C. Prehofer, W. Kellerer, R. Hirschfeld, H. Berndt, K. Kawamura: An Architecture Supporting Adaption and Evolution in 4G Mobile Communications Systems, JCN, Vol. 4 No. 4, December 2002.

An Architecture Supporting Adaptation and Evolution in Fourth Generation Mobile Communication Systems

Christian Prehofer, Wolfgang Kellerer, Robert Hirschfeld,
Hendrik Berndt, Katsuya Kawamura

Abstract

A major challenge for next generation mobile communication is capturing the system architecture's complexity with all its internal and external dependencies. Seamless integration of heterogeneous environments in all system parts is a key requirement. Moreover, future systems have to consider the different evolution cycles of individual system parts. Among those, services are expected to change the fastest. With respect to these considerations, we propose an overall architecture for next generation mobile communication systems. It covers all system parts from wireless transmission to applications including network and middleware platform. Our approach focuses on adaptability in terms of reconfigurability and programmability to support unanticipated system evolution. Therefore, we consider abstraction layers which consist of adaptable cooperating components grouped by open platforms rather than rigid system layers. In addition to that, we introduce cross-layer cooperation allowing an efficient use of the available resources. Specific scenarios illustrate the feasibility of our approach.

Keywords

4G architecture, mobile system evolution, network architecture, service architecture, middleware architecture, active networks, open systems, software adaptation, reconfigurability

I. Introduction

This article presents an overall concept for a fourth generation (4G) system architecture focusing on adaptability. Starting with today's technologies we can identify several major trends resulting in changes to systems requirements. For this, we propose a new unifying, integrated architectural approach.

4G is expected to be a heterogeneous environment whereas second generation (2G) and third generation (3G) systems are characterized by a single type of air interface. Currently, network systems are still developed as closed systems. Only for services and applications has third party provisioning started recently. This is expected to be extended in fourth generation mobile networks where different parties are able to contribute both user applications and network components [1][2].

With fast progress in radio and network technologies, we envision interactive, high-quality multimedia applications on ubiquitous mobile devices for the fourth generation. Most of these new applications and services are difficult to predict beforehand. In the presence of open platforms and third-party participation in service creation, we assume the application and service landscape to change much faster than today.

Seamless integration of heterogeneous environments on all system levels is regarded to be a key requirement for 4G. Future systems have to consider the different evolution cycles of individual system parts where services are expected to change at the highest pace. Our architecture for next generation mobile communication systems is motivated by these considerations.

So far, most architectures center on required functionality and its separation within a layered system structure, e.g., [1][2][3][19]. They are not considering system adaptability and

evolution capabilities required for mobile systems. For those requirements our approach offers a unifying system architecture that covers all system layers from radio transmission to applications.

We advocate open platforms on every layer and not only open interfaces between layers. To support adaptation, future systems must allow for dynamic exchange of interacting components including the change of interfaces. Fixed interfaces clearly limit this adaptability to the available methods. Open platforms provide a base, on which platform components including interfaces could be dynamically exchanged. Interface discovery and negotiation is one of the basic functionalities of such an open platform.

Our system architecture is represented by abstraction layers, grouping open platforms that do not reflect strict functional dependencies like in traditional layering. Compared to the layers of the Open Systems Interconnect model of the International Standards Organization (ISO/OSI), our architecture is more tailored towards implementation technologies. For example, conventional layer 3 functionality can be found in several platforms, i.e., in the forwarding engine, the protocol processing engine or in the smart card. In our approach, we hence consider these platforms with their components separately and examine their dependencies.

Open platforms permit the installation of new components with new interfaces for interactions between any platforms beyond abstraction layers, which are called cross-layer interfaces. This is important, especially for optimising services in mobile networks due to wireless link and mobility characteristics. For example, in order to efficiently adapt the content representation in terms of resolution and data rate, streaming applications need information from different layers of the network beyond the pure data transport. In time notification about handover would enable applications to react immediately by adapting to the new (e.g., low bandwidth) situations and prevents data loss. In this paper we examine some major driving forces for adaptability in our system architecture. In particular, we show that adaptability can be used to introduce new cross-layer interfaces.

Moreover, openness and adaptability, not only at the service level, but with respect to the whole system architecture, will invite third party vendors to evolve the system parts and/or to provide services as it unfolds and is therefore key to viable solutions.

The structure of the paper is as follows. First, the system concepts of our architecture will be described in Section 2. We introduce abstraction layers including the open platforms of a future mobile communication system and cross-layer interfaces. Next, the open platforms as the main architecture building blocks will be described in more detail. Section 3 describes the components of the middleware abstraction layer and Section 4 focuses on the networking abstraction layer. Two scenarios illustrate the adaptability provided by our architecture in Section 5. The conclusion summarizes the concepts and discusses related work.

II. Architectural Concepts

In this section, we introduce the main concepts used for our system architecture. We first detail our concept of abstraction layers and the notion of cross-layer interfaces. Then we discuss adaptability concepts and open platforms, which can be used to introduce the cross-layer interfaces.

II.1 Abstraction Layers

The scope of 4G systems will range from radio transmission mechanisms to applications visible to the end-user. Abstraction layers encompass transmission, networking, middleware (including service support), as well as applications and services. This is shown in Figure 1.

