Comparision of Softswitch-Architectures Using Parlay as Application Interface

J. Stadler[1], I. Miladinovic[1], R. Pailer[1], and M. Vucic[2]
[1]*Vienna University of Technology, Institute of Communication Networks, Favoritenstrasse 9/388, A-1040 Vienna, Austria; {Johannes.Stadler, Igor.Miladinovic, Rudolf.Pailer}@tuwien.ac.at*

[2]*SIEMENS AG Österreich, Gudrunstrasse 11, A-1100 Vienna, Austria*

Abstract In the last years the convergence between circuit switched and packet switched communication networks has become more and more important, not at least because of the growing Voice over IP or rather Multimedia over IP tendency. Currently, network providers from the classical circuit switched world have to think about enabling interconnection to the IP domain over appropriated gateways. Another important issue is the centralization of service logic to make it usable from different types of networks. If this is done over open interfaces, so called 3rd Party Service Providers could offer their services in a more flexible way. Hybrid Components, called softswitches, could fulfil both, gateway functionality and access to centralized applications using standardized interfaces. One possibility for such an interface could be the Parlay API. In this paper we propose three different softswitch architectures supporting connectivity to the PSTN as well as to the IP world. They differ in the placement of network specific components and in the way of gateway control. In each case Parlay is used between the networks specific Call Control and the independent application.

Keywords: Softswitch, Parlay, Gateway, Services, SIP, MGCP

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35584-9_19

O. Martikainen et al. (eds.), *Smart Networks*

1. INTRODUCTION

Due to the growing importance of packet switched technology in contemporary communication networks, there is a promising tendency of convergence between the world of telecommunication and that of data communication. As much as the transport of voice or more general multimedia data over packet switched networks provides an enormous opportunity, it also implies incalculable risks for current circuit switched network providers.

Beside their PSTN services as main business, they have to decide about the development of new packet switched networks, in many cases from the scratch. Huge investments would have to be made for hardware and software, new groups of users have to be addressed and new, more powerful services have to be invented to get back the high costs.

To reduce the financial effort, it would be a goal to utilize equipment, which is applicable for both domains. Furthermore, services should have to be developed only once. Obviously this reduces the cost of the creation of such services and moreover it enables a new class of applications, based on the interaction of different types of communication networks.

So it seems possible that for the years of migration, systems for hybrid network handling could be a meaningful alternative. Using current architectures, which we know from PSTN switches, this is not easily possible. In most cases such devices are designed in a more or less monolithic manner. The software is tightly coupled to the hardware mostly, interfaces are rarely standardized and applications are hardly reusable.

This implies two further problems. Because of the lack of an open service interface only the switch developers are able to develop applications for a certain piece of hardware. This affects the spectrum of available services and increases the time to market. The second issue is that it is impossible to offer network capabilities to so called third party service providers. So only network providers have the ability to act as service providers. Figure 1 shows this conventional architectural approach.

Above considerations motivate the redesign towards a more flexible and more open architecture. Figure 2 depicts such an approach, which is commonly called a softswitch. The International Softswitch Consortium (ISC), formed in 1999, offers the industry a global coordination and promotion of these issues.

A softswitch enables the combination of applications, call control and hardware from different vendors as well as an abstraction of the applied hardware to the call control layer and of the communication network to the applications, by the usage of open interfaces.

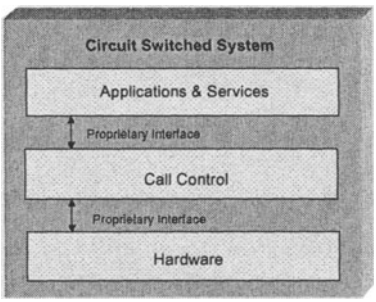

Figure 1. Classical Circuit Switched Architecture

Different realizations of such softswitches are imaginable. Due to a wide range of requirements, it is not possible to fulfil all of them without any shortcomings. In this paper we propose and compare three possible architectures of different softswitch realisations, which have in common that they are all based on the Parlay APIs [1], which are designed as an interface for the open separation between the service logic and the call control layer.

