LEABHARLANN CHOLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH | TRINITY COLLEGE LIBRARY DUBLIN
Ollscoil Átha Cliath | The University of Dublin

Terms and Conditions of Use of Digitised Theses from Trinity College Library Dublin

Copyright statement

All material supplied by Trinity College Library is protected by copyright (under the Copyright and Related Rights Act, 2000 as amended) and other relevant Intellectual Property Rights. By accessing and using a Digitised Thesis from Trinity College Library you acknowledge that all Intellectual Property Rights in any Works supplied are the sole and exclusive property of the copyright and/or other IPR holder. Specific copyright holders may not be explicitly identified. Use of materials from other sources within a thesis should not be construed as a claim over them.

A non-exclusive, non-transferable licence is hereby granted to those using or reproducing, in whole or in part, the material for valid purposes, providing the copyright owners are acknowledged using the normal conventions. Where specific permission to use material is required, this is identified and such permission must be sought from the copyright holder or agency cited.

Liability statement

By using a Digitised Thesis, I accept that Trinity College Dublin bears no legal responsibility for the accuracy, legality or comprehensiveness of materials contained within the thesis, and that Trinity College Dublin accepts no liability for indirect, consequential, or incidental, damages or losses arising from use of the thesis for whatever reason. Information located in a thesis may be subject to specific use constraints, details of which may not be explicitly described. It is the responsibility of potential and actual users to be aware of such constraints and to abide by them. By making use of material from a digitised thesis, you accept these copyright and disclaimer provisions. Where it is brought to the attention of Trinity College Library that there may be a breach of copyright or other restraint, it is the policy to withdraw or take down access to a thesis while the issue is being resolved.

Access Agreement

By using a Digitised Thesis from Trinity College Library you are bound by the following Terms & Conditions. Please read them carefully.

I have read and I understand the following statement: All material supplied via a Digitised Thesis from Trinity College Library is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of a thesis is not permitted, except that material may be duplicated by you for your research use or for educational purposes in electronic or print form providing the copyright owners are acknowledged using the normal conventions. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

A Policy Based Framework for Real Time Charging in Next Generation Networks

by

Brian Lee

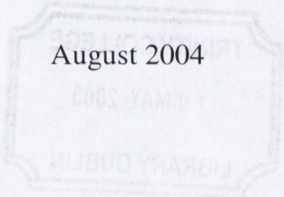
Submitted to the Department of Computer Science in partial fulfilment of the requirements of the degree of

Doctor of Philosophy

at

The University of Dublin, Trinity College

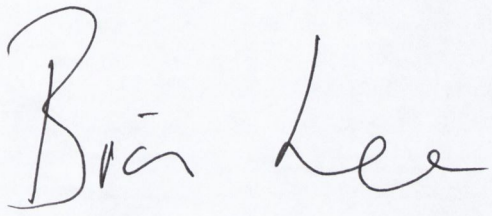
August 2004



TRINITY COLLEGE
19 MAY 2005
LIBRARY DUBLIN
TH0815
7568

Declaration

I declare that the contents of this thesis are entirely my own work and that the thesis has not been submitted for a degree at this or any other university heretofore. I agree that the library of the university may lend or copy the thesis upon request.



Brian Lee, August 2004

Acknowledgements

I would like to thank my supervisor, Dr. Donal O'Mahony for his support and help over the long years it has taken to complete this work. He has always been available for advice and guidance and it has been very rewarding to have had him as my supervisor.

I would also like to thank my parents for making it all possible in the first place by creating the opportunity for me to embark on a career in engineering. I dedicate this work to the memory of my mother, Katherine. She was ever a source of inspiration and encouragement.

Finally I would like to thank Loretto and the rest of the gang for all their support and sacrifice over all the years. We have a lot of weekends to catch up on!

Abstract

The telecommunications landscape is undergoing a period of dramatic change. This change has been ongoing for a number of years and is driven by a combination of industry deregulation and technology advance. The net effect of these two trends has been to greatly increase the variety of services on offer, as well as the number of providers in the marketplace.

A near term next generation network (NGN) is emerging characterised by a rich set of services and a dynamic and competitive marketplace where innovation and time to market will be critical success factors. Improved quality of service (QoS), multimedia services such as IP telephony, multi-party gaming etc. and delivery of value added content from third party service providers will become the norm.

Changing consumer-provider relationships, competitive pressures and new e-commerce technologies will accelerate the use of real time payment and prepayment techniques. This will increase the demand for real time processing of charging data. .Powerful, new, charging support systems will be needed to enable real time charging of these new services.

Current approaches to IP service charging systems development are an evolution of the traditional PSTN approach to charging. While these systems represent an advance on the state of the art they remain based on the "call data record", (CDR), paradigm and will not provide the scalability and flexibility needed for NGN charging. A more radical approach is needed which pushes elements of the charging process, e.g. rating, down into the network layer and allows for a more distributed and scalable charging system.

The central theme of this thesis is that the use of programmable networking technologies, specifically *active networking* and *policy management*, can provide a distributed charging system solution which meets the above demands. Charging logic is distributed to leverage the increased processing power of network nodes and enables real-time interaction between the charging and payment systems, thereby

eliminating the need for CDR generation for many types of service. It also supports the flexible creation and deployment of new charging schemes to enable the rapid introduction of innovative new services. The principal contribution of the thesis is the development of a programmable framework, PEACH (Programmable Environment for Accounting and Charging), that facilitates the creation and deployment of charging service logic to support real time charging of new services. PEACH comprises a programming model, defined by the APPLE (Accounting Policy Programming Language) language, and an execution environment for the definition and execution of charging logic and policies.

Novel features of PEACH include:

1. The capability to define arbitrary distributed charging architectures to support the evolving communications marketplace. This is enabled by the use of active networking techniques to define the required architectural functional entities and to allow the distribution of charging logic in programmable active nodes as required in the network layer to support real time charging. One such architecture is defined to model the needs of the anticipated NGN.

2. A programming model, based on APPLE, which provides a single, data centric, formalism for the definition of both charging service logic and charging policy. APPLE combines and extends aspects of several existing programmable networking approaches to facilitate the manipulation and sharing of data between charging policy rules, data repositories and PEACH charging nodes.

3. The capability to easily incorporate new service families to the charging system. This is enabled by means of a mechanism known as a *context*. The platform facilitates speedy importation of context specific vocabularies (tariff parameters, events and other data) which are used by context charging logic (modules and policy rules). The importation process automates the generation of the platform program code needed to allow context charging logic to seamlessly access, transmit and manipulate policy parameters.

Table Of Figures

Chapter 1

1 Introduction

Major changes have occurred in the telecommunications landscape in the last twenty years. The settled, even static, business world of the public switched telephony network (PSTN) has experienced major upheaval. The end result is an environment characterised by a diverse range of new services and a large number of new service providers. These changes have been brought about by a combination of factors. Deregulation of the telecom industry has taken place on a world-wide basis. This has attracted new entrants to the service provision marketplace. Also new services have been enabled by new technologies. This has created new markets and accelerated the trend toward a more competitive marketplace. Examples of this trend are cellular telephony and the emergence of the world wide web (WWW) and the underlying Internet into the public telecom domain in the mid 1990's.

1.1 Service Mania

The Internet in particular is having a major impact on the evolution of the public telecom environment. This is due not only to the inherent flexibility and economy of packet based networks to transport multiple data types but also to the fundamental Internet design philosophy of pushing intelligence to the network edge. These factors, combined with open and accessible programming models and standards, are creating a rich and dynamic service marketplace. While initially focused on data services such as the Web, e-mail and data transfer, recent trends in IP technology are positioning the Internet as the platform for a future, media rich, person to person 'next-generation network', (NGN). Improvements in quality of service (QoS) such as the differentiated services (Diffserv) and integrated services (Intserv) efforts enable services such as

real time voice and video to be provided on IP. Furthermore, protocols such as RTP (Real Time Protocol and SIP (Session Initiation Protocol) allow for multi-media related services to be introduced. Moreover technologies such as GPRS (General Packet Radio Service), or 2.5G, UMTS (Universal Mobile Telecommunication Service), or 3G, and WLAN (Wireless Local Area Network) are bringing the mobile Internet to the masses and will enable the same set of services to be offered over both wired and wireless networks. Services such as mobile multimedia, online gaming, location-based information, wireless e-commerce and so on are anticipated to be common place in near term wireless networks.

As the range of services evolves so too does the structure of the market place in which they are offered. New types of service provider arise leading to new forms of alliance and competition. In the early days of the Internet providing basic access was the main business for Internet Service Providers (ISP's) and this remains a staple for most of them today. There are a number of different service provider types in the Internet access market place namely ISP and NSP (Network Service Provider). The NSP provides a wholesale service to other service providers and does not have a business relationship with the end user. Similarly the world wide web has allowed a new set of business relationships to be created in the provision of content and other services to end-users/consumers across the Internet This has given rise to a new player, the Application Service Provider (ASP). Typically an ASP provides enterprise customers with access to front and back end office applications across the Internet (cf. www.salesforce.com) . The benefit to end users is that they don't always have to purchase the latest software packages and are relieved of having to maintain these packages on an ongoing basis. A variant of the ASP concept is the so-called Communications ASP (CASP) who provides enhanced communications services such as video conferencing or presence enabled communication services on top of basic Internet access (e.g. www.tellme.com., www.webex.com). In general ISP's and ASP's will need to partner in order that application services can be delivered with the required speed and user perceived quality of service. As the marketplace for IP services grows and the relationship between ISP and ASP becomes more of a fact we are likely to see the emergence of *service portals* i.e. web sites which offer on-demand IP services which are combined offerings from both ISP and ASP. Service portals could be operated by an ISP or by any third party acting as a service broker.

The emergence of a variety of service provider types will increase the likelihood of a dynamic relationship between customer and service provider whereby communications services are increasingly purchased on a once-off basis rather than as part of a contractual relationship between supplier and consumer. The increasingly competitive marketplace will further emphasise this trend as service providers seek to differentiate themselves by providing innovative services and to enable the rapid deployment of these services by allowing users to dynamically provision services using Web interfaces. E-commerce technologies also have a key role to place in advancing this changing relationship, [Mess99]. Trading protocols, intelligent agents and micro-payment techniques, [Peir99], allows for a dynamic relationship between service purchaser and service provider, even for fixed users. Scenarios can easily be envisaged in which a service user conducts a 'reverse auction' with a number of suppliers (or consortia) to purchase a telecommunication session [Mess99].

1.2 Charging System Requirements

A natural consequence of the increased complexity of service offerings and the related marketplace has been a knock-on increase in complexity in both the networks that provide these services and the support systems that provision and manage them. In particular this thesis is concerned with the impact on the support systems used for charging of IP services. A number of issues need to be considered.

Firstly networks and services are becoming more diverse. Although IP is emerging as the de-facto standard for both network and application layer services a variety of technologies are used to provide network services. DSL (Digital Subscriber Line) and ATM (Asynchronous Transfer Mode) are widely used in access networks in the wireline world. The situation is perhaps more diverse in the wireless world where GSM, GPRS, UMTS and WLAN are (or will be) used to provide not just access to backbone networks but also complete services in their own right. On the application side "bundled" service offerings based on a combination of network service and application service are also increasing the melting pot of services on offer. The range of services that may be created in this way is limited perhaps only by the imagination of service provider marketing departments. Charging systems clearly need to be able to deal with a wide variety of services and technologies. They need to be easily adapted to new network technologies. They must be inherently flexible to allow for

definition of new, as yet unimagined, services.

Secondly the rate of change in the marketplace is very rapid. Shrinking margins and new technologies, such as SIP, will push service providers to introduce innovative new services in ever decreasing lifecycles. It must be possible to introduce charging schemes and tariffs for new services quickly in order to allow these new services to be enabled in the network. Nor is a service tariff likely to remain static. Depending on competition and other marketplace pressures service providers may want to experiment with pricing models or structures and to vary in real time the way in which services are charged for. This situation exits already today and will become more pronounced in the NGN.

A third factor is the increasing complexity of networks. Network topologies and technologies are becoming evermore elaborate. One consequence of the move toward the NGN is an increase in the number and type of network nodes. Exposing network interfaces to allow application service providers (ASP) access to the networks also introduces new network element types resulting in yet more protocols and network nodes. Network services are further complicated by the introduction of multiservice networks with different types and levels of QoS. The consequences for charging systems is that they must be able to deal with many different types of information from many different network elements, and in many cases with much more information. Indeed according to one estimate GPRS (2.5G) networks will generate up to forty times as much call charging data as a corresponding GSM (2G) network [Sur01] [1]. Charging systems will also need to be distributed to deal with the increased network load and variety of technologies.

Fourthly there is a trend towards the real time processing of charging and billing information. This is happening for a number of reasons. The introduction of "on-demand" services is enabled by technologies such as web based automated provisioning and, further out, by e-commerce technologies such as those outlined above [Mess99], [Peir99], e.g. trading protocols and intelligent agents. Introducing

---

[1] Already in current networks the need to deal with the diversity and volume of information has led to the introduction of an IP "mediation layer" for collecting network, primarily, charging data. A mediation network element collects data from a variety of sources (routers, media gateways, application servers etc), aggregates or normalizes this data into a uniform format and forwards the data to the charging and billing systems for further processing

on-demand services means that billing systems have to be updated immediately with customer details and customer details authenticated [LuLa99]. Furthermore where the relationship between customer and service provider is transitory rather than long term there is a need for the charging system to calculate and present the charge in real time. Internet services are also more variable in nature than PSTN services. A customer can for example change the level of QoS dynamically during a call or can vary the volume of data by several orders of magnitude in a very short period. This allows users to clock up high charges very quickly and increases the risk for fraud. It also complicates the credit management of prepaid and debit card models [LuLa99]. In this method of payment, currently widely used in mobile networks, up to the minute (or even second) charge calculation is essential. Such calculations are more difficult in IP based networks because of the aforementioned possibilities to quickly change QoS or volume of data flowing. Furthermore in a multi-service network a charging scheme may be based on a number of QoS parameters, e.g. jitter and bandwidth, which vary dynamically, making the calculation of charges even more difficult. Introducing the possibility of using pricing to regulate network usage in times of congestion as described above introduces the possibility of dynamic tariffs and widely varying prices for network service. Dynamic charge calculation would seem to be the only feasible means to offer services in this case as users will want to be instantly aware of any major price fluctuation in their service. All of these possibilities mean that the charging systems will need more and more to provide real time support for service provisioning and usage. One impact is the need for charging systems to be distributed in order to deal with increased processing load in a real-time charging environment. Additionally in near term mobile networks one can envisage scenarios in which charging state may need to migrate from node to node to support user mobility.

1.3 Charging System Evolution

The telecom industry is responding to the above challenges by gradually moving to a usage-based billing model for many new services and consequently is developing a billing system to support this move. In essence the emerging IP service industry is adopting many of the practises of the traditional PSTN. Network elements are being instrumented to measure IP data flows. New charging function elements such as

mediation platforms are being deployed and new standards are being defined to record data for IP services [IPDR02]. While this emerging charging/billing infrastructure has its origin in the PSTN it is clear that many changes are being made to adapt to the challenges of the IP service market place. Service diversity and the trend toward real-time processing of data for service provisioning, in particular, are placing great demands on the charging systems e.g. there is a need for real-time charge calculation to meet the needs of pre-paid billing and to prevent service fraud. This is leading to the billing system architecture becoming fragmented or unbundled as key functional components are separated out from the previously monolithic batch system. A good example is the rating engine[2]. These were traditionally embedded in backend billing systems but nowadays can be found as stand alone entities. Rating engines are rule based and are flexible enough to facilitate introduction of new billing models. As a corollary to unbundling, billing systems are also becoming more distributed with functionality being migrated from the higher layers of the billing system toward the network especially into the IP mediation layer.

While these trends are steps in the right direction towards meeting the charging requirements outlined above, (chapter 1.2), they do not go far enough. IP charging systems are still based on the PSTN call data recording (CDR) paradigm, though they are now known as IP data records or IPDR's. Generating IPDR's, [IPDR02], for every session and perhaps many times during a session and transferring that data to the charging systems is very demanding of network bandwidth and charging system processing. Bearing in mind the estimation that 2.5G and 3.G networks may generate up to 40 times the amount of charging data that 2G networks did, [Sur01], there will clearly be a tremendous amount of network bandwidth needed to process IP services in real time. This raises serious questions about the ultimate scalability of this new charging architecture. Also the round-trip delay between the traffic reporting and the charging system analysis may cause revenue leakage and decrease customer satisfaction. Neither does this new charging really meet the needs of bundled service provision and charging. The charging system architecture is inflexible and forces interworking by means of IPDR definition in one or both partner billing systems. The

[2] Rating engines are used to calculate the costs associated with service usage.

formal definition and interaction necessary for these steps slows down the ability of service providers to quickly deploy new service offerings.

To truly meet the needs of the next generation network a fully distributed charging/billing system is needed. In particular it must be possible to distribute charging processing to the network layer. Real time interaction between the charging system, payment system and user data repositories will eliminate the need for generating and communicating large amounts of charge data through the network. It will also remove the need for storage of large amounts of charging data. Furthermore distributing the charging to the network layer will take advantage of the huge amounts of computing power in network elements and, ultimately, in the increasingly powerful mobile terminals/handsets. This distribution will lead to more scalable and flexible charging solutions as charging logic can be deployed throughout the network and it will no longer be necessary to communicate large amounts of data through the network. New capabilities and features may be introduced at points where they are needed. Customised services may be introduced on a selective basis. The current limitations on interworking between charging systems due to IPDR definition need to be removed to make bundled service charging easier to achieve.

This thesis examines the use of programmable networking technologies to provide flexible, distributed charging systems that can fully meet the needs of next generation charging.

1.4 Programmable Networks

A programmable network is a network into which new features may easily be added by either the service provider or trusted third parties. Programmable networks provide direct support by the use of API's, to allow programmers to "rapidly create deploy and manage novel services in response to user demands", [Camp99]. The goal of programmable network research is to facilitate the rapid introduction of new services and features into a network as well as to allow flexibility to introduce customised services and features. These techniques have been used for both network management and end-user service provision. There are a variety of approaches to programmable networking. Two such approaches, *active networking* and *policy-based management* are of particular interest in this thesis.

Active networks are intended to allow programmability at the packet level, [Calv99]. In addition to data, packets traversing an IP network may carry programs which can be executed in active node "execution environments" thereby introducing new features or services into a network. While having similar goals a broad range of technologies and methodologies has been used in different research programs.

Policy based management is the use of rule based policy to manage network behaviour. Policies reflect the business goals of the network or service provider. Policy can be expressed at several levels of abstraction. Policies which directly express business goals are normally formulated in every day human language such as English e.g. "Gold level service users will pre-empt bronze level service users at times of network congestion". A rule is a condition-action statement that determines the appropriate action to take under certain conditions. The use of rule based policies to manage and control networks is not new, [Wies94] [Lein89], though interest in the area has been spurred recently by the work on QoS policy management in the IETF [IETF00a], [Raj99].

1.5 Goals and Contributions

This thesis investigates the uses of programmable networking techniques to meet the following goals:-

- The creation of a distributed charging system where service logic can be deployed as needed in the network layer and hence contribute the goals of scalability and performance.

- The creation of a flexible charging system to allow the definition of a wide variety of charging schemes to support the diversity of the existing marketplace and to allow for the introduction of future services.

- To allow charging policies to be defined and easily changed to meet the technological diversity of the NGN and to be adaptable for future technology introductions.

- To support the introduction of real-time charging

Policy based management enables the definition of flexible pricing models and charging

schemes. Pricing models associated with services reflect the business policies of the service provider. In a charging context policy specifies the how charging should be applied for a particular service under different conditions. Policies may be customised or adapted to meet the needs of selected customer segments or even individual customers. Expressing pricing model tariffs using interpreted policy formalisms facilitates the rapid introduction of new pricing models and easy modification of existing policies.

The active networking paradigm is attractive for the design of a distributed real time charging system to meet the needs of present and future IP service networks. Active networks facilitate the design of flexible architectures to meet the needs of the evolving marketplace. Active network techniques enable the distribution of charging logic in a number of active nodes in the network. These nodes form a *virtual active network* and can collaborate to provide charging.

The principal contribution of this thesis is a programmable framework that enables the definition and distribution of charging policies and logic thereby meeting the needs for a distributed charging system outlined in chapter 1.3.

The framework includes a number of novel features designed to optimise the development of next generation charging systems. These include:

- The capability to model open-ended, distributed, charging system architectures. The modeling approach is based on the definition of a number of well-defined functional entities. These entities act as computational nodes that cooperate as a virtual active network to enable charging of services. This allows for the distribution of charging logic in the network layer to meet performance and scalability requirements. Because the modeling approach is based on active networking, new functional entities can be defined at will. Thus the use of active networking brings an unprecedented level of flexibility to the definition of charging system architectures and will allow deployed systems to be easily adapted to meet any future need. The work in the thesis defines one such model that is predicated on a vision of how the marketplace will evolve (cf. Chapter2). The model is adaptable to multiple service provider scenarios.

- A programming model, based on APPLE, which provides a single, data centric,

formalism for the definition of both charging service logic and charging policy. APPLE is specifically defined to meet the needs of charging systems by allowing easy manipulation and sharing of data between policy rules and by providing built in support for data retrieval from network directories and databases. As such APPLE represents an advancement of the state of the art in the application of programmable approaches to charging/billing systems. APPLE distinguishes between *modules* and *rules*. Modules are the executable entities in the charging active nodes. Modules receive, and respond to, service related events and allow customized service charging logic to defined. Rules define what may be charged and what tariff should be applied. Programmable rules allow flexible tariff definition.

- A programming environment which is optimized to the needs of charging systems. The key contribution of the programming environment is the ability to easily incorporate new service and technology families to the charging system. This is enabled by means of a mechanism known as a *context*. The context mechanism facilitates speedy importation of family specific vocabularies (tariff parameters, events and other data) which are used by context charging logic (modules and policy rules). The importation process automates the generation of the platform program code needed to allow context charging logic to seamlessly access, transmit and manipulate policy parameters. The context mechanism allows new levels of customisation and extensibility in the application of programmable networking approaches to charging support systems.

These features are described in more detail in later chapters.

The structure of the thesis is as follows:

Chapter 2 examines how current communications networks are evolving due to the influence of new technologies, regulatory and market trends. A model for a near term next generation network is introduced. This model provides a framework for analysis and a description of new communication services and the service marketplace.

Chapter 3 describes pricing and accounting in legacy and current networks. It examines recent research in the area as well as emerging trends in the industry. It

highlights the complexity of charging in the NGN and makes the case for a distributed, flexible charging system, optimised for real-time charging.

Chapter 4 examines the state of the art in research approaches to programmable networking. It describes current research to network charging based on programmable networking and explains the benefits of using programmable networking techniques for NGN charging.

Chapter 5 describes PEACH. The charging functional architecture is introduced and the design and implementation of PEACH environment are described. The APPLE policy language is described in detail.

Chapter 6 describes implementation of charging functions for a number of case studies including providing QoS on a Differentiated services (Diffserv) network, downloading of MP3 files from a content provider and provision of VoIP telephony services.

Chapter 7 contains a summary and conclusions and describes possible future work.

Chapter 2

2 Next Generation Network

2.1 Introduction

The flux in the telecom industry caused by the ongoing technological and marketplace evolution has given rise to a diverse and complex public communications network. Indeed it is more accurate to consider the 'global communications network' as a patchwork quilt of networks of varying technologies rather than as a single monolithic entity. Many older technologies, which meet the needs of their marketplace, will not be replaced until market economics judge an update to be profitable. At the same time newer technologies are continually appearing as candidates for inclusion in the public communications network, though the rate of change of the network varies substantially from place to place. Nonetheless the network *is* changing and certain common trends in its evolution can be identified.

Foremost amongst these trends is the emergence of IP as the dominant network protocol for the transfer of *all* types of information. IP has emerged as the sole 'to the desktop' network service and is also increasingly replacing alternative technologies, such as ATM, as the choice for network switching.

A corollary to the previous point is the emergence of 'Voice over IP' (VoIP) type services. Voice remains the most profitable service in the network today and much work has been done to ensure that packet based networks will continue to provide reliable voice services. However VoIP is viewed not just as a replacement for the PSTN telephony service but, more importantly, as an enabler of a host of new multimedia communication services and services which combine WWW technology with real-time multimedia communication e.g. multiparty gaming.

Wireless technologies are also increasingly important both in mobile networks and in 'last mile' broadband fixed networks. Wireless LAN technology is being deployed in hot-spots in airports and other high density 'islands'. Wireless metropolitan area networks (MANs) are being touted as economical alternatives to DSL and cable.

The PSTN is being adapted to interwork with the Internet to provide a hybrid telephony network. This form of interworking can be expected to continue for many years. However very little investment is being made in circuit switched technology network equipment.

Optical technologies have undergone huge advances in the last ten years and massive investment in this technology has been made in metropolitan and long haul networks in that time. The net effect of all this investment has been to increase the bandwidth available in these networks and to drive down the price paid for bit transmission.

Spurred on by these trends a Next Generation Network (NGN) is beginning to slowly emerge from the diversity of the present network. We describe our view of this network in the remainder of this chapter. Given that change is the only constant and bearing in mind that networks will be ultimately shaped by market pull rather than technology push it is impossible to predict with certainty when this NGN will fully emerge. Certainly the bursting of the dot com bubble has removed much hype from the marketplace and forced revised predictions for the growth of new services. At this point in early 2004 signs are emerging of a slow up-tick in the marketplace and many forecasters are predicting accelerated take-up of new services during 2004/2005. It is the judgement of the author that it will take 10 years to reach the full promise of the mobile Internet but that progress toward that point will increase steadily during that period and the NGN will emerge over that timeframe.

This thesis examines the effects on the charging infrastructure arising from new interactive services in the NGN.

2.2 NGN Taxonomy

In the mass of detail surrounding the modern public network it can be difficult to relate different aspects of the network, (services, technologies etc.) to each other. In order to provide a better understanding and to provide a framework for comparison and contrast, a reference model for the NGN is presented.[5]

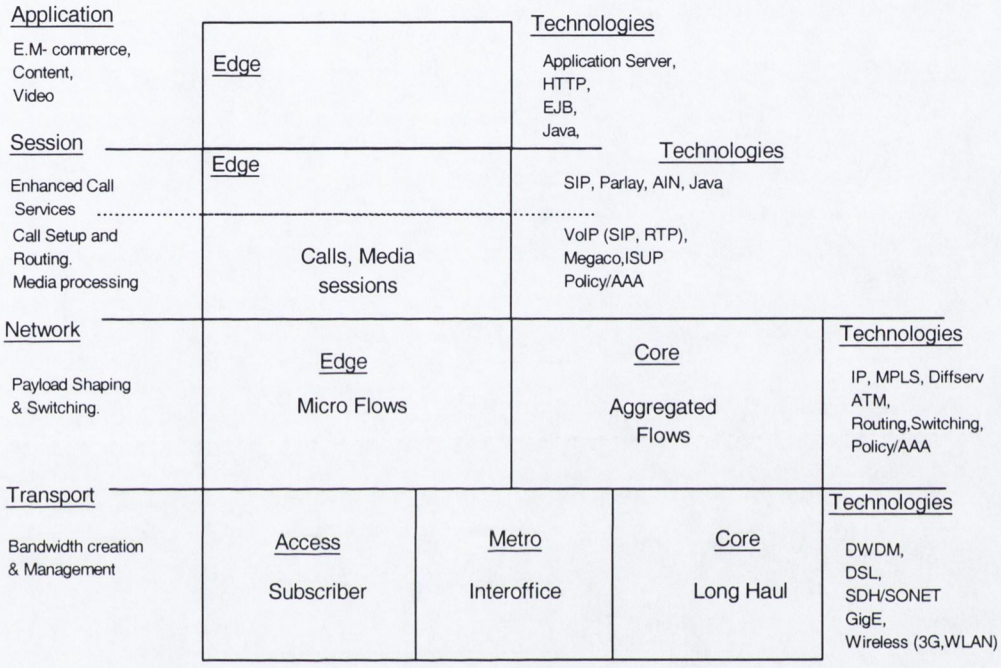


Figure 2-1Next Generation Network Reference Model

The vertical axis of the framework describes the category of service to be found in the NGN while the horizontal axis describes the category of network that will be found. The framework is separated into a number of distinct layers. Each layer is well defined in terms of the functions it contains, and the services it offers both to the end-user and the layer above it i.e. a layer may consume services offered by a lower layer to provide it's own end-user services. The framework also indicates categories or subdivisions of functionality of each layer.

---

[5] This reference model is based on a functional framework suggested by Clavenna and Heywood [Clav01] to categorise optical networking equipment

A complementary, topological, view of the reference model is shown in Figure 2-2 below

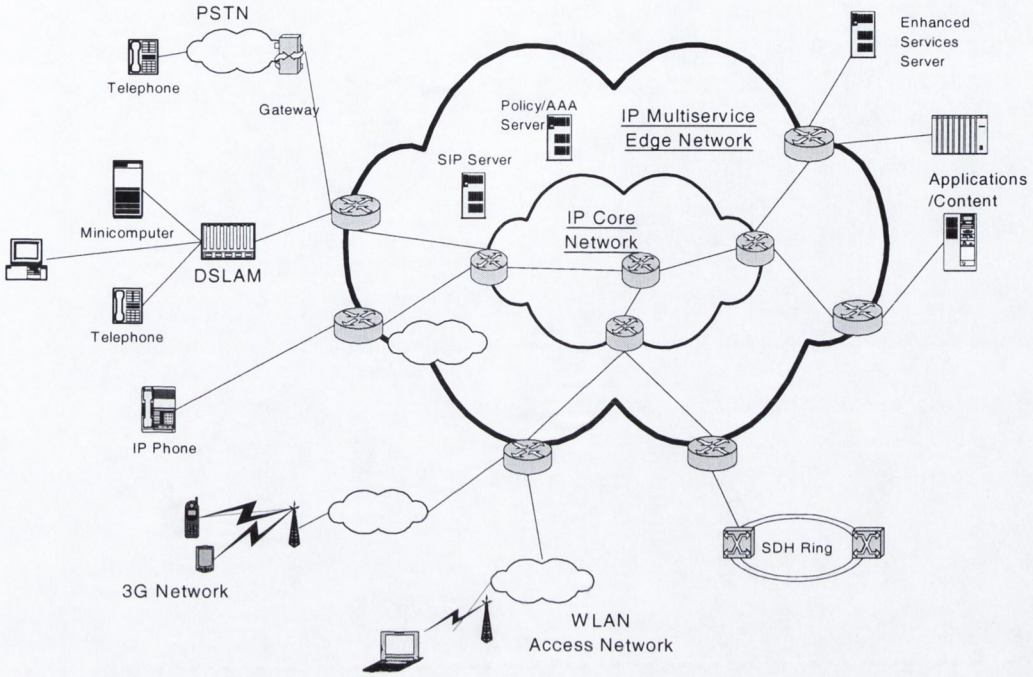


Figure 2-2 Next Generation Network

A number of points can be observed from the above diagrams:

- A variety of end-user services will be enabled by the NGN. These services will be provided by different types of service provider e.g. basic connectivity services are provided by network service provider's (NSP), while application services e.g. multiparty gaming, may be provided by application service providers (this issue is examined in more detail later). Therefore the NGN as defined in this work encompasses all functionality used to provide networked services and not just the network and transport layers.