In addition to interfacing between adjacent abstraction layers, we consider cross-layer cooperation. This is essential regarding different rates of change in different parts of the system.

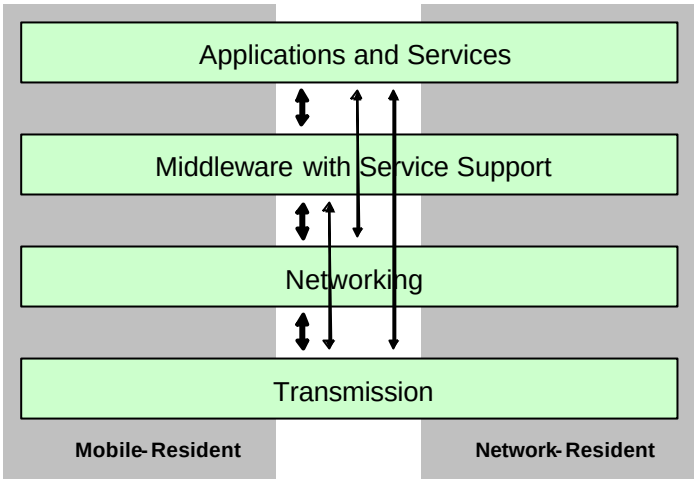

Figure 1: 4G System Abstraction Layers

Each layer can be configured separately and independently via configuration interfaces (not shown explicitly in Figure 1). These interfaces are used to either parameterize existing components of each layer, or to add, exchange, or remove components, through the well-defined interfaces supported by each component.

II.2 Cross-Layer Interfaces

Telecommunication systems are traditionally classified in layers, based on the ISO/OSI. This model is used to show the necessity of abstraction for modularity and for managing the complexity of current networking systems.

However, in mobile networking, one often needs specific and possibly medium-dependent interfaces between the layers, e.g., as discussed in [17]. Cross-layer interfaces define interfaces between different platforms across the layers, which exchange information beyond the standard interfaces between the layers. Cross-layer interfaces can be within, between or beyond adjacent abstraction layers. Although interfaces between adjacent layers are in general preferable, there can be the need for efficient and direct interaction within an abstraction layer or beyond adjacent abstraction layers. For our proposed architecture, cross-layer collaboration means interfacing between platforms and platform components.

In the classical view, cross-layer interfaces may break modularity and the important concept of abstraction. The main benefit of this abstraction is that layers can be exchanged modularly, e.g., a different air interface should not impact OSI layer 3. However, in mobile networks, OSI layer 3 and above often need direct interfaces to OSI layer 2, e.g., for handover support. More examples are discussed in [15][16][18].

Cross-layer interfaces may for instance enable direct signaling of events from lower-level to higher-level platforms or components. They can be characterized by their primary direction (like from upper or lower layers) and by the information that they provide (like notification or configuration). We consider the following types of cross-layer cooperation:

- Additional information, for instance regarding

  - transmission parameters like channel coding technique, use of forward error correction (FEC) and link-layer retransmission settings
  - application characteristics, e.g. status information or source coding
  - the network, e.g. possible bandwidth or delay variations
  - the user (e.g., user preferences, user context)
- Event Notification
  - Notification of changes, e.g. of a network service
  - Notification of anticipated changes in the network services, e.g., handover or quality of service (QoS)-degradation
- Configuration using interfaces (e.g., setting parameters for wireless interfaces and also for the updating of the components (reconfiguration)

Cross-layer interfaces are difficult to handle for the following reasons:

- Cross-layer interfaces are often media or application specific
- Cross-layer interfaces change due to the independent evolution of all abstraction layers involved
- Cross-layer interfaces are difficult to maintain and violate the modularity of the layers

This can be alleviated by our adaptability concepts as discussed below.

II.3 Adaptability Concepts

Flexible adaptation follows a long-term technology trend towards configurable or programmable platforms. This is due to the progress in the development of semi-conductor and software technologies. For instance, for middleware, execution environments [23][24] are now well established. For lower layer technologies like network processors or radio chip sets, a higher degree of programmability is currently a major trend.

Our notion of adaptability is broad in order to cover all system aspects from reconfigurable radio parameters to applications. Due to this broad scope, different forms of adaptability are considered. The supported adaptability could be classified in the following way with respect to configurability and programmability:

- Configuration, either before system startup (e.g., during development or deployment) or at runtime,
- Parameterization of software as the classical way to introduce flexibility without the need for software updates,
- Complete software update or exchange, with firmware update of devices as a classical example, which often requires devices or servers to be made unavailable for some period of time,
- Partial software update in a complex software system, where there are often no open interfaces with the result that most of the expected system behavior has to be reevaluated, with possible service interruptions as another consequence,
- Installation of components in an open platform with an interface design that ensures proper functionality and that supports seamless service evolution,
- On-demand installation of mobile software components that are carried as part of the exchanged data in a communication relationship and that directly influence the

exchanged data. This is similar to the active networks capsules approach [29] or to the mobile code in Jini [30].

In the following we focus on open platforms, although other forms are important in practice as well. The last item, the capsules approach, offers much higher flexibility regarding the location of the execution of the code, but also poses very high risks with respect to security. For this reason, we do not focus on this approach in this paper.

