

## NGN und Mehrwertdienste – Technische Lösungen

Armin Lehmann • Ulrich Trick • Steffen Oehler

**Bei Netzbetreibern und Diensteanbietern besteht zukünftig ein großer Bedarf, schnell, einfach und kostenoptimiert neue Dienste anbieten zu können. Hauptgrund hierfür ist, dass auf Basis der durch NGN (Next Generation Networks) gegebenen neuen Dienstmöglichkeiten neue Einnahmequellen erschlossen werden müssen. Nachdem im vorangegangenen Heft die Anforderungen, die verschiedenen Netze und die Funktionen von Application und Media Servern erläutert wurden, folgt hier die Fortsetzung zu Lösungen, Standardisierung und Service Delivery Platform (Dienstbereitstellungsplattformen).**

Die heute wesentlichen Techniken zur Entwicklung und Bereitstellung von Mehrwertdiensten können in zwei Gruppen von Lösungen untergliedert werden, die anhand der vorhandenen Programmierschnittstellen unterschieden werden: Mit „Low Level API (Application Programming Interface, Programmierschnittstelle)“ werden jene Techniken gekennzeichnet, die direkt auf Servern ausgeführt werden. Dagegen setzt das „High Level API“ auf Middleware auf, d. h. auf einer die Eigenheiten und Komplexität des darunter liegenden Netzes vor den Anwendungen verbergenden zusätzlichen Software-Schicht.

Zu den Lösungen mit Low Level API gehören:

- SIP-Servlets;
- CPL (Call Processing Language);

- SIP-CGI (Common Gateway Interface);
- JAIN (Java API for Integrated Networks);
- Scripting oder Software in C, C++, .NET oder Java;
- proprietäre API, beispielsweise Skype-API.

Lösungen mit High Level API sind:

- Parlay/OSA (Open Service Access), Parlay X;
- CSE (Customized Application for Mobile Enhanced Logic Service Environment);
- Web Services;
- OSE (Open Mobile Alliance Service Environment).

Auf die oben genannten proprietären API soll hier nicht näher eingegangen werden, da diese nicht auf offenen Standards beruhen. Da reine

Scripting- bzw. Software-Lösungen zu unflexibel, nicht plattformunabhängig bzw. nicht standardisiert sind, werden auch sie hier nicht näher beleuchtet.

### Lösungen mit Low Level API

#### SIP-Servlets:

Servlets sind Java-Programme, die auf Servern laufen. SIP-Servlets sind im Prinzip HTTP-(Hyper-

text-Transfer-Protocol-)Servlets, die um eine Programmierschnittstelle zur SIP-Kommunikation erweitert wurden. Die Ausführungsumgebung in Form eines sogenannten Servlet-Containers stellt ein SIP-Anwendungsserver (SIP Application Server, SIP-AS) zur Verfügung. Er enthält u. a. den SIP-Protokoll-Stack und Timer für zeitgesteuerte Ereignisse.

Empfangene SIP-Nachrichten werden entsprechend dem Inhalt einer speziellen Konfigurationsdatei, dem Deployment Descriptor, gefiltert. In der Folge werden die mit dem gewünschten Dienst korrespondierenden SIP-Servlets ausgeführt. Bild 1 zeigt den Aufbau eines SIP-Anwendungsservers mit SIP-Servlets [1].

### Auf einen Blick

**Entwicklung und Bereitstellung von Mehrwertdiensten in NGN bedingen neue Vorgehensweisen der Netzbetreiber und Diensteanbieter. Unterschiedliche Lösungsansätze zeichnen sich ab; ihre Vor- und Nachteile müssen sorgfältig abgewogen werden, denn sie beeinflussen den Aufbau der „Service Delivery Platform“.**

Die Vorteile der SIP-Servlets sind, dass sie in Threads, nicht in Prozessen laufen und persistent sind, was eine höhere Ausführungsgeschwindigkeit mit sich bringt. Servlets bieten ein hohes Maß an Sicherheit, da sie innerhalb des SIP-Serverprozesses laufen und daher nur über den Server selbst erreichbar sind. Weiterhin bieten sie alle Vorteile der Java-Technik, wie z. B. die Plattformunabhängigkeit.