- User services are defined at the edge of the network. The network "edge" is considered to be a combination of edge-routers and various servers used for authentication, authorisation and accounting (AAA) and/or service specific support nodes e.g. SIP servers. In the network core traffic is aggregated and

there is no individual user information. This architecture has evolved as it is considered to be the most scalable.

- End user services are increasingly being defined in terms of IP. The reason for this is simple. IP allows uniformity of service across heterogeneous networks and devices. And heterogeneity will be a hallmark of the NGN. (Some connectivity end-user services will be defined in terms of other technologies e.g. circuit switched voice or GPRS bearer services).

- Wireless technologies are becoming evermore widespread. This raises a number of questions that need to be solved, including how to provide seamless hand-off and roaming across heterogeneous wireless networks.

- Many different network types will be used to gain access to IP service networks. These include DSL, cable, WLAN, GPRS, 3G and ad-hoc/mesh networks.

The remainder of this chapter discusses aspects of the NGN in more detail, especially those that are pertinent to the provision of on-demand end-user services, since this type of service is the focus of the research of this thesis.

2.3 Access Networks

Access networks, by definition, enable end-users to access the IP edge and core networks and the services provided by these networks. Many different types of access network technology exist and the variety seems to grow continuously, especially in the wireless arena.

Fixed access networks include the PSTN, ISDN, DSL and cable as well as a variety of wireless technologies.

Access over the PSTN is via dial-up modem and service delivery is limited by the relatively narrow bandwidths, typically 56kbps, provided. ISDN networks can provide up to 128 kbps and offer a more seamless interaction, though dial-up is still needed. The primary function of the PSTN is still, of course, the provision of basic circuit-switched telephony (which is not included in the scope of the current research)

DSL and cable are largely superseding the PSTN as the wireline access networks of choice. DSL refers to a family of technologies that run over PSTN copper lines and can deliver data at rates up to 51 Mbps (Very high data rate DSL – VDSL) though delivery rates of 1.5 to 2 Mbps are more common (Asymmetric DSL). DSL is widely deployed worldwide and deployment rates are accelerating. Cable television networks are an alternative access method to the NGN and provide (downstream) rates of 6-9 Mbps. Cable access is particularly popular in the US.

Wireless technologies are usually associated with mobile networks but wireless also plays an important part in providing fixed network access. Networks of this category are sometimes referred to as "broadband wireless access", (BWA), networks (Note: previously the term "Wireless in the local loop",(WLL), was also widely used). Historically BWA networks were based on proprietary solutions. Recently the industry has begun to move towards using standard solutions and seems to be converging on IEEE 802.16, [IEEE16], and its derivatives, as the technology of choice for BWA networks. 802.16 offers speeds of up to 120 Mbps. 802.16 variants offer mesh networking (801.16a) and mobility (802.16e).

Mobile networks have evolved from cellular telephony based on GSM and other 2G technologies. The current state of the art networks offer data capability in the form of GPRS networks while 3G networks based on UMTS and cdma2000 technologies are beginning to be deployed. 2.5G and 3G networks offer circuit switched telephony as an evolution of 2G but are also architected to provide IP based communication services i.e. the so called "All IP" approach. Over the lifetime of the NGN IP services will be the primary services delivered by these networks. 3G networks can provide access rates of up to 2Mbps but in practise rates of up to 384kps are expected to be the norm - from a both a cost and efficiency of resource usage point of view.

In addition to cellular systems other mobile wireless technologies are being deployed. Foremost amongst these are wireless local area networks. WLANs based on the IEEE 802.11, [IEEE11], standard in particular have become very widespread during the past two years. WLAN's offer access rates of up 54 Mbps although average usage rates are likely to be at rates of less than 10Mbps. In contrast to mobile telephony networks WLANs can be deployed by anyone and are widely

used in both home and enterprise scenarios. They are also being deployed in many public spaces such as cafes, airports etc. where they can provide access to NGN services to "road warriors".

There is a general agreement that, from a wireless perspective, current access networks will evolve to a fourth generation network, (designated as 4G or "Beyond 3G" B3G) based on IP services and providing the capability for seamless hand-off and roaming across heterogeneous networks, [OMah03], [Lach03]. There is not universal agreement on the rate of evolution of toward 4g networks. O'Mahony, [Omah03], notes that there are two opposing points of view on how this evolution will occur. The telecommunications industry envisages that 4G should incrementally evolve naturally from 3G without any major architectural changes taking place while the research community believes that a more fundamental and radical approach is needed and enabled by current technologies. In this latter view the IP edge and core network will totally displace the PSTN, 2G and 3G networks. Ad-hoc networking will be used to create networks for local communication and to provide links to fixed access points to the core network.

It is the authors' point of view that, irrespective of the technological merits of either side, the evolutionary approach will, for economic reasons, predominate in the near to medium term. Take up of 3G networks has not been as great as anticipated and will, in all likelihood, never approach the penetration previously anticipated. WLANs will be increasingly deployed. IP will be the basis of service provision and mobility. Ad-hoc networks based on 802.11, 802.16a or other radio technologies will be deployed in a variety of contexts including public access to the IP core. All of these will co-exist. However significant investment has been made in 2.5 G and 3G technologies and network operators will want to recoup this investment. Nor is it proven that WLAN radio technologies can provide a robust solution for large-scale voice and multi-media services. (Indeed the standard for QoS in 802.11 networks (802.11e) has not yet been approved.) Significant challenges remain to be solved to ensure interrupt free handover when IP based handoff is used between radio cells. Undoubtedly all of these will be solved in time but it is premature, in my view, to predict the demise and disappearance of 3G.

2.4 Network Layer

The network layer offers connectivity and routing services i.e. users can transfer information to, or between, addressable entities. Typical technologies are IP, ATM and Frame Relay and units of information are packets, cells and frames.. ATM is currently widely deployed and will continue to exist in networks for some time to come. With the exception of some managed connectivity services for large customers ATM is not visible to end-users. Furthermore ATM is being superseded by IP in the network infrastructure and consequently the role of ATM will be minimal in the next generation network. We therefore restrict our discussion to IP services.

IP today offers a single network service, or "bearer service" in ISDN terminology, based on best-effort data transmission. Packets are transferred on a first come first served basis. Best effort service has been very successful, as can be inferred from the runaway growth of the Internet in recent years. Best effort service is not however generally considered adequate to transfer real-time information such as video and voice. To enable these types of information to be transferred in a next generation network IP will have to offers multiple types of network service i.e. the Internet will evolve to a so-called multiservice network. Stated another way IP will have to offer different levels of Quality of Service or QoS [Ferg98]. QoS essentially means that some packets will be given priority over other packets i.e. higher priority packets will be transferred ahead of lower priority packets. A variety of proposals exist for the implementation of QoS in IP networks, [IETF94], [IETF98].

As a QoS enhanced IP network becomes the bedrock for mainstream communication services a number of other issues will have also to be addressed. Service usage will become more dynamic as users will wish to allocate network resources for time-spans ranging from seconds to hours. A variety of resource provisioning mechanisms will be needed to meet these requirements ranging from service provider manual provisioning, through customer self provisioning using Web portals to on demand allocation using signalling protocols from user terminals. Additionally service providers will want to generate revenue from offering enhanced services. This will lead to the need for a policy infrastructure to authenticate, authorise and account for service usage i.e. an AAA infrastructure,

[Metz99]. In order to allocate resources, police and charge users it will be necessary to distinguish individual user packets in the network, Packets will be categorised into '*flows*' to allow these needs to be met.

A variety of criteria can be used to distinguish flows. Routers in the network will need to be augmented with *packet filters* to perform flow categorisation. Because the volume of packets increases tremendously as one moves toward the core of the network most filtering will be done in the routers at the edge of the network. Aggregation of flows will take place as packets move into the core. So we distinguish between edge network micro flows which track individual user packets and core network macro flows which track classes of packets.

End-users services will, for the most part, be based on the classification and treatment of microflows in the edge network. Typical services will be bearer services based on a combination of QoS, bandwidth, treatment of non-conforming packets and, course, pricing. VPNs will also be an important service and auxiliary services such as firewall, encryption and transcoding may also be offered.

Configuration of traffic conditioning elements represents flow-state in the network. Scalability is achieved because state is maintained only in the edge of the network. Differentiated services (Diffserv) may be allocated for a range of lifetimes ranging from seconds (dynamic) to days/weeks/months (static). Static configurations will be installed by the service provider via management systems or by the customer via a web portal. Dynamic configurations will normally be installed via signalling protocols between the customer host and the network. To enable customer based provisioning, whether static or dynamic, it becomes necessary to maintain information about the customer in the network. This information includes details on the customer's service level agreement (SLA) i.e. what services the customer has subscribed to and how traffic is to be treated. It also includes information on charging etc. Each time a customer requests a new service allocation the network database is consulted to ensure that the request is allowable. A new *policy management* infrastructure has been introduced to enable QoS provisioning, [Raj99]. Customer related policy information is stored in a policy server in the

network[6] Requests for new service allocation are made to this server by network elements e.g. edge routers or Web Portals. If the request is allowed, the policy server then configures the network element/edge router with the flow information. A network element configured in this way is sometimes known as a policy enforcement point (PEP). A new protocol called Common Open Policy System (COPS) has been designed to meet these policy configuration requirements [IETF00]. RSVP has been identified as the most suitable signalling protocol for dynamic request handling. Figure 2-3 below shows how an RSVP signalled request will be handled in the network. The RSVP request is sent to the BB from the host. If the user's SLA permits the flow to be admitted the BB will configure the traffic conditioning elements in the edge route router to allow the end-user application traffic to be transmitted. The BB then signals the end-user host that it may begin to transmit data. The edge router polices the user's data to ensure it conforms to the agreed traffic profile. After the user is finished sending data it may signal the BB to

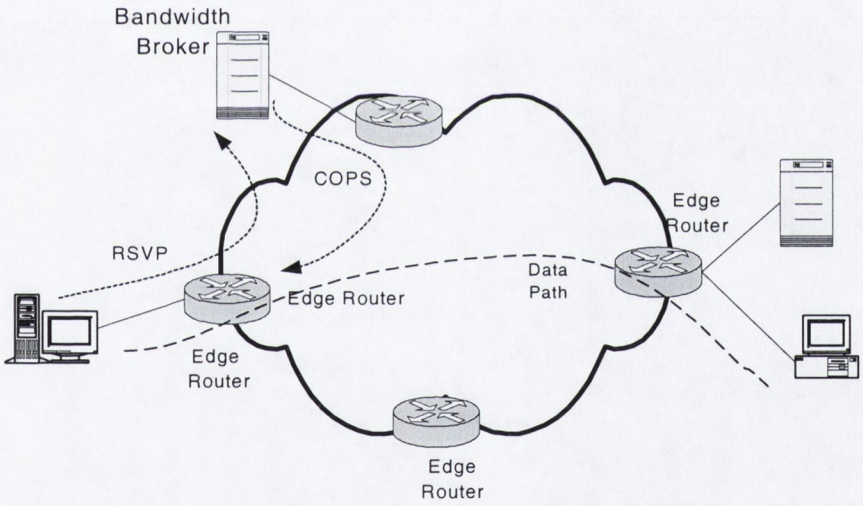


Figure 2-3 Diffserv - signalled resource allocation

remove the traffic conditioning.

---

[6] In Intserv a policy server is known as a Policy Decision Point (PDP) while in Diffserv it is known as a Bandwidth Broker (BB).

2.5 Session Layer

This layer is all about communication between two or more parties, human or otherwise. Instances of such communication are referred to as "sessions", (the meaning of session will vary depending on the particular service). Historically 'plain old telephony service', POTS, in the PSTN, has been the main such service. A telephony session is known as a "call".

In its simplest form communication services over IP is telephony over IP, also known as Voice over IP (VoIP). However VoIP has come to symbolise a plethora of new services which can be enabled by the flexibility of IP itself as well as the combination of voice with IP based applications (i.e. the enhanced services referred to above), [Poly99]. (Note that the term VoIP as used in the remainder of the dissertation generally refers to session related multi-media services rather than pure telephony).

2.5.1 VoIP Services

VoIP services will enhance telephony by enabling communication using multiple media rather than just voice. In particular video telephony will become commonplace. Video indeed can be used in different ways and video streaming from application servers to end user terminals represents another variation of VoIP communication. In order to enable VoIP a number of new protocols have been invented. These include

- Session Initiation Protocol (SIP) – SIP is used to establish multiparty associations or session and is the bedrock of many of the new service's which are envisioned. SIP is an HTTP like protocol and is implemented in the IP application layer. SIP servers will be distributed in edge networks and will serve to locate users and invite them to network communication sessions

- Real Time Protocol (RTP) – RTP is an application layer IP protocol which is used for media transmission. RTP sessions are simplex, unidirectional, transmission between IP end points. Media processing is performed, for

the most part, in network end points such as user terminals and RTP is carried transparently through the network.

- Real Time Streaming Protocol (RTSP) – RTSP is an application layer protocol used for streaming media. RTSP sessions are also simplex and unidirectional.

An important architectural feature of this network is that SIP is independent of the actual media used and the protocols and network resources which are used to transport the media. This leads to a separation of the *session* from the connection and can be visualised in Figure 2-4 below

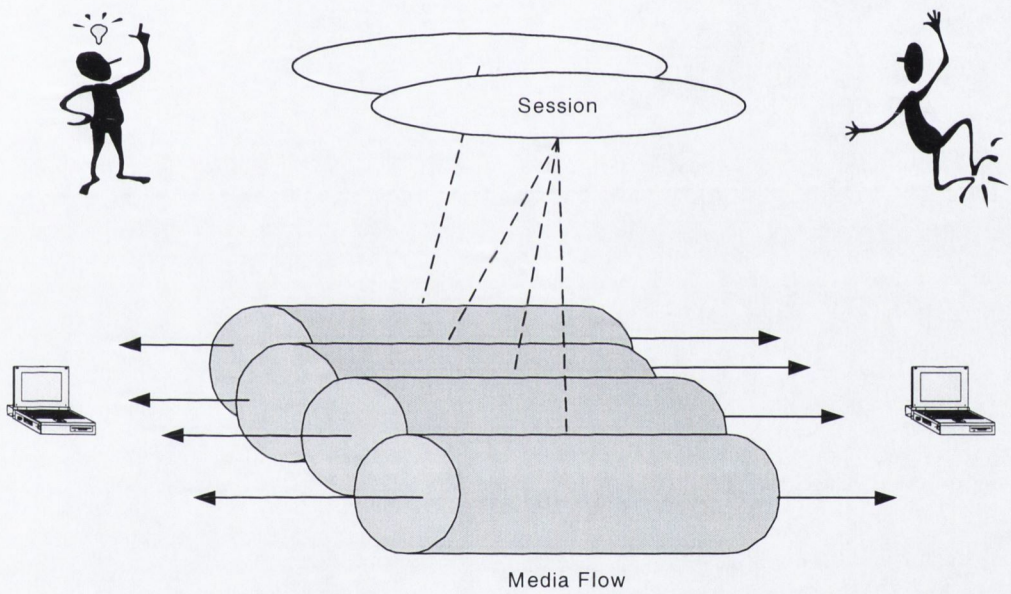


Figure 2-4 VoIP Session and Flows

Pure VoIP services of course involve IP from end point to end point. Figure 2-5 shows how these services are implemented on an "all IP" network.

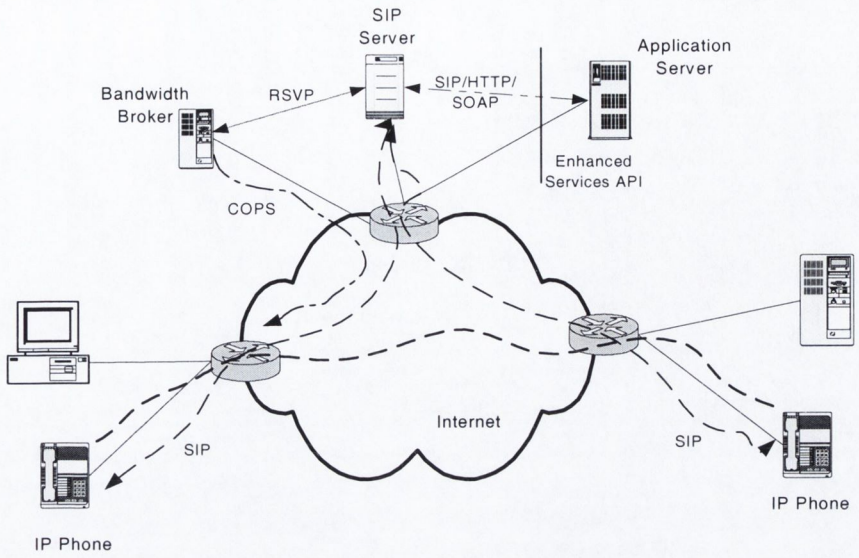


Figure 2-5 Native VoIP Service

This picture shows clearly how services can be layered on top of each other. QoS resource allocation in the network is implemented as before (Figure 2-3) and the SIP VoIP service then builds on this layer. The SIP server now controls the bandwidth broker to allocate/deallocate QoS resources. Enhanced services, which are built on top of SIP, are implemented in application servers.

2.5.2 Enhanced Services

SIP is independent of any specific service and can be used to create sessions of any kind. This means that SIP could be used for creating services such as multiparty gaming, multi-party messaging services etc. Further because SIP, RTP etc are IP applications SIP based services can be offered by service providers other than network service providers[7].

However network service providers are well positioned to offer enhanced services based on SIP especially when these services are built on combination of network

[7] Business models for provision of services in general are discussed later in this chapter.

related functionality, such as media transfer and mobility, with Web based technologies.

Representative services that can be provided on this infrastructure include:

- Telephony type enhanced call handling such as call forwarding etc.
- Multi-party, multimedia conferencing including collaborative working sessions using whiteboard and joint document authoring.
- Enhanced messaging such as instant messaging, multimedia messaging (voice, text, video, data) and unified messaging (text to speech, speech to e-mail).
- Presence enhanced communication e.g. multiparty conferencing session is not set up until all parties are available or call completion on busy i.e. calling part is contacted when called party has completed first session
- Multiparty entertainment services such as gaming. This can include both fixed and mobile users.
- Location based services for mobile subscribers e.g. targeted advertising, traffic related information and so on.

A number of initiatives have been made to define standard, de facto or de-jure, mechanisms and API's for control of network resources to allow enhanced new services, [Jain03], [Parl03], [OSA02], [Rose99]. While taking different approaches the aim is generally the same in all cases i.e. to define a network, and vendor, independent service interface that will allow rapid creation and deployment of enhanced services. Typical network services that are modelled in the interfaces include:

- Call Control
- Connectivity Management
- Generic Messaging

- Mobility
- Presence

These service subsets are called "service capability feature", SCFs, in 3GPP Open Services Architecture terminology [OSA02]. The service interface architecture is depicted in Figure 2-6. SCF's are software interfaces to existing network elements . SCF's may well be implemented on either application servers or the network element that contains the actual network service.

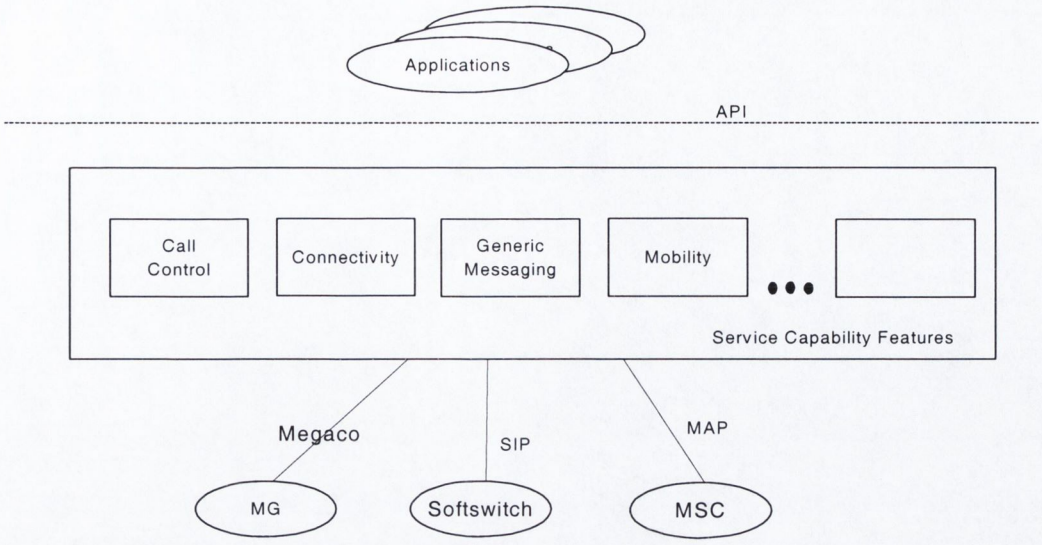


Figure 2-6 Enhanced Service API's

The interfaces are generally based on some form of remote procedure call mechanism, such as Corba or Java Remote Method Interface (RMI) , or Web technologies such as SIP, HTTP, SOAP, SMTP etc.

For emerging cellular networks the service API concept allows services to be built to take advantage of new technologies such as SIP and SOAP while at the same time to build on legacy infrastructure such as CAMEL in 3G networks.

Houssos et al., [Hous02], take this one step further and argue that enhanced API's will be necessary, but not sufficient, for the provision of value added services. They envision that networks will in fact need to be *reconfigurable* in order to enable service differentiation for 3G and 4G mobile services. In this view reconfigurability encompasses the entire service provision domain extending from the mobile terminal through the network infrastructure to the application service layer. They posit the existence of a reconfigurability platform that will allow 3rd party enhanced service providers to deploy services across the mobile operators' network and to optimise their use of the mobile network, in line with commercial agreements, to provide services to end-users.

2.6 Application Layer

This is something of a catch-all for providing application functionality over the Internet. As can be seen from the previous section this layer to some degree overlaps with the enhanced services sublayer of the Session Layer i.e. a subset of application services can tightly coupled with basic network services to offer enhanced communication services.

More generally services in this category deal with the provision of various types of "content" or business/commerce transactions across the network. These services can be offered independently to the network and information is carried completely transparently the network or the service offering may be loosely coupled with network services. Service provision models are more fluid in this layer and a variety of service provider relationships may exist.

The first category includes today's WWW surfing. In this case the network offers just a connectivity service which may be best effort or, in the future, based on higher levels of QoS. Loosely coupled services may be content bundled with a particular network QoS and made available to the end user by the third party service provider. Loosely coupled services may make use of network services other than transport. In particular third party service providers may capture information about the user from the network to flavour their service offering or service providers may make use of network billing system to charge for service usage. The best example of this is the very successful I-Mode service offered by NTT. NTT charges service providers nine percent (9%) of the total session costs and for this NTT bills the individual user.

"Content" can represent just about anything from ring-tones, football results, MP3 files to sexually explicit images. Other categories of services are transaction-based services that involve users purchasing goods across the Internet. These transactions are viewed as a revenue generation possibility by mobile service providers, who envisage being able to tap into the m-commerce information stream.

2.7 Mobility Management

Mobility management allows a user to roam between networks while allowing new connections to be set up or to maintain exiting connections. There are a number of different types of mobility in the network and a number of different solutions. Broadly speaking we can distinguish between *personal* mobility and *terminal* mobility[8][Schu00].

Personal user mobility refers to the ability of a person to access his subscribed services in any location independent of the terminals. It is based on the use of a unique personal identity.

Terminal mobility refers to either:

1. The capability of a user terminal to move its network point of attachment between sessions. Hence a user terminal may appear at different point in a network or even in different networks

2. The capability of a user terminal to move between network points of attachment *during* a session. This type of mobility involves handoff/hand-over between network attachment points and is most often associated with wireless/cellular networks.

All cellular technologies solve the mobility problem in technology specific ways. Most use the concept of a Home Location Register (HLR) database to keep track of users roaming in other networks and the concept of a Visitor Location Register (VLR) to keep track of the location of users roaming in the visited network. Various methods are used to find and track a user's precise location in a visited network e.g. paging techniques.

In order to enable roaming between heterogeneous networks IP mobility techniques are needed. Mobility in IP networks can be categorised as macro mobility or micro mobility, depending on the scope of movement of the user. Micromobility refers to

---

[8] Actually we can consider distinguish other types of mobility viz: service mobility and network mobility. Service mobility ensures that user can access the same network services where ever they are and is enabled by user profiles. Network mobility is a relatively new concern and refers to the (vehicular) movement of LANs or personal area networks (PANs) e.g. on trains or cars [IETF04]

the movement of a user between a collection of subnets owned or administered by a single authority (known as an administrative domain). Macromobility refers to roaming between administrative domains.

Mobile IP is a network layer protocol designed by the IETF [IETF96] to support mobility in IP networks. Mobile IP uses the concepts of Home Agent (HA) and Foreign Agent (FA) which act much the same as HLR and VLR in cellular systems. A roaming mobile host registers with an FA in the visited network which in turn associates with the HA. All subsequent packets destined to the host are routed via the HA and FA. Mobile IP provides good support for mobility between administrative domains but is inefficient to support mobility where frequent handover is needed as it introduced significant delay and signalling overhead. A number of new protocols have been designed to solve these problems. They include HAWAII, Cellular IP and Hierarchical Mobile IP amongst others [Camp01]. As may be guessed they adopt a variety of approaches to the problem while all use Mobile IP to track terminal mobility between administrative domains. Session state, (e.g. QoS, security details, accounting details etc) may need to be migrated rapidly between access points as users move. Micromobility is as active area of research and standardisation.

Mobility can be significant from a charging point of for a number of reasons:

- It can be used a basis for charging for network services i.e. the tariff applied to a network user may depend on his location.
- As users move in the network, session-state may also need to move for both intra-domain and, perhaps, inter-domain scenarios.
- User location may be used as the basis value added services e.g. a restaurant may push lunch advertisements to a set of users in a particular subnetwork at midday.

2.8 Service Marketplace

The next-generation marketplace will be service rich with a huge variety of services on offer, including communication services, information services and content. IP in

the network and end to end in the user terminals will enable great flexibility and innovation in service creation and delivery.

Worldwide deregulation of telecommunications markets in the last twenty years has led to the appearance of many service providers in the fixed line and cellular telephony marketplace. Competitive local exchange carriers (CLEC's) in the U.S offering services based on DSL are but one example of this phenomenon. The emergence of the Internet as a global communication medium has accelerated this marketplace fragmentation with service providers providing both basic internet access (ISP's) and applications and content on top of this basic access (application service providers ASP's).

As the range of services evolves so too does the structure of the market place in which they are offered. The opportunities afforded by new technologies will further increase both the number and type of service provider. Examples include

- wireless ISP's (WISP's) using WLAN technology to offer Internet access who typically locate at 'hot-spots' such airports
- mobile Virtual Network Operators (MVNO's) who lease network capacity from large cellular operators and handle subscriber management and billing
- communication ASPs (CASP's) offering enhanced communication services using enhanced service API's

Competitive pressures will lead to specialisation and new business models based on wholesale and partnering relationship between service providers will emerge. One example could be co-operation between an ISP and a CASP to offer multimedia conferencing. In this scenario the ISP operates a Web *services portal* where a range of enhanced services are provided. A customer uses the portal to access the CASP service system and specifies the other parties to the call. The CASP's system then sets up connection with the required QoS through the network and bridges the parties into the conference. The scenario may enhanced by specifying that the conference be set up to occur at some future time or by the use of presence services to set up the session only when all parties are available.

The relationship between service providers can be placed in the context of the NGN taxonomy – cf. below.

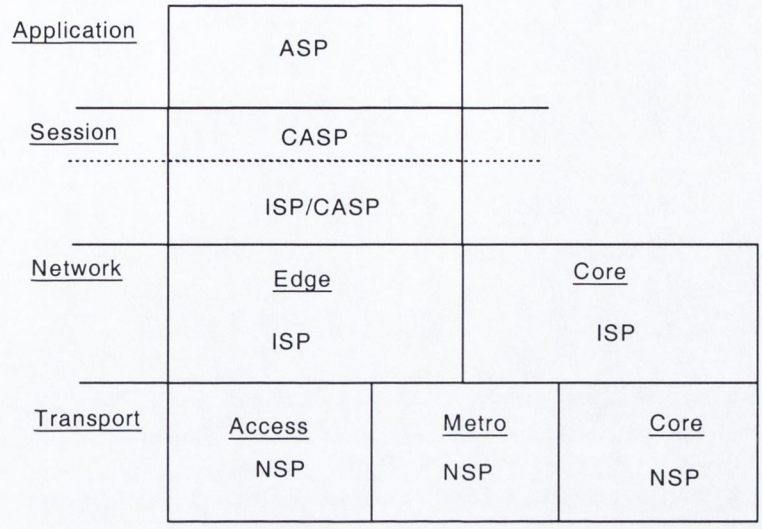


Figure 2-7 Service Provider Taxonomy

The bundled service scenario above can be extended to the provision of basic QoS services by the use of e-commerce technologies [Mess99]. In this scenario user may contact *a services broker* to request quotes for a QoS session. The broker can contact sets of network providers, who may form temporary syndicates, to obtain quotes for the connection. The broker then presents the results to the end user who can select a service based on QoS, price and so on. In this example the user may pay the broker (postpaid or prepaid) or may use e-commerce *micropayment* technology to pay in real time for the service, [Peir99]. Taken to its limit the user could conduct competitive bidding by e.g. holding a reverse auction. This example represents a follow on from the existing prepaid service scenario and leads to network connectivity being purchased as a good rather than a service. Micro-payment technology can extend this example to the provision and purchase of many different types of service including information services and content.

New forms of service provisioning will also come about as service providers seek to make it simpler and easier for end-users to create, deploy and use new services and hence generate new revenues. End-users will be able to create 'on-demand' services via WWW based service portals, [LuLa99]. Typical services may be on-

demand provisioning of bandwidth, creation of VPN's, setting up of presence enabled multiparty conferences and so on.

2.9 Implications for Billing and Charging

The NGN will pose a number of challenges for charging and billing of systems. This thesis is concerned with charging for on-demand, session-based, services. Some of the major issues are:

1. The layered nature of the NGN will result in different service provider types

   There will be many different services and no service provider will provide services at every layer. There will be a lot of competition in the marketplace.

2. Competition will force service providers to experiment with different business models.

   The choice of business model has perhaps the most fundamental effect on the charging and billing system e.g. if services are free there is no need for a charging system! This topic is explored in more detail in the next chapter.

3. Service providers will co-operate to provide bundled services.

   Services in upper layers of the NGN i.e. enhanced communication services and application layer services will in many cases be provided in partnership by 3rd party service providers and network service providers. This implies co-operation between the charging and billing system for the collection of charging data and remittance of monies between the partners: the exact form will depend on the charging/billing approach. One example is the Parlay charging API that enables payment to be made between service providers billing systems, [Parl03].