Although adaptability is an important requirement, we should also consider its implications. In many cases, the additional flexibility can lead to higher initial cost. However, the price of a system replacement due to inflexibility is significantly higher. The main cost factors, among others, are more expensive hardware than initially needed and the development of a reliable programmable environment. In addition, the request for flexibility might lead to more complex designs for the introduction of variation points and indirection layers. Yet the bit advantage of programmable hardware and flexible software platforms is that they can be used in larger application areas supporting a seamless system evolution. This leads to much better economies of scale.

II.4 Open Platforms and Cross-Layer Interfaces

In this section, we introduce our notion of open platform and show its relation with the system architecture and cross-layer interfaces. An abstraction layer as shown in Figure 1 may consist of several open platforms providing the actual computational behavior. Each such platform consists of a stable and minimal platform base, plus several platform components, which can be added or removed to address different requirements at different times (Figure 2).

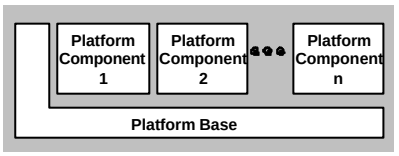

Figure 2: Open Platform with its Base and Components

We claim that the adaptability achieved by this approach is a central concept in future mobile networks, because flexibility is a key requirement for future mobile services.

Additionally, we propose adaptability mechanisms to dynamically introduce cross-layer interfaces. The main idea is as follows: Instead of static, built-in cross-layer interfaces, the components providing the interfaces are installed when needed. This is illustrated in Figure 3. Of course, this requires discovery of interfaces, since some components with interfaces may already be installed.

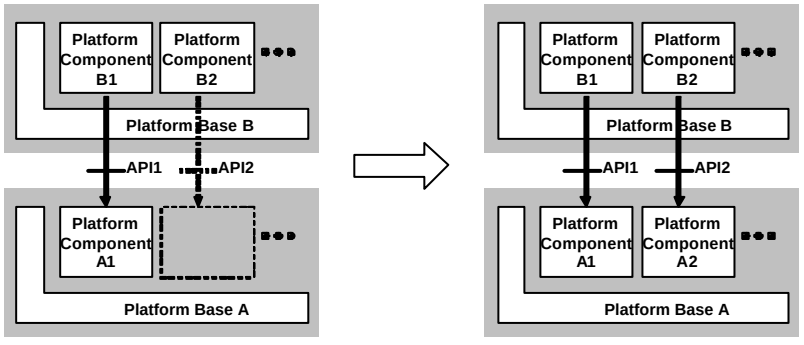

Figure 3: Programmable Cross-layer Interfaces

By dynamically installing components providing the interfaces, we can avoid the shortcomings of static interfaces as described above. This concept is ideal for medium specific interfaces, since the components can be installed in a medium specific way. For instance, different interfaces can be installed for each medium, depending on the desired services. This notion of service deployment architecture has been investigated in the area of active networks in [13][29], but not in the context of cross-layer cooperation.

III. Middleware and Application Abstraction Layers

In this section, we describe platforms that are included in the middleware (including service support) abstraction layer in our overall architecture. In addition, we briefly sketch the applications. Our main goal in designing these platforms for the next generation mobile system is, from the service provider point of view, the effortless development and provisioning of new and customized services. From the system operator point of view, the middleware must support high flexibility, which is essential for maintenance and evolution.

III.1 Network Resident Middleware

On the network resident side of a mobile communication system, the *middleware abstraction layer* is basically represented by three platforms: the local execution platform, the distributed processing platform, and the services platform (Figure 4). The arrows show general dependencies between the platforms and their components, which can also reach beyond the borders of the abstraction layer.

The *local execution platform* is used to decouple most of the middleware components and services/applications from characteristics and specifics of traditional functions implemented in the network nodes and switches. Candidates may include the Java platform (J2ME, J2SE, J2EE) [23] or the .NET platform [24], plus additional third-party components or libraries addressing multimedia, security, transactions, persistence, etc.

The *distributed processing platform* provides an abstraction for interaction between different system parts, residing either on a single or on multiple devices. Mechanisms supported by a distributed processing platform can range from simple remote procedure calls to more complex stream, call, or resource control interfaces.

Technologies, specifications, and protocols to be associated with the distributed processing platform include classical middleware solutions like the Common Object Request Broker Architecture (CORBA) [25], emerging technologies like the Simple Object Access Protocol (SOAP) [27], telecommunication interfaces like in Parlay [6], data communication stream interfaces, and advanced support services like naming, trading, or events.

The *services platform* offers session semantics to any service requesting such. A session describes the meaningful context of service execution and therefore provides coordination functionality. Sessions maintained by the service platform can be separated mostly into access sessions, service sessions, and connectivity sessions. This includes components managing user profiles or context-awareness. Example specifications addressing this can be found in the Telecommunication Information Networking Architecture (TINA) Service Architecture [4][5] and in the Telecommunication Service Access and Subscription specifications (TSAS) from the Object Management Group (OMG) [26]. Extensions to the Parlay Framework API [6] to provide additional session semantics would be conceivable as well.