#### Call Processing Language (CPL):

CPL ist eine Programmiersprache auf der Basis von XML (Extensible Markup

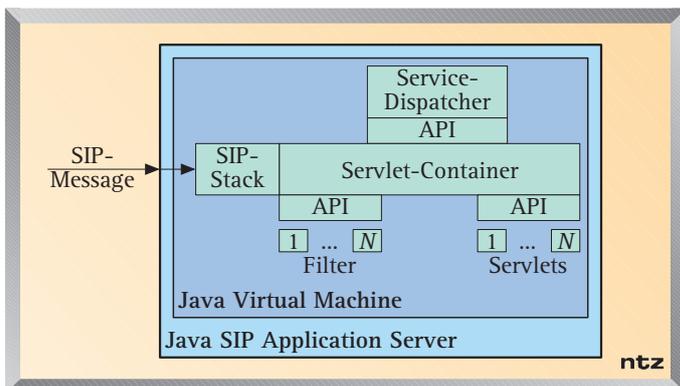
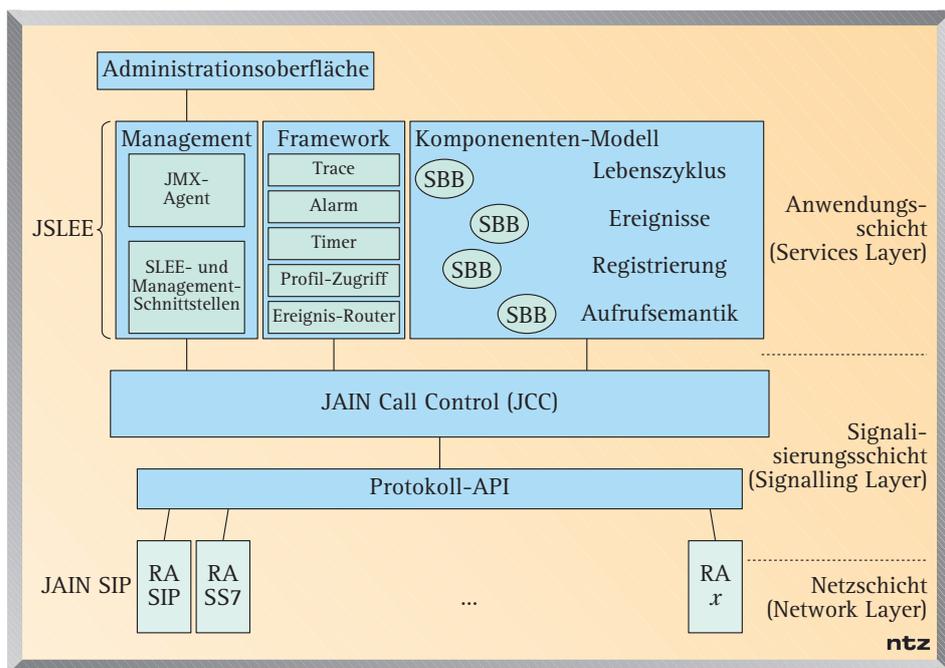


Bild 1. Java SIP Application Server mit SIP-Servlets



**Bild 2.** JAIN-SLEE-Architektur

Language), die zur Beschreibung von Multimediadiensten entwickelt wurde [2]. Sie zeichnet sich durch die Unabhängigkeit vom Betriebssystem und dem Signalisierungsprotokoll aus. Durch Einschränkungen, die im Befehlssatz hinterlegt sind, können nur die dort bekannten Aktionen ausgeführt werden. Dadurch bietet CPL ein hohes Maß an Sicherheit. Proprietäre Erweiterungen des Befehlssatzes sind möglich, diese können aber die Sicherheit wesentlich beeinträchtigen. Mit grafischen Editoren können auch die Endnutzer selbst in CPL ihre eigenen Dienste programmieren; dies kann ein wesentlicher Vorteil sein.