4. E-commerce technologies will change the relationship between customer and service provider

   Today users are bound to a particular service provider. New technologies that

allow competitive bidding for network access and real-time payment allow NGN services to be purchased as a good rather than a service. This implies the charge calculation must be made in real-time, which may be difficult for a complex service, and may place great demands on the charging system. Charging systems may need to be distributed.

5. The NGN will be a very diverse and heterogeneous network.

   There will be many different network and device technologies. There will also be many different types of service. The charging and billing systems must be able to deal with this diversity and must be easily adaptable to new services and technologies

6. The NGN will be very complex

   Networks and technologies are becoming evermore elaborate with more network types and more network nodes. The consequences for charging systems are that they must be able to deal with many different types of information and with much greater volumes of information. In fact it is estimated that GPRS networks may generate up to forty times as much charging data as GSM networks, [Sur01]. This will place great demands on the charging system to move this data around and to process it. Charging systems will need to be distributed in order to deal with the processing load.

7. Mobility will be more complex than before.

   Users will be more mobile. There will be more of them and services will be more complex resulting in greater complexity for handoff. Mobility will have impacts on the charging system, as a user's location may be a factor in charge calculation. Furthermore where real-time charges must be calculated it may be necessary to migrate a user's session state as the user moves through the network. This will have implications on where charge calculations are performed.

8. There will be a trend toward real time processing of charging data.

   Evidence for this has already been seen in the influence of e-commerce technologies. The availability of on-demand services will mean greater interaction between service provisioning and AAA systems. Internet traffic can increase by an order of magnitude in a very short time that can be bad news for both consumer and service provider. Real time charge calculation will be needed where the service usage is very variable to protect both parties. Again this implies the need for some form of distributed charging system to deal with the potential processing load.

From all of this we can see that the NGN will place great demands on the billing and, in particular, charging infrastructure. We do not believe that current approaches to charging can provide the flexibility that is needed to provide charging of on-demand session based services. Neither can they scale to meet the demands that will exist. We believe new thinking is needed. However before we can explore new ideas it is necessary to consider current pricing and charging approaches in order to i) evaluate their potential impact on future NGN charging and ii) to consider their suitability to provide solutions to the charging needs. We will return therefore to the discussion of NGN charging needs and an approach to meet these needs at the end of the next chapter.

Chapter 3

3 Pricing and Charging for Services

3.1 Introduction

The rapid growth of the communications environment in the last decade has triggered a vigorous debate about how usage of services should be priced and charged/accounted[14] for.

The definition of charging used in this document is the collection of charging data and the calculation of charges based on that data. That is a somewhat broader definition than those currently in use in the industry where charging refers to collection of data only. We do not differentiate between the terms accounting and charging. The terminology will sometimes be ambiguous as we in some cases refer to charging in an NGN context and in other cases refer to term as used by other people. It should, hopefully, be clear from the context which meaning applies.

This chapter presents an overview of current industry approaches and reviews emerging industry trends and significant research contributions. The focus of the discussion is on pricing and accounting for end-user on-demand session related services and as such, the issues of inter-provider pricing and accounting are not considered.

We consider firstly the factors that influence pricing and review a number of pricing approaches, including the use of pricing for traffic management. Several research approaches are introduced. We then review current and emerging approaches to NGN accounting. The implications of pricing models and charging approaches are reviewed in the context of the NGN. Shortfalls in current approaches are identified and a new approach to charging for session related NGN services is described.

---

[14] Historically the term 'accounting' was used in the Bell networks in the North America while 'charging' was used by European manufacturers and operators.

3.1.1 Influencing Factors

Several factors have influenced the debate on pricing and accounting including:-

- The need for governmental regulation of the Internet. According to McKnight, [McKB97], "the Internet policy community until recently chanted the mantra of ' no regulation of the Internet' to allow it to continue to grow unfettered by societal concerns". On the other hand economists are "almost offended by the thought that the Internet could somehow be providing a free lunch". Pricing for Internet access and services has been a key component of this debate.

- The statistical multiplexing nature of the underlying packet switched technology. This means the Internet is in effect a shared resource for user traffic. Congestion can, and does, occur when too many users simultaneously try to access the network. This can lead to what economists term a "tragedy of the commons" [15] [McKB97]. Pricing has been suggested as a mechanism to control congestion.

- The emergence of the Internet as an e-commerce marketplace for trading of a wide range of goods and services.

- The emergence of the Internet as a converged voice and data communications network which will in the longer term replace the PSTN as the universal communications network of the 21st century.

- The directly contrasting philosophies (both technical and commercial) of the PSTN and the Internet. In contrast to the PSTN the Internet was founded on a philosophy of openness and on a policy of interoperability. Its underlying technical approach is that of "dumb network, smart edge device". These features of the Internet are very conducive to easily adding new, intelligent, devices to the Internet thereby growing the community of users and services.

All of these factors has led to many pricing models being suggested and deployed. Some models are highly technical and academic and are unlikely to see actual use while others

---

[15] Used to describe abuse of a shared resource e.g. when villagers overgraze the common pasturage ultimately grass can not grow and hence the villagers will have no feed for their animals.

are in widespread use. It is impossible to say what form of pricing will dominate and it is highly likely that a wide of variety of models will be in use for a long time to come.

3.2 Pricing Models

Faulhaber [Faul95] has attempted to characterise the various types of pricing models which are likely to apply in an Internet based marketplace, for both communication services and e-commerce services. His characterisation also takes account the historical antecedents of the various players who shape and form the marketplace. He identifies four main categories viz.

- *The telephony model* – this model applies to the Internet as a provider of basic communication services such as telephony. Common forms of pricing from the telephony world apply and include issues such as access pricing, congestion pricing and so on. Rappa, [Rapp04], terms this form of pricing the "utility business model" i.e. a combination of subscription and usage pricing and believes it will emerge as a form of pricing in future "on demand " computing services.

- *The movie model* – this originates from the television and cable industry and is based on the broadcast model. Here connectivity and content are free and the content provider recoups revenue by advertising. This model can apply to the provision of many goods in the Internet and is fact beginning to emerge in some places. Of late the cable industry has been embracing a 'pay-per-view' approach (more like a cinema or theatre) and this will probably spread on the Internet as well.

- *The library model* – this is the view from the early Internet. Information is held to be a public good (right) and is provided free. Pricing of any kind is to be abhorred and must be resisted. Access and usage must be subsidised by someone else.

- *The shopping mall model* – in this model the Internet is viewed as a virtual mall where users 'hang out' (i.e. connect to the network). Goods and services are purchased from vendors in the mall and paid for on a subscription or transaction basis. Users access the network ('mall') for free and the cost of

connectivity is paid for by the vendor who, in turn, pays the network provider for connection to the network (i.e. for his virtual store). The vendor recoups the network connectivity cost from the sale of goods.

Faulhaber concludes that all types of model are likely to apply concurrently in the Internet depending on the good or service being purchased and the approach taken by different vendors. He also notes that marketing departments will have as much, if not more, influence than either economists or engineers about pricing and that unless the Internet becomes fully regulated, commercial providers are very likely to have the final word. This view is also shared by DaSilva, [Das00], who notes that the pricing of network services is "primarily a marketing and strategic decision rather than an engineering decision".

The position taken in this thesis is that Faulhaber is correct and that variety of pricing models will apply for the provision of both e-commerce and network services. Some network services will be based on usage pricing and will need accompanying accounting mechanisms. Competitive, commercial and technology factors will create a market for real-time based charging and payment for some services. New charging solutions will be needed to meet this need. The provision of such real-time solutions is the subject of this work. The remainder of this chapter focuses on pricing and accounting for network services.

McKnight and Bailey [McKB97] note that it can be difficult to get agreement on a definition of terminology in this area, especially as regards usage based pricing. They provide a number of definitions of Internet pricing structures, which are applicable to pricing in multi-service networks. These include:-

- Flat rate Pricing

  With flat rate pricing ("all you can eat") a user is charged a fixed fee regardless of how much bandwidth he uses. The fee is independent of the configuration or speed of the connection. A major benefit of flat rate pricing is that complex billing systems are not necessary. It is easy to provision and administer. It is easy for users to understand and it encourages usage as users do not have to pay more for extra usage.

- Capacity-based pricing

  Here the user also pays a fixed-fee but the fee is proportional to and varies with the bandwidth of the connection. Capacity based pricing is the most common form of pricing in the Internet today. Capacity based pricing differentiates between classes of users and so is a form of price discrimination and so helps network providers generate more revenue. The main weakness of capacity based pricing is that it does not track a users actual usage of the system but rather his expected usage. As such it can not be used to influence user behaviour and can not, therefore, be used to reduce congestion. While the billing model is slightly more complicated that flat rate pricing a complex accounting system is not needed.

- Usage Sensitive Pricing

  Pricing is sensitive to the usage of the network. Users may be charged according to a number of criteria e.g. by volume (bytes or bits sent), QoS used, time-of –day and so on. Usage sensitive pricing requires an accounting system in order to track actual use of the network resources. Usage sensitive pricing is common in current telephony networks. One difficulty that may arise with usage sensitive pricing in packet switched networks is that users may be faced with variable bills at the end of each month. Pricing models, especially those related to QoS, may tend to be more complex. Usage sensitive pricing may therefore be more applicable with a prepaid payment model or a real-time charging model i.e. where users can see the accrued charges as they occur. McKnight suggests such a solution may be the only way to implement an Integrated Services QoS model [MckB97].

All forms of pricing are applied to day, often in combination though flat rate and capacity based pricing are the main forms of pricing for Internet access. In Europe network access is often a combination of a flat-rate charge (to the ISP) for access to the Internet and a usage based price (to the PSTN service provider) for dial-up facilities.

3.3 Network Service Pricing

3.3.1 Cost Models

As noted above the pricing of network services is largely determined by business and marketing criteria and will most likely continue to be so in the future [Faul95], [Das00].

Prices therefore may not be directly related to costs. Prices are undoubtedly influenced by costs, as well as by the overall economic and regulatory regime. However as Clark, [Clark97], suggests, an understanding of service provider costs can lead to a better understanding of pricing.

Leida, [Leida98], describes a comprehensive cost model for ISP's operating under two scenarios

- Offering basic Internet access where web browsing is the only activity
- Offering a VoIP service in addition to web browsing.

Leida defines the model for a mix of residential and business subscribers including analogue dial up, ISDN dial up and leased line (56kbps and T1). The model aims to be as representative as possible of the real world situation. It defines a network for a backbone ISP and a number of related access ISP's. The modeled network spans nine cities in the US and contains a representative number of tier 1 (backbone) and tier 2 (regional/local) POPs network The model also defines the switching, routing, modems and other equipment, including leased lines for inter-POP connection, needed to build a network of this size. Prices for equipment and line rental are based on real-world prices, as applied in 1997. Also the mix of subscribers and assumptions about usage patterns is based on typical ISP figures from that time. All customers are assumed to be existing customers i.e. the model does not describe startup or installation costs.

He defines five cost categories

- Capital equipment
  This includes routers, modems, switches, LANs, servers and so on. The cost for equipment is a one-time, up front, fixed cost that is converted into a monthly recurring cost.
- Transport
  This includes leased line costs for router interconnect, T1 customer leased lines, and T1 leased line costs to connect the analogue modem bank to the local exchanges for dial up subscribers. The costs are accrued monthly.

- Customer Service

  This entails providing technical support to customers. The cost is proportional to the number of subscribers. Costs are calculated as a fixed amount per subscriber per month.

- Operations

  This includes network maintenance, facilities costs (heat, rent etc) and billing costs. These costs are related to the size of the network and the number of subscribers and represent a fixed cost.

- Other Expenses

  This includes sales, marketing and general administration. These costs are assumed to be related to revenue.

The pricing structures applied are flat rate, for dial in, and capacity based, for leased lines. Actual rates are based on real-world commercial ISP rates applying in 1997.

He finds that the no particular cost category dominates and that there is substantial variation in the distribution of costs across the mix of subscribers. For the basic access scenario the primary sources of cost are sales and marketing (28%) , customer service (26%) and transport (24%). For the VoIP scenario the primary costs are transport (28%), with customer services and sales/marketing both at 26%.

For the basic access scenario the model indicates that revenues are some 15% less than costs and that ISPs are losing money. The situation becomes even worse for the VoIP scenario. Here the cost increase is double the revenue increase and ISPs stand to lose even more money. Leida concludes that ISPs may be forced to consider usage based pricing to recover costs and make a profit.

An alternative cost model is defined by Mackie-Mason, [MMV94]. He categorizes network service costs as

- The fixed cost of providing network infrastructure
  This includes line/network rental, routers and support staff costs
- The incremental cost of adding a new connection
  This includes costs for access lines and switching equipment

- The incremental costs of adding network capacity
  This includes providing new lines, routers and staff.

- The incremental cost of sending extra packets.
  This cost is essentially zero when the network is not congested.

This analysis differs from the former in a numbers of respects. Firstly it is more abstract in that a number of cost categories are analysed from first principles rather than a drill down into any particular category. Secondly the costs associated with adding new subscribers are considered. Thirdly a new type of cost is introduced. This is the *social cost* of sending a packet when the network is congested. This is a concept from the field of economics and relates to the cost that one user imposes on another user when the first user gains access to a common resource thereby denying access to the second user. This is closely related to the issue of congestion pricing and discussion on this particular cost is deferred to the next section.

According to [MMV94] costs associated with network infrastructure can be recovered by a flat rate charging scheme where the rate charged may vary from customer to customer depending on what each customer is prepared to pay. The incremental cost of adding new subscribers is recovered by charging each customer a once-off installation fee. The costs associated with network expansion are best borne by a congestion pricing scheme as network expansion is only warranted when congestion occurs and only those users who are prepared to pay for such usage should bear the cost of expansion. [MMV94] proposes one such pricing scheme-'smart packet' pricing-which is discussed in the next section.

It is moot to ask that since most costs are fixed why should any pricing structure other than flat rate apply? Common sense would seem to indicate that this is the case yet Leida, [Leida98], clearly shows that flat rate based revenues did not recover costs in the US market c. 1997. Intense competition seems to the reason. The need for service differentiation is echoed by Lucas, [LuCo99]. He notes that pure bandwidth services are becoming commoditized and prices are falling accordingly. He advocates service differentiation and usage based billing to generate profits. Gong and Srinagesh, [Gong97] note that, for unbundled bearer services, competition in an undifferentiated commodity may not be feasible and that the long term trend "may lead to disinvestment in networks and the re-establishment of monopoly or price cartels and oligolostopolic

pricing". The unbundled bearer services market seems to be moving towards increasing efforts at differentiation. They argue that an unbundled bearer service market may be possible only if accompanied by some form of regulatory mechanism.

The conclusion seems to be clear i.e. that if service providers are to survive and prosper in a deregulated and competitive marketplace they need to differentiate themselves from their competitors by providing higher margin services, including the use of different pricing models to realise greater revenues.

Further support for usage based pricing comes from the INDEX trial, [Edel99]. In this trial a number of users were provided with differentiated Internet access i.e. they could access the network at different speeds and for different prices. Users were provided with a panel on their PC to enable the choice to be made and charges were displayed in real-time. Users could thus control their spending in real time. Edell found that

1. Demand is very sensitive to price and quality
2. Differences in demand amongst users are large and persistent
3. The pricing usage parameter (e.g. by volume or time) had a large impact on demand.

He concluded that flat-rate pricing does not promote the most beneficial use of the network from a user perspective (i.e. it is not economically efficient – see discussion on this subject in the next section) and that in the longer term flat-rate pricing will limit broadband access. He suggests that, based on the INDEX results, the cost of access to users would be significantly lower and quality would be higher in a usage based scheme than in flat rate schemes.

3.3.2 Broadband Service Pricing

3.3.2.1 Using Pricing for Traffic Management

It is fair to say that the majority of papers published in the last 10 years or so on the subject of network pricing have been concerned with the role of pricing in traffic

management in both single service and multiservice networks[16]. Generally researchers have tended to address one of two issues:-

- the use of pricing for congestion control in best-effort networks , or,
- the role of pricing to govern choice of QoS levels by users in a multiservice network.

Keshav, [Kesh97], provides a definition of traffic management that allows us to view both approaches as part of a larger, unifying, framework. Traffic management is defined to be

"the set of policies and mechanisms that allow a network to efficiently satisfy a diverse range of service requests, ...[it] subsumes many ideas traditionally classified under congestion control... [and] is more general because it includes other mechanisms such as scheduling and signalling that are unrelated to congestion control".

Keshav further points out that traffic management can be applied at a range of time scales. The table below, adapted from [Kesh97] defines these time scales and indicates some of the mechanisms that can be applied at each time scale.

[16] As one might expect, a single service network provides one level only of QoS while a multiservice network provides more that one level of QoS

Table 3-1 Timescales and mechanisms of control

Time Scale	Mechanism
Less than one round trip time (cell/packet level)	Scheduling and buffer management
	Regulation and policing
	Routing (datagram networks)
	Error detection and correction
One or more round-trip times (burst level)	Feedback flow control
	Retransmission
	Renegotiation
Session (Call Level)	Signalling
	Admission control
	Service pricing
	Routing (connection-oriented networks)
Day	Peak-load pricing
Weeks or longer	Capacity planning

From a pricing point of view data in the table serves a number of purposes:-

- It allows us to view pricing as one of a range of traffic management mechanisms.
- It shows how different forms of pricing can be applied on different time-scales.
- It covers both connectionless and connection oriented networks and shows how different time-scales (and hence different pricing forms) apply to different network types i.e. to connectionless (e.g. cell/packet level) and connection oriented (e.g. session/call level).

There are two key terms in Keshav's traffic management definition that warrant further analysis. These are *diversity of service requests* and *efficiency*.

Traditionally the Internet has been used for transfer of data. In general data applications e.g. file transfer, e-mail etc., are tolerant of delay while being largely intolerant of loss. Many new types of application will use the next generation Internet including applications that involve the transfer of voice, video and other media. Some of these

applications will require bounded delays in order to be useful. To meet these requirements priority will have to be given to some packets during periods of congestion i.e. the network will have to provide differentiated level of (quality of) service to its users. Such a network is termed a multi-service, or integrated service, network, [Shen95] and is said to have different levels of QoS. There is an alternative view which argues that by sufficiently over provisioning (i.e. providing enough bandwidth) a best-effort network delays will not occur and integrated service networks are not necessary [Odly98]. In this argument QoS will only be needed in strategic places such as radio links. This argument has some merit given the huge amount of optical fibre laid in the last few years. Further the deployment of QoS enabled networks is taking longer than anticipated, given the recent downturn in telecommunications. However it is highly probable that a QoS enabled network will be deployed in the medium to longer term and the most likely scenario will have a mix of QoS and no-QoS enabled parts of the network.

Efficiency, in the context of traffic management, has a dual meaning. It refers to both *network efficiency* and *economic efficiency*. Network efficiency relates to the utilisation of the network resources to the highest degree possible i.e. carrying as much traffic as possible on a given set of resources with minimal delay to the set of users. Economic efficiency on the other hand relates to how satisfied users are with their experience of the network. Economics posits the notion of a *utility function* to quantify consumer's preferences. A utility function is a way of assigning a number to a set of "consumption bundles" such that the more preferred bundles get a higher number [Vari99]. The greater the value of the utility function. the greater the degree of user satisfaction. A related economic concept is that of *social welfare* which has to do with the satisfaction of a community of users with any particular economic system. A *welfare function* provides a way to rank different distributions of utility amongst the community of users. Since an economic system has finite set of resources then allocating a resource to one user means that resource is no longer available for another user. Optimum use of the system's resources is achieved when social welfare is maximised. Such a condition arises when no consumer can be made better off without another consumer being made worse off by an equal amount. An economic system exhibiting such behaviour is said to be *Pareto efficient*.

In networking this situation applies at times of congestion – in a non-congested network it is possible to increase one user's welfare without adversely affecting another user's welfare. Sending a packet during periods of congestion imposes a social cost on other users i.e. Susan can't send her packet because Bill has sent his. Bill therefore should pay a price equal to the (social) cost his packet imposes on Susan. This price is said to be the *marginal congestion cost* of admitting a packet to the network. A network is optimally efficient when the price a user pays for admitting his packet equals the marginal congestion costs. Economic efficiency mandates that users who are willing to pay more should be admitted before users with lower willingness-to-pay. Price is in effect a proxy for a users utility function and utility function's measure a user's sensitivity to changes in QoS according to DaSilva [DaS00], implying that a utility function may be considered a measure of how much a user is willing to pay for QoS. Achieving economic efficiency has been a goal of much of the early work on network pricing for traffic management for both IP and ATM based proposals.

One of the earliest and best-known proposals dealing with congestion pricing was the 'Smart Market Mechanism' from Mackie-Mason and Varian [MMV94]. In this proposal a user indicates to the network on a packet by packet basis how much he is willing to pay, by means of a "bid" field in the packet header. The user would have a range of prices depending on the application. The network would admit all packets whose bid price exceeds the current cut-off amount i.e. marginal congestion cost of sending the next packet. Users do actually pay the price they bid i.e. they pay the bid price of the next highest price packet not admitted to the network. Clearly this is an economically efficient solution as the users with the highest preference (as expressed by their bid price) are being admitted. The congestion cost would need to be calculated over some fairly short period. The smart market mechanism represents pricing on the time scale of a packet or burst in Keshav's model in Table 3-1 and is an example of *dynamic pricing* i.e. varying the price for network access in accordance with the degree of congestion in the network.

Dynamic pricing for Internet services has also been studied. Mackie-Mason et al., [Mack97], propose a scheme termed "Responsive Pricing" in which prices for network services are varied periodically. The price charged depends on the state of congestion in the network and is zero, or close to zero, at times of no congestion. Active participation

by users of the services in a central feature of responsive pricing. Users react to price changes in real time by varying their traffic patterns to adapt to changes in prices. When prices rise users will either defer sending data or will choose to pay more to have their data sent. Those users who value the service most will send data while those who value it less will wait until prices decrease. The network thus operates in an economically efficient manner. Responsive pricing relies on intelligence in user terminals to interact with the network to enable user participation that would otherwise not be very feasible. A variety of pricing schemes could be employed in responsive pricing. Mackie-Mason describes two schemes, one in which prices are varied based on buffer occupancy at the entrance to the network and on based on the 'smart-packet' approach. Responsive pricing has also been applied in ATM networks, [Murph95].

The use of pricing to govern access to QoS in a multi-service network has been studied by a number of researchers including Shenker [Shen95], Parris [Parr92] and Cocchi [Cocc93]. A number of researchers have also studied pricing for traffic management in multi-service ATM networks including Low and Varaiya [Low93], Veciana and Baldick [Vec98], and Songhurst and Kelly, [Song97]. A common thread in much of this work is that usage based pricing, (and in some cases dynamic pricing) , will be needed in order to:

- provide proper incentives for use of the multi-service network
- achieve optimal efficiency in the use of the network.

Shenker argues that in an integrated services network, i.e. a multi-service network, usage based pricing polices which are QoS sensitive must be applied on a per user basis in order to achieve economic efficiency and to provide proper incentives to users to choose the most appropriate service level. If such pricing discrimination is not applied then users will automatically choose the highest level of service available and performance sensitive applications will suffer and hence efficiency will be lost. Charging higher prices for higher level services ensures that such behaviour does not happen. He also argues that usage based pricing is also needed in order to eliminate reselling of bandwidth which can reduce efficiency, a point echoed by Songhurst, [Song97]. In fact Shenker argues that for real-time connections a multi part tariff is needed to adequately provide service. This tariff consists of an access charge, a charge

for the establishment of the reservation, a charge for the duration of the reservation and a charge for the actual usage. For best-effort traffic a QoS sensitive per-packet charge is levied. He goes on to note that major changes will be needed to the network infrastructure to accommodate accounting infrastructure.

Low and Varaiya, [Low93] propose to alter prices for connection reservations in an ATM network on a periodic basis in response to variations in demand i.e. dynamic pricing. Prices are set so as to maximise economic efficiency. Prices are adjusted periodically based on demand in the previous period. Veciana, [Vec98], extends Low and Varaiya's work by using pricing to regulate demand in a statistical multiplexed ATM network using the concept of effective bandwidth.

3.3.2.2 Pricing Schemes

The focus of broadband pricing research changed in the latter part of the 1990's. This change of direction was motivated primarily by the seminal paper by Shenker et al. [Shen96]. In this paper the authors argue that the attainment of economic efficiency may not be achievable or desirable and that the emphasis of research should be on structural and architectural issues. The relationship between a users satisfaction and the congestion costs of individual packet is almost impossible to quantify. This is because a user is normally concerned with end to end performance of the network for a number of packets that make up an application transmission. Further optimal prices may not under all circumstances recover network costs. Also optimality is often not the only goal for pricing and pricing policy needs to be more flexible i.e. policy should not be built into the network. A variety of pricing schemes may be needed. With a focus thus on architectural issues the paper introduces a pricing approach called "Edge Pricing". Edge pricing is concerned with *where* charges should be calculated rather than *how* they should be calculated. As the name implies edge pricing advocates calculation of prices at the access point in the network only. It argues against charging scheme's which entail calculation at distributed points in the network. The authors agree that network congestion should be factored into pricing but only on the basis of 'expected congestion' i.e. a form of QoS sensitive time of day pricing. Pricing is not changed based on the actual state of the network but on the expected state of the network based on historical trends. Edge pricing is neutral about the pricing model used; though the

authors do introduce a new variant of pricing referred to as *capacity pricing*. The paper also goes on to explore a umber of open issues such as charging for multicast groups and charging receivers (as opposed to senders).

Capacity pricing is further developed by one of the authors of the previous paper, David Clark [Clark97], in which he defines a pricing model called "Expected Capacity Pricing ". In this scheme a user subscribes to a network service which is defined by a particular traffic profile. The traffic profile defines the 'amount' or capacity of service that the user purchases. Capacity is normally defined in terms of a filter that is applied to the traffic. The filter can be based on a variety of traffic parameters such as an average or peak rate, delay, burst size, jitter etc. Traffic from a user is policed to ensure compliance with the traffic profile. Traffic that exceeds the profile can be treated in a number of ways; it may be dropped, given a lower service class, charged at a higher rate and so on. Traffic profiles can be defined for a range of time horizons and traffic rates allowing a wide variety of services to be offered. Expected capacity refers to the fact that the network's offers a capacity that the user expects to receive rather than a capacity that the network guarantees to provide. This is related to the statistical nature of network traffic. Expected capacity defines the service that a user expects to receive under congestion conditions. A service provider will need to dimension his network based on the expected capacity service he has contracted to provide. Pricing for expected capacity service's is an extension of the current capacity based pricing model but with pricing applied per individual profile. Users pay a fixed rate per period, e.g. calendar month, for a certain capacity. All traffic that meets the profile is transmitted while traffic that exceeds the profile will receive a lesser treatment. Clark describes a method for implementing an expected capacity service based on the notion of 'in' and 'out' packets and which forms the basis of the Diffserv Assured Forwarding per-hop behaviour [IETF99].

A somewhat similar scheme is described by Odslykzo,[Odly98a]. This is Paris Metro Pricing (PMP)[17]. Odslykzo analysed the contemporary Internet topology, structure and traffic patterns and concluded that over-provisioning the Internet may continue to be the best, simplest, and cheapest way to provide network service. He shows that when networks are lightly utilised (of the order of 5%) then all traffic types needs are met by

---

[17] The title is based on a pricing scheme applied in the Paris metro. Passengers could buy either a 1st or 2nd class ticket. The more expensive 1st coaches were more lightly loaded and provided seating to those who were prepared to pay.

best effort service. He feels QoS will be needed in only a few places in the Internet. PMP may be used to regulate traffic on those occasions when congestion does occur. In PMP the network is divided into a number of logical channels, each of which contains a, not necessarily similar, fraction of total network capacity. Each channel offers a single, best-effort, network service. However each channel has a different price than the other channels with prices rising progressively from zero, or near zero, to some maximum price. Pricing is based on packet size. When congestion occurs on a channel users may chose to send their data on a higher priced, more lightly utilised, channel. Thus users who value their service will pay more and the scheme therefore nods in the direction of optimal pricing.

3.3.3 Analysis of Network Service Pricing

As noted at the outset of this chapter the debate on pricing has been motivated by widely varying concerns, reflecting the differing backgrounds of the many protagonists involved. A number of trends seem to be clear:

- The NGN marketplace is still in evolution. The early enthusiasm, (some would say hype), of the late nineties has been moderated by the economic downturn of 2001-2003 and the excessive spending on networking technology (cf. the 3G spectrum bidding debacle in Europe). Nor has the take-up of new services been as rapid as the industry would have liked. However the underlying trend is still one of change. Telecommunication spending is once again increasing and the NGN remains on a course towards a competitive multiple service, multiple service provider marketplace. The basic issues relating to the range of pricing and business models still apply, albeit over a longer time frame. It is thus too early to conclude which, if any, will be the dominant pricing model. In fact it is our contention, as stated at the outset of the chapter, that a variety of pricing models will apply in the NGN.
- Competition in a deregulated marketplace will see increasing movement toward usage sensitive pricing models. These will be particularly applicable for higher margin services and where markets are especially competitive. We believe also that usage based pricing will be used in many cases for QoS related services particularly where QoS services are bundled as part of higher layer end-user services.

- We do not believe that the use of pricing for traffic management on short time-scales, (seconds) will become widespread. The cost/benefit rationale seems to be tenuous. It is however possible that dynamic pricing could be applied in areas where resources are severely constrained and subject to rapid change e.g. in wireless access networks. The use of pricing for traffic management on longer time scales, e.g. time of day pricing, is quite likely to be adopted. We agree with Shenker at al, [Shen96], that architectural issues are of importance and in particular that pricing (and hence accounting) will be applied at the network edge.

3.4 Charging in Legacy Networks

3.4.1 Charging in Telephony Networks

3.4.1.1 Postpaid systems

Telephony networks here includes the PSTN and circuit switched cellular telephony networks, sometimes called Public Land Mobile Networks (PLMN). The accounting structure of course reflects the pricing model used. In general accounting is done by generating a *call detail record*, CDR, in the telephony switch, for each telephone call that is to be charged for. A CDR as its name implies is a set of data that gives information to a billing system to allow a charge to be generated for the call. Typical data to be recorded in the CDR includes

- Information about the called and calling user
- Time and date of the call
- Duration of the call
- Any extra services which were during the call

In fact a great deal more information may be generated in the CDR in addition to the above depending on the particular call and the policy of the network operators. The data is assembled during the course of the call and the CDR is generated normally at the end of the call, though if the call goes on for a very long time, interim records may be generated periodically. When generated the CDR is stored on persistent storage in the

telephony switch.

Figure 3-1 below shows the process of accounting for legacy networks. The diagram is valid for both fixed and mobile networks. The PSTN is composed of a switching hierarchy that is based on local area switches interconnected nationally by a number of backbone transit switches. Countries are in turn connected by large international switches. CDR's are generated in both local and transit switches. For a particular call the location of CDR generation depends on the several factors including the destination of the call.