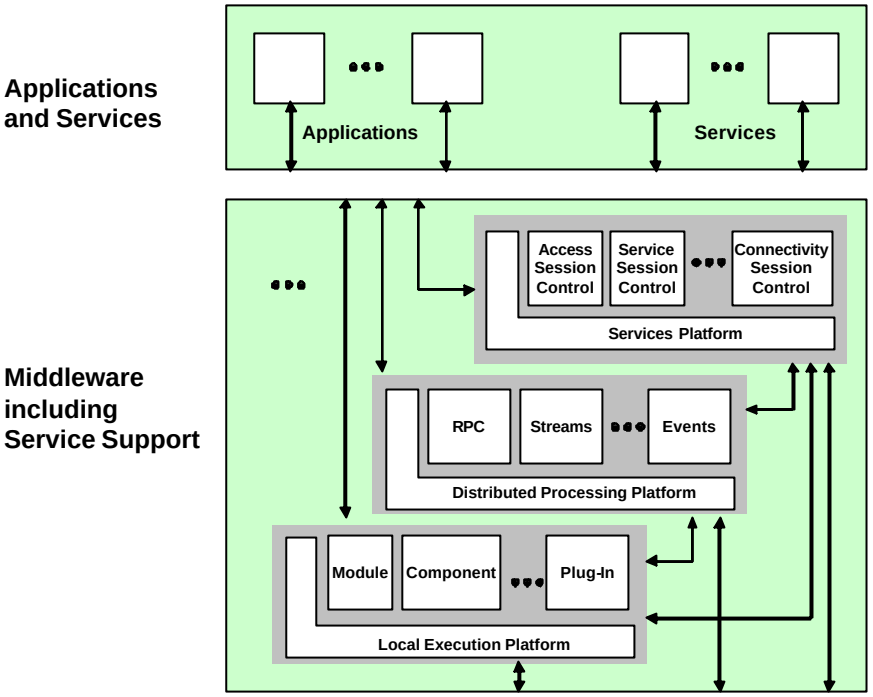

Figure 4: Middleware (Including Service Support) Platforms and Applications

In order to fulfill the expectations of operators and service providers, the main requirements on these platforms from the application point of view are

- Hiding internal system complexity, e.g. for introduction of new applications,
- Seamless support for heterogeneous environment,
- Reliable, uninterrupted service even during adaptation,
- Flexibility provide personalized solutions,
- Protection of user data.

III.2 Mobile Resident Middleware

Mobile resident middleware, residing on the mobile device, is typically restricted compared to our middleware abstraction layer in Figure 4. This is mainly due to the resource limitations in mobile devices. Furthermore, a large variety of different mobile devices with different characteristics have to be supported. Mobile resident middleware must be reconfigurable and adaptable with respect to extension elements of particular platforms or the addition or removal of entire platforms, depending on device characteristics, user profiles and preferences, and software requirements of the client resident part of services. In addition to this, mobile resident middleware includes the smart card middleware platform as discussed later together with the smart cart networking platform in Section IV.2.

The need for on-demand downloads of middleware software artifacts makes the security model a core part of the system's architecture. Such security model has to support the interests of at least the user, the operator, the manufacturer, and the actual service provider.

Technologies of interest in the area of mobile middleware are for example J2ME [28] and Mobile Station Application Execution Environment (MExE) [12], both intending to provide a small execution environment for resource-constraint mobile devices.

III.3 Applications and Services

The *applications and services abstraction layer* resides on top of the middleware abstraction layer, with well-defined access mechanisms to interact with open platforms on the layers below. Applications are considered to be executed in the local execution platform making use of other platforms' functionality whenever necessary for the application purpose. For example, they make use of the distributed processing platform in case they are part of a distributed system, spawning more than one process in the network. In general, all communication related services might take advantage of this platform that provides interfaces to standard communication protocols.

Applications are regarded as system parts that can act as clients not only of platform components but of other applications, too. In this way, examples for applications are billing, profile management, media conversion, as well as chat rooms, tour guides, and emergency locators. We regard the term services to describe communication-oriented applications that require particular session semantics, offered by the services platform, e.g., videoconference or computer supported collaborative work.

IV. Networking and Transmission Abstraction Layers

In this section we describe those platforms of our overall architecture that are grouped in the networking abstraction layer. For those platforms, we aim for technologies, which add open interfaces and computing environments, including the radio and networking layers, to enhance existing network nodes and mobile devices. This will allow operators to deploy new radio technologies, protocols and services in order to support novel, not yet anticipated services. The key technologies are the following:

- Software defined radio technologies in reconfigurable user equipment [7] will allow for unprecedented flexibility to add new functionality. Mobile devices will not only have open platforms for installing new applications, but also the radio parameters and the stacks will be programmable.
- Programmable, active network nodes [8] will enable fast deployment of new services by using open interfaces to the network resources. With this technology, new protocols can be installed on the network elements, which use the lower layer resources for creating new services.

IV.1 Network Infrastructure Nodes

A generic architecture for the network elements of a mobile network is shown in Figure 5.