Because Parlay supports developments for the IN, network providers are enabled to reuse their legacy services, which seems to be one of their base requirements. Chapter 2.1 gives a brief introduction into Parlay, its functionality and its advantages for the usage in softswitch architectures.

The scenario, which we have chosen for explanation of the several approaches is a call setup between circuit switched and packet switched domain. Due to this reason, the usage of a media gateway, which is the entity sending, receiving and converting data in an appropriate way, is necessary. For the control of such media gateways, IETF'S Media Gateway Control Protocol (MGCP) [9], which we describe in chapter 2.2, is proposed.

For the purpose of call control in packet switched networks, different protocols are available. Nowadays it is not certainly clear which one will be able to survive or to become industrial standard, but it seems to be the Session Initiation Protocol (SIP) [7], also created by the IETF. 3rd Generation Partnership Project (3GPP) decided in favour of SIP as call control protocol in UMTS, starting with release five. Therefore, our scenarios are based on SIP, which makes them applicable to conversions between PSTN and the mobile domain without any changes. For a better understanding chapter 2.3 overviews the Session Initiation Protocol briefly.

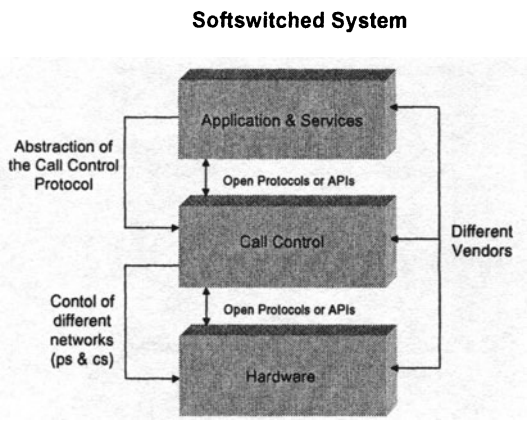

Figure 2. Softswitched Architecture Approach

In chapter 3 we describe three different design principles, which differ in the place where the entities for the generic call control reside, either common or distributed, and whether the application has to control the media gateway or if it works autonomously. Chapter 4 summarizes the conclusions.

2. RELATED PROTOCOLS AND TECHNOLOGIES

2.1 Parlay

Parlay [1] is a forum of many of the key players in telecommunications and computer industry with the goal to define object oriented APIs that allow third party application developers to access network resources and functionality in a generic and technology independent way.

The Parlay APIs have their origins in the Intelligent Network (IN) architecture [2] that is widely used in today s telephony networks. The IN service framework, however, is not an open platform for application development and IN service creation is a time consuming task and restricted to the network provider domain. The Parlay forum learned from the deficiencies of the IN architecture and separated the un-trusted service provider domain from the trusted network provider domain.

Furthermore, Parlay service capabilities are defined far beyond the scope of simple telephony and include the possibility to create powerful applications that combine the strengths of data networks with multimedia communications. A Parlay client application can be developed and run by Third Party Service Providers outside of the Network Providers core domain.

The Parlay APIs between those domains manage access to the network resources and are divided in the Framework Interfaces for security, service registration, service discovery, service subscription and management, and the Service Interfaces for the control of specific network capabilities like multimedia call control, mobility, messaging, QoS management or user interactions. Currently, the Parlay APIs are available in version 2.1 and interface definitions are published for Microsoft's DCOM and CORBA.

For the next phase of the Parlay API specification that is expected for the end of this year further extensions of the existing APIs and new network service APIs are planned like generic charging and billing, enhancement of user profile & subscription data handling, policy management, new framework interfaces, management support, new generic application interfaces and mobile E-pay.