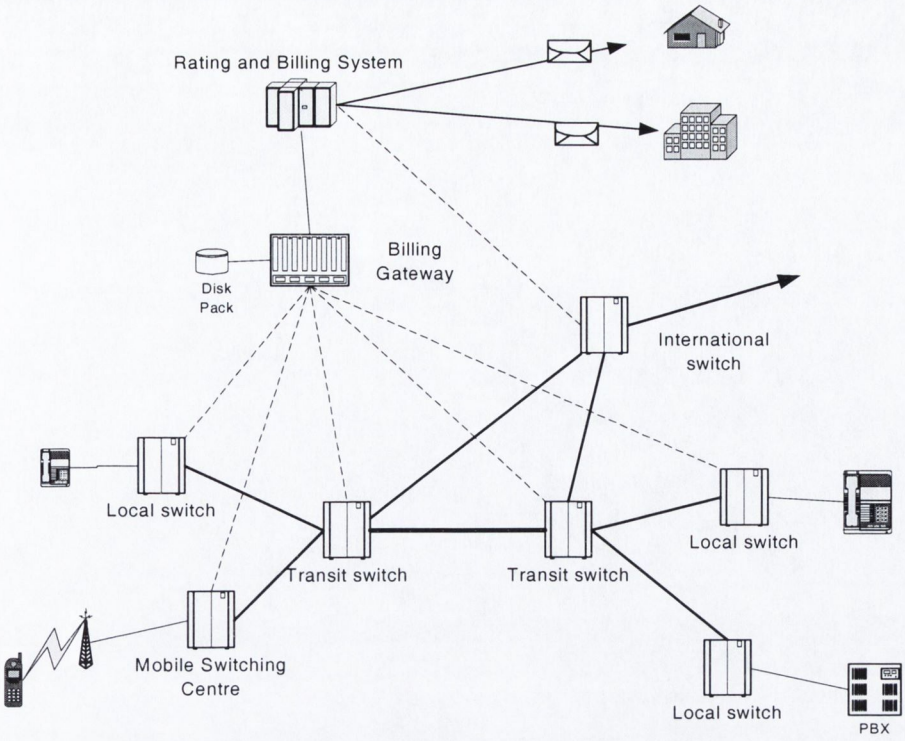


Figure 3-1Accounting in Circuit Switched Telephony Systems

CDR's are periodically transferred to a *billing gateway.* A billing gateway serves as a point of *mediation* between the billing system and the telephony network The billing gateway performs a number of functions. It hides the complexity of the underlying network from the billing system. Since a telephony network is normally multi-vendor and there may be many network elements in addition to switches then a wide variety of CDR types may be produced. A billing gateway serves to normalise or convert the different formats into a single CDR format that the billing system expects. Individual switches may use different file transfer protocols and the gateway adapts to these protocols when needed. The gateway also serves as an intermediate storage transfer point i.e. network elements may not have much persistent storage and may transfer their CDR's to the gateway for intermediate storage. Some network elements e.g. international switches in the diagram above, may connect directly to the billing system but that is not usual.

The billing system in turn collects data from the billing gateways for call charge calculation and invoice preparation. The call charges are calculated by a *rating* system based on the information in the CDR. The rating system contains a series of *rules* which are used for calculating the charges and implement the *billing policy* of the service

provider. When charges are calculated they are posted to the users account and periodically an invoice is sent to the user.

CDR's are always generated when usage sensitive pricing is used and it follows that CDR's are not needed when flat rate pricing for local calls is used. Very often they are generated anyway as telephony providers may use the data for purposes other than billing.

A variation of the above scenario occurs when a cellular telephony subscriber visits or roams to, a network owned by a different service provider cf. Figure 3-2 below

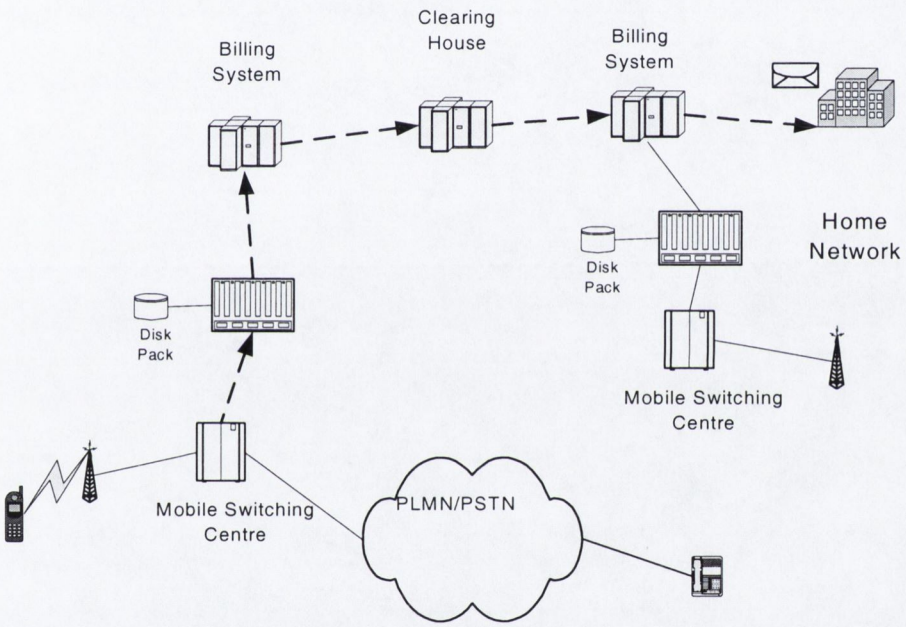


Figure 3-2 Accounting for Subscriber Roaming

In this scenario a subscriber has roamed from his home network (on the right) to a foreign network (on the left). The CDR's are generated by the service provider in the visited network, in the manner already described. They are later transferred via a *clearing house* to the billing system of the user home network service provider. The clearing house is a trusted third party who may operate the service commercially. The CDR's are transferred in a standard format such as TAP3, [GSM03].

3.4.1.2 Prepaid Systems

The "Prepaid" payment mechanism has emerged in recent years particularly in mobile networks. In this case the user purchases an amount of service usage before he actually uses the service. This has many benefits for network operators, not least of which is having revenue on hand quickly. Prepaid billing models also means changes to the accounting system. A number of means exist to implement prepaid billing [Lin01]. The most common form is based on an Intelligent Network (IN) approach, described below in Figure 3-3

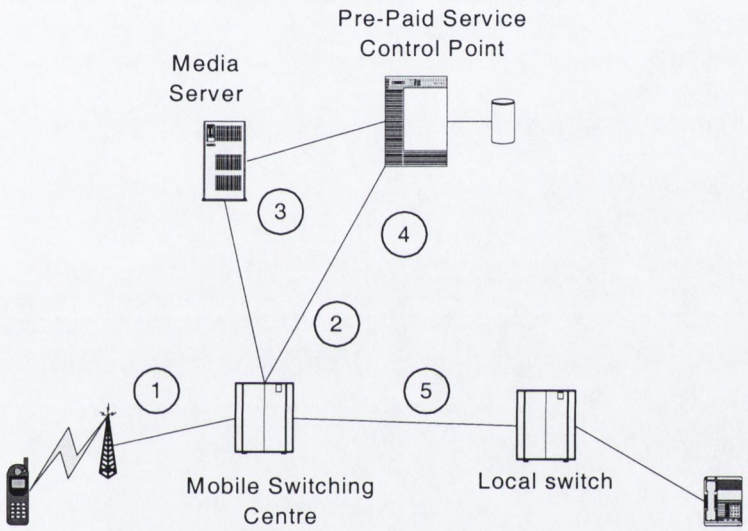


Figure 3-3 Prepaid Billing in Cellular Networks

The sequence of events is as follows:

1. The user requests to make a call

2. The Mobile Switching Centre recognises the call as prepaid and launches a query to the prepaid service control point (PSCP). The query contains the users mobile number, called number etc.

3. The PSCP may connect a media server (voice announcements etc) into the call if further information needs to be collected from the user e.g. PIN.

4. The PSCP then rates the call based on the user and call data, including remaining prepaid balance. As a result of this rating it calculates a timeout for the call.

5. The PSCP authorises the MSC to complete the call. In the normal case the user completes the call and hangs up. The PSCP updates the balance at the end of the call. (In the event the timer runs down the PSCP may instruct the MSC to terminate the call or may inform the user via a message from the media server that the call is about to terminate).

There are a couple of significant differences when compared to postpaid billing. Firstly with prepaid billing there is normally no CDR generated[18]. Secondly the call is rated in real time i.e. as the call is taking place or soon thereafter. Also a variety of payment methods are used to purchase the service. Some prepaid systems allow user to 'top-up' their credit during a call by means of a credit/debit card.

One interesting form of prepayment is based on use of the mobile handset to perform rating [Lin01]. This is the standardised Advice of Charge (AoC) supplementary service that is specified for GSM Phase 2, cf. Figure 3-4

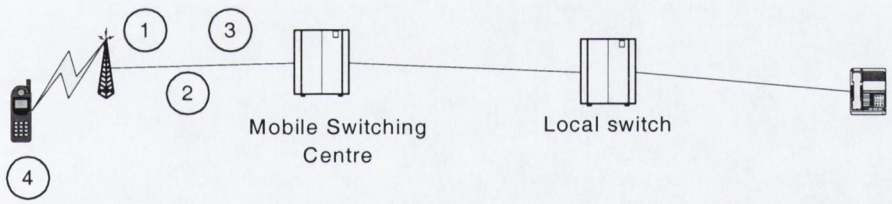


Figure 3-4 Handset Prepaid Billing

This service uses on the SIM card in the GSM handset. A prepaid service centre downloads a simple executable program to the SIM card when the user subscribes to this service. The card also maintains the remaining balance

During usage the normal sequence of usage is as follows:

1. The user requests to set up a call.

2. The MSC performs a charging analysis and establishes a rating plan. It transfers rating data to the MS

3. The call is set up

4. During the call the handset decrements the remaining balance according to the rating parameter. Note that today this supports only duration based billing. If the balance is exceeded the handset disconnects the call. The tariff parameters may be updated during the course of a call if e.g. the user changes location.

This service is not widely used today. Further it is not very secure as the encryption is not specified and so the balance or debit mechanism may be tampered with. Nonetheless the service is step further on the road to real time billing.

3.4.2 Emerging IP Charging Architecture

As new network and application services are being deployed there has been a drive by service providers and billing equipment suppliers to put billing systems in place to allow service providers to capitalise on new revenue opportunities. Traditional suppliers of billing equipment to the telephony market are adapting their systems to deal with new data services. Many new players have also appeared. In parallel data networking equipment is also be adapted to produce statistics on usage e.g. NetFlow from Cisco, [Cisc03]. Solutions tend to be proprietary.

However a generalised architecture for accounting for IP based has been proposed. This is based on the network data management specification that is prepared by the IPDR industry consortium, [IPDR02]. This specification defines

- A structure for IP detail records i.e. IPDR's, for recording IP service usage.
- a reference model for data usage recording for IP services i.e. set of logical nodes and interfaces for exchanging IPDR's
- a protocol for transferring IPDR's to billing systems and other decision support systems. This protocol is based on XML.

---

[18] One form of prepaid billing does use a CDR [Lin01]. This is based on so-called 'Hot Billing' whereby the CDR is transported to the PSCP and rated in real time, after the call.

Mediation systems play a major role in the reference model. A mediation system aggregates, normalises and correlates usage data before transferring the data to the billing system. The mediation system also translates configuration commands from the billing/decision support system to the network elements.

Figure 3-5 illustrates this flow

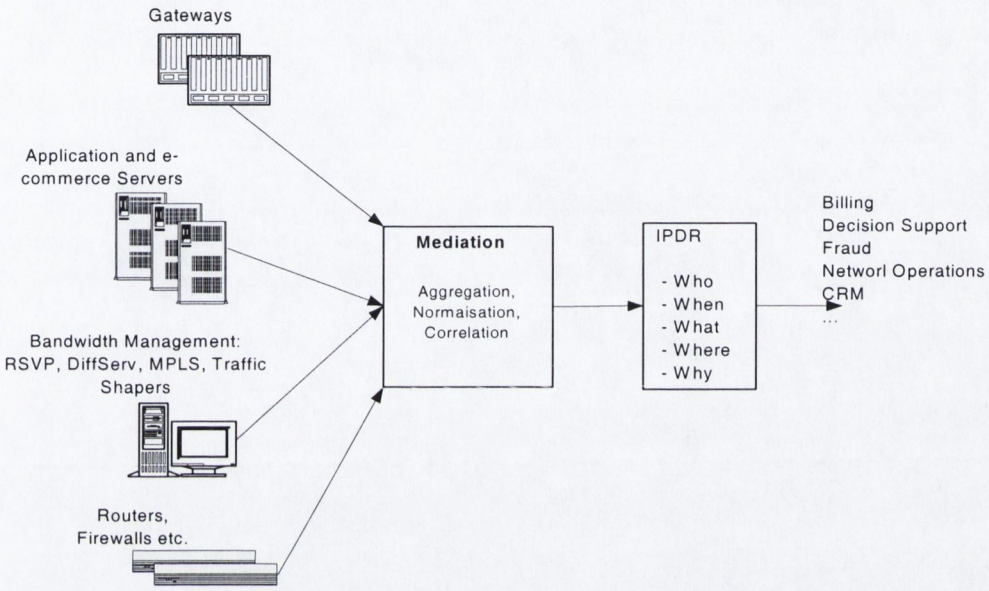


Figure 3-5 IPDR Service Accounting

The IPDR specification is intended to cover all types of services including network and application services including e-commerce and content. IPDR generalises the concept of network element as service element to describe any equipment node viz : "*Many classes of Service Elements exist: Network routers, VOIP switches (and) servers, ASP applications servers, etc. The IPDR reference model applies to any type of Service Element that is capable of generating accountable usage records (i.e. a record of which services were provided to which consumers). An SE is a generalised superset of Network Elements (NE).*" ,[IPDR02]. The model elements essentially define a framework that can easily be extended to deal with a wide range of services. Several

service examples are defined to show how the model can adapted including Video on Demand, Voice over IP, Internet Access and so on. The model also is designed to enable inter-service provider IPDR's and transfer of IPDR's for roaming subscribers.

Although clearly based on and similar to the PSTN billing model the IP services billing model is subjected to greater demands. IP services will be more complex and will change more quickly. Billing for QoS can be quite complex as we have already seen from an earlier discussion and even when approached from a non-academic view billing is not simple. Lucas, [LuCo99], describes methods for measuring end to end QoS parameters such as drop rate, delay and delay variation by means of special probes, diagnostic packets and monitoring RTCP packets. Lucas proposes to map a weighted combination of all QoS parameters on a single QoS metric, q with a value given by $0 < q < 1$, where 0 = worst case QoS and 1 = best case possible. This simplifies the perception of QoS for users and allows a service provider to simplify the QoS charging formulation. He argues further that if the QoS promised is unattainable the price should be discounted.

In addition to the complexity of QoS usage based IP services will demand that the billing system become real-time based, [LuLa99], for reasons of fraud, customer self service and credit management. Because the range of services on an IP network can vary widely and because data transmission rates can vary by orders of magnitude in a very short time frame there is far more scope for fraud in IP systems than traditional telephony networks. Authentication and authorisation of users is needed instantaneously. In the competitive marketplace customers will provision services dynamically via web portals and this will result in network elements and customer management systems being configured in real time in order to make the service available. The accounting and billing system must also be updated in real-time so the allocated service can be properly measured and billed for. Further in such a dynamic environment prepaid methods are likely to be more prevalent. Given that IP services are more complex and dynamic than POTS they are consequently more difficult to price and rate. It is important that such services be rated in real time especially where prepaid relationship and customer self-provisioning is used. Billing systems will have to move from a batch oriented legacy style of operation to a more dynamic and real time interaction with the network and service elements. Events and messages will be

exchanged between the network and the billing system. Mediation platforms have a key role to play in this interaction according to Lucas, [LuLa99]. They provide a standardised interface to the billing system and map this interface to individual network elements. According to Lucas there will be hundreds or even thousands of events entering the billing system every second. It will be necessary to use parallel processing architecture and powerful databases in the design of the billing system. Furthermore it will be advantageous to distribute data. He suggests that relatively static data (e.g. users account information) should be distributed to local data area while all dynamic usage data should be handled centrally. Real time billing systems should complement the flexibility of the underlying IP service network.

Another trend toward real time processing is the anticipated use of a network service provider to provider billing for content services by attaching the cost to the IP services charge. Today some network service providers e.g. NTT Docomo with i-Mode, offer this facility to content providers. The content providers billing system forwards transaction CDR's for a customers usage of their service e.g. MP3 downloads. The network providers rating system is configured with rules for this service and calculates charges accordingly. In a postpaid model these charges are then attached to the subscribers telephony bill. In the NGN where IP services dominate there will be increased co-operation between service providers in this manner. As prepaid, real time billing systems become more common this interaction will in turn become real time and less dependent on IPDR generation. The Parlay group, which is engaged in the definition of enhanced service API's discussed earlier, has defined an interface between a network service provider billing system and an application/content service provider to enable real time payment for content and application service purchase, [Parl03].

3.4.3 Analysis of Legacy Charging

A number of observations can be made about legacy charging systems in the context of demands to be placed on such systems in the NGN.

1. There is an increased trend towards real time processing of charging data

   The need for real time processing was previously identified in the discussion on the effects of the NGN services and technology on charging system and

that need is clearly reinforced here, in a number of ways. Firstly the move towards prepaid charging greatly increases interaction between the billing/charging system and the network and introduces the need for real time charge calculation. Secondly, charging for IP services introduces more dynamism between the billing/charging infrastructure and the network than telephony accounting. This is for reasons of fraud, customer self service and credit management, [Lula99]. Thirdly the confluence of prepayment or micropayment technologies and the delivery of application services as a bundled offering means that service providers charging and billing system will co-operate in real time to charge and bill users. We believe that these trends will accelerate in the NGN and that real-time processing will become the norm rather than the exception.

2. There is a trend towards unbundling, or distribution, of the billing system.

   Mediation systems are playing an increasingly important role in the collection and processing of charging data and are handling much of the real time interaction that is needed with the network, thereby offloading the backend systems. Rating engines are being unbundled from the back end billing system as the need for real time charging increases e.g. they are being deployed in prepaid service control points (PSCP). Finally an interesting trend is the deployment of rating functions to the handset as described by Lin, [Lin01]. In the NGN the trend toward unbundling will accelerate. Charging and billing functionality will be need to be placed directly in the network in order to get the response time needed for real time processing.

3. Charging is heavily CDR centric

   Although the prepaid model does not mandate the generation of CDR's current charging approaches are very much based on the processing of CDR's/IPDR's. We believe that this will not be sustainable in the NGN given the push towards real time data processing and the huge volume of IPDR's that can be generated due to service complexity in the NGN.

4. The prepaid architecture is billing system centric

   In current prepaid systems the usage data is pushed toward the PSCP to calculate the charge. The PSCP then maintains session charging state for the duration of the session and interacts with the MSC to handle charge related events. We do not believe this type of architecture will scale in the NGN for two reasons,

   i) there will be a much greater number of prepaid users in the NGN and this increase the load on the PSCP and on the network and

   ii) services will be much more complex and event driven and will necessitate more interaction between the network and the PSCP

We will return to a consideration of what these trends mean for NGN charging systems later in this chapter. It is necessary first however to consider the potential impact on NGN charging arising from current and recent research approaches to charging.

3.5 Charging Research and Standardisation

Pricing has, as we have seen, attracted a great deal of attention during the last decade. Charging for services has also been an area of active research, often in conjunction with pricing. This section contains a review of the more prominent and pertinent of the charging schemes suggested in the literature. Discussion and analysis of the merits of the various efforts is deferred until the last section of this chapter.

An experimental scheme to measure usage of TCP connections is described by Edell et al, [Edel95]. This was carried out in the University of California, Berkeley. Each administrative domain was connected to the Internet via a billing gateway and a billed link. When a user attempts to establish a TCP connection to the Internet the billing gateway contacts the user and confirms that the user is willing to pay for the connection. The connection is established when/if the user confirms payment and the traffic volume is subsequently metered. The charging architecture features a number of components that enable users to be identified and payment to be handled autonomously. The charging system devised for TCP monitoring was later used as the basis of the INDEX experiment,[Edel99], which has been reported on earlier. The system was the first implementation that gave users real time feedback on network pricing. The authors

conclude that an accounting system is needed if pricing is to be used for efficient allocation of resources and fair cost recovery.

A similar connection oriented approach is adopted by the CATI project but applied in this case to RSVP flows, [Stil98]. RSVP is extended to carry pricing policy objects.

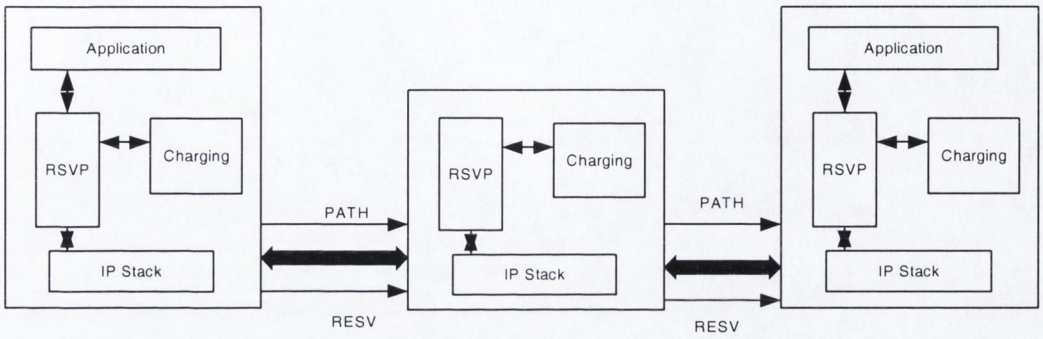


Figure 3-6 Using RSVP for Charging

The PATH and RESV messages carry the pricing objects along the path. Users, both senders and receivers, indicate willingness to pay and charging for resources may occur at every node along the path from sender to receiver. Prices are calculated at each hop along the path and different pricing models may be applied at each node. The user is paying for a "slice of bandwidth" on the outgoing link. Sender or receiver payment may be applied. The scheme has been prototyped on a testbed of routers and hosts connected over a 10 Mbps Ethernet. Stiller found that the scheme could be implemented very efficiently for relatively complex pricing models and reported a processing overhead of just 2.3% on a router dealing with 100,000 IP telephony sessions.

The IST CANCAN project focused on charging for ATM services, [Mor99]. It proposed a model based on *session* and *traffic* contracts. A session is the duration of usage of a service while a traffic contract applies at the connection level i.e. for a single media flow This is entirely similar to the session/flow model outlined earlier in our discussion of the NGN session layer.

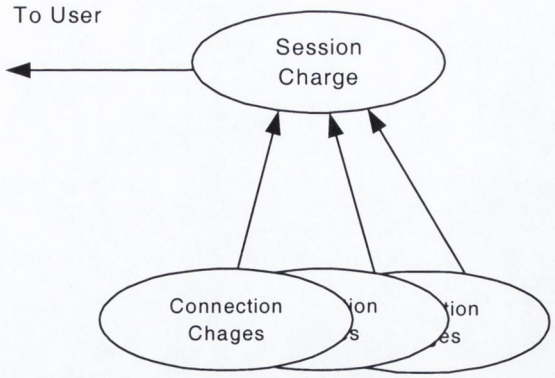


Figure 3-7 CANCAN Session Charging Model

A charge may be attached to each or any of the traffic or session contracts. A charge is calculated by a charging function, which uses charging parameters, (how much used), tariff parameters, (unit cost per resource) and other service related parameters. CANCAN produced a prototype implementation for ATM services based on a variety of pricing schemes including that of Songhurst and Kelly described earlier [Song97].

Koutsopoulou et al, [Kout04] provide an overview of the approaches to charging, accounting and billing in the IETF and the 3GPP standardisation efforts.

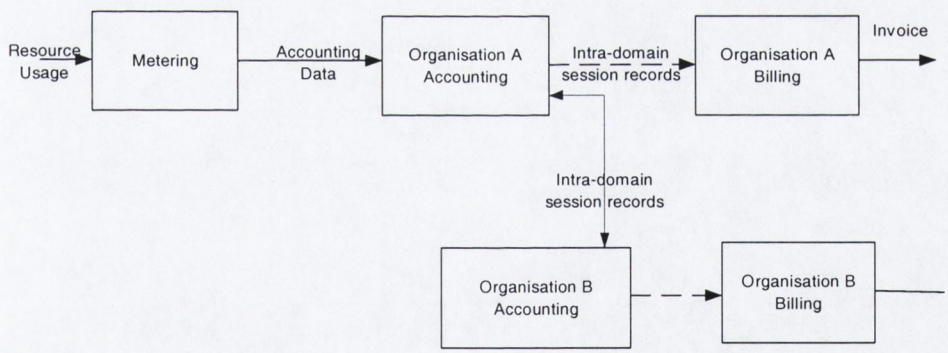


Figure 3-8 IETF Charging Functional Model

They describe the IETF in terms of the functional model of Figure 3-8. Metering is the function that captures all the data relating to network resource consumption. The RTFM working group has defined an architecture for the specification and capture of flow data, [IETF99a]. The captured data is transferred using an accounting protocol. DIAMETER,

[IETF03], and RADIUS, [IETF00a] are the most commonly used accounting protocols. Data transfer may be initiated by the network element or by the accounting server. The accounting server is responsible to forward these records to peer entities in the case of roaming terminals. In the case where different organisations are involved in the accounting, the records will cross-domain boundaries (inter-domain accounting). Otherwise the records are transferred to the billing function in the home domain which produces the bill and invoices the user. FTP or a similar protocol is used to transfer the records to the billing function.

The 3GPP architecture is somewhat different.

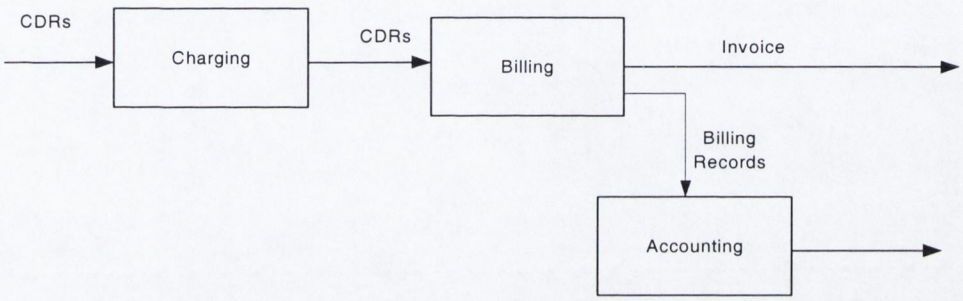


Figure 3-9 3GPP Charging Functional Model

We can see immediately that the terminology is different. The charging function collects information relating to a chargeable event from several nodes. The charging function produces the CDRs and forwards them to the billing function using FTP or similar. The charging function is implemented by two entities designated the charging gateway function (CGF) and the charge collection function (CCF). Both of these are a form of mediation function, the CGF is used for GPRS nodes while the CCF is used for the IP Multimedia System (IMS) domain. The billing function calculates the charges and presents the bill. In the case of a roaming user the billing record(s) are transferred to the home network provider using the TAP3 protocol referred to earlier, [GSM03]. CDR's are transferred from the network elements using the GPRS Tunnelling Protocol (GTP). Many different types of network element generate CDR's depending on the service. 3GPP has recently added two new functional entities to deal with content charging. These are the subscriber content charging function (SCCF) and the content provider

charging function (CPCF). These handle the interaction between an application service provider charging system, the mobile network provider charging system and the end-user when the network providers charging system is involved for charging for content.

The above approaches are for the most part largely based on view of a static network where the pace of change is not too rapid. Several recent research efforts have recognised the networks will be very dynamic in the future and that flexibility will be the order of the day. We now consider a number of such approaches which, while they are very different in many ways, have in common that they are based on the use of programmable networking technologies.

Hartano and Carle, [Hart99], describe a policy management based approach to billing for QoS and IP differentiated services used in the EU Susie project. This framework is also been incorporated as part of the IETF AAAARCH working group as RFC3334, [Carl02]. This is a layered architecture shown in Figure 3-10 below. Each layer represents specific basic functionality of a generic charging system. The left-hand "column" represents policies that drive the activities of the right-hand "column" which represents the processing functions in the network and billing system which perform the actual donkey work of the charging/billing cycle. There is an implied dependency in the layering with each layer triggering the selection and execution of lower layer policies. The metering layer extracts data from flows. The collecting layer extracts data from the metering layer and may read data from multiple meters. It forwards this data to the accounting layer that consolidates the data into data records. These data records are then sent to the charging layer for calculation of charges. The charges are posted to a users account and an invoice is forwarded in due course by the billing layer The architecture is generic in that not all components may be needed for every system e.g. if flat-rate pricing is used only the billing layer is needed.

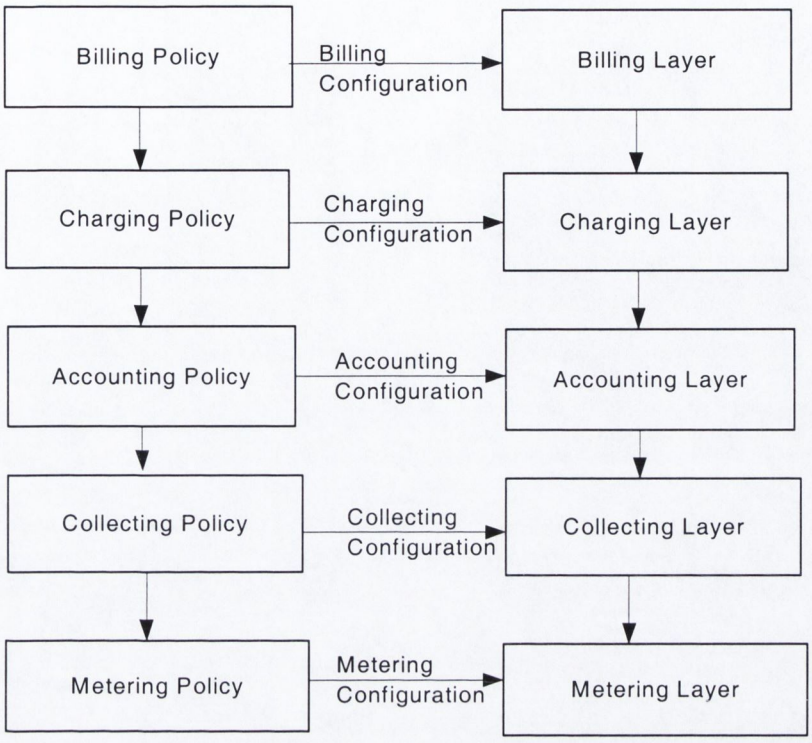


Figure 3-10 Billing System Framework

The architecture does not mandate a particular policy implementation but does imply that rule based policy syntax is used. The framework has been implemented as part of the EU SUSIE project, [Hart99], for charging of IP services. Data records are collected from meters (based on the Real Time Flow Measurement architecture, [IETF99a]), and forwarded to the accounting and charging layers for processing. No performance results are reported.