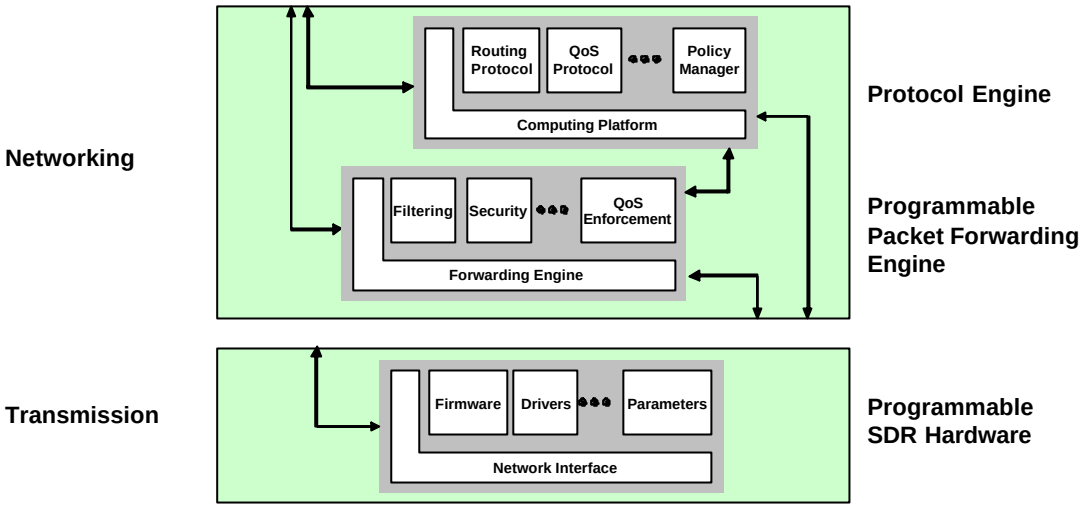

Figure 5: Network Side Networking and Transmission Platforms

In this architecture, we consider three platforms, each programmable with configurable components:

- The *computing platform* serves as a general-purpose platform for processing stateful protocols, e.g. routing, QoS signaling or connection management.
- The *forwarding engine* is in the data path of a network node and it connects network interface platforms, e.g., by a switch matrix. This engine can be implemented as dedicated hardware or as a kernel module of common operating systems. The forwarding engine is programmable for performance-critical tasks, which are performed on a per-packet basis.
- The *network interface platforms* are medium specific for different wireless or wired standards. They can be configured or programmed to adapt to new physical layer protocols or for triggering events in higher layers.

The main component interfaces shall follow existing or upcoming standards (e.g. [9]) or forthcoming Internet Engineering Task Force (IETF) [10] results.

From the operation point of view, the main requirements for the platforms are:

- Reliability for uninterrupted services, in particular when updating services.
- Remote management as it is important for the central configuration in large networks.
- Security with respect to attacks from outside. Since we assume that the network is owned by operators, the main security risks arise from external interfaces.

We assume a that a configuration manager is present in the middleware abstraction layer, which is able to install new components in the lower layers. Since the configuration shall be performed remotely, the distributed processing platform is suitable for this purpose and may also offer transaction services for complex reconfigurations.

IV.2 Mobile Device Architecture

The main platforms of the mobile device-side architecture of Figure 6 are

- *Smart card middleware platform* which provides subscriber identities and a highly secure execution environment.

- *Smart card networking platform* which provides networking functions like addressing and authentication.
- *Programmable radio platform* which is designed for one or more radio standard families.
- *Native operating system* platform, which provides real time support, needed for stacks and certain critical applications, e.g., multimedia codecs.

The main requirements from operator and manufacturer point of view are

- Survivability, e.g., robustness to misconfigurations, failures, or misusage.
- Security with respect to end-user, operator, and manufacturer requirements. For instance, the manufacturer may be liable that the device does not misbehave with respect to radio emission. The operator is mainly interested in reliable services and billing. The user wants to be assured of the device integrity and service availability.
- Mass-market optimization of hardware and software platforms balanced with time-to market and flexibility or upgrade requirements.