2.1.1 Advantages of using the Parlay APIs in a Softswitch architecture

The Parlay APIs have initially been proposed for providing 3rd party service developers with a secure interface towards telecommunication resources of a network service provider. Towards the application program only a highly abstracted, object oriented and fully distributed call-processing model is visible. Existing solutions usually offer access to one specific switching resource (e.g. an IN Switching Control Point – SCP), where the calls and call legs are abstractions of the same network technology. A call that crosses networks boundaries is therefore only visible up to the gateway where the call leaves the Parlay controlled domain.

We argue that this classic Parlay usage model fits very well in the proposed Softswitch architecture where the call control layer is separated from the application and service logic layer by open and standardized interfaces. The Parlay APIs facilitate the implementation of Softswitch architectures by a mature security and management concept, by an object-oriented design and by the distributed nature of all of the interfaces.

We propose that apart from this classic 'network technology centric' usage model the Parlay APIs can be used to present a call model towards the application logic where a single instance of a call object can have call legs that use different access technologies. The Parlay Multi-Party call model defines the call leg as an abstraction of a signalling relationship between an end-point and a call object. Call legs are modelled as independent interfaces that are implemented by distributed objects (e.g. by using CORBA technology). This means that the Parlay specification allows call legs that belong to one call to be implemented by objects that run on different physical machines. A network resource that offers access through some

specific technology (e.g. circuit switched ISDN, analogue lines, SIP, H.323) can host a Parlay call leg that represents and controls this half of the call. The call legs are associated with a call that runs for example on a separate call-processing server. In this architecture a signalling gateway between different network technologies is no longer necessary as its function is implicitly carried out by a common Parlay call processing model. Thus, call set-up procedures and call control tasks are unified and simplified through a common and distributed call model. We have shown that a complete mapping of Parlay's Generic Call Control Service functionality towards Session Initiation Protocol (SIP) [7] messages is possible, so that Parlay can be used for control of a modern IP based call processing system [3], [4].

2.2 Media Gateway Control Protocol (MGCP)

Telephony gateways are network elements between circuit switched (e.g. Public Switched Telephone Network (PSTN)) and packet switched networks (e.g. Internet) that basically perform two types of tasks: signalling translation and media translation. The European Telecommunications Standards Institute (ETSI) project Telecommunications and Internet Protocol Harmonization over Networks (TIPHON) founded a decomposition model for telephony gateways, where the gateway is divided into three entities – a Signalling Gateway (SG), a Media Gateway (MGW) and a Media Gateway Controller (MGC). Nevertheless, the decomposed gateway appears to the outside as a single gateway. The gateway decomposition does not impose a physical breakdown, but rather a logical perspective. The goal of this logical perspective is to allow the signalling that enables services to be normalized into a media control interface, thereby separating services from the underlying media handling. The key advantages of this decomposition approach are the higher degree of scalability and maintainability the resulting gateway solutions provide and the opportunity to introduce new services more quickly. [5]

Each of the three entities of a decomposed gateway has to perform different tasks. The SG mediates between circuit switched (e.g. Signalling System 7 (SS7) or Integrated Services Digital Network (ISDN) [6]) and packet switched (e.g. Session Initiation Protocol (SIP) [7] or H.323 [8]) signalling protocols. The MGW focuses on the audio signal translation function, performing conversion between the audio signals carried on circuit switched networks and data packets carried over packet switched networks. Finally, the MGC acts as a manager within the telephony gateway and instructs the MGW according to the signalling information received by interfacing with the circuit switched network via the SG on the one side and

with the packet switched network via a signalling device such as a SIP server or an H.323 gatekeeper on the other side.