**SIP-CGI (Common Gateway Interface):**  
SIP-CGI stellt eine sprachneutrale Schnittstelle dar, die Interaktionen auf SIP-Anwendungsservern mit Programmen bzw. Skripten ermöglicht [3]. Vorteil hierbei ist, dass prinzipiell alle Programmier- bzw. Skriptsprachen zu verwenden sind, solange diese durch den SIP-Anwendungsserver ausgeführt werden können. Die Daten der eintreffenden SIP-Nachrichten werden an die ausführenden Programme übergeben. Da CGI-Skripte normalerweise die gleichen Zugriffsrechte auf Ressourcen des Servers wie andere Serversoftware haben, sollten aus Sicherheitsgründen Dienste nur durch den Dienstanbieter selbst erstellt werden. Ein großer Nachteil von SIP-CGI kann die relativ geringe Ausführungsgeschwindigkeit sein, vor allem wenn Skriptsprachen wie z. B. Perl benutzt werden.

### JAIN (Java API for Integrated Networks):

Mit JAIN wird das Ziel verfolgt, ein Framework für hochportable und konvergente Dienste zur Verfügung zu stellen. Dafür gibt es eine Anzahl sogenannter Network API wie z. B. JAIN SIP oder JAIN ISUP (ISDN User Part). Diese sind Low Level API, da ein direkter Zugriff auf alle Einzelheiten der Protokolle ermöglicht wird. Das JAIN-Konzept baut gemäß Bild 2 auf drei Schichten auf:

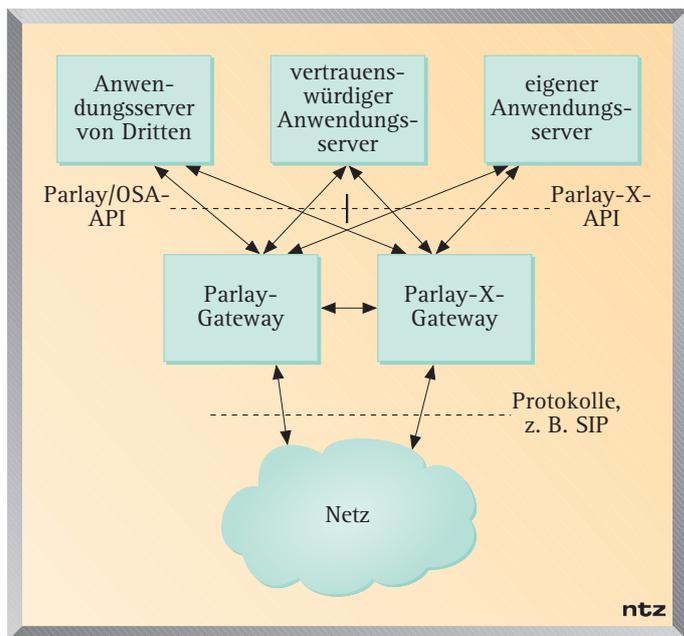
- **Netzschicht (Network Layer):** umfasst Schnittstellen (Resource Adaptor, RA) zu verschiedenen Netzen.
- **Signalisierungsschicht (Signalling Layer):** bietet die erforderliche Vermittlungslogik (Call Control).
- **Anwendungsschicht (Service Layer):** repräsentiert die Dienstlogik und Ausführungsumgebung (JSLEE, JAIN Service Logic Execution Environment) und optional das SCE (Service Creation Environment).