Travostino takes a more radical step with the "Active IP Accounting Co-processor Environment"(AIACE), [Trav00]. AIACE embeds a dedicated accounting processor in network nodes to perform real time processing of accounting data. Real time processing will allow:

- Packets to classified and filtered into flows. This can include aggregation into application specific flows such as H.323, SIP sessions etc.

- Enable prepaid billing to be implemented
- Distinguish accounting records that belong to different VPN and apply VPN specific policies.
- Dynamically adapt data from meter X format to accounting policy Y layout.
- Advertise unused resources back to the accounting server so that e.g. discounting may be applied
- Minimise fraud by monitoring usage in real time.
- Enable real-time trend analysis by dynamically monitoring meters.
- And so on.

AIACE allows accounting "plug-ins" to be dynamically loaded onto the coprocessor. Plug-ins represent specific accounting functions or applications and are akin to active applications in the active network architecture. AIACE is therefore a form of active networking approach to charging for network services. The AIACE architecture is described in Figure 3-11. The *virtual accounting device* is a platform independent set of accounting base functions and resources. The *classifier API* applies rules and actions to the accounting records. The *plug-in* represents a primitive action upon a record or set of records. Plug-ins may be concatenated (pipe and filter) to perform composite actions on records. Plug-ins can be loaded on demand. New plugins, rules and actions can be added to the framework via the *publisher* but only after *admission control* has validated the entrant.

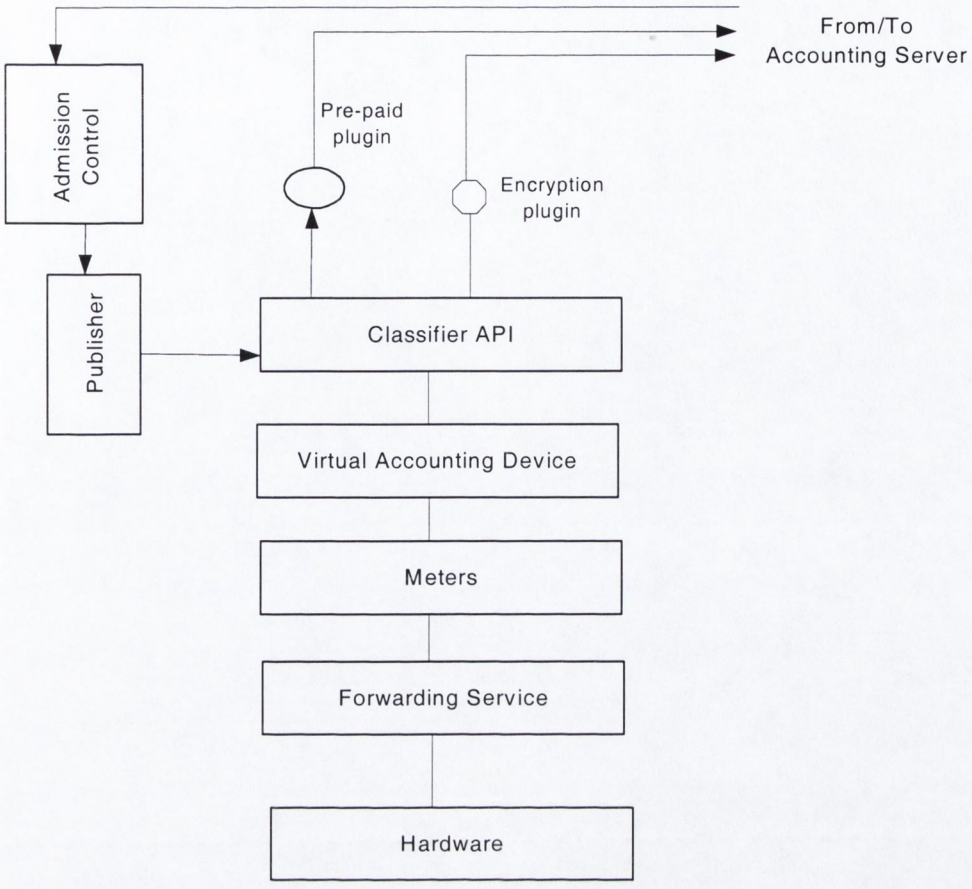


Figure 3-11 AIACE co-processor

The node co-processors communicate with and are managed by an accounting server. The server functions as a policy and plug-in repository and distribution point as well as carrying out more traditional accounting services.

Another approach to accounting based on "active tariffs" is described by Briscoe et al., [Bris00]. This is an innovative and radical approach to charging and proposes to push all accounting and charging on to end-customers hosts. Charging is on a per packet basis and tariffs, which are active Java objects, are disseminated to end-user via multicast, a multicast group corresponding to a particular network service. Metering and accounting policies and resources on these machines interact with the active tariffs to carry out the billing cycle. To allay security concerns simple random audits are periodically carried out to verify authenticity of user accrued charges. The provider can access the users machines remotely and can generate billing reports at a range of frequencies e.g. every

day, every hour, every few minutes. These reports can be compared against metering which is carried out in the edge router enabling auditing to take place. Since only a few customers need be audited at any one time the load on the edge router is low.

Metering, accounting, rating and payment functions are located on both end user machine and service provider infrastructure. The accounting functions consolidate results from possibly a number of meters. Results from the two accounting systems are reconciled to ensure that an agreed understanding of network usage has been achieved.

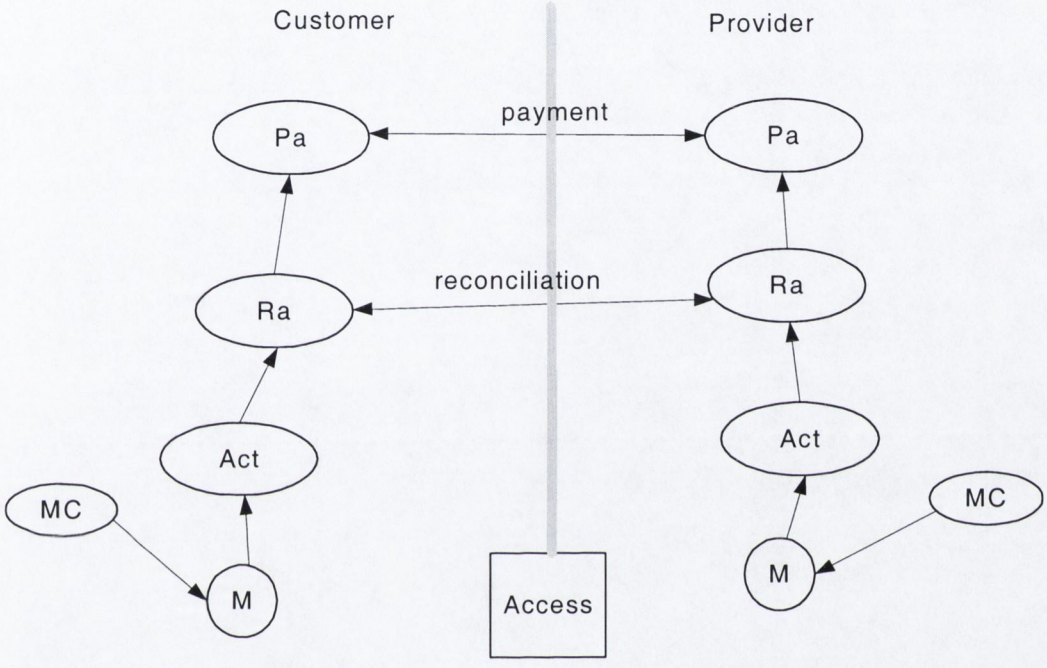


Figure 3-12 Metering, accounting and payment architecture

The accounting architecture is shown in the figure above. The Meter controller (MC) configures the meter while accounting (Act) functions consolidate the data. The Rating (Ra) function calculates charges and the payment (Pa) function later ensures payment takes place satisfactorily.

Because tariffs are disseminated as active objects and the frequency of dissemination can be varied this pricing architecture leads naturally to the use of dynamic pricing i.e. varying tariffs to reflect the network state. Briscoe envisages that dynamic pricing may indeed become quite common and that users who prefer to have price stability may in

fact have to pay for such a luxury. He notes that under a regime where dynamic pricing is common then charging could suffice as the sole form of traffic management. This work has formed part of the subsequent IST project M3I which looked at using pricing to control QoS, much as described above, [M3I03].

Bellavista et al., [Bell02], describe a charging approach based on the use of mobile agents, specifically to meet the challenges brought about by mobility in the NGN. These challenges include location tracking, coordination of remote resources, monitoring and charging of resources in each location the user moves to. Further accounting should be possible without the need for continuous communication between the remote accounting entity and centralised accounting managers.

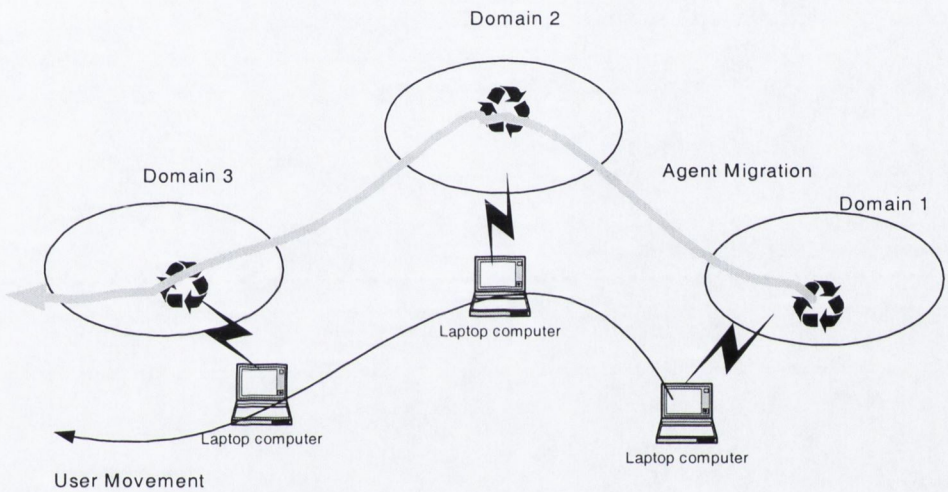


Figure 3-13 Mobile Agent Charging

They have implemented a charging framework (PUPA or Portable Usage Based Pervasive Accounting) on top of a mobile agent platform. They describe a scenario – the Downtown Visitor Assistant wherein a tourist to a city moves around the city centre using a PDA like device to give location specific information. They implement the charging solution as a couple of mobile agents, a configuration agent (CA) and a gathering agent (GA). Once initiated the CA migrates from domain to domain as the user moves through the domains (Note: a domain is assumed to correspond to particular coverage are e.g. a WLAN domain). The CA has responsibility to configure charging related parameters but also to configure any service specific elements that may be needed e.g. media transcoders. When the session is finished a GA is sent along the path

to collect charging data that are then transferred to the billing system so that charges may be calculated.

Finally Koutsopoulous et al., [Kout02] describe a charging framework which is part of the service reconfigurability platform earlier introduced by Houssos, [Hous02]. As with Houssos the charging framework is used to support the configuration of the mobile network to enable 3rd party ASP's to introduce value-added services. New business models are needed and greater co-operation is needed between ASP and mobile NSP's to enable these models. Furthermore the evolution of mobile terminals and service adaptability aspects introduces new requirements and complexity to future charging systems. ASP's should be able to able to dynamically apply pricing policies to service usage and they should be easily able to add or modify tariffs for the transport and/or content part. This will need an open API between the provider's charging systems. Koutsopoulous proposes a charging, billing and accounting (CAB) service as an extended OSA/Parlay SCF which will allow ASP's to modify the charging system to meet the above requirements (and more e.g. automated policy based division of revenues). The CAB will allow the ASP to configure charging related policies e.g. tariffs, configuration of network meters, collection of charging session information, the deployment of advanced charging services etc. This paper does not directly propose a particular charging approach. Rather it describes a charging framework to support the demands for flexibility and adaptability in what will be a very dynamic and heterogeneous NGN.

3.6 Implications for NGN Charging

We have, at this stage, considered a number of factors that will have a direct influence on the NGN charging system. These include

- NGN networks, technologies and services
- NGN marketplace including pricing models and the effects of competition
- Legacy accounting system and their evolution toward the NGN
- Research activities in both accounting and pricing

We have already listed and discussed some of the main trends that we believe will come to bear upon and shape the NGN charging system. In this section we recap the main points and consider also possible influences from accounting research efforts discussed in the previous section. In our view the main conclusions are

1. There will be much more real time processing of charging data, including charge calculation, in the NGN

   This is due to

   - use of e-commerce technologies facilitating real time payment
   - increased use of the prepaid model
   - IP networks are more dynamic and will need more interaction with charging system to relay changes in network state.
   - for reasons of fraud, customer self management when provisioning on-demand services.

2. A variety pricing models will be used and usage based pricing will become more common.

   Service providers will experiment with different pricing models. Usage pricing will be used to raise margins especially for high margin prices. This will increase demands on the charging system

3. There will be much greater volumes of charging data generated in the NGN

   The NGN will be a much more complex network that today. There will many network types and many more network nodes. Service usage will generate more data than today e.g. up to 40 times more in GPRS than GSM according to some sources, [Sur01].

4. The NGN charging system will have to be very extensible to deal with service diversity and network heterogeneity

   New technologies are appearing all the time. The NGN will be in a constant state of change and it will be important to be able to extend the charging system as the network itself expands.

5. The NGN charging system will have to be very adaptable i.e. reconfigurable, to deal with changing service provider priorities.

   The NGN marketplace will be dynamic. Service provider will want introduce new pricing models and charging schemes at short notice.

6. The NGN charging system will have to provide well-defined interfaces to allow interaction with 3rd party charging systems.

The consequences of the above effects are

1. Current CDR charging systems will not scale to provide the solutions needed. Relying on CDR centric systems will result in the charging system and the network being swamped by very large volumes of data. We believe this will prove to be untenable. The only way to reduce the data volume will be to restrict the range of pricing models to be applied which is counter to other requirements.

2. Prepaid charging architectures will not scale.
   Today session charging state is maintained in special service control nodes that are part of the billing system. This solution will not work in the future because services will be more complex and the number of interactions between billing system and the network will increase.

3. Current charging systems are not architected to handle the volumes of real time charging sessions that will be anticipated in the NGN.
   Mediation systems and prepaid service control points are simply not designed to deal with real time charging on the scale required.

New solutions are needed to meet the requirements for charging of services which need real time interaction between the network and the charging system. As we have seen earlier current billing systems are becoming unbundled e.g. rating engines are being decoupled from the backend billing system and located on mediation engines. While this is a step in the right direction it will not provide the solution needed for the reasons outlined above. We propose a distributed charging system in which some elements of

the charging /billing system are placed in the network layer. In particular we propose that the rating engine be collocated with the data metering function. This will greatly reduce the amount of charging data that will need to be moved around the network. This represents a new architectural approach to provide the flexibility and scalability for real-time charging of complex network services. Rather than bringing the data to the processing logic we are bringing the processing logic to the data. Further when real time payment or prepayment is used no CDR's need be generated. Also with real time payment there is no need for a long-term relationship between customer and service provider. This reduces the need to store customer data. This in turn makes the solution yet more scalable.

In addition to the need for real time processing the charging system has to provide flexibility and adaptability. Service providers will want to be able to quickly redraft or remove charging schemes. Lucas, [LuCo99], has shown that the interaction between the charging system and the network service will greatly increase for IP services. Thus, charging schemes will increasingly be defined in terms of the events generated by the service logic. For example, if the QoS provided on a video link changes the tariff applied will also need to change or if the location of a mobile user changes then a tariff switch may need to be applied. In the NGN much of the intelligence resides in powerful application servers. This allows for complex, diverse and novel services to be created. It must be possible to easily create charging schemes that mirror these services and that can adapt to the events that can be generated by the newly created service logic. Just as a new service programs the network so too must it be possible to program the charging system with charging logic to respond to service events.

Thus, the twin needs of real-time charging and service diversity point to the need for a programmable solution for an NGN charging system to handle charging of complex services.

The central contribution of this work is a charging system framework based on programmable networking technologies that will

1. allow for the rapid creation and deployment of novel charging schemes to provide the flexibility required to introduce new services

2. provide a framework for the easy integration of new network technologies and protocols

3. enable the service rating function to be distributed to arbitrary network nodes to allow real-time charging to scale in the NGN.

The framework, called PEACH (Programmable Environment for Accounting and Charging), will be based on a combination of two programmable networking technologies - *active networking and policy management.* These technologies are explored in more detail in the next chapter and a rationale is provided for their choice. Briefly policy management allows charging schemes to be formulated as rules which enables the easy creation of new pricing models while active networking allows charging logic to be executed at selected nodes in the network and provides a framework for a distributed charging system. The framework will allow for the definition of a number of *logical nodes* for the execution of charging logic. The framework does not impose any restriction on the physical nodes in which these logical nodes may be located. Thus the charging nodes may be located in edge routers, or adjunct processors to edge routers or they might be located on special purpose layer-4 type switches which are used for measuring and charging network traffic such as the "Encharge" solution from P-Cube, [Pcube04].

The key features of PEACH include

- An architecture for interactive session-based NGN charging.

  PEACH defines a functional model for multi-layer charging based on well defined logical nodes viz. charge coordination point (CCP), charge analysis point (CAP) and the charge execution point (CEP). These three entities allow PEACH cater for charging for any NGN service, whether in a single layer, or bundled across multiple layers.

- A language and programming model for charging policy definition

  PEACH is programmed by means of the APPLE (Accounting Policy Programming) language. Apple allows for the definition of both policy rules, (*Rule* construct), and active nodes (*Module* construct), and provides a unified syntax for the definition of charging policy and charging logic. The language

allows for migration of modules between active nodes.

- An execution environment for policy evaluation and communication

  APPLE defines a virtual machine for the PEACH execution engine (EE). The EE enables charging rules and charging modules to be evaluated and allows communication between individual nodes in the charging architecture and between the nodes and their external environment. The EE allows charging policies, logic and parameters (data) to be added via a mechanism known as a *context*.

We contend further that none of the approaches we have seen to date can meet the requirements of real time charging in the NGN as outlined above. We have rejected CDR based approaches to real time charging. This rules out any of the current commercial offerings, though these systems do provide a good deal of flexibility in defining charging schemes. Nor do the standard approaches, IETF and 3GPP, outlined in [Kout02] either solve the problems based as they are on CDR/IPDR's. The policy approach described by Carle, [Carl02], does indeed offer much of the functionality needed by the NGN. It is policy based at all levels and provides the architectural adaptability and policy flexibility to incorporate new technologies. We believe this is the correct approach to take and that policy-based solution's will become more apparent over time. However Carle's work does not provide enough to meet real time charging needs. It is primarily a description of an architecture. No detail is given on how charging schemes can be created or modified. An implementation of the architecture was made as part of the EU SUSIE project. This was based on the collection and processing of CDR's and so does not provide what we believe will be required for real time charging. Carle's work is nonetheless a significant contribution and will provide an architectural framework in which to place and relate other contributions. (In fact we use the architecture to place our own work in context later in this document).

The framework described by Koutsopoulous, [Kout02], complements our own work. Its purpose is to make the network charging system more accessible to ASP's. While both frameworks share this goal, the emphasis placed on it differs substantially between the frameworks. For Koutsopoulous it is the main goal while for us it is a secondary issue. Koutsopoulus is not concerned directly with the issue of real-time charging or the

mechanics of rating and the framework is described in the context of batch billing (i.e. CDRs). Thus while both approaches recognise the need for flexibility in a dynamic and competitive marketplace they are addressing separate aspects of the problem.

The PUPA platform, [Bell02], addresses issues associated with mobility. This is certainly an issue for the NGN and Bellavista's contribution is a novel approach to the problem. Again the focus of the work does not address the main issues identified above and as such does not provide an over all solution. We recognise that mobility of charging logic may be needed to deal with user and/or terminal mobility and have incorporated a solution for that in PEACH.

The CANCAN charging model, [Mor99] is directly relevant to our approach and is implicitly recognised in the layered NGN reference model. We recognise the need for this type of approach and have incorporated support for it in PEACH. Thus while the CANCAN model coincides with our own view it should be seen as a component of a larger solution. The work of Edell, [Edel95], [Edel99], was an early implementation of a usage based charging architecture. It was conducted mainly on an enterprise network and its scope is limited when compared to the NGN. The goal of Edell's work was to examine how end-users would respond to different pricing models, rather than to support real-time charging. Real time charging was important in his work but as a means to an end rather than being the end itself, which is the case in our work. It provided early evidence for the feasibility of usage based charging and made a significant contribution to the study of user involvement in the charging process by allowing users input to service selection and by providing feedback on charging costs. Our framework incorporates some of the lessons of INDEX (user feedback, usage based charging) and the two efforts should be seen as complementary.

Stiller's work, [Stil98], is complementary to our own. The main thrust of his work is to examine how RSVP can be extended to enable real-time billing to be embedded in the network layer and as such shares a common goal with our own work. It examines charging of network layer media flows and shows how these charges may be incorporated into higher layer service usage of the network (in this case H.323 IP telephony). The approach is shown to work for a couple of charging schemes, based on dynamic pricing models. Since PEACH is essentially about providing support for

deploying a variety of charging schemes directly in the network the two research efforts can be seen to be addressing different aspects of the same problem and hence directly complement each other. It should be noted that it is our belief that charging will take place at the network edge rather than in every hop along the route, as Stiller suggests. However there is no fundamental restriction that would prevent this possibility in our framework. In fact the opposite applies as PEACH provides support for locating charging nodes anywhere in the network.

The work of Bricoe, [Bris00, M3I03], and Travostino, [Trav00] are probably the closest in design to PEACH. Comparison between the systems is deferred until a detailed description of PEACH has been given, in the next chapter.

The handset prepaid model reported by Lin, [Lin01], is a very interesting development and takes distribution of the charging system its fullest extent. The model Lin describes is for a simple charging scheme, duration based pricing, and handset support for complex NGN charging schemes will require considerably more processing power from handsets. For this reason, and for the reasons outlined in the previous paragraph, we have focused PEACH on deploying charging logic in the network. None the less it is very feasible that these problems will be overcome in time and that deploying charging logic to handsets will be achievable.

Our conclusion then is that while much worthwhile research has taken place, none of the presented solutions can meet the needs for real-time NGN charging, for the reasons outlined, and that a new approach is needed. We propose a charging framework, PEACH, based on programmable networking technologies, specifically policy management and active networking. We review these technologies in the next chapter.

Chapter 4

4 Programmable Networking

4.1 Introduction

This chapter gives an overview of programmable networking technologies that are relevant to PEACH. Several approaches and contributions are described, though no critical comparison is made between them. Comparison of these approaches with the PEACH approach is deferred until the next chapter. The purpose of this chapter is therefore educational. A discussion of the benefits of programmable networking to NGN charging is made after the technologies have been described.

There has been an active thread of research over the last decade to facilitate the rapid creation and deployment of services in public networks by making networks programmable. The idea is to provide a programmable interface in network nodes to expose the resources, policies and mechanisms of the nodes and by so doing to create a means to construct or refine new services based on these elements, [Calv98]. This allows trusted third parties to tailor or create new services and allows experimental services to be easily deployed. Programmable networks link network-aware applications and application aware networks by means of a programmable interface, [Slo99].

Chen, [Chen99] list some of the benefits a programmable network allows:

- Rapid introduction of new types of service e.g. new multimedia services
- Implementation of traffic control algorithms to better support QoS
- Enhanced internetworking e.g. coding gateways.
- More flexible network management

Examples of services that could be created include stock quote dissemination, online auctions, [Weth98], packet filtering and transcoding, [Anco98].

4.2 Programming Models

A variety of approaches to programmable networks exist. Sloman, [Slo99], identifies the following:

Active networks – are intended to allow programmability at the packet level, [Calv99] [Calv98]. In addition to data, packets traversing an IP network may carry programs which can be executed in active node "execution environments" thereby introducing new features or services into a network.

Open signalling (OpenSig) -is based on network elements providing open application programming interfaces to trusted third parties to allow them develop application and services. OpenSig envisages a layered functional network model. Each layer provides a well defined services to the layer above via an API. Open signalling API's are often implemented with remote procedure call (RPC) semantics. The best known example is based on the Xbind work undertaken in Columbia, [Laz97], and this research was indeed the beginning of the OpenSig approach. An attempt was made to standardise the OpenSig approach in the IEEE in the P1520 working group, [Bis98]. The goal of P1520 was to define a multi-layer network reference model and a set of agreed programming interfaces between the layers. The reference model is shown in the diagram below:

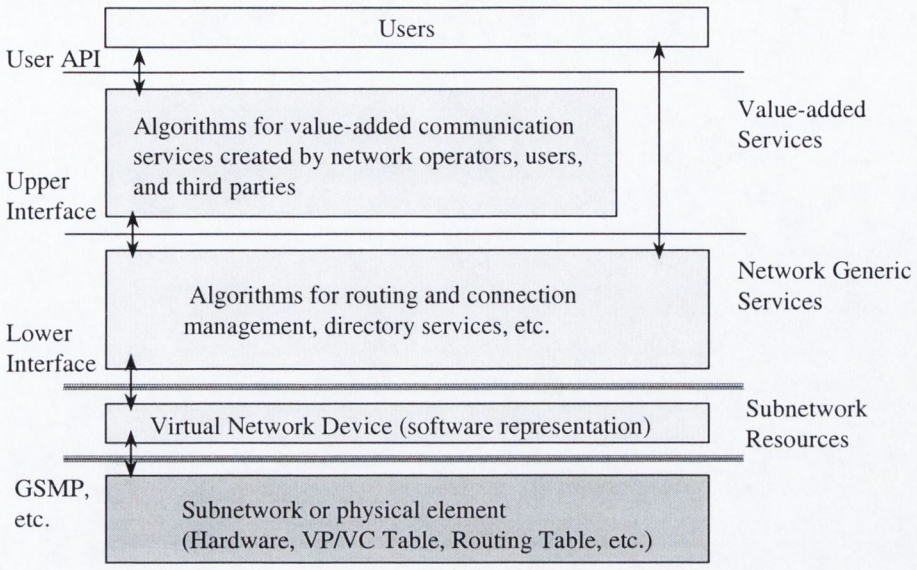


Figure 4-1 P1520 Reference Model

Much of the current work in development of the enhanced service sublayer of the session layer, (cf. Fig 2-7) is based on OpenSig principles including Parlay, [Parl03], and JAIN, [Jain03].

Mobile Agents – are programs that can move between nodes in a network, carrying state, which act on behalf of a network user, [Pham98]. The goal is to introduce flexibility into the network while at the same time reducing network traffic. Sloman notes that mobile agents are often associated with hosts or servers within the network rather than routers or switches in the network itself, [Slo99].

Management by delegation (MBD) – entails downloading code to network nodes to perform complex management tasks as e.g. testing particular equipment, [Yem91]. Traditional network management is based on a "smart manager" – "dumb agent" paradigm where all decision making is located in the manager and only data is transferred between manager and agent nodes. MBD extends the traditional manager-agent paradigm by incorporating more intelligence into the agent node and allowing tasks to be delegated to the agent.

Interpreted policy –policies are "rules governing the choices of behaviour of a system", [Slo02]. Policy rules reflect the business goals of service providers and are applied selectively at nodes in the network to ensure these goals are met. Policy rules

are often implemented as statements in an interpreted language, [Ston01], [Slo99]. Policy programming of networks is examined extensively later in this chapter.

In addition to the above approaches programmable networking has also been suggested for the provision of enhanced services for Internet Telephony [Rose99].

4.2.1 Service Composition

A theme which runs through much of the above work is that *of service composition* i.e. how statements in a programming language can serve as a method of building, or *composing*, complex services from more elemental programs called *components*, [Anco98]. A component is viewed in general terms and may range from a simple programming statement to a code module with complex behaviour. A service is formed by putting together components using the sequence control mechanisms of a composition method, (the syntax and semantics of creating services from components).

[Anco98] lists a number of features of composition methods that can be used to analyze the various approaches. These include:

Sequence control:- determines in what order components are executed. Some examples include sequential execution, concurrent execution and recursion.

Shared data control:- determines how data is sheared between components. Methods include parameter passing, shared data objects and message passing.

Binding time:- refers to the instantiation or selection of a component for execution. This can occur at either compile time or run time.

Invocation method:- the events that cause a composite service to be executed. For active networks this is normally the arrival of a packet. In other cases invocation can be triggered via a call from another component or may be triggered by a network event.

Division of functionality:- This refers to how the computational logic is shared between the invoking agent e.g. packet and the node where the invocation takes place. Approaches can vary from having all content in the invoking mechanism, as in the Active Network Transport System (ANTS), [Weth98], active network approach, to having all content in the node as in open signaling. An 'in between approach' which has

proved popular is to transfer a script like program in the packet or message and have this script invoke services resident on the node. This approach is used in active networking with the PLAN language, [Alex98] and in mobile agents,[Pham98]. Pham refers to this approach as 'smart network' and contrasts it with a 'smart message' approach in which the computational logic is contained in the message.

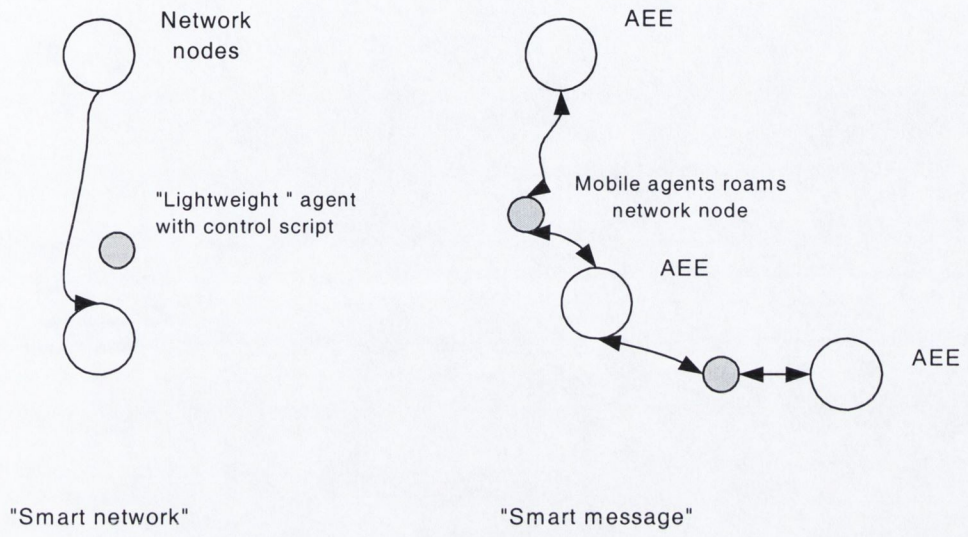


Figure 4-2 Mobile Agent Migration

In a smart network approach intelligence ("agents") reside at the network nodes and control scripts, which are regarded as "lightweight" mobile agents are downloaded to invoke node resident components. In a smart message approach the messages are mobile entities that travel between network nodes in order to perform tasks. Agents are received and executed in "agent execution environments" (AEE) which reside in network nodes. The smart message paradigm is clearly conceptually similar to *the active packet* approach from active network research, though the message is generally delivered in the management plane with the former and via the data plane in the case of the latter.