The Media Gateway Control Protocol (MGCP) [9] serves as an internal master/slave protocol between the MGC (master) and the MGW (slave) of an ETSI/TIPHON compliant decomposed telephony gateway. MGCP retains simplicity by utilizing an Application Programming Interface (API) that is made up of a fairly small set of eight commands, which are grouped into three basic command sets: device management, connection control and in-band signalling handling. Commands may take numerous arguments and are transmitted using plain American Standard Code for Information Interchange (ASCII) text. Like SIP and H.323, MGCP also relies on already established standards such as the User Datagram Protocol (UDP) [10] for transmitting messages between the MGC and the MGW (with the expectation that transmission reliability will be managed by the specific implementation), the Session Description Protocol (SDP) [11] for negotiating the media aspects of a call (e.g. ports and Internet Protocol (IP) [12] addresses, voice coding methods and other connection type parameters), the Real-time Transport Protocol (RTP) [13] used by the MGW for handling the transport of media streams, and the Internet Protocol Security (IPSEC) [14] protocol for providing secure connections.

The benefits of MGCP are manifold: the potential to control very large deployed systems, the opportunity of simple integration into SS7 networks, which gives greater control and throughput in handling calls and finally the fact that gateways utilizing MGCP can co-exist in networks with other SIP or H.323 compliant entities. One of the weaknesses of MGCP is the time-consuming implementation of the protocol for smaller applications.

MGCP owes its origin to the confluence of the Simple Gateway Control Protocol (SGCP) [16] and the Internet Protocol Device Control (IPDC) [17] protocol. Nevertheless, MGCP is not the final word on gateway control. The IETF combined forces with the ITU to endorse the MEGACO/H.248 [18] standard in late 2000. MEGACO/H.248 draws heavily from MGCP plus introduces several enhancements, as for instance: support for multimedia and multi-point conferencing enhanced services, improved syntax for more efficient semantic message processing, TCP [15] and UDP message transport options, text or binary encoding and an expanded definition of packages. Accordingly, MEGACO/H.248 can be considered as a richer approach to gateway control, nevertheless implementation is a lot more complex.

2.3 Session Initiation Protocol (SIP)

The Session Initiation Protocol (SIP) [7] is an application layer protocol, used for signalling in IP networks. It was developed by the Multiparty Multimedia Session Control (MMUSIC) working group of the IETF. SIP allows establishment, modification and termination of all types of sessions. It is a text-based, HTTP-like peer-to-peer protocol with two basic types of messages, requests and responses. SIP messages also carry information about the session. For VoIP applications and multimedia conferencing applications this information is usually placed in the Session Description Protocol (SDP) [11]. In that case the SDP packet is a part of the SIP message, called body. Another important IETF protocol for VoIP is the Real-time Transport Protocol (RTP) [13]. It is used for the transport of real-time media data (voice and video) in an established session.

There are two types of entities in SIP, SIP User Agents (UA) and SIP network servers. A SIP UA could be seen as end device and acts either as user terminal or as automated connection endpoint, for instance a call answering machine. Network servers are used for call routing and for maintaining call states, as well as they could be enabled to perform different kinds of applications. They are divided into three different types, the proxy server, the redirect server and the register server. A SIP proxy server is an application layer router that forwards SIP messages after address resolution. SIP redirect servers reply incoming SIP request with a new user location. After receiving a response from a SIP redirect server, a SIP UA will try to reach the called party at the new location.

Call setup in SIP is quite simple. Suppose there are two users (A and B) and user A wants to set up a call to user B. User A sends an INVITE request to B. If B wants to accept the call, he replies with OK response. The last step is the sending of an acknowledge (ACK) request by A, after the response arrives from B. As we can see, SIP uses a three-way handshake (INVITE, OK, ACK) for the call setup. If one party wants to tear down the call, it sends a BYE request to the opponent UA, which replies with an OK response and the call is terminated.

3. DIFFERENT SOFTSWITCH ARCHITECTURES USING PARLAY

In this chapter we propose and compare three possible softswitch architectures, which differ in some specific parts. The scenario we used to explain the functionality is the same for every approach. A user in the PSTN wants to call another one, who has configured a call routing application in

the way that the call is forwarded into the IP domain, more exactly to a certain SIP address.