JSLEE spezifiziert eine universelle Ablaufumgebung für Telekommunikationsdienste [4]. Sie setzt dabei auf drei Kernbereiche [5]: Das Management wurde als JMX (Java Management Extension) [6] standardisiert und verwaltet die Dienste, ihre Bereitstellung und die Profile. Die Dienste selbst werden als kombinierbare Service Building Blocks (SBB) realisiert. Ihr Aufruf und ihr Zusammenspiel werden vom sog. Framework gesteuert. Die JAIN-Technik lässt sich auch nutzen, in

dem die Resource Adapter (z. B. JAIN SIP) mit SIP-Servlets anstelle von JSLEE verwendet werden. Die besonderen Vorteile der JAIN-Technik sind ihre Flexibilität, ihre Plattformunabhängigkeit und eine relativ hohe Leistungsfähigkeit.

## Abkürzungen

3GPP	Third Generation Partnership Project
AAA	Authentifizierung, Autorisierung und Abrechnung
API	Application Programming Interface
BSS	Business Support System
Camel	Customized application for mobile network enhanced logic
CGI	Common Gateway Interface
Corba	Common Object Request Broker Architecture
CPL	Call Processing Language
CSE	Camel Service Environment
ETSI	European Telecommunications Standards Institute
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IM-SSF	IP Multimedia Service Switching Function
IMS	IP Multimedia Subsystem
IN	Intelligent Network
ISC	IMS Service Control
ISUP	ISDN User Part
JAIN	Java API for Integrated Networks
JCC	JAIN Call Control
JMX	Java Management Extension
JSLEE	JAIN Service Logic Execution Environment
MMS	Multimedia Messaging Service
NGN	Next Generation Networks
OMA	Open Mobile Alliance
OSA	Open Service Access
OSE	Open Mobile Alliance Service Environment
OSS	Operations Support System
RA	Resource Adaptor
S-CSCF	Serving-Call SessionControl Function
SBB	Service Building Blocks
SCE	Service Creation Environment
SDP	Service Delivery Platform
SIP	Session Initiation Protocol
SMS	Short Message Service
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SS7	Signalling System # 7
TISPAN	Telecoms & Internet converged Services & Protocols for Advanced Networks
UML	Unified Modelling Language
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
XML	Extensible Markup Language



**Bild 3.** Parlay/OSA- und Parlay X-Architektur

## Lösungen mit High Level API

### Parlay/OSA (Open Service Access) und Parlay X:

Die Parlay-Techniken [7] sind in UML (Unified Modelling Language) spezifiziert. Allgemein wurden hier verschiedene Standardschnittstellen definiert, d. h. keine Anwendungsserver (AS), sondern eher Gateways. Ein Parlay/OSA-Gateway setzt Protokolle wie z. B. SIP in das sogenannte OSA-API um. Darüber können

dann Anwendungsserver mittels meist Corba-basierter (Common Object Request Broker Architecture) Schnittstellen angesprochen werden. Das Gateway selbst dient hierbei als Middleware, um einen gesicherten und abstrahierten Zugriff auf Netzfunktionen zur Diensterealisierung anzubieten. Ein großer Vorteil dieser Technik ist, dass Anwendungsserver (AS) von Dritten sicher an ein NGN/IMS angebunden werden können, da OSA selbst die Erkennung (Discovery), Authentisierung, Registrierung und eine Zugriffssteuerung bietet;

S-CSCF (Serving-Call Session Control Function) bietet dies nicht für solche „fremden“ Anwendungsserver.

Die alternative Lösung, Parlay X, beruht auf der Web-Services-Technik, auf die weiter unten noch eingegangen wird. Beide Techniken können auch kombiniert benutzt werden.

Die wichtigsten Vorteile dieser beiden Lösungen sind das hohe Maß an Sicherheit und die Kombinierbarkeit von Web Services mit Parlay/OSA-Funktionen. Ein Nachteil ist die geringe Anzahl von Softwareentwicklern mit Corba-Kenntnissen.

