists of dedicated modules for TCP [RFC792], IP [RFC791], ICMP [RFC793] and ARP [RFC826].

While implementing TCP / IP, it was more important to optimise and enhance the various mechanisms of the framework, than getting the entire features of TCP / IP. The implementation was therefore designed to provide only the main functionalities of a well-working TCP / IP stack.

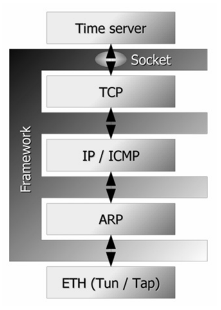

Figure 2: TCP / IP implementation in the framework

The TCP / IP stack is implemented in a modular manner; each module corresponds to a certain layer in the protocol stack, a TCP module, an IP / ICMP module and an ARP module. Native (C / C++) or existing legacy configurations, pure Java and mixed configurations are possible and can be exchanged during runtime. Additionally, test applications, which are based around the BSD-socket-API, have been ported for validation purposes.

The possibility to base the implementation on generic reusable parts from the framework library, made the implementation of new modules much easier.

Since each TCP / IP stack is based on a network device, the UNIX TUN / TAP device has been chosen to emulate a real Ethernet network device. Figure 2 shows how the stack is implemented in the framework.

4 Agent-based monitoring towards organic computing

To achieve the requirements of self-configuration, self-healing, self-protection and self-optimisation, we have combined the framework architecture for reconfigurable protocol stacks with an intelligent multi-agent system.

An agent is an encapsulated computer system that is situated in some environment and is capable of flexible, autonomous action in that environment in order to meet its design objectives [Je00]. Most researchers agree on the following features of intelligent agents: *autonomy, adaptiveness, collaborative behaviour,* and *mobility* [FY99]. A multi agent system (MAS) in turn can be defined as a loosely coupled network of entities that work together to make decisions, or solve problems that are beyond the individual capabilities or knowledge of each entry [ZZ04].

The intelligence of the monitoring agent is packed in a robust structure of a finite state machine (FSM). Both, the states and actions of the FSM can be changed at runtime, providing the agents with the ability to adapt them to a changing environment.

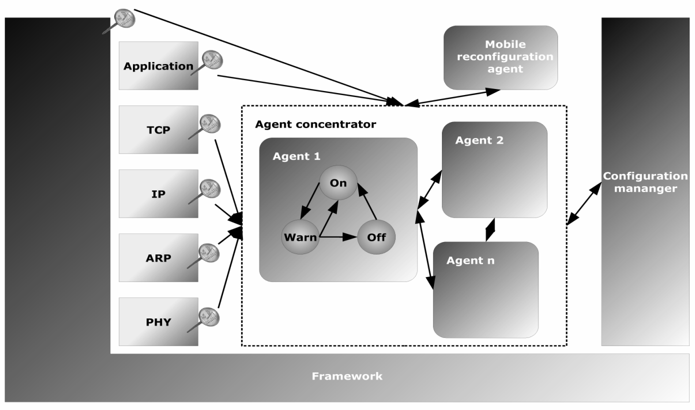

Figure 3: Agent-based monitoring framework

Specific sensors are situated in the framework to obtain runtime information (observer pattern) from the protocol stack. In this way, an agent can dynamically subscribe to the sensors of our protocol stack framework. An agent can request a list of all available sensors as well as agents. In combination with the observer pattern, this allows one agent to subscribe / unsubscribe to the conversation with other agents, allowing single-cast and multi-cast communication. Furthermore it allows agents to become sensors for other agents as well.

As seen in Figure 3, the agents obtain data from sensors and software probes, such as: number of sent and received bytes, packet size, retransmission and error information from different network levels as well as information about user actions.

Using their internal intelligence (rules) and by communicating with each other, they evaluate the data, update their own status and try to fulfil the self-healing tasks by taking actions and fixing configurations (closed-loop operation). Due to the flexible architecture of this system, heterogeneous decision making algorithms can be implemented.

If the agents determine, that a problem cannot be solved by themselves, they try to connect the terminal-based configuration manager (or even the network provider) and send a request for a new system configuration. In case of external help (network provider) a mobile configuration agent is sent to the terminal. This mobile agent can update the monitoring system so that the local agents can make the correct decisions in the future.

Furthermore and to enhance prediction of system behaviour, the user activity should be followed. The agent system then will try to define a user profile. Application level agents monitor the user / system interaction.

5 Current state and conclusion

The described reconfigurable protocol stack framework has been validated by various implementations and configurations of the TCP / IP protocol suite and has been proven stable and helpful. Lower protocol stack layers (such as for WLAN and Bluetooth) are currently under examination and will be included. As described herein, the framework is currently being enriched with intelligent agents fulfilling self-monitoring and self-healing tasks, making them important components of the framework. This envisaged monitoring approach provides a robust and dynamically runtime-extendable architecture towards an organic terminal. A mobile terminal protocol stack, acting in a closed loop with fewer connections to the network provider or to the user, is achieved. The reconfiguration of the system, to current network demands through mobile agents, makes it possible to enhance the performance of mobile devices. This leads to a smaller burden for the user in managing the software complexity.

References

[FY99] Feldman, S. and Yu, E.: “Intelligent agents: A primer.”. http://www.infotoday.com/searcher/oct99/feldman+yu.htm. October 1999.

[HP91] N. C. Hutchinson and L. L. Peterson: “The x-Kernel: An architecture for implementing network protocols”. IEEE Transactions on Software Engineering, 17(1): 64-76, Jan. 1991.

[Je00] Jennings, N. R.: “On Agent Based Software Engineering”. Artificial Intelligence, Vol. 117, 2000, 277-296.

[JSW98] Jennings N. R., Sycara, K., Wooldridge, M.: “A Roadmap of Agent Research and Development, Autonomous Development, Autonomous Agents and Multi Agent Systems”. Vol. 1, 1998, 7-38.

[OP92] S. W. O'Malley and L. L. Peterson: “A dynamic network architecture”. ACM Transactions on Computer Systems, 10(2): 110-143, May 1992.

[RFC791] Postel, J.: “Internet Protocol”. RFC 791, USC / Information Sciences Institute, 1981.

[RFC792] Postel, J.: “Internet Control Message Protocol”. RFC 792, USC / Information Sciences Institute, 1981.