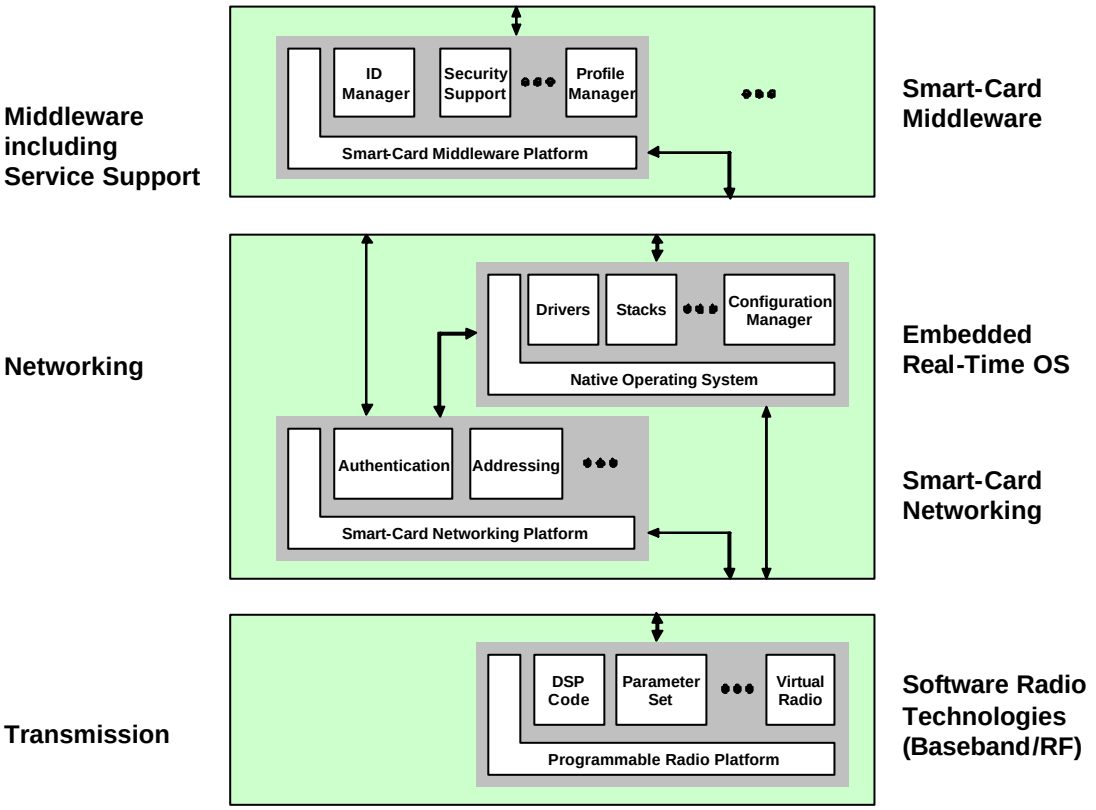

Figure 6: Mobile Device Architecture

Service deployment and control of reconfigurations is complex since there is a split of responsibility between operator and manufacturer. For instance, the manufacturer has to provide or digitally sign appropriate low-level code for reconfigurations. On the other hand, the operator is interested in controlling the configuration to fit the user and network needs.

For the wide-spread adoption of reconfigurable mobile devices, open interfaces are essential. The MExE security model with different security domains for operators, manufacturers and un/trusted third parties is an example for access control [12].

V. Adaptation Scenarios

The following scenarios illustrate the adaptability offered by our approach for next generation mobile communication networks. Both examples demonstrate certain aspects of adaptation mechanisms in combination with cross-layer cooperation.

V.1 Context-Aware Handover

In many cases, a handover can be optimized based on information about the user context. As an example, we consider the optimal selection of a new access point during hand-off (see Figure 7):

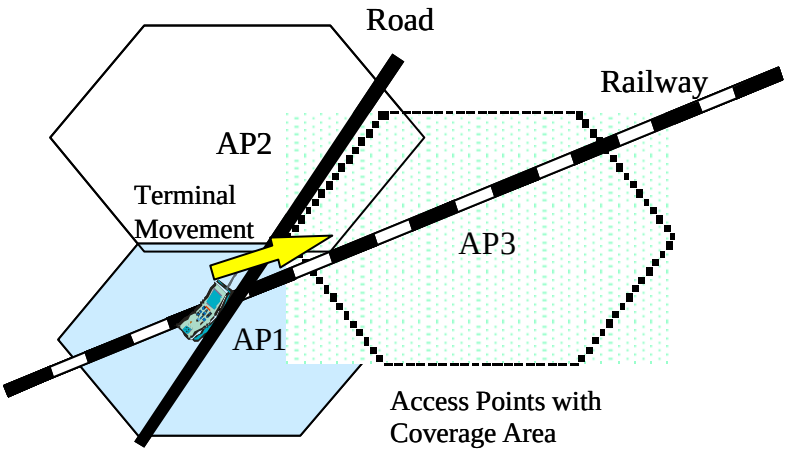

Figure 7: Context-Aware Handover Prediction

In this scenario, the terminal moves into an area covered by two access points AP2 and AP3. The challenge is to decide which access point to choose. State of the art are many algorithms based on signal strength analysis or on available radio resources. Even if one access point is slightly better regarding these local measurements, the decision may not be the best. For instance, if the terminal in the above figure is on the train, it is obviously better to handover to AP3, even if AP2 is better reachable for a short period of time.

In many cases, the handover can be optimized by knowledge of terminal and user context, user preferences and the prediction of terminal movement. For instance, if the terminal is located in a car or train, its route may be constrained to certain areas. The terminal profile may contain the information that the terminal is built in a car. Alternatively, the movement pattern of a terminal may suggest that the user is in a train.

The main problem is that handover decisions have to be executed fast. The terminal profile and location information is usually available on a central server in the core network. Retrieving this information may be too slow for hand-off decisions. Furthermore, in some cases the radio conditions during hand-off may be poor and hence limit such information exchange.

A solution is to proactively deploy a context-aware decision mechanism onto the operating system or middleware platform of the terminal, which can be used to assist hand-off decisions. A typical example is the information about the current movement pattern, e.g., through knowledge of train or road routes. For implementation, different algorithms can be deployed on different layers, depending on the context information. For instance, the information about the movement is available on the middleware layer of the network and can be pushed into the

terminal. Notice that this implementation needs a cross-layer interface, which collects the information from different layers to make an optimized decision.

Similar scenarios have already been investigated in [17], where a middleware was presented which provided a hook for a function, which determines the best next access point. In [18], algorithms based on a static user profile are presented which select among different networks for handover.