The application resides in the service provider's domain and offers two access points. The first enables the user to configure his service via a HTTP interface, for instance to decide to which SIP UA a call to his PSTN number should be forwarded. The other, offering the Parlay API, is used by the call control components to access the application, e.g. for the purpose of address translation or for the interaction towards the media gateway controller.

Call control components are separated in two parts, the generic and the specific call control part. The generic call control part represents the call towards the application, but is independent of the underlying signalling network. It interacts with the specific call control, which represents a certain call leg towards the protocol stack of the corresponding network via an open interface.

The media gateway controller gets its tasks either by the application itself using the Parlay interface, or via the generic call control part. In that case it works autonomous, as described in chapter 3.3, performing the signalling towards the different networks by its own. It controls the media gateway, which actually does the media conversion between the involved networks and deals with the appropriated transport protocols or mechanisms.

As mentioned above, the call should be forwarded from the PSTN domain into the SIP domain. Due to this, ISUP and SIP protocol stacks take place beneath the specific call control parts.

We compare the architectures in the degree of complexity in the application versus that in the generic call control. We suggest that application programmers should not necessarily have to understand call or network specific parts. The required traffic over the open interface towards the service implementations is another relevant point, because such interaction is probably remote and so it deteriorates call setup time intensely. Moreover, it is reasonable that the application exists only once, whereas different call control facilities have to interact with it. So we can say it is important to have very low traffic here. Finally, it might be important that the single blocks could be independent, which means that the ideal case is being able to combine components of different vendors.

3.1 Common Generic Call Control Unit

The most important detail of this architecture is the necessity for only one Generic Call Control (GCC) unit. It communicates on the one hand with different protocol specific Call Control (CC) units and on the other hand with the application as shown in figure 3. The interface between the GCC unit and the application is Parlay. Note that the application makes usage only

of the GCC and therefore its Parlay implementation requires only support for the generic call control interface and of course for the framework.

The GCC unit has to maintain states for every protocol involved in a call, what makes it rather complex. It represents the call towards the application while the single call legs are maintained by the specific CC components. This causes a distributed management of call states.

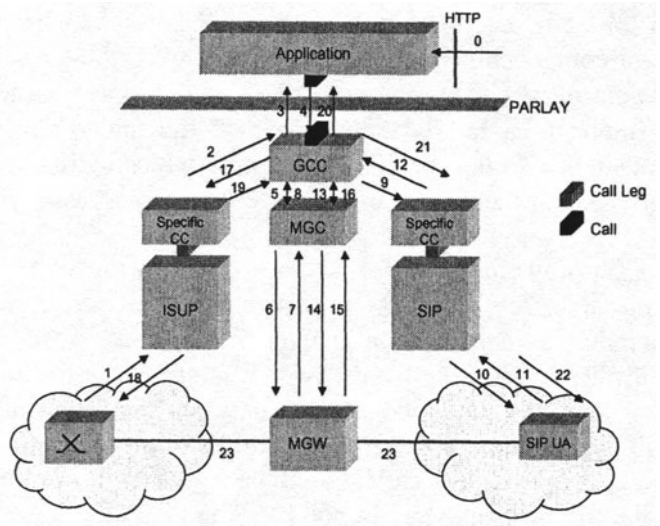

Figure 3. Call Setup Scenario with common Generic Call Control Unit

Consider an incoming call from the ISDN side. Let us assume that the destination for this call is placed in the IP domain. The following sections give a detailed explanation of this scenario shown in figure 3. For the remaining two architectures we will describe only the differences to this part.

At first, after the deployment and setup of the application, the subscribed users are enabled to place some user related configurations, e.g. over HTTP (0). Afterwards, a call-setup request coming from a PSTN switch is delivered to the ISUP protocol stack of the softswitch (1). It is forwarded to the protocol specific CC entity, which generates a call leg for this call, and further to the GCC unit (2), which creates a call and contacts the application with origination address, destination address and some other parameters (3). The application finds the new destination address (where the user has defined to receive this call), generates a call representation and sends a route request to the GCC unit containing the new address (4). In our case it is a SIP address. The GCC unit recognizes the type of the address and knows that it is a PSTN-IP call.