The notion of service composition is not restricted to programmable network research and is discussed in [Oust98] and [Schn99] which view scripting languages as a method for building composite services by combining together a number of predefined components. [Oust98] refers to scripting languages as "glue languages" which are primarily intended for plugging together components. [Schn99] introduces the idea of a "component framework" which consists of a generic architecture and an extensible set of

components. An architecture defines the mechanisms ("connectors") and rules for combination and configuration of component's and the mechanisms for extensibility including component inheritance, addition of new components and so on.

Although both mobile agents and active networks feature mobile execution units traversing a network of active nodes there is a great degree of difference in the scope of execution capability between both. Active packets are mostly concerned with customising packet handling for a users connection in the network. They work within the network, on the network itself, [Chen99]. Mobile agents, on the other hand, typically have more intelligence and autonomy and are more often directed to mobile computation (i.e. on the hosts rather than in the network nodes).

4.2.2 Network Programming Languages

Mobile agents are most often implemented with Java because that language offers many advantageous features such as dynamic class loading, portability, multitasking and interoperability, [Pham98]. Several approaches have combined mobile agents with distributed object middleware such as Common Object Request Broker Architecture (CORBA) and Java Remote Method Invocation (RMI), [Breu98]. Mobile agents are transferred from node to node via RPC and may also communicate via message passing using an RPC foundation though other metaphors such as events may also be used, [Pham98]. Security is a major issue for mobile agent systems and many experimental systems rely on Java security features. Mobile agents have been applied to the implementation of Intelligent Network type services,[Breu98], including Call forwarding and Virtual Private Network (VPN) services. In both examples the mobile agent is a script i.e. operates in a 'smart network' framework.

Active networks have been implemented in a variety of languages including functional languages such as Caml [Alex98] and procedural languages such as Java (ANTS), [Weth98]. PLAN (Programming Language for Active Network), [Alex98], is a lightweight simple language which is intended for writing active packet programs. It is strongly typed and has simple data and control structures so that it can be easily interpreted or implemented. Figure 4-3 shows the ping routine implemented in PLAN. The program is injected into the network with "src" and "dst" as arguments. When evaluated on a node which is not the destination it invokes the built in function

OnRemote to transfer the program to the next node. If this is the destination node then the function ack() is evaluated directly on the source, bypassing the usual ping reverse path.

```
fun ping(src:host, dst:host) : unit=
      if (not (thisHostIs(dst))) then
         OnRemote( |ping| (src,dst),
            dst,getRB(),defaultRoute)
       else
         OnRemote(|ack|(),
            src,getRB(),defaultRoute)


fun ack(): unit = print("Success")
```

PLAN

Figure 4-3 PLAN program example

Another example of a script like language is CPL, [Rose99] that has been designed to program enhanced Internet telephony services, e.g. Call Forwarding.

```
<call>
  <location url="sip:smith@phone.example.com">
        <redirect />
</call>
```

CPL

Figure 4-4 Internet Telephony Service

The CPL program in Figure 4-4 implements unconditional call redirect. CPL programs are constructed as decision graphs. The individual nodes in the graph are the primitives of the language. Each node may have several outputs and each output leads to anode further down the (directed acyclical) graph (DAG). Because of the DAG structure CPL is based on an XML syntax. Language primitives include switching nodes (e.g. *time-switch*), location nodes(*location* keyword) which indicate the next node to be contacted, signalling actions (*proxy, redirect* etc) which causes signalling messages to be transmitted through the network.

4.2.3 Node Architecture

Both mobile agents and active networks rely on the concept of *execution environment* (EE), [Calv99], [Pham98]. An EE provides the resources and support that agents need in order to be execute, communicate and migrate through the network. An EE is analogous to a Unix shell and provides an interface through which a end-to-end network services can be constructed. An EE thus exports an API or virtual machine that users can program by directing packets or agents to it, [Calv99]. The Active networking community attempted early on to define a standard architecture for active

networking so that interoperability between different research efforts could be

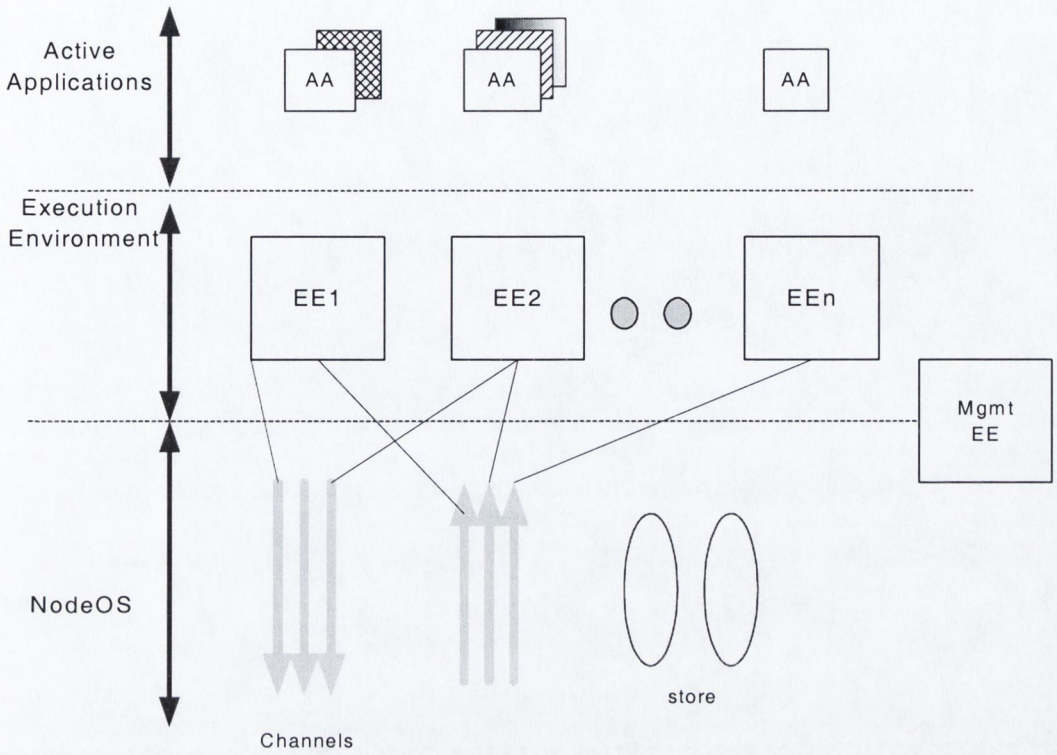


Figure 4-5 Active Node Architecture

achieved. The main components of the architecture are shown in Figure 4-5. An active network consists of a set of "active nodes" which co-operate to provide a network service. Each active node provide a node operating system (NodeOS) and one or more execution environments. A NodeOS is responsible for allocating node resources including computing cycles, storage etc. Channels in the NodeOS are well defined mechanisms to pass packets between nodes. Channels consist of physical transmission media plus higher layer protocols such as TCP, UDP etc. A number of different EE's can exist in a node. An EE provides a virtual machine for the execution of arriving active packets. EE's are defined, in part, by the types of service composition method used. An EE can request channels from the NodeOS and the channel is configured to forward packets to that EE. Network services are provided by 'active applications',(AA) which program the EE virtual machine to provide end-to-end services, [Calv99]. An AA is the program that is loaded either via the active

packet, mobile agent or via a management interface. The AA exercises the EE virtual machine via the EE API.

4.2.4 Evolution

Increasingly all approaches are being used for both service provisioning and network management. Active networking has been applied to network management in the Smart Packets project [Calv98], the active engine architecture, [Raz00], and the somewhat similar Active Distributed Management (ADM) architecture, [Kaw00].

Chen, [Chen99], sees the various network programming approaches as evolutionary steps on the road to a fully programmable network. He sees an MPLS enabled network as the first such step. An open signalling network represents the next step and allows third parties to program network functions via a well-defined interface such as JAIN and Parlay. Beyond that active packets will serve to allow users to customise packet-forwarding services, or create new network services. Mobile agents represent the ultimate step on the road as a myriad of co-operating agents roam the network to carry out users needs.

This section has given an overview of active networking and mobile agent approaches. We next consider another form of programmable network: policy management.

4.3 Policy Based Network Management

4.3.1 Overview

Policy management is a general term that describes how the resources and services of the network are allocated and managed in order to best meet the strategic and business needs of the service provider. *Policies* formulate and express these goals in context specific terms e.g. security policies, QoS policies, routing policies, traffic management and indeed charging and accounting policies. Policy can be expressed at different levels of abstraction. At the highest level business policy will be formulated in a natural language such as English e.g. "All traffic admitted to the network from Network X will be routed via Network Y". Ultimately policies at this level must be transformed into one or more lower level policies which can be implemented in network nodes and expressed in a more formal, perhaps, proprietary syntax.

A policy is defined as an aggregation of *policy rules* [Stras98]. A policy rule is composed of a set of conditions and a set of actions. The outcome of the rule depends on which if any of the conditions is satisfied and the subsequent triggering of actions.

The use of rule based policies to manage and control networks is not new, [Wies94] [Lein89], [Clar89] though interest in the area has been spurred recently by the work on QoS policy management in the IETF [PCIM01]. Policy management has also been widely applied for network and systems management in distributed computing systems, [Wies94], [Slo94].

An excellent survey and comparison of Internet related policy languages is given in [Ston01]. They examine policies from three general points of view:

- how they are used
- how they are triggered
- the level at which they are applied

Usage policies specify which network capabilities are used to provide services and how they are used i.e. what permissions and restrictions apply. Policies triggers may be *static* or *dynamic*. Static policies are applied in a predetermined way according to a set of predefined parameters e.g. "Streaming video is not allowed between 9 a.m and 12 noon". Dynamic policies are triggered in response to changing network conditions such as changes in network load. Policies may be applied to traffic of differing granularity including flows of different end-point granularity, service aggregate, application level and so on.

4.3.2 Policy Languages

[Ston01] categorises policy languages as *network* policy languages and *traffic flow* languages. Network policy languages specify policies at a network level and address issues such as routing and QoS. Traffic flow languages are lower level than network policy languages and deal with inspection and selection of individual packets flowing through network elements.

4.3.2.1 Routing Policy Languages

Policy rules have been used in telephony networks since the mid-1970's to control areas such as routing, authorization and charging [Eric97]. In this instance a policy was implemented as a tree like data structure and policy for a particular network area (e.g. routing) was captured as a table of individual policy rules. A typical routing policy example is shown below:

{RC=12; BR=CL-10&-15 ;R=SNHU;SP=111; COT=2}
{RC=12; R=SFSHU ;SP=112, COT=0;}

This policy states for traffic towards a particular destination (RC=12) certain categories of user (10 < CL < 15) should be directed on a certain link (R=SNHU) with no charging while all other users should be directed on an alternate link (R=SFSHU) . The policy is installed at a certain node in the network and implements, in that node, a high level policy that says "Traffic from emergency services operators should have priority over other traffic at all times ". Obviously the policy must be similarly expressed in all traffic nodes in order to implement a *network* level policy. The condition clause in the above rule is captured by the BR parameter which effectively represents an 'IF' statement.

Early work on Internet policy languages was also mainly concerned with routing [Lein89], [Clar89] and [Alae99]. Routing policy languages are used to specify restrictions and permissions associated with network paths or routes. Typically a routing policy will specify which routes through its network an ISP wishes to advertise to its peers. RPSL is the routing policy language currently in use on the Internet. The earlier proposal from Clark [Clar89] allowed for a wide range of network paths to be specified. A fundamental concept in this work is that Internet resources are divided into a number of administrative regions (AR). The policy language specifies permission and restrictions associated with a path or "policy route " across a network of AR's. The policy rules (or "terms") are expressed as

((Hs, ARs, ARent),(Hd,ARd,Arexit),UCI,Cg)
where: Hs = source host
ARs = source autonomous region
ARent= entry AR
Hd= destination host address
AR=destination AR
Arexit = exit AR (last hop)

UCI = User Class Id (e.g. University, Hospital, etc)
Cg= any global conditions

The first two elements specify a source and destination endpoint . The user class identifier can represent a wide range of user type while the global conditions element can represent any general condition that may be applied. A example that could be applied to an arbitrary network containing a number of AR's is :

((*,4,-),(*,6,*) best_effort, no_charge)

when applied at say AR2, this rule says "allow only best effort traffic from hosts directly connected to AR4 to pass if their destination goes through AR6 and apply no charge to such traffic".

[Ston01] extends this path based approach to policy specification by introducing a new language called the Path Based Policy Language (PPL). PPL is intended to allow for the association of arbitrary policies with complex and flexible network path specifications. In particular PPL is intended to give support for Integrated and Differentiated Internet services. A policy rule in PPL is expressed as

policyID<userId>@{paths} {target} {conditions} [{action_items}]
which can be expressed as
"policyId created by <userId> allows the target class of traffic
to use paths only if {conditions} are true after {actions_items} are performed"

```
Policy7<Betty>@{<1,*,5>} {traffic_class=accounting} {day != Friday : priority=5}
```

An example (taken from [Ston01]) is

which states "On all paths from node 1 to node 5 accounting class traffic will be lowered priority 5 except on Friday". [Ston01] argues that using paths as a fundamental building block in specifying policies allows great control and flexibility in the specification of *network* level policies.

4.3.2.2 PDL

Further examples of the use of policies for control and management of telecommunications networks is given in [Kohl00]. Policies here are expressed using a language called PDL (Policy Definition Language). PDL is based on concepts from action description languages in Artificial Intelligence (AI) and active database languages. Policies are expressed as a sets of rules of general form:

event **causes** *action* **if** *condition*

Events are generated by network elements and compared against network conditions. Actions are triggered if the conditions in the policy are met. Events may be primitive or complex. A primitive event is a named event defined in a policy rule while a complex event is an aggregation of two or more primitive events e.g.

ce1 = pe1 & pe2 & pe3

i.e. ce1 occurs only when all pe_i occur. PDL is implemented in Java. Events in PDL clauses are the names of Java classes which represent system external real world events. Actions are implemented as remote procedure calls. In addition to policies the PDL framework includes a workflow scripting language which allows action sequences to be defined and activated from policies. Polices are stored and executed in a policy execution environment . The policy execution environment is distributed and policy based management is implemented as a series of cooperating policy enabled nodes called *policy elements*. Policy elements cooperate using actions and events. Policy elements may be embedded in network elements such as routers or in policy management nodes such as policy servers.

4.3.2.3 Ponder

The Ponder policy language [Dam00] derives from early work by Sloman and others, [Slo94]. Ponder supports a range of policy types. *Authorisation* policies are used to specify access control permissions to objects. *Obligation* policies specify what activities must be carried out on a set of target objects . Obligation policies are normally triggered by events. *Delegation* policies specify which actions a policy subject may delegate to others. *Refrain* policies specify what a policy subject must refrain from doing. Ponder allows *meta-policies* to be specified to determine which policy to apply in the event a policy conflict should occur.

Ponder also provides support for organising policies. Policies may be aggregated to form a *composite* policy. Two composition mechanism's are the *group* and *role* constructs. A group construct, as the name implies, is used to group together a set of policies that have some semantic relationship and need to be instantiated together. A role construct defines a set of policies that apply to a position in an organisation. A role construct could for example be used to define the authorisation policies applicable to a network administrator. Ponder also supports the concept of *domain*. A domain is a means of grouping objects that need to be managed. The domain concept can be used to partition a large network into smaller, more manageable, entities. Partitioning can be made using a wide variety of criteria including object type, geographical area etc.

Ponder allows parameterised policy types to be defined and specific instances of these new policies types to be later created with values supplied to the policy formal parameters. An example taken from [Dam00] illustrates this.

```
type
  oblig allocBwT(subject m, target o) {
        on perfDegradation(bw, source)
        do bwReserve(bw+10)
    } // allocate bandwidth
```

This statement defines a new policy type *allocBwT*. The policy allows extra bandwidth to be allocated in a network element when network congestion occurs. The *target* keyword specifies the network element where the action is to occur while the *subject* keyword specifies who initiates the policy. The *on* keyword specifies the event which causes the policy to be triggered and the *do* keyword specifies the action to be taken.

Instances of the policy are created as shown below:

```
inst
      oblig   p1 = allocBwT( nwAdm,  site1/edgeRtr)
}
```

Ponder is a declarative language and could be implemented on a variety of platforms e.g. operating systems such as Windows 2000 or Linux and in distributed

programming environments e.g. Java or CORBA. Ponder has been applied to the management of Diffserv networks, [Lymb02].

4.3.2.4 Traffic Flow Languages

Traffic flow policy work includes PAX, [Pax00], and SRL [IETF99b]. PAX is used to specify bit patterns to match against packet headers to select individual packets for processing. It is intended for packet classification purposes for use in network edge devices. PAX only specifies conditional part of a policy rule and does not indicate what action is to be taken.

SRL is used to select traffic flows for measurement or 'metering' in the terminology of the Real Time Flow Measurement (RTFM) group of IETF, [IETF99a]. RTFM defines a set of "attributes which can be used to categorise network traffic". SRL rulesets are programs for a 'Packet Matching Engine' virtual machine which processes the packets at run time i.e. extracts attributes from the current packet, compares them against conditions in the rulesets and invokes appropriate actions if a match occurs. SRL is used to identify if a packet is part of a flow which is of interest and if so to save information required to identify the flow and finally to accumulate quantitative data for the flow. SRL is a procedural language and allows flows of arbitrary granularity to be specified. An example SRL program is shown below:

```
# Program to identify and count www packets
define www = 80;
define IP =1;
if SourcePeerType == IP save; else ignore;   # ignore non IP packets
if DestTransAddress == www save, store  FlowKind = 'W';
save  SourcePeerAddress/32;
save DestPeerAddress/32;
count;
```

This policy identifies and counts packets directed towards the WWW.

4.3.3 IETF Policy Model

The IETF has defined a modelling framework to enable policies to be specified for management of IP networks [PCIM01]. This model is based on the Common Information Model (CIM) defined by the Distributed Management Task Force (DMTF) for management of networks and services [DMTF04]. CIM defines a set of common objects such as networks elements, network devices, application and services. The model is intended to be extended as needed to define other classes for specific management areas e.g. policy management. PCIM defines a set of structural classes and associations that allow policy based

rules of the form " if <condition> then <actions>" to be modelled. The core classes of

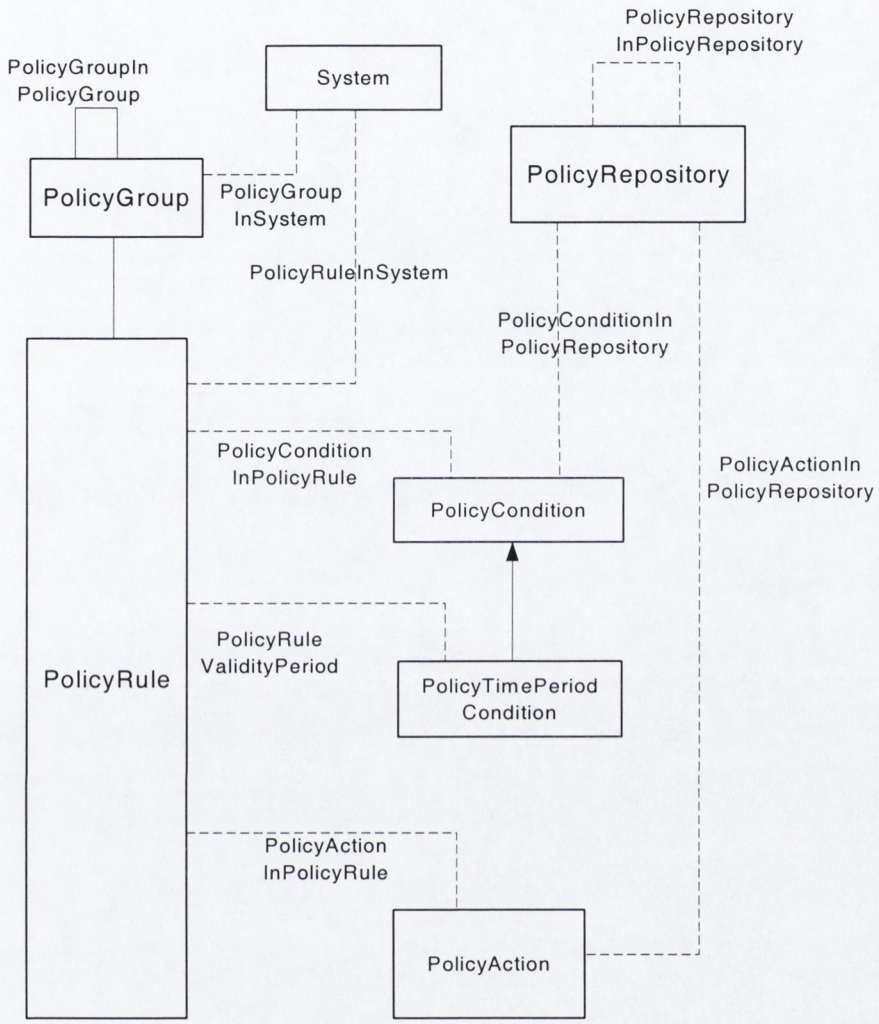


Figure 4-6 IETF Policy Core Information Model

the model are shown in Figure 4-6[24]. Individual conditions can be combined conjunctively (AND'ed) or disjunctively (OR'ed) to form compound conditions in policy rules. Policy rules can also be grouped to form higher levels of reusability.

[24] In fact some extensions and small changes have been made to the model in RFC3460. In particular the PolicyRepository class has been replaced. The original is however retained here as the essence of the model remains unchanged.

The IETF has defined a mapping of the model from the CIM schema to an LDAP representation [PCLD01]. The LDAP schema representation of policy rules is very verbose, as can be seen from Figure 4-7, [Slo02].

```
if (SourcePort==MyWebServerPort) then Color DSCP=5


LDAP Schema
--------------------

 Objectclass:qosPolicyRule
        Type:1
        Direction:out
        Priority:1

        Objectclass:qosColorPolicyAction
                DSCPValue:5

        Objectclass:qosColorPolicyCondition
                Type:Integer OID
                Operator:"=="

                Objectclass:qosPolicyVariable
                        Name:SourcePort
                        Type:Integer OID

                Objectclass:qosPolicyConstant
                        Name:MyWebServerPort
                        Type:Integer OID

                        Objectclass: qosPolicyNumberValue
                                Type:Integer OID
                                PortValue:80
```

Figure 4-7 LDAP Policy Rule Example

As Sloman notes this representation is not intended for human interpretation. The representation will be instead be generated as output from a graphical tool or compiled from a higher level language.

In order to apply policy management in a particular management domain the core model must be extended with domain specific classes that capture the semantics of that area. This has been done in the first place for QoS management, where an information model has been defined [QPIM01]. The example in Figure 4-7 is based on the QoS model.

Policies will be stored as objects in a directory service that will allow policies to be distributed at strategic repositories throughout the network. A policy architecture has been defined to allow policies to be retrieved and applied

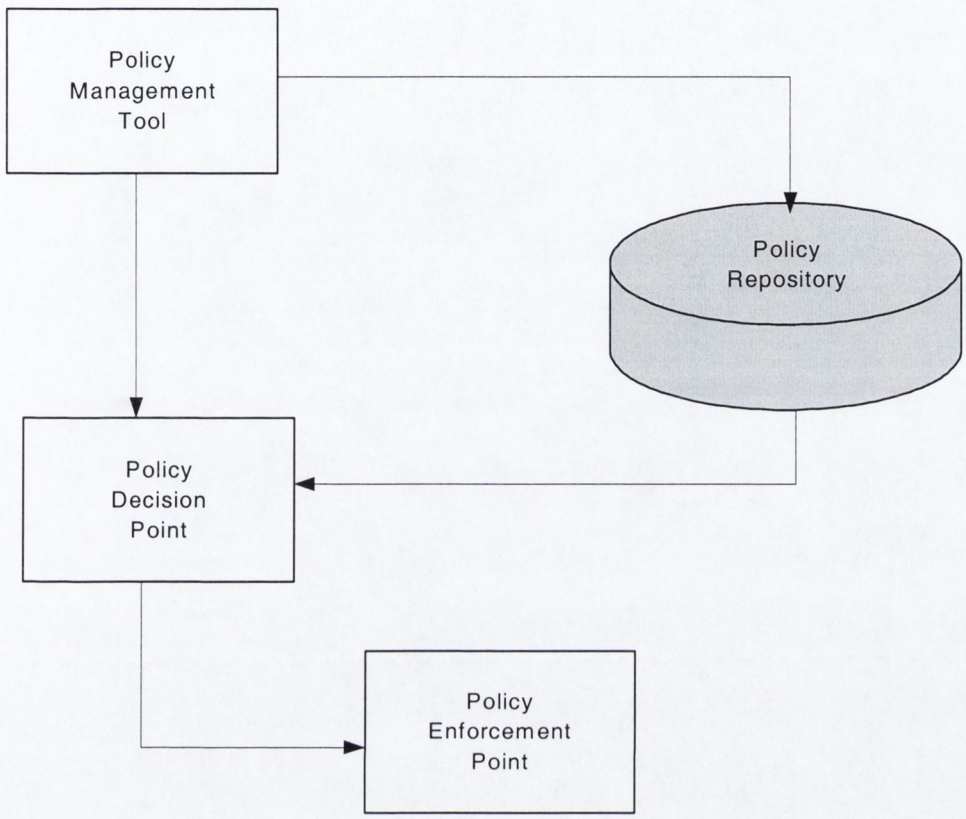


Figure 4-8 IETF Policy Architecture

Policies are retrieved from the repository by a policy server, a Policy Decision Point (PDP), The PDP decides if the policy should be applied and if so informs the Policy Enforcement Point (PEP). The PEP then interacts with the network device or element to enforce the policy. The entities in Figure 4-8 are logical and may be combined physically in a particular implementation. The interaction between PEP and network element is not standardised and may be implemented as wished by vendors. The other interfaces are standardised and COPS, (Common Open Policy System) is proposed as the protocol of choice for deploying policies between PDP and PEP, though other protocols such as HTTP, SNMP or DIAMETER could be used. As Sloman notes,

[Slo02], event triggering is not supported explicitly in PCIM. Instead policies are triggered by events such as packet arrival at a router which will cause a PEP to launch a query to the PDP.

4.3.4 DEN-ng

Strassner, [Stras03], has consolidated and extended a number of the above approaches in the "DEN – new generation"(DEN-ng) standardisation. DEN-ng is significantly different from previous work both in ambition and implementation. Strassner takes the view that the scope of policy management today is too limited to be useful in real world network management. He believes that a more holistic view is needed: specifically PBNM needs to be extended to consider *users* and *process*.

DEN-ng recognises that there are a number of users of the PBNM system in a service providers' organisation e.g. business users, network administrators etc. Ultimately the goal of the business is to make money and so businesses must define and implement network services according to their own business processes and policies. Business policies and goals must be captured in the PBNM system and decomposed/translated from an abstract form into concrete network and device specific policies that implement those business policies. Business users, network administrators and technicians will all have different views of the network and it is vital that a PBNM system capture all these views and be able to translate between the different levels. DEN-ng considers a network to be modelled in a hierarchy of views cf. Figure 4-9.

Strassner call this the policy continuum.

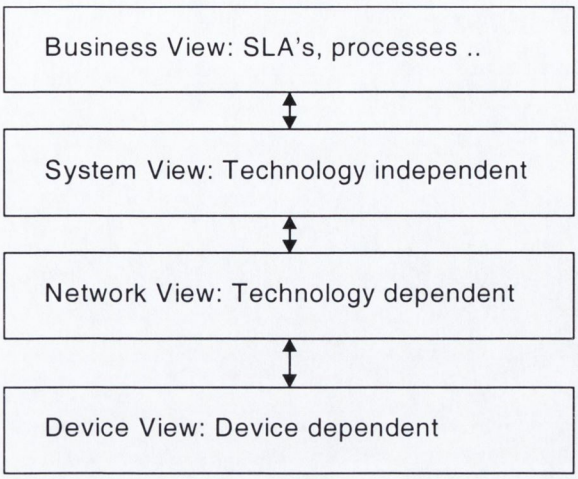


Figure 4-9 DEN-ng Policy Continuum

Different policy rules will exist at each level of the continuum and translation between policies at different levels will be needed. Policies do not exist in a vacuum in a network management environment. Policies tell what to do but not how to do and so they must be accompanied by process. In DEN-ng process is seen as an inherent part of the PBNM system. Policies select which processes to use and processes carry out the task. Policies also control which users are authorized to carry out specific tasks.

DEN-ng recognizes that, in order for a PBNM system to be effective, there is a need for a common way to share information between users, policies, and applications. In other words there is a need for a common information model to represent policy information and information that policy acts on, monitors and manages. The DEN-ng information model describes policies, business, service and network entities in a unified manner. It is based on the use of UML and is what Strassner terms a federated model, meaning it takes information from other sources. DEN-ng is a *layered* information model. It defines a common framework for a number of core models. Core models define generic entities for a particular conceptual area e.g. policy, service and resource. Specialized extensions to these models can then be added for management of particular domains e.g. MPLS VPN. DEN-ng has been heavily influenced by Ponder, [Dam00], and has included a set of languages with its models. Thus one can use different languages to express policies at different layers of the

policy continuum. This recognizes that different notations are appropriate to address different levels of abstraction when formulating policies.

Policy based management is clearly an active field of research with many different approaches and many open questions. One of main issues, it seems to us, is the very broad definition of policies and rules. For example a rule may express a business goal in plain English or a configuration of a very specific QoS mechanism in a router. There is a need to set the various concepts and levels of abstraction in relation to each other. DEN-ng seems to be the best current approach but even DEN-ng has many gaps. We do not believe that there is an ultimate, unifying, policy formalism/language and we believe that a number of languages will be needed. Declarative rules based languages are best suited to expressing higher level goals at the upper layers of the policy continuum. As one moves down the continuum the amount of information to be processed and the amount of equipment to be acted on increases. Therefore, procedural languages will be more applicable as the level of policy detail increases. That is not to say declarative languages may not be used but we do not believe that current declarative policy languages are the best approach to expressing lower level equipment management policies.

4.4 Benefits to Charging

Policy based management has much promise to reduce the complexity of managing the NGN. One of the foremost reasons to use it is that it allows for the direct formulation of service provider business objectives as high level policies and the subsequent translation of these policies to lower level system and network focused. It is difficult to find a more direct expression of a service providers business goals than a pricing model. It is not so much a question therefore, of why one should use policy management for charging as how one can possibly avoid using it. Of course to be usable in the network, high level charging policies (pricing models) have to be transformed to network and device level policies (charging schemes). Depending on the approach one takes polices are expressed directly in a language or can be defined as part of an information model. Carle, [Carl02], has outlined a policy architecture that describes all levels of the charging/accounting/billing systems that are likely to be found in the NGN. There is a direct similarity between this architecture and Strassner's policy continuum, though the hierarchies are subtly different since in

Carles case there is different functionality at each layer while in Strassner's case the function is the same but at a different level of abstraction at each layer. Nonetheless applying DEN-ng to the billing system architecture would, we believe, yield significant benefits to develop a holistic policy management solution for charging and billing in the NGN. Such an approach is beyond the scope of this thesis as we are concerned only with charging. In the context of the policy continuum PEACH policies define charging schemes and must be seen at the system and network level of that architecture.