V.2 Network-Aware Application Adaptation

In the following application example, we illustrate the functionality and cooperation of the abstraction layers defined above. We take a video streaming application with a content adaptation server as an example, as illustrated in Figure 8. A mobile user wants to receive a real-time video stream while moving through different network situations, e.g., leaving a Wireless LAN (WLAN) covered area and switching to a lower data-rate cellular network. In such case, the presented solution prepares for a soft handover and appropriate content adaptation as soon as possible. Since the reaction to handover is too slow, IP-multicasting of the video stream is used to ensure continuous delivery.

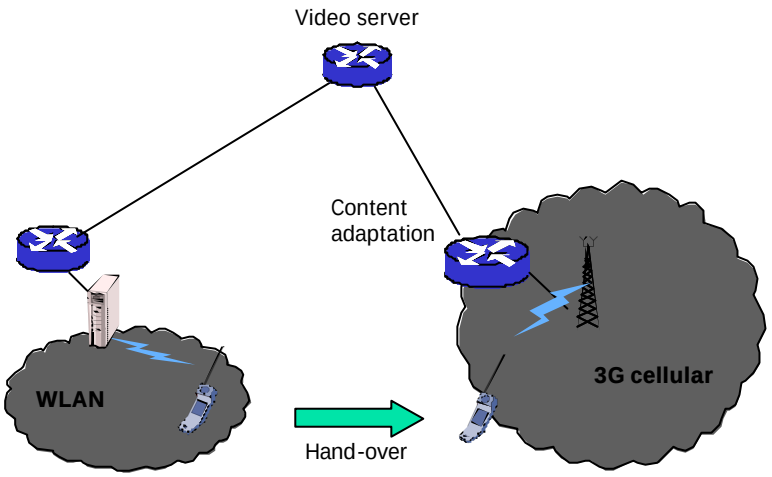

Figure 8: Video Streaming Adaptation for Handover

To provide this "seamless service", several new components have to be integrated into the system on several abstraction layers on both terminal and network side. These include

- Session management components for access, service and connectivity session control in the network-side service platform, which control the service context and interact with both the end-user streaming application and the network components,
- Multicast mechanisms in the network computing platform that divert a media stream to different routes for different access networks,
- A content-adaptation service for low data rate networks as an service on top of the middleware, and
- Handover aware-components in the terminal-side programmable radio platform.

The availability of those components allows one to perform the above sketched example in the following steps: First, video streaming has to be established over WLAN. Therefore, the end-user streaming service starts an access session by interacting with the service platform's

access session controller. The access session controller registers with the network-side service session controller via the remote procedure call (RPC) mechanism of the distributed processing environment and a service session is set up by the service session controller to maintain the service context (e.g., for creating and maintaining status and billing information). Connectivity is provided by the network-side forwarding and transmission platform components. It is controlled by the the connectivity session controller according to the required QoS. Video streaming is started over the WLAN network.

Now, in our example, the user is moving to the border of the WLAN coverage and the application has to react on the change in the network. Immediately, the terminal-side programmable radio platform components predict handover to the cellular network. This announcement is forwarded via cross-layer interfaces to the distributed processing platform and the service session and connectivity session controller of the service platform. These can now prepare for the forthcoming change of QoS.

The appropriate platform components of the network- and terminal side network layers are informed to prepare soft handover, i.e., two connectivity sessions are maintained until the new session has been re-established on the new link. Furthermore, the lower layers are requested to notify the service session control about the new location of the terminal as soon as possible. With this information, the new QoS after the handover can be determined.

The platform components of the network layer of the cellular network notify the service session management of new terminal registration in the cellular network. The service session control informs the adaptation service and the respective end-user service in the terminal, which is included into the data path. The video server sends two different streams, one to the old network and one to the adaptation service, to address both sides of the handover.

Then, as the registration in the new network is completed, the terminal can receive the adapted video stream via the cellular network. When the connectivity session in the cellular network is established or if the connection in the WLAN network is lost, the service session control and the network platform components interact and cut off the old connectivity session. For this, the service session control will instruct the video server to issue only one video stream to the new network.

This example shows the interaction between the session control of the higher layers and the handover control of the lower layers. With this cross-layer cooperation, the right kind of handover and synchronization for the access network change and for the video session transfer can be achieved.

VI. Conclusion

Our approach consists of an overall architecture for future mobile communication systems based on a set of open platforms supporting adaptation and evolution. We have also shown that well designed cross-layer interfaces are essential for wireless networks. Cross-layer cooperation has not yet been considered systematically as an architecture ingredient. To enable cross-layer interfaces an overall framework is needed that allows dynamic insertion and deletion of new interfaces.

Recently, adaptability itself has received increasing importance in the Beyond 3G research community. Earlier versions of our approach [20][21] have been incorporated the World Wireless Research Forum (WWRF) [22], where two special interest groups work on identifying research items for (service level) adaptability and lower layer reconfigurability, respectively.

In the past, performance was critical for many technologies supporting programmability or adaptability. We believe that the processing power of mobile devices and network elements will soon be sufficient for flexible adaptability concepts. Furthermore, the introduction of widely used platforms can lead to a significant reduction in total cost. Key issues in future mobile communication systems are raised by reliability, security, system and configuration management concerns.