### CSE (Camel Service Environment):

CSE beruht auf Camel-IN-Funktionen (Customized application for mobile network enhanced logic – Intelligent Network) in Mobilfunknetzen. Mit CSE wird die Nutzung vorhandener IN-Dienste auf Camel-Basis ermöglicht. Um die im CSE bereitgestellten Dienste in NGN/IMS (Next Generation Networks/IP Multimedia Subsystem) nutzen zu können, bedarf es einer Protokollumsetzung. Diese übernimmt die IM-SSF (IP Multimedia Service Switching Function). Hier werden die beiden Protokolle SIP (an der ISC-Schnittstelle, IMS Service Control) und CAP (Camel Application Part) ineinander konvertiert. Typische Dienste des Camel-IN sind z. B. SMS (Short Message Service), MMS (Multimedia Messaging Service) und Televoting [8, 9]. Der Vorteil dieser Technik liegt ausschließlich darin, bereits entwickelte Camel-gestützte Dienste weiterhin nut-

zen zu können. Neuentwicklungen mittels dieser Technik sind kostspielig und daher nicht zu empfehlen.

### Web Services:

Web Services sind verteilte Software-Anwendungen, die sich an der SOA (Service Oriented Architecture) orientieren, Bild 4. Die Web Services verwenden standardisierte Schnittstellen (WSDL, Web Service Description Language) und Protokolle (SOAP, Simple Object Access Protocol). Web Services sind unabhängig von der Programmiersprache, der Ausführungsplattform und dem Transportprotokoll (z. B. SIP oder HTTP). Auf der Basis von XML-Nachrichten vereinen Web Services verteilte und objektorientierte Programmierstandards.

Besonders hervorzuheben ist, dass Web Services untereinander kompositionsfähig sind: Ein Web Service kann mit einem anderen zusammenarbeiten, um einen komplexeren Dienst zu ermöglichen. Aber auch die Nachteile der Web Services sollen nicht unerwähnt bleiben. Aufgrund des Overheads, der durch Protokolle wie z. B. SOAP entsteht, leidet die Leistungsfähigkeit [10].

### OSE (Open Mobile Alliance Service Environment):

Die Open Mobile Alliance (OMA) spezifiziert offene globale Standards für netzunabhängige Anwendungen, vor allem für Mobilfunknetze. Hauptanforderungen sind die Unabhängigkeit von Betriebssystemen, Ausführungsumgebungen, Programmiersprachen und Herstellerplattformen sowie die Interoperabilität zwischen Geräten und über Netze hinweg (Roaming), zwischen Infrastrukturen und Diensteanbietern. Dabei sollen bereits vorhandene Techniken wie IMS genutzt werden [11]. Die OMA definiert in OSE im Wesentlichen sogenannte Enabler (ähnlich den SBB der JAIN-Technik) und Applikationen. Ein Beispiel für einen Enabler ist die Präsenz-Funktion. OSE spezifiziert, wie die Enabler zusammenarbeiten und wie diese über standardisierte Schnittstellen ihre Ressourcen bereitstellen.

Die OSE soll einen einfachen, sicheren und geschützten Zugang zu den Netzressourcen ermöglichen. Die Anwendungen realisieren unter Nutzung der Enabler die eigentlichen Kommunikationsdienste. Dabei kann die Realisierung z. B. mittels Anwendungsservern innerhalb der OSE oder außerhalb („3rd party“) vorgenommen werden.

Dipl.-Ing. (FH) Armin Lehmann ist in der Forschungsgruppe für Telekommunikationsnetze an der Fachhochschule Frankfurt/M. tätig.  
E-Mail: lehmann@e-technik.org



Prof. Dr. Ulrich Trick leitet die Forschungsgruppe für Telekommunikationsnetze an der Fachhochschule Frankfurt/M. E-Mail: trick@e-technik.org



Dr. Steffen Oehler leitet bei Detecon International in Bonn die Gruppe Packet Network Architecture & Services. E-Mail: steffen.oehler@detecon.com