If policy management allows convenient representation of business goals and flexible manipulation of charging schemes then active networking allows the (active) charging schemes to be deployed and executed on (active) charging nodes at selected locations in the network. Active networking allows for the definition of arbitrary architectures and ensures great flexibility in mapping these architectures to physical entities. This contributes to the generality of the approach and its applicability to a wide range of services and network types.

Chapter 5

5 A Charging Framework

5.1 Introduction

We have seen in previous chapters that new demands will be placed on charging and accounting systems in next generation networks and have argued that programmable networking technologies can be used to provide solutions to some of these demands. In this chapter we develop a specific proposal to meet these needs.

The core elements of this proposal are

- A charging reference model which describes charging and accounting functional entities. These logical entities are considered to be computational entities and can communicate and co-operate as an active and programmable network to provide charging solutions

- A programming model, and language, for the definition of charging programs and policies. The language is known as APPLE (Accounting policy programming language)

- A programming platform to support the development, deployment and execution of charging policies and programs in a distributed environment. The platform has been designed to be easily adaptable to different network technologies and service families. The platform is designated PEACH (Policy execution environment for accounting and charging)

5.2 Scope of PEACH

It is important to understand that PEACH does not of itself provide a complete solution to the charging and billing process. Rather it is an element of an overall solution and as such depends on the existence of other billing system elements in order that a complete solution be provided.

The billing framework proposed by Hartano, [Hart99], and discussed previously, provides a convenient frame of reference to describe the scope of PEACH in the context of an overall solution. A modified version of this diagram is shown below

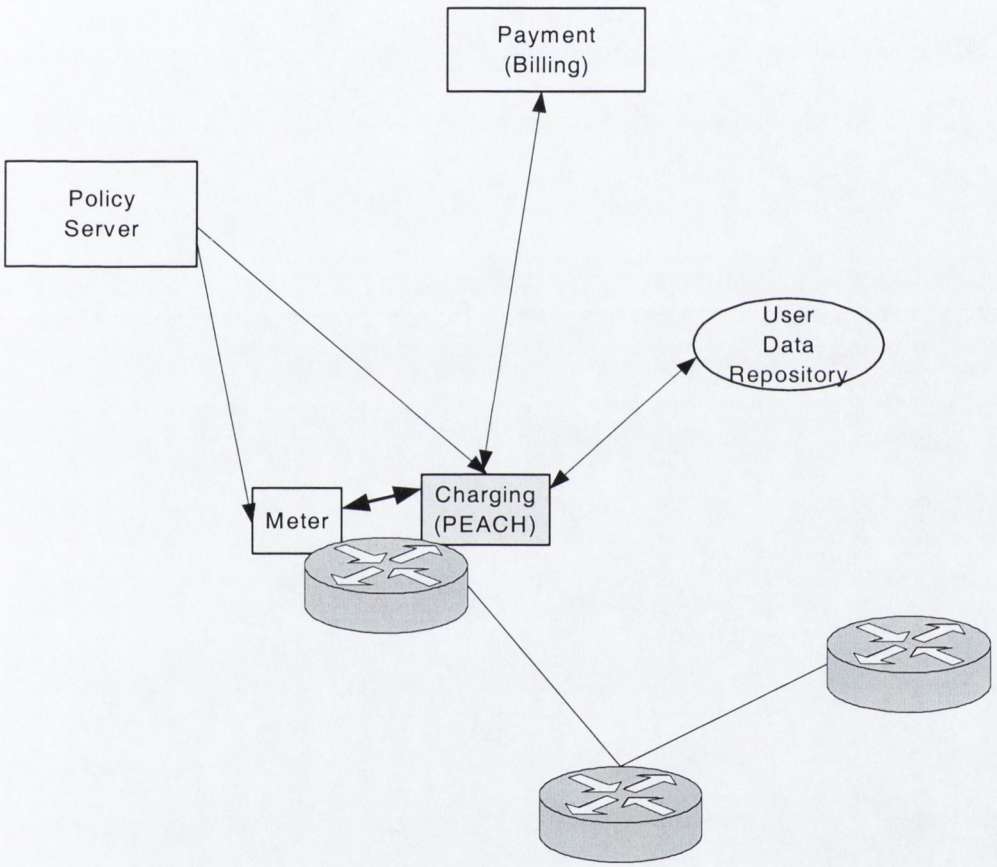


Figure 5-1 Scope of PEACH in Billing Framework

In this diagram PEACH implements the *charging* function. The *collection* and *accounting* functions have been incorporated into the interaction between PEACH and the *metering* function and between PEACH and the user data repository. The *billing*

function is represented as a real time payment mechanism. Policies are deployed from a policy server.

All of the elements above must be present in order to present a complete solution for real time charging and billing. The exact implementation of the various elements will depend on the service scenario. For example metering for IP QoS may involve programming a dedicated traffic measurement integrated circuit with a specific metering policy based on e.g. SRL and thereafter collecting data from a hardware counter while metering MP3 streaming may involve counting the number of tracks played using a Java applet or other software agent.

5.3 NGN Charging Reference Model

When developing a charging reference model for the anticipated NGN we need to bear in some of the issues previously highlighted, namely:

The NGN will be service rich. This will apply to the provision of communication/connectivity services as well as to the provision of content and value added services.

The service marketplace will be very competitive and there will be many types of service provider. Services may continue to be provided by a single service provider or bundled service offerings may be co-operatively provided by a number of service providers. This can apply for provision of QoS related services as well as value-added services

The NGN will be a layered network with each layer using capabilities and services from the layer beneath in order to deliver end user services. Each layer can offer end-user services as well as supply services to upper layers. Very often end-user services are provided by service providers in different layers co-operating e.g. a videoconferencing service provided by a CASP/NSP combination. Services may also be made available by providers in the same layer acting in concert.

Network service providers are well positioned to provided charging and billing facilities to value-added service providers, cf. NTT in the case of I-Mode, and can in fact generate substantial revenues from this service

With these thoughts in mind we propose a model for service charging based on the notion of layered networks and of loose coupling between charging entities on each layer. These ideas are illustrated in Figure 5-2 below:

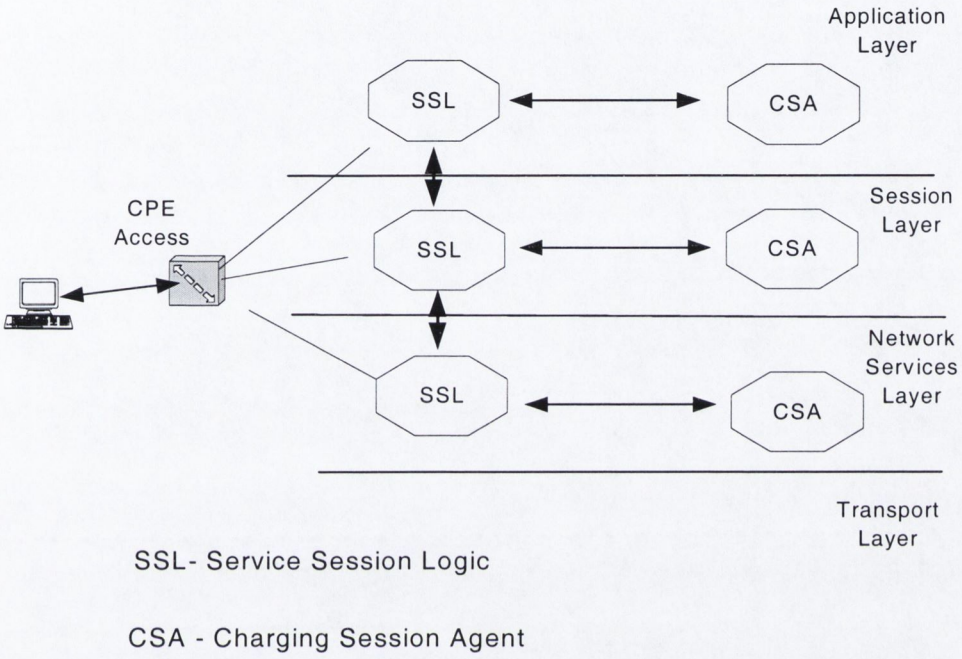


Figure 5-2 Charging Reference Model

The model is shown applying to the network, session and application layers. In fact the model is general enough to apply to any number of service layers.

The notion of *service session* is fundamental to the model presented. A service session is the period in which a service is invoked. This concept applies at all layers, not just the session layer. The logic needed to provide the service on a particular layer is depicted in Figure 5-2 encapsulated in the SSL logical entity. SSL's interact between layers when services are bundled. For each SSL entity there will be a corresponding charging session entity i.e. Charging Session Agent (CSA) functional entity. Because the nature of a service i.e. SSL is so variable the charging framework must provide firsthand support for a flexible definition of charging sessions i.e. a particular CSA will be dependent on and determined by the interworking SSL

A second important principle is that the interaction between charging session agents on different levels must be loosely coupled. There is a need for some degree of coupling since the charging for a service at one layer may be influenced by its usage by a service in a higher layer. We introduce the concept of a *'charging origin'* to allow this indirect interaction. The charging origin is used to influence or determine how charging is applied for a particular service. This allows different pricing models to be applied to the same service for different categories of customer. The use of a charging origin allows for the charging functional entities on different layers to be unaware of each other and to evolve independently while at the same time allowing for interdependent charging to be applied. Charging origin information is provided to the charging functional entity from the service session logic.

Consider a VoIP session as an example of how this might work. The CSA on the VoIP service layer knows that one or more media session will be needed in the network layer. The CSA passes a particular charging origin to the SSL on the VoIP layer. This token is then passed from this SSL to the SSL on the media flow layer and ultimately passed to the CSA on the media flow layer, where the appropriate charging is applied. Thus a flow with a particular type of QoS may be charged for as a standalone service or perhaps have no charge attached if it is part of a VoIP service (assuming the VoIP service is charged for). The setting and distribution of charging origin tokens must of course be co-ordinated across the different CSA's, either by a single provider or between multiple providers if the service layers are owned by different providers. An alternative approach could be for the VoIP SSL to directly reserve resources from the network layer SSL and to pass the charging origin to the network layer CSA via the network SSL.

For real time charging the charge session agent periodically calculates a charge according to the given charging scheme and outputs this charge to an arbitrary charge accumulation agent. The identity of the charge accumulation agent is passed from the SSL at invocation time.

A charging session agent can be decomposed into a number of constituent functional entities as shown in Figure 5-3 below.

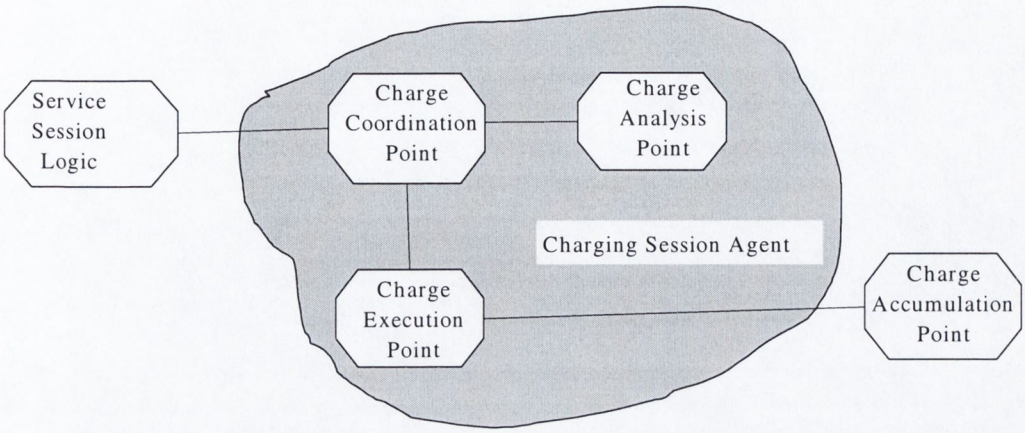


Figure 5-3 Expanded Charging Agent

The constituent functional entities of the CSA are:

- *Charging Analysis Point (CAP)*- This entity is a policy server and contains the policies which determine how charging should be applied for a particular service. It functions in a request/response mode. It is similar to the PDP functional entity in the IETF policy model.

- *Charging Session Coordinator (CCP)* – This entity defines the charging session handling for a particular service. It is stateful and is event driven. It is the main point of interaction with service session logic and coordinates the other charging functional entities during a session.

- *Charging Execution (CEP)* – This entity carries out the actual charging for service usage i.e. it carries out the real time rating and charge calculation and periodically outputs the computed charge to a designated destination. It interacts with the service session logic to read data for charge calculation.

The diagram also introduces a new functional entity – the charge accumulation point., (CMP). This entity serves a receiver of the calculated charge. It is not strictly speaking a part of the charging system. It is rather a part of the billing system and will vary depending on the service provider. Further there will be only one such entity in

instance of user session and it will be located on the uppermost layer which invokes the service. The reference model does not mandate where the CMP is located and it may in some implementations be incorporated with the CCP if appropriate e.g. for prepaid billing.

5.4 PEACH

PEACH is a framework which implements a distributed charging processing environment that provides the features and capabilities of the charging reference model described above. PEACH is a programmable network framework and is based on the related fields of active networking, mobile agents and policy-based networking. PEACH incorporates and combines elements of these, and other, fields to introduce a novel approach to network service charging.

The main features of PEACH are:

- A programming language, APPLE, which allows the rapid definition and introduction of charging policies and the definition of charging agent task logic.
- Direct language support for the distributed charging functional architecture
- A computing environment which provides support for policy rule execution and for communication between PEACH nodes and the external network environment.

Extensive customization facilities exist to adapt PEACH to different network and service types. This is achieved by a PEACH mechanism known as a *context* which allow service specific data and service components, a la active networking, to be introduced as needed. The context provides the definition and instantiation of the 'shared data' between the APPLE rules and logic and avoids the need for a common information model to share information

Architecturally PEACH can be considered as a three layer framework as depicted in Figure 5-4

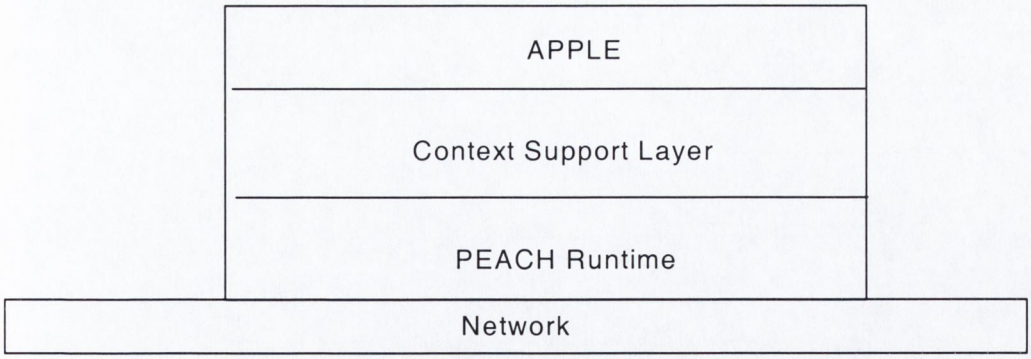


Figure 5-4 PEACH Architectural Framework

APPLE provides a service composition framework. The statements in the language are used to bind together service components, or *policy components* in APPLE terminology, and service charging data, known as *policy parameters* in APPLE, to form policy rules and active charging agent tasks. The context layer contents will vary from application to application. Finally the PEACH runtime layer provides support for program execution and communication. The context and runtime layers together encapsulate the PEACH execution engine.

Policy parameters include network service information, user and customer equipment details, billing information, network state information and so on. Parameters can be fetched from just about anywhere including the network, OSS data storage including databases and directories, customer equipment, co-operating service providers data systems etc. Parameters are fetched into the policy space using policy components .

PEACH is implemented in Java.

5.5 Policy Language

PEACH provides a programming language, APPLE (Accounting Policy Programming Language), to define charging policies and charging agent task logic. APPLE also provides direct language support for the distributed charging architecture introduced earlier.

APPLE provides two constructs to encapsulate program content. A *Rule* defines a specific policy for a particular charging situation. A *Module* defines active task logic for charging agents. It encapsulates state and co-ordinates with other modules to

provide dynamic charging services. The charging model functional entities are represented in APPLE by modules. Rules and modules are active applications (AAs) in active networking terminology.

Rules represent static polices and capture business goals and pricing models. They are not standalone executable elements and must be invoked from within a module though rules may invoke other rules. Typically a charging policy is implemented as a chain of rules where the primary, or root rule, is triggered from within a module.

Data is shared between rules by means of a shared data space or blackboard type data structure [Papa98], though it must be stressed that APPLE is not a full-blown coordination language in the sense described by Papadopolous. High level policies, whether charging related or otherwise, tend to be expressed in the form of independent or semi-independent rules. The shared data space approach in APPLE is primarily designed to directly facilitate the expression of such policies by allowing loose coupling between APPLE policy rules. Policy rules exchange data only via the shared data space. This reduced coupling between the policy rules allows rules to be easily changed or replaced and hence supports rapid introduction of new charging and pricing models to support the new services. A second benefit of the shared data framework is that mechanics of fetching of parameter data is entirely hidden from the policy rules. Modules and rules place and retrieve data on the shared space by means of the language statements

inp(*policy parameter1, ... policy_parameterN*);

outp(*policy_parameter1, ...policy_parameterN*);

The example below illustrate these features of the language:

```
Rule CheckToD {
    inp(time_of_day);
    if (time_of_day < 8)
        && (time_of_day >18)
    then
      todCharging=#no_charge ;
    else
       todCharging=#tod_charge;
     outp (todCharging);
}
```

This policy simply says that no charging is applied for a particular service between six o'clock in the evening and eight o'clock in the morning.

Because the bulk of charging policies involve extensive comparison of data APPLE incorporates statements to make these comparisons easier as shown in the

```
Rule IpQ_getEFcsch{

  inp(originForCharging);

  switch(originForCharging)
  {
     between(1,10) :
        chargingScheme =#flat_rate;
     between(11,20) :
        chargingScheme=#no_charge;
     between(21,100) :
        chargingScheme=#EF_time_volume;
     in (101,102,103,104):
        chargingScheme=#EF_volume;
     gt 500:
          chargingScheme=#no_charge;
      default:
        chargingScheme =#flat_rate;
  }

  call CheckToD;
  meter="ipQosMeter";
  outp(chargingScheme,meter);

} // end IpQ_AnalyseEF
```

next example. This example determines the charging scheme/pricing model to be applied for a particular QoS based on the particular customer or category of customer requesting the service i.e. represented by the originForCharging policy parameter. The example also illustrates how a policy rule chain may be created, in this case the current rules invokes the previous CheckToD ruling using the

call *policy_rule* ;

statement. This causes the particular rule to be invoked and executed, after which control is returned to the calling rule at the statement following the call statement.

APPLE also provides support for retrieving data from network databases or directories by means of

inpk(*entity, policy_parameter1, policy_parameter2 ...*);

The data source is configured as part of the PEACH runtime system. Data is retrieved from attributes of 'Entity' with names 'policy_parameter1' etc and placed in variables of the same name in the local context of the calling rule or module.

Modules are active tasks that implement the functional entities of the charging reference model. Data sharing and control between modules is implemented by message passing. The language provides support for message passing by means of the statements:

send(*node_Address, application, message, policy_parameter1, ..policy_parameterN*)

inputEvent { *event1 -> {}; .. eventN ->{};*};

The 'message' transferred by the send() statement from module A is received as an 'event' in the inputEvent statement in module B. The policy parameters transferred with the message are retrieved in module B by inp() statements. This points out an important aspect of APPLE namely a uniform method of handling data in both rules and modules. The purpose of doing this is to provide a uniform programming model to the policy designers.

A module wishing to pass a message to an application on another PEACH node must know the address of that node. Exactly how that address is obtained is dependent on the application.

Node addresses are defined by PEACH uniform resource locators i.e. PEACH URL's. URL's have the form

PEACH://host:port/interface

The host part of the address is a DNS address while port is used as in HTTP etc. When present the interface defines the *role* of this address i.e. what functional entity this address represents. An example is

PEACH://bilbo.bagend.shire:39001/cap

which identifies the charge analysis functional entity on a particular machine.

The interface portion of the address is used if the underlying communication system needs it e.g. as is the case with distributed object systems.

URLs may be stored in directories or databases and retrieved by the inpk statement e.g. inpk(neId, capUrl, cepUrl) retrieves URL's for a charge analysis point and a charge execution point for a particular network element.

APPLE provides support for direct service component invocation by means of the *action* statement.

action(*service_component.called_method, policy_parameter1,.policy_parameterN*);

The action statement is very general and service components can vary widely in their function. Policy parameters produced or returned by *action* invoked service components must be fetched from the shared data space into module local execution space by an *inp* statement.

The following APPLE example illustrates the use of the above statements

```
Module  AnalyseCharging {

        // message recipient info.
        application = "aTrafficProgram";
        message = "chargeResults";
        capUrl="bilbo.bagend.shire:39001/cap";

        // do analysis
        call IpQ_getEFcsch;

        // get analysis result and return to caller
        inp(todCharging, chargingScheme);
        send (capUrl, application, message, todCharging, chargingScheme);

        // wait for next message and process
        inputEvent {
                close -> {
                        action( Logger.write, application);
                        exit() ; // clean up and die
                 }
         }

} // end Module
```

In this simplified example the module AnalyseCharging is invoked when a charge coordination point sends a message requesting analysis. The first three lines of the module identify the application and message to which to send the analysis results. After this the module calls the appropriate charging policy rule. It then retrieves the analysis results and sends them to the receiving application i.e. another APPLE module in the node identified by *capUrl*. After this the module enters an event loop to wait for the next message. In this case only a single message is shown *close* which causes the module to log a message and then exit.

A so-called "root module" exists for each context. This module is invoked each time a message is received which is not destined to a specific module instance. The root module analyses the incoming message and invokes the appropriate module (type) to handle that message. Thereafter all messages destined for that module instance are directed to the module instance by the run-time system. The root module is in essence a map between message types and module types.

There is a problem of modularity or information hiding in the example above. The calling module has to have a knowledge of the parameters being manipulated by the

policy rules. This can lead to complexity, handling errors and problems with maintenance. To avoid this an extra statement is provided to allow the policy rules to specify policy parameters that may be sent in a message, independently of the calling module. This is enabled by the *pack* statement .

pack(destUrl, policy_parameter1 ...., policy_parameterN);

destUrl is a built in variable which must be preloaded with the actual destination PEACHURL. Parameters are then associated with the message to be sent and are transmitted to the destination when the *send* statement is invoked in the calling module.

In addition to message passing between modules APPLE also allows modules to receive events from arbitrary sources. These could be for example, events relating to the state of the network, time-out events, or any other factor in the external environment. We refer to events of this type as *ns-events* (network and system events). Ns-events are received by means of the *inputEvent* statement. In order to receive such events the module must subscribe to a particular *event channel.* It does this by means of the subscribe statement

subscribe(*event_channel, event_name, event-filter*);

The *event channel* identifies the source of the expected event. Many types of event may be received from an event channel and hence *event_name* identifies the particular event expected. A module may choose to receive only a subset of events of a particular event type. It does this by including an event filter which the event channel applies to incoming events. Filter expressions are relational compositions of fields in the ns-event e.g.

(network_load > 5000) && (time <= 18)

Filters are expressed as strings and are evaluated by the event channel.

In addition to ns-events APPLE also supports the concept of policy events (ala PDL), or so-called *p-events. (Note: not implemented in the current version)*

APPLE also supports code mobility. It is possible to move a module from one node to another and then instansiate the module on that node. It is also possible to transfer state (policy parameters) when migrating the module. This is enabled by two language statements : *move* and *start*.

move(*PEACHURL, module_name*) ;

causes the module identified by to be installed on the node designated by PEACHURL. The complete code for the module is migrated. The move statement causes a handle to be returned, policy parameter *functionId*, which is used later in the *start* statement to start execution of the module.

start(*PEACHURL, functionId*);

Two different statements are used to allow give more flexibility to the process by allowing the installation and invocation by different nodes. The most typical example is for a charging analysis point to install a module and for a charge coordination function to subsequently cause execution of the installed program. Both actions can however be given from the same module. State is migrated by use of the pack statement.

APPLE is a weakly typed language similar to many scripting languages and indeed in certain regards APPLE can be viewed as a scripting language. APPLE variables can be bound to any of the primitive types at run-time i.e. integer, float or string. This has many advantages. It enables new parameters to be easily added and is therefore very valuable to modify a run time system.

APPLE is an interpreted language and programs are evaluated at run time.

5.6 Context support

In order to be used in a particular network service environment PEACH must be 'primed' with service specific policies and data. This is achieved via the context feature described above A context is a PEACH concept that defines a service category for which charging policies and modules may be specified. A PEACH node can support multiple simultaneous contexts.

Examples of contexts are "IP QoS services", "ATM_bearer_services", "SIP VoIP services", "Video_on_Demand services".

A context defines a family of policy parameters, policy rules and modules and service components and perhaps context events. These elements are defined in an XML context configuration file. An example context file for IP QoS services is shown in Figure 5-5

```
?xml version="1.0"?>
<context name="IpAccounting">
<protocol>prr</protocol>
<ns-event_list>
        <ns-event  name="timeout" producer = "Timer"  >
                <field name = "timeFilter" type = "String" />
        </ns-event>
</ns-event_list>
<policy_param_list>
        <policy_param name="dscode" typeNo = "5001" keyed = "no" keyEntity ="">
                <field name = "dscode"  type="int" defval ="6" />
        </policy_param>
        <policy_param name="svc_category" typeId = "5002" keyed = "no" keyEntity ="">
                <field name = "svc_category" type = "String" defval="EF" />
        </policy_param>
        <policy_param name="peakRate" typeId="5003" keyed = "no" keyEntity ="">
                <field name = "peakRate" type = "int" defval="500000"/>
        </policy_param>

.....
</policy_param_list>

<policy_rule name ="IpCcf" type="ap">
<![CDATA[ Module IpQ_Coordination {

 inp( dscode, peakRate, jitter, neId,originForCharging);
 inp( neId, snet, dnet,dport, sport, custId)
 inpk(neId, capUrl, cepUrl);
 dayOfWeek=4;
 application="IpAccounting";  message="analyseCharging";
 send(capUrl, application, message,
     neId, dscode, peakRate, jitter, originForCharging,
     snet, dnet, sport, dport, custId, dayOfWeek);

 inputEvent
 {

   chargeResult ->
   {
      inp(analysisResult);
      if (analysisResult == "dynamic_charging") then
      {
         inp(functionId);
         start(cepUrl, functionId);
      }
      inp(sourceURL);
      app="DsTmf"; mess = "tmChargeResult";
      send(sourceURL, app, mess, analysisResult);

   } // end chargeResult

 } // end inputEvent
} // end Module IpQ_Coord

]]>
</policy_rule>
...
...
</policy_rule_list>
</context>
```

Figure 5-5 Context XML file for IP QoS services

This file defines a context called "IpAccounting" and its associated family of policy parameters (dscode, svc_category, peakRate) , modules and ns-events. This particular context provides charging and accounting data and modules for IP QoS related services. The example includes typical policy parameters and the module for the IpAccounting charging session agent (IpQ_Coordination).

A simplifying assumption in PEACH is that because data is being processed in real time it is necessary only to manipulate primitive data rather than structured data. There are thus no entities or relationships visible in APPLE rules or modules and thus an information model of charging domain entities is not needed. This is further the case since the policies are not modelled as objects. The context mechanism enables the basic data type vocabulary to be defined and thereby obviates the need for a complex object oriented schema e.g. such as CIM.

A context must also provide the policy components that these rules, modules, parameters and events refer to. These policy components are a combination of hand coded and automatically generated programs. Policy components must be provided to source policy parameter data. Policy parameters may, in principle, be drawn from arbitrary sources and the PEACH context designer has complete freedom to extract data from any suitable source. This in turn puts the onus on the context designer to provide the appropriate policy component. PEACH does provide in built support for automated generation of policy components to fetch data where parameters are derived from an underlying policy protocol.

Each context has an associated policy protocol. The policy protocol is used to transfer data between nodes. As such a policy protocol is independent of contexts and many contexts can use the same policy protocol. PEACH is designed to support different policy protocols. This is done in order to be able to adapt PEACH to multiple network technologies. All policy protocols however must conform to a pattern. They must be able to transfer data transparently i.e. the protocol has no inherent knowledge of the meaning of the data it carries. The data objects in the policy protocol are referred to as *protocol objects*. Policy protocols must support delivery of the data to an agent or handler at the destination node. Many current Internet management protocols fit this model including COPS, RADIUS, DIAMETER and SNMP. In principle any of these protocols can be used to transfer data between the nodes. As may be inferred from the

use of these protocols the relationship between PEACH charging functional entities is similar to the PEP – PDP relationship defined for IP QoS policy modelling, [QPIM01]. PEACH provides a default policy protocol call **Prr** (PEACH Request Reply protocol).

It is an assumption in PEACH that the bulk of policy parameters will be derived from the underlying policy protocol. PEACH provides for the automated generation of policy components for policy parameters which are derived from underlying protocol objects. The mapping between policy parameters and protocol objects is specified in an XML file. An example of such a file that maps IpAccounting context parameters to the Prr protocols is shown in Figure 5-6

```
<?xml version="1.0"?>
<context name="IpAccounting">
<protocol name = "prr" />
<compile order="yes" />
<mapping_list>

        <policy_param name="dscode" >
                <policy_object name = "trafficClass" />
                <field name = "dscode" />
        </policy_param>
        <policy_param name="peakRate" >
                <policy_object name = "dsFlowSpec" />
                <field name = "peakRate" />
        </policy_param>

        <policy_param name="MTU" >
                <policy_object name = "dsFlowSpec" />
                <field name = "mtu" />
        </policy_param>

        <policy_param name="jitter" >
                <policy_object name = "dsFlowSpec" />
                <field name = "jitter" />
        </policy_param>
..
..
        <policy_param name="custId" >
                <policy_object name = "custInfo" />
                <field name = "Id" />
        </policy_param>
</mapping_list>
</context>
```

Figure 5-6 IpAccounting context policy parameters to Prr protocol objects mapping

The context definition file and the parameter-to-protocol object mapping file are used to compile the policy components which are then used at run time by the modules and rules to extract data from and insert data to the policy protocol.

Using these features it is very simple and easy to introduce new policy parameters into a context vocabulary. Of course it is more difficult to introduce new protocol objects as this entails modifying the underlying policy protocol and this may have impacts on a number of network nodes especially when the policy protocol is a standard protocol such as COPS. Where the protocol is non standard, such as Prr, then it is easier to introduce changes. In general changes to underlying protocols are not made very frequently, and protocol objects once defined are assumed to have a relatively long shelf life.

To support the run-time retrieval of policy parameters from policy objects PEACH also automatically generates helper classes to enable the policy components from the policy protocol. All data objects carried in a policy protocol must be specified in an XML file. These XML definitions are then used to generate the policy object helper classes. An example file for the Prr protocol is shown below in Figure 5-7.