For calls between different kinds of networks (e.g. circuit switched and packet networks) the usage of a Media Gateway (MGW) is necessary. It converts the media streams between two networks. Therefore, the GCC unit contacts a Media Gateway Controller (MGC), which tries to occupy a trunk between the MGW and the PSTN switch (5-8). If this does not succeed, a busy signal is sent to the PSTN user and the call setup ends. Otherwise the GCC unit contacts the SIP specific CC unit with the previously resolved destination address (9). The SIP specific CC unit creates the second call leg of this call and forwards the call request to the SIP stack.

The SIP stack sends a SIP INVITE request to the destination address (in this case it is the address of a SIP UA) (10). The SIP UA indicates an incoming call. Supposed the user wants to accept this call, the SIP UA sends an OK response to the SIP stack of the softswitch (11). This response is forwarded to the SIP specific CC, which notifies the GCC unit (12). The next step is to inform the MGW about the media settings of the callee via the MGC (13-16).

Afterwards, the GCC unit informs the ISUP specific CC unit that the call-setup was successful (17). This information is forwarded to the ISUP stack and further to the PSTN switch (18). Finally, the ISUP specific CC notifies the GCC unit (19) that appraises the application (20) and the SIP specific CC (21) that the call-setup succeeded. The SIP CC forwards this information to the SIP stack and a SIP ACK request is sent to the SIP UA (22). After completion of the call-setup, media data exchange (voice, video) starts (23).

The big benefit of this architecture is that the application communicates only with one entity (GCC unit). Therefore, the programmer of the application does not need to care about underlying network issues. He simply always delegates this duty to the common GCC unit. This abstraction simplifies service creation, which reduces time to market. Another important benefit is the low traffic over Parlay. We suppose that function calls over this API would be remote in most cases. Economizing this traffic enables better call setup times.As a drawback of this architecture we notice that the GCC unit, protocol specific CC, protocol stack and MGC must be obtained by the same vendor, because of proprietary interfaces, and a more or less distributed state-machine between the CC entities. It means that there is no possibility to combine protocol stacks of different vendors. Besides that, we should remember the complexity of the GCC unit.

3.2 Distributed Call Control

In this architecture (see figure 4) there are no proprietary interfaces, due to the usage only of the open Parlay API between all protocol specific

components and the application. Every protocol specific component contains a GCC unit and is therefore able to communicate with the application.

For the scenario mentioned in the above section, it is important that in this architecture the GCC unit of the ISUP stack (note that is not a protocol specific CC unit) contacts the application directly (2) over Parlay. The application contacts the MGC that orders the MGW to occupy a trunk to the PSTN switch (3-6). After that, the application performs a call-setup (7-10) into the SIP domain, contacts MGC to configure callee's media settings (11-14) and notifies the GCC of ISUP stack that the call-setup has succeed (15). A call representation is created in both GCC (ISUP and SIP) and in the application.Comparing this architecture to the one of chapter 3.1, we notice that protocol specific components can be obtained from different vendors, because there are no proprietary interfaces between them and every unit is enabled to have its own state machine.