We argue that conventional system design in terms of building blocks and open communication interfaces is not sufficient for future mobile networks. With open platforms, we can significantly reduce the effort to introduce new services and reduce the complexity of 4G systems including all services. In this way, we can complement the lengthy standardization and development process for mobile communication networks with a solution for easy deployment of new functionality provided by individual interfaces. Hence adaptability in open platforms on all layers can be a significant breakthrough for deployment of 4G mobile systems.

References

[1] A. Jamalipour, and S. Tekinay (eds.), *Special issue on Fourth Generation Wireless networks and Interconnecting Standards*. IEEE Personal Communications Magazine, October 2001.

[2] ITU-R WP 8F: *Vision, framework and overall objectives of the future development of IMT-2000 and of systems beyond IMT-2000.* Preliminary draft, October 2001.

[3] H. Yumiba, K. Imai, and M. Yabusaki: *IP-Based IMT Network Platform.* IEEE Personal Communications Magazine. October 2001.

[4] L. Kristiansen: *TINA-C Service Architecture*. Version 5.0. TINA-C, 1997.

[5] H. Berndt, T. Hamada, and P. Graubmann, *TINA: Its achivements and its future directions*, IEEE Communication Surveys, Vol. 3, No. 1, 2000.

[6] The Parlay Group: http://www.parlay.org

[7] N.J. Drew, M.M. Dillinger: E*volution toward reconfigurable user equipment.* IEEE Communications Magazine, Vol. 39, Issue 2, Feb. 2001.

[8] A. T. Campbell, M. E. Kounavis, and R. R.-F. Liao: *Programmable Mobile Networks*. Computer Networks, Vol. 31, No. 7, pg. 741-765, April 1999.

[9] Proposed IEEE Standard for Application Programming, http://www.ieee-pin.org/

[10] IETF Working Group: Forwarding and Control Element Separation. http://www.ietf.org/html.charters/forces-charter.html

[11] Mitola, J., III, *SDR architecture refinement for JTRS*, MILCOM 2000. 21st Century Military Communications Conference Proceedings, vol.1, Oct. 2000

[12] 3GPP Technical Report 3GPP TS 23.057 V4.4.0 (2001-12): *Mobile Station Application Execution Environment (MExE) Functional description.* 2001.

[13] M. Bossardt, L. Ruf, R. Stadler, B. Plattner: *A Service Deployment Architecture for Heterogeneous Active Network Nodes*. Kluwer Academic Publishers, 7th Conference on Intelligence in Networks (IFIP SmartNet 2002), Saariselkä, Finland, April 2002.

[14] E. A. Brewer, R. H. Katz, Y. Chawathe, S. D. Gribble, T. Hodes, N. Giao, M. Stemm, T. Henderson, E. Amir, H. Balakrishnan, A. Fox, V. N. Padmanabhan, S. Seshan: *A network architecture for heterogeneous mobile computing*. IEEE Personal Communications, Volume 5 No 5, Oct. 1998.

[15] Y. Fang, A. B. McDonald: *Cross-layer performance effects of path coupling in wireless ad hoc networks: power and throughput implications of IEEE 802.11 MAC*. Performance, Computing, and Communications Conference, 2002. 21st IEEE International, 2002. Page(s): 281–290.

[16] B. Raman, P. Bhagwat, S. Seshan: *Arguments for cross-layer optimizations in Bluetooth scatternets.* Symposium on Applications and the Internet, 2001. Page(s): 176 –184.

[17] M. Kounavis, A. Campbell: *Design, Implementation and Evaluation of Programmable Handoff in Mobile Networks.* Mobile Networks and Applications 6, 443-461, 2001 2001 Kluwer Academic.

[18] H. J. Wang, R. H. Katz, J. Giese: *Policy-Enabled Handoffs Across Heterogeneous Wireless Networks*. In WMCSA 99, New Orleans, LA, 1999.

[19] B. G. Evans, K. Baughan: *Visions of 4G.* Electronics & Communication Engineering Journal, Volume12 Issue 6, Dec. 2000.

[20] R. Hirschfeld, W. Kellerer, C. Prehofer, H. Berndt: *An Integrated System Concept for Fourth Generation Mobile Communication*. Eurescom Summit 2002.

[21] W. Kellerer, C. Prehofer, R. Hirschfeld, H. Berndt: *Configurable 4G System Architecture*. 6th WWRF meeting in London, 2002, http://www.wireless-world-research.org/WWRF6/wwrf6.htm

[22] WWRF, Wireless World Research Forum, http://www.wireless-world-research.org

[23] Java Community Process, http://www.jcp.org/

[24] Microsoft .NET -- http://www.microsoft.com/net/

[25] CORBA, http://www.corba.org/

[26] Object Management Group, http://www.omg.org/

[27] SOAP, http://www.w3.org/TR/SOAP/

[28] Java2 Micro Edition, http://java.sun.com/j2me/

[29] David J. Wetherall, David L. Tennenhouse: *The ACTIVE IP Option*. Proceedings of the Seventh ACM SIGOPS European Workshop, 1996.

[30] Jini Network Technology Archive – APIs and Specs, http://java.sun.com/products/jini/