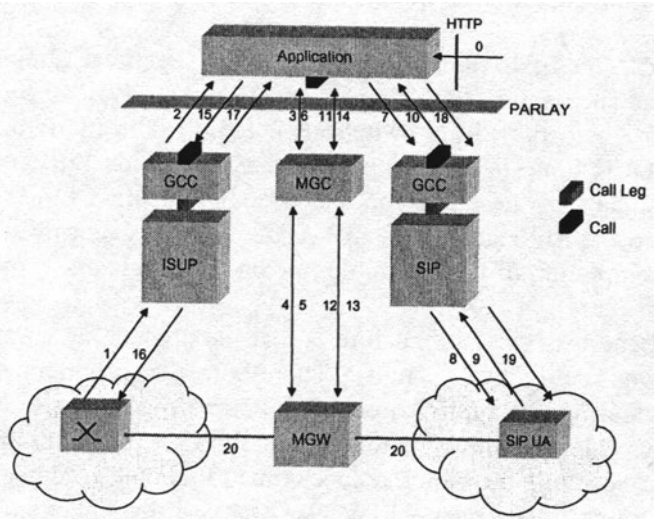

Figure 4. Call Setup Scenario with distributed Call Control

On the other hand, the traffic over Parlay is more than twice intensive, which increases call-setup times significantly. Another difference is that the complexity of the GCC has been moved to the application, which means that the application has to recognise the type of the call (e.g. IP-PSTN call) and to use the corresponding GCC unit. Furthermore, the application has to maintain states for different GCC units in a call. This complexity has to be implemented in every application (we assume independent applications), what considerably increases the developing time.

3.3 Distributed Call Control with external Gateway (Pseudo Softswitch)

The last architecture we propose is very similar to the architecture shown in the section above. The difference here is that there is no MGC component in the softswitch itself, which demands the usage of an external entity. Due to this reason we call this architecture pseudo softswitch. The call-setup is quite different in comparison to the two other architectures (see figure 5).

After receiving a call-setup request from the ISUP, the GCC of the ISUP side contacts the application (2). The application recognises that this call should be routed to the IP network and forwards it over ISUP stack to the MGC (3, 4). Note that a call is created in the application and in the GCC of the ISUP stack.

The MGC reserves resources from the MGW (5,6) and sends a SIP INVITE request to the SIP stack of the softswitch (7). Afterwards, the GCC of the SIP side notifies the application that determinates the address of the callee and contacts the SIP GCC (8, 9). Another call is created in the application and in the GCC of the SIP stack. The callee receives a SIP INVITE request and replies with an OK response (10,11). Now the SIP stack informs the MGW over the MGC (12-15) and notifies the application and the callee that the call-setup has succeeded (16,17). Finally, the MGC reports the call-setup results to ISUP stack, which forwards this report to the application and to the PSTN switch (18-20).

In this architecture we have tried to avoid drawbacks of the above section by reducing the traffic over Parlay and by making the application less complex in terms of call control. There are two calls in the application now, which can be maintained separately. This makes the application more flexible and reduces the developing time. Furthermore, the application does not need to communicate with the MGC. Protocol specific components are independent from each other and so they could be obtained from different vendors.One drawback is the usage of an external MGC component. It means that this softswitch does not support MGCP and therefore it cannot route calls between different networks without an external MGC component. Furthermore, the application has to perform the mapping between gateway addresses and user addresses.

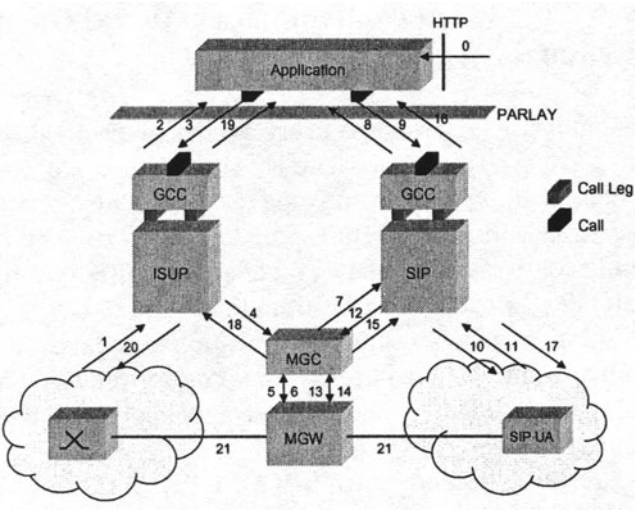

Figure 5. Call Setup Scenario with distributed Call Control and external Gateway

4. CONCLUSIONS

The world of telecommunication networks is rapidly converging. Network providers, especially circuit switched network providers, are facing the problem that they have to decide about taking the step into the packet switched domain. Such a step might be coupled with huge investments. To reduce these high costs and risks in the periods of migration, more flexible hybrid architectures might be helpful. They should enable the usage in, and the interconnection of, different types of networks in an open manner. Architectures commonly known under the term softswitch fit these requirements and, moreover, offer the possibility of network-independent service creation. In this paper we have shown three different possible realisations of such an architecture, each with characteristic advantages and drawbacks.

The design principle of chapter 3.1 enables the network provider to offer his network capabilities to so-called third party service providers in a suitable manner, because here the creation of services is very easy and the traffic to the application is very low. Network providers' own gateways could be fitted in this architecture as well. The necessary changes, mainly in the call control layer, could be severe, and so we call this an offensive architecture in terms of convergence.

Realisation 3.2 and 3.3 differ only in the controllability of the gateway. If a network provider runs his own gateways, he will try to prefer the

distributed call control instead of the pseudo softswitch. Both architectures have in common that here the legacy systems could mostly be reused. With the usage of Parlay, existing IN applications are utilizable, and, except for the integration of Parlay, no changes in call control are necessary. Support of other networks and extension of functionality could be added step by step in module manner. So we call this approach a moderate architecture in terms of convergence.

REFERENCES

[1] The Parlay Group, "Parlay Specification 2.1", http://www.parlay.org

[2] ITU-T, "Q12xx, Intelligent Network", ITU-T Standards

[3] S. Bessler, A.V. Nisanyan, K. Peterbauer, R. Pailer, J. Stadler, "A Service Platform for Internet-Telecom Services using SIP", 6th IFIP Conference on Intelligence in Networks, SMARTNET 2000, September 2000.

[4] R. Pailer, J. Stadler, "A Service Framework for Carrier Grade Multimedia Services using Parlay APIs over a SIP System", 1st ACM Workshop on Wireless Mobile Internet, WMI 2001, July 2000.

[5] Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON): Network Architecture and Reference Configurations; Phase II: Scenario 1 + Scenario 2. TS 101 313 V0.4.2. February 1999.

[6] Telecommunication Standardization Sector of the International Telecommunication Union: Integrated Services Digital Networks (ISDNs). Recommendation I.120. March 1993.

[7] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg: SIP: Session Initiation Protocol. IETF RFC 2543. March 1999.

[8] Telecommunication Standardization Sector of the International Telecommunication Union: Packet-Based Multimedia Communications Systems. Recommendation H.323. November 2000.

[9] M. Arango, A. Dugan, I. Elliott, C. Huitema, S. Pickett: Media Gateway Control Protocol (MGCP) Version 1.0. IETF RFC 2705. October 1999.

[10] J. Postel: User Datagram Protocol. IETF RFC 768. August 1980.

[11] M. Handley, V. Jacobson: SDP: Session Description Protocol. IETF RFC 2327. April 1998.

[12] J. Postel: Internet Protocol. IETF RFC 791. September 1981.

[13] Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson: RTP: A Transport Protocol for Real-Time Applications. IETF RFC 1889. January 1996.

[14] S. Kent, R. Atkinson: Security Architecture for the Internet Protocol. IETF RFC 2401. November 1998.

[15] J. Postel: Transmission Control Protocol. IETF RFC 793. September 1981.

[16] M. Arango, C. Huitema: Simple Gateway Control Protocol (SGCP) Version 1.1 Draft. IETF Internet Draft. July 1998.

[17] I. Elliott: IPDC: Media Control Protocol. IETF Internet Draft. August 1998.

[18] F. Cuervo, N. Greene, A. Rayhan, C. Huitema, B. Rosen, J. Segers: Megaco Protocol Version 1.0. IETF RFC 3015. November 2000.