Protocol objects are derived from a common base and provide a standard interface for all protocols. Each particular protocol must then implement the interface for its own particular set of protocol objects. Providing a standardised interface allows the PEACH run time system to work in a protocol independent manner and allows for protocols to be changed without affecting the run-time system.

```
<?xml version="1.0"?>
<protocol name = "prr">
<client_list>
        <client name = "peach" number ="" />
</client_list>
<policy_object_list>
        <policy_object name="NE" typeId = "1001">
                <field name = "neId"  type="String" defval="Annagh"/>
        </policy_object>
        <policy_object name = "Application"  typeId="1002">
                <field name = "RSVP_App" type = "String" defval="VoIP"/>
        </policy_object>
        <policy_object name="custInfo" typeId = "1001">
                <field name = "Id"  type="String" defval="bob@lsoft.ie"/>
        </policy_object>
        <policy_object name="intfLoadState"  typeId="1003" >
                <field name = "load"  type="int"  defval="0"/>
        </policy_object>
        <policy_object name="trafficClass" typeId="1004">
                <field name = "dscode"  type="int" defval="6" />
        </policy_object>
        <policy_object name="afData" typeId ="1005">
                <field name = "qos_class"  type="int"  defval="1"/>
                <field name = "dropPrecedence"  type="int" defval="2" />
        </policy_object>
        <policy_object name="dsFlowSpec"  typeId = "1006">
                 <field name = "mtu" type = "int" defval="600" />
                 <field name = "jitter" type = "int" defval="90" />
                 <field name = "peakRate"  type = "int" defval="500200"/>
                 <field name = "averageRate" type = "int" defval="90000" />
                 <field name = "lossSensitivity" type = "int" defval="1" />
                 <field name = "burstSize" type = "int" defval= "150000"/>
        </policy_object>
        <policy_object name="StartTime"  typeId = "1007" >
                 <field name = "startTime" type = "int" defval= "1030" />
        </policy_object>
        <policy_object name="dsFlowId"  typeId = "1008">
                 <field name = "FlowType" type = "int" defval = "1"/>
                 <field name = "dest"  type = "int" defval="4444444444" />
                 <field name = "destMask" type = "int" defval= "666666666" />
                 <field name = "source" type = "int" defval="5555555555"/>
                 <field name = "sourceMask" type = "int" defval="6666666666"/>
                 <field name = "dstPort"  type = "int" defval="3455"/>
                 <field name = "srcPort"  type = "int" defval="5678"/>
                 <field name = "directionality" type = "int" defval="3"/>
        </policy_object>
        "
        ....
        </policy_object_list>
</protocol>
```

Figure 5-7 Prr Definition File

5.7 Architecture

An architectural view of PEACH that shows the main building blocks and API's is shown in Figure 5-8.

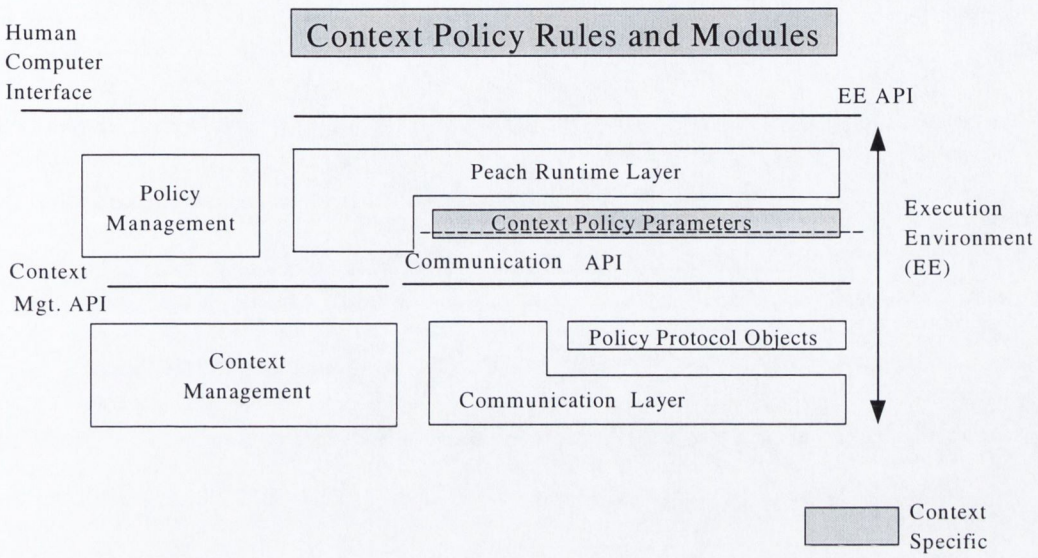


Figure 5-8 PEACH System Structure

The greyed areas shown context specific components and modules. This diagram shows the total set of PEACH functionality and includes both runtime and development components. The "Context Management" block is used both at runtime and development time.

Considering the PEACH system in terms of the active networking node architecture of Figure 4-4 we can note that PEACH does not make as sharp a distinction between the EE and the node operating system, or NodeOS, as is done in active networking. The primary reason is that PEACH is not designed to provide architecture support for many forms of active networking. Rather its purpose is to support one specific implementation of active networking to solve a specific problem i.e. charging policy management.

The APPLE language and its associated interpreter provide the API and runtime platform for the EE while other elements of the run-time system and the

communication system can be considered to implement the NodeOS. APPLE rules and modules are the active applications (AA) of the PEACH system.

The PEACH Runtime subsystem is essentially the 'engine' of a PEACH node and provides support for

- Implementation and interpretation of APPLE.
- Loading and initialisation of the PEACH node.
- Location, dynamic binding and activation of service components i.e. enabling *service composition* The location path is normally found on the local storage system but PEACH does support downloading of information across the network
- Transparent mapping of policy parameters to protocol objects.
- Session management and allocation and scheduling of resources.
- Reception and distribution of events.
- Interaction with network directories and databases.
- SDK support for customisation

The runtime subsystem interacts with the Communication subsystem and the Context Management subsystem via generalised API's. The Runtime subsystem is context independent and is not affected when contexts are added or changed.

The Communication Subsystem primarily provides support for:

- Policy protocol handling.
- Event handling
- SDK support for customisation of both policy protocols and events.

For policy protocol handling it provides generalised interfaces to the runtime subsystem which are protocol independent. This interface must in turn be specialised

for each policy protocol implementation. The primary interface to the communication system is via a *Policy Protocol Handler*, or PPH. One PPH is assigned, per node, to handle inter-nodal communication between an AA (i.e. module instance) running on two separate nodes. If an AA communicates simultaneously with a number of nodes then a PPH is allocated to it for each of the nodes with which it interacts. The second main interface that the communication subsystem offers is the *Policy Protocol Object* or PPO. The PPO interface allows the runtime subsystem to transparently read and write policy protocol data to and from the PPH.

The communication system also provides support for definition, receiving, filtering and forwarding of events. It provides general interfaces for event producers and consumer that can be customised by event channels to adapt for different event categories. It provides a base event type that may be specialised for different actual events. It also provides support for event filter evaluation via a language interpreter. The event filter language is a subset of APPLE.

The Context Management subsystem provides two main functions:

- Parsing and loading of the various context and protocol XML files
- Storage of runtime context data.
- SDK support for context customisation. This includes base classes and interfaces for policy parameters.

The subsystem is used both at (context) development time and at runtime.

The PEACH Management subsystem is used for context development and policy deployment and management. Context development includes design and test of AA's i.e. policy rules and policy modules, generation of policy parameter components, generation of protocol object components and design of general policy components (i.e. service components).

The structural view of the PEACH system can be complemented by a runtime or process view of the system. This is shown in Figure 5-9 below.

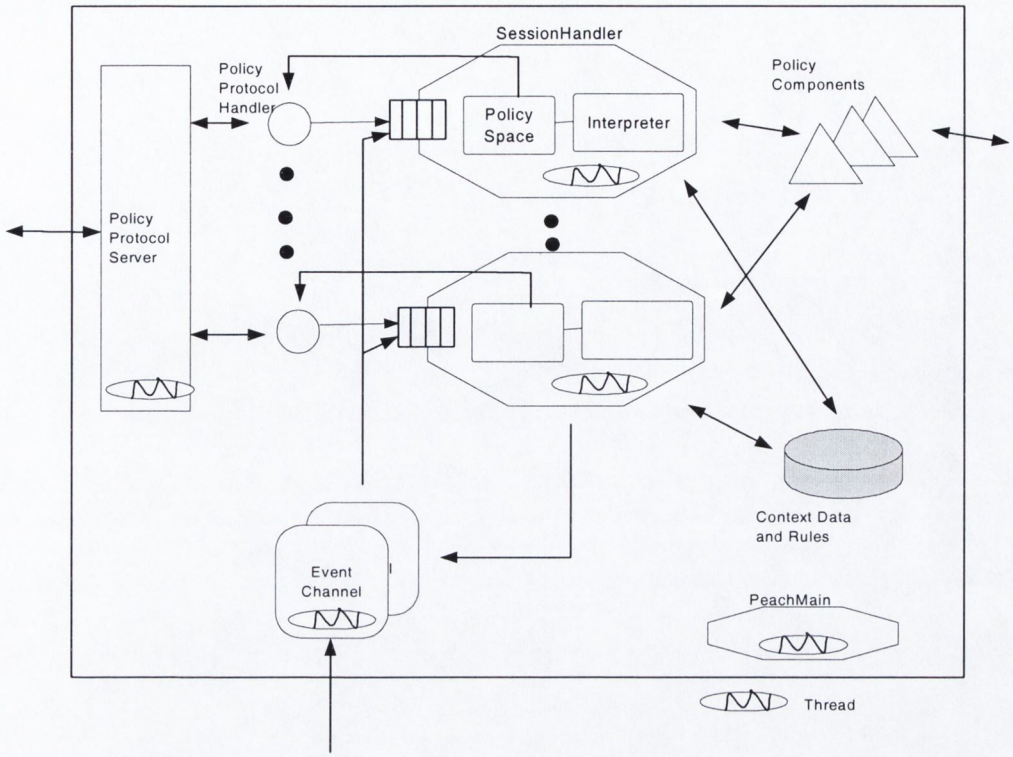


Figure 5-9 PEACH Node Runtime View

Typical operation is as follows:

1. PEACHMain handles startup of the system. Context information is loaded by PEACH at start up. This information is stored in the Context Data repository. A PEACH node has the capability of handling many different contexts at the same time. PEACHMain runs in it's own thread of control and can be used to stop or restart the system and for various other management operations including installing and starting migrated modules.

2. When a message is received at a PEACH node the policy protocol server determines if this is a new request for service. If so it allocates a new policy protocol handler and creates a new session handler.

3. The session handler runs in a new thread and it in turn creates a new instance of the APPLE interpreter and a *Policy Space* and allocates other resources. A policy space is the conceptual representation of the shared data metaphor used in APPLE. This object manages all runtime interaction between the APPLE environment and the external environment, including other subsystems. The

thread of control is indicated by the octagon figure in the diagram above. A separate thread of control is allocated to each new session.

4. The policy protocol server then creates an event object to handle the incoming message and inserts this event into the session handler event queue.

5. The context to be activated is contained as data in the incoming message and the session handler reads the context *boot module* from the repository and activates the module via the interpreter. The boot module maps the incoming message to the actual AA module that is the target of the message.

6. The AA module then continues to execute. The runtime system manages interaction with policy components, context data repository and other subsystems as the AA statements are executed.

7. The AA may subscribe for events from one or more events channels. The event channel in turn has to ensure that it receives the external events and a subscription from an AA may cause an underlying subscription from the event channel to an event producer elsewhere in the network. The exact details of any event channel are outside the scope of the PEACH platform. Each event channel runs asynchronously in its own thread.

8. Incoming messages and events are queued to the session handlers. This ensures that the correct temporal ordering of events is maintained. The policy space object communicates directly with the policy protocol handler when sending messages.

5.8 Implementation

PEACH is implemented in Java.

5.9 Comparison with State of the Art

We believe that the work herein significantly advances the state of the art in the application of programmable technologies to the issue of NGN charging.

There are a number of similarities between the work here and the policy management research efforts presented earlier. In Stone's, [Ston01], terminology APPLE is a "network policy language". As such it can be compared with Ponder and PDL rather

than the lower level policy languages. APPLE incorporates some features from both languages, in particular the capability to handle simple and composite events, which are common in both those languages. APPLE also borrows some minor features such as the 'action' statement from Ponder. However there are significant differences between the languages also. Ponder and PDL are general-purpose policy languages while APPLE has been designed specifically for the distributed NGN charging system. APPLE thus incorporates a number of features that support manipulation of charging data and communication between distributed nodes. These include

- The notion of a "shared data space", or blackboard type structure, for sharing data between policies and modules. This simplifies the writing of policies and makes policies more loosely coupled and reusable and hence makes it easier to create new charging schemes

- Direct language support for creating data look up tables and for retrieving data from these tables. Again this simplifies the preparation of policies.

- Primitives to enable communication between nodes and to migrate modules between nodes.

Further APPLE combines all these features in a unified syntax. These features combine to make APPLE significantly more suitable to handle charging system requirements than any other policy language[26].

Because APPLE provides direct support for distributed nodes it allows the creation of arbitrary virtual active networks. This in turn means that APPLE can facilitate the creation of arbitrary charging architectures. We have described one such architecture in this chapter that supports our vision of real time charging in a multi-layered NGN. But that is only one such architecture. We are free to create new charging architectures if the needs of the NGN change. This is a powerful feature and greatly enhances the adaptability of APPLE to meet ongoing NGN charging needs. We do not

---

[26] Conversely APPLE does not function as well as the other languages for general purpose use. Nor is it intended to .

believe that this degree of flexibility is provided by any other approach we have reviewed.

In terms of the DEN-ng policy continuum APPLE is located at the system or network layer. It is used for defining charge schema rather than pricing models. We have chosen a direct language approach rather than an information modelling approach for APPLE. This is simply because we regard information modelling as an overly elaborate approach to express charging policies. While today charging data is standardised the increase of real time processing in the NGN will lessen the need for standardisation. Using an information modelling approach would have made the definition and execution of policies more difficult. Neither does such an approach lend itself to expressing polices to manipulate data. It is feasible that APPLE could be incorporated as a language constituent of a DEN-ng charging model, but this does not change the point being made here. Rather it serves to embed APPLE as part of a larger holistic charging and billing policy system.

APPLE is procedural while PDL and Ponder are declarative. This again reflects the purposes for which the languages were designed. APPLE is designed to be stateful to support charging sessions while Ponder and PDL are meant to express atomic policies that do not retain state. The procedural paradigm is very familiar to millions of programmers worldwide and this facilitates the creation of policies. Declarative languages do have advantages in many cases but we believe a procedural language better meets the needs of NGN charging.

The PEACH framework has also been designed from the beginning to provide support for extensibility and adaptability. The context mechanism means that widely varying service types based on multiple technology types can be simultaneously active in a PEACH instantiation. Contexts allow us to import technology and service specific charging parameter vocabularies and the PEACH SDK allows for quick and easy incorporation of these parameters into the platform. PEACH is also architected to be policy protocol independent and this means it can work with legacy networks and allow new protocols to be introduced without affecting the operation of the charging logic. No other policy based approach that we have seen gives this degree of flexibility and adaptability.

The works of Briscoe, [Bris00, M3I03], and Travostino, [Trav00] are the nearest programmable networking based charging approaches to PEACH that we have come across. PEACH is very comparable with AIACE and they share many features and goals in common. As with PEACH, AIACE pushes much of the charging and accounting system down into the network layer. Unlike PEACH AIACE also includes the metering/accounting part of the overall system. AIACE also seeks to reduce the amount of charging traffic through the network although it differs from PEACH as it still produces CDR's albeit with reduced data. As presented AIACE is primarily an architectural approach and no information is given on how policies are defined or how "plug-ins" are deployed and activated. It is thus difficult to gauge how flexible AIACE is for particular service scenarios. A further limitation is that AIACE does not address the issue of interoperation between service providers to charge for bundled services. AIACE in its current form can not therefore provide the flexibility and support for a competitive NGN marketplace.

Briscoe's work also has much in common with PEACH including the concept of "active tariff" and the distribution of charging logic into lower layers. Briscoe goes a step further than PEACH by moving logic to the customer's equipment. This raises issues of trust and security. PEACH does not depend on the use of customer machines nor is it exposed to their misuse. The scope and goals of Briscoe's work is different to PEACH. PEACH is intended to provide a flexible, extensible and scalable solution to real time charging in a multi-provider NGN and while Briscoe is also concerned to provide flexibility, his major goal is to use the charging/pricing system for traffic management by means of dynamic pricing. He also produces data records to log network usage. PEACH is a more general and suitable solution for real-time service charging. Briscoe's work is an innovative and promising approach that may in the longer term yield significant benefits but it does not for now meet the needs of NGN charging.

Chapter 6

6 Evaluation of the Approach

6.1 Introduction

This chapter is intended to show that the PEACH system meets a number of the principal requirements of NGN charging systems which were identified earlier in this work and to demonstrate the key contributions of the PEACH platform.

The summary main requirements of NGN charging systems are

1. To enable the creation of a range of pricing models to support competitiveness in the marketplace and to facilitate the introduction of a wide variety of services in a number of heterogeneous networks.
   This requires key attributes of *flexibility* and *adaptability*

2. To support real time charging.
   This requires key attributes of *scalability* and *distribution.*

PEACH proposes to meet these requirements by moving the charging/rating function to the point in the network where the data is collected i.e. into the network elements and by basing the solution on a programmable networking techniques viz. active networks and policy management. A programmable networking approach ensures PEACH has the key required attributes highlighted above.

A more detailed set of requirements and platform features needed to fulfil the two summary requirements above is given in

. The table lists the complete set of features and platform characteristics and indicates which features are validated by which case study.

Table 6-1 PEACH Features mapped to case studies

Attribute	Case Study		
	Diffserv QoS	SIP VoIP	MP3 Streaming
Marketplace Support			
Pricing model diversity	✔	✔	✔
Service diversity	✔	✔	✔
Service bundling		✔	✔
Service provider interworking			✔
Charge Allocation	✔		
Real time charging	✔	✔	✔
Platform Attributes			
Platform extensibility	✔	✔	✔
Computing performance	✔		
Architectural flexibility	✔	✔	✔
Mobility Support	✔		

The features/requirements listed in the table have been identified and discussed in earlier chapters.

A important point to grasp here is that many of the features and characteristics are demonstrated by a combination of case studies rather than by a single case study. The approach to validating PEACH is therefore based primarily on an analysis of a combination of the case studies. Some features are however illustrated in the context of a particular case study.

Three case studies are presented. These are

- Diffserv QoS connectivity

  This case study examines the provision of basic connectivity services over a Diffserv network. It is important for a number of reasons;

  - it is a network layer service;
  - QoS is an important building block for upper layer services;
  - it is sufficiently complex to demonstrate many of the main platform features.

- VoIP Telephony

  This case study is based on VoIP service based on SIP. It is a session layer service. It is important because it is a reasonably complex service that combines with the network (lower) layer QoS service to form a bundled service offering

- MP3 Streaming

  This case study features delivery of MP3 music files from an ASP to a consumer across the network. It is an application layer service that also bundles network layer QoS service to form a composite service based on co-operation between two different service providers

The case studies are chosen to cover all of the layers of the NGN service model introduced in chapter two.

6.2 Charging for Internet QoS

6.2.1 Motivation

This case study focuses on the charging of IP point-to-point QoS bearer services, essentially as described in chapter 2. QoS charging is important for a number of reasons:

- QoS and charging are closely linked as we have seen from the literature review in chapter 3. Pricing, and hence charging, has a key role to play in ensuring that users are incentivised to choose the most appropriate level of QoS to meet their needs. Otherwise users are likely to choose the highest level of QoS available with the follow effects of congestion, loss and delay [Shen95]. Also as bandwidth becomes a commodity, service providers view QoS as a key enabler for service differentiation and hence increased revenue generation [Leida98]. It is important to show how the charging framework can be used to provide QoS charging solutions

- QoS services will be bundled with higher layer services to provide composite end user services. Charging for such services may involve interaction between charging systems of different service providers. An understanding of how the QoS charging system operates will be needed in order to demonstrate the composite service charging scenario.

The case study is the most complex of the three and is used to demonstrate many of the essential features of the charging framework architecture. In addition to contributing to the demonstration of system global features the case study is designed to highlight a number of individual features of the PEACH system in more detail. These are shown in Table 6-2 below. This table is a subset of the overall feature set presented above.

Table 6-2QoS Case Study Individual Features

Attribute	Diffserv QoS
Marketplace Support	
Charge Allocation	✔
Platform Attributes	
Computing performance	✔
Mobility Support	✔

The case study is designed to illustrate *charge allocation* by examining a number of charging schemes for QoS connectivity provision. These include

- Sender based charging for simplex data transmission
- Sender based charging for duplex data transmission.
- Split charging – sender charged for uplink, receiver charged for down link
- Receiver based charging – receiver charged for uplink and downlink

These charging schemes are described in more detail below.

The case study is also designed to examine the *performance* of the system under a number of distribution scenarios. The goal of this part of the experiment is to obtain a feeling for the computational feasibility of the approach rather than to obtain optimised performance figures. This will be described in more detail in the Analysis section of this case study

Finally the case study is also designed to show how support for *mobility* related charging can be implemented. Support for mobility can take a number of forms as we have seen from some of the previous work viz. handset prepaid billing, [Lin01], and the mobile agent charging approach described by Bellavista, [Bell02]. In the first example the charging function responds to events from the network to change the charging tariff depending on the user location. The charge calculation is simple i.e. a number of charge units per unit time. The second example is more elaborate and the charge function is implemented using mobile agent technology in which the charging function migrates through the network as the user moves. This case study does not directly implement user mobility but demonstrates the capability of PEACH to implement mechanisms similar to the above which can be used to support mobility related charging.

6.2.2 Scenario

The case study is specifically focussed on customer provisioning of Diffserv connectivity services. A user is connected to an ISP who offers Internet access with a range of different Diffserv QoS values. Users enter into agreement with the ISP for

connectivity for a particular QoS. A service level agreement (SLA) encapsulates the service contract and contains information such as customer details, authentication information, maximum bandwidth allowed, pricing/charging details etc. SLA's are created via a Web interface. Details of the SLA are stored in the service provider database. Once an SLA is put in place a user can request connectivity to the Internet. The user must specify the traffic profile details for the request in terms of e.g. connection endpoints, average and maximum bandwidth and so on. The request for connectivity is signalled in this example using a Web interface. The request for service, (Resource Allocation Request – RAR) will be directed to a bandwidth broker server in the service provider management system. The bandwidth broker will check the user SLA details and if the request is within limits will allocate the resources and allow the connection request and update the SLA to track the allocated resources. If the SLA would be exceeded the request is denied. When the connection is removed the bandwidth broker is informed and the SLA is updated. Any number of simultaneous requests may be admitted as long as the SLA is not breached.

The scenario is implemented using the actual network shown Figure 6-1. All the devices are PC's running either Linux or Windows. Boyne and Nore are edge routers on a null Internet. Erne and Slaney are hosts while Shannon acts as an application server. Boyne, Nore, Shannon and Slaney run Linux while Erne runs Windows 2000. The following functional elements are shown in the diagram

- BB server – a bandwidth broker server which manages all interactions for service management and allocation of resources.

- BBR – an edge router client which allocates resources on the edge router under command from the BB server.

- SLA client and RAR client – used by the user host to set up an SLA and to request resource allocation.

- PEACH functional elements, Charge Control Point (CCP), Charge Analysis Point, (CAP), and Charge Execution Point, (CEP), cf. Figure 5.3 –CCP and CEP are allocated on the edge routers

The bandwidth broker, (BB server), and its associated components i.e. the SLA client, RAR client and edge router client, (BBR), are based on an implementation from the University of Kansas, [Sund99]. The original system is implemented using C, cgi scripts and Perl. The original bandwidth broker, [BB98], used a resource allocation protocol called the Bandwidth Broker Transfer Protocol, [BBTP98]. These components were modified to implement the current case study and the edge router client was totally rewritten in Perl. A complete database of customers, services, network topology and SLA's was created and implemented in MySQL, [MYSQ04]. The Linux Traffic Control, (TC), functions were used to implement Diffserv. The charging system components were implemented using PEACH.

Figure 6-1 Diffserv Test Network

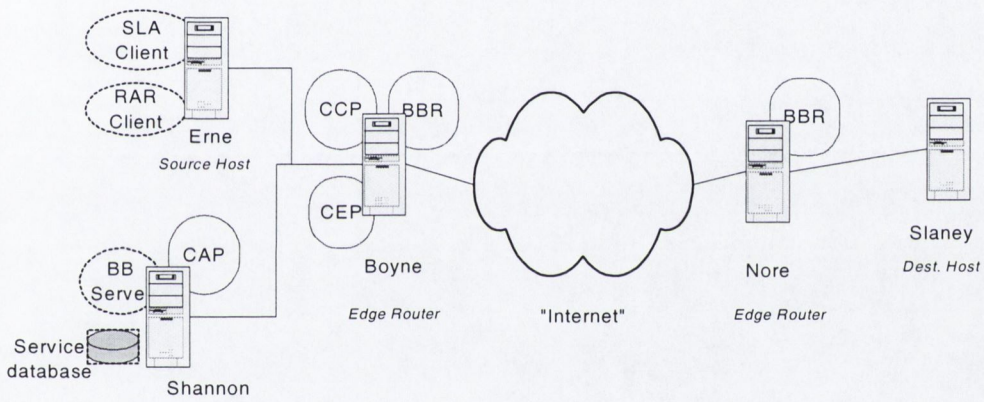


Figure 6-1 shows how the functional components of the case study were allocated to the test bed computers. The components were allocated in this manner to make best use of the computing resources. This setup was used for the majority of the experimental work though the configuration of charging elements was changed during the performance measurement part of the case study.

A simple test scenario based on a user transferring files between hosts (Erne and Slaney) across the Internet was used to examine the charging cases. The user chooses a Diffserv QoS to ensure that the files are transferred with a minimal time delay. The SLA was defined using the SLA client. A screen shot of the SLA client is shown in Appendix A. The SLA is applied for transport of traffic across the "Internet" between Boyne and Nore. The bandwidth broker database is updated with the customer and

SLA details. When the user wishes to transfer files from Erne to Slaney the following sequence of steps occurs:

1. The user requests that a flow/pipe be created with the Diffserv QoS - Expedited Forwarding (EF) 'per-hop-behaviour', QoS between Erne and Slaney. This is done using the RAR client on Erne. A screenshot of the RAR client is shown in Appendix A. The RAR client signals the bandwidth broker server, on Shannon, with the request. The bandwidth broker server checks SLA resource allocation permissions in the service database and assuming the resources are available, requests the charge coordinator (CCP) to check what type of charging is to occur for this resource request.

2. The CCP in turn requests the charge analysis function (CAP) to analyse the charging scheme policies for this service. The CAP determines the policy to apply and, assuming that the service is to be charged, installs a charging module on the edge router (CEP) to carry out the actual charging. This is effected by the APPLE **move**() primitive. The CAP then informs the CCP that charging can proceed.

3. The CCP activates the CEP charging module using the APPLE **start**() primitive. The CEP instantiates a meter on the edge router, Boyne, to measure time and volume according to the desired charging scheme. The CCP informs the bandwidth broker that charging analysis is finished and the session can proceed.

4. The bandwidth broker server then instructs the edge router client, (BBR), on Boyne to allocate resources on the edge router. It then informs the user RAR client on Erne.

5. The user then ftps the desired files from Erne to Slaney

6. The CEP thereafter periodically reads the meter and calculates and outputs the charge. This charge would normally be transferred to a payment system and/or user display. However for this series of experiments we displayed the charge in a console.

This sequence of events describes the transfer of file over an "uplink" from Erne to Slaney. The sequence is essentially similar to transfer files from Slaney to Erne with

some exceptions. The BBR on Nore is used to allocate resources since the 'pipe' is now directed from the other end.

6.2.3 Charging

PEACH charging for this scenario is implemented by the set of charging modules and rules shown in Appendix B, Part 1. Four charging possibilities were examined in the case study.

- Sender charged – uplink charging only
- Sender charged – uplink and down link
- Split charging – sender charged for uplink and receiver charged for downlink
- Receiver charged – uplink and downlink

The roles of "sender" and "receiver" were taken on by the Erne and Slaney hosts. It is normal when charging a certain party to instantiate the charging execution function (CEP) on the edge router nearest the user's host, as shown in Figure 6-1. Thus for receiver based charging the CEP would normally be instantiated on the receiver side edge router, Nore. However for the purposes of the experiments in this case study we have implemented all charging on the sender side edge router, Boyne.

The case study is based on the use of Diffserv Expedited Forwarding for transfer of files. A range of charging schemes can be selected for charging of file transfer. This is implemented by the following charging policy rule which is part of the ruleset of the CAP

```
Rule IpQ_getEFcsch{

inp(originForCharging);
switch(originForCharging)
{
   between(1,10) :
      chargingScheme =#flat_rate;

   between(11,20) :
      chargingScheme=#no_charge;

   between(21,100) :
      chargingScheme=#EF_time_volume;

    default :
      chargingScheme=#EF_volume;
}

outp(chargingScheme);

} //end rule
```

Figure 6-2 Rule to determine charging model to be applied

Each SLA is allocated an 'originForCharging' which determines what charging treatment will be accorded to the SLA. In the case study only one scheme was actually implemented. That is based on charging for time and volume, #EF_time_volume in the rule above. That charging scheme results in a charging module being installed and started in the CEP (on Boyne) by the CAP (on Shannon This is the IpQ_Eftandv charging module). This is in effect agent migration as a complete program and its associated state is transferred from one node to another and is reactivated on the second node.

6.2.3.1 Sender based charging-uplink only

The following extract from IpQ_tandv implements charging

```
timeoutRep->
{
  //action(IpQosMeter.readVol);
  inp(upLinkVolume);
  charge=charge+((upLinkVolume/volChargeAmount)*volTariff)
        +(timeChargePeriod*timeTariff);

  send(ccpUrl, application, message,charge);

}
```

Figure 6-3 IpQ_EFtandv charging function extract

The tariffs are determined by the CAP and transferred along with the module. This code shows the charge being periodically calculated for upLinkVolume and the resulting charge being transferred to the CCP, which runs on Boyne also. In the case study we displayed the charge total in a console window on Boyne. (Normally of course the charge would be posted to a database or prepaid system etc.)

6.2.3.2 Sender based charging - bidirectional

To extend the charging to bi-directional is relatively trivial. IpQ_tandv is modified as shown in the following code extract:

```
inp(timeChargePeriod,volChargeAmount, neId, timeTariff, timeFilter,
   volTariff, balance, meter);
...
....
inp(upLinkVolume, downLinkVolume);
charge=charge+((downLinkVolume/volChargeAmount)*volTariff)+
      ((downLinkVolume/volChargeAmount)*volTariff)+
      (timeChargePeriod*timeTariff));
```

It is quite easy to extend the charging scheme to have different tariffs apply in both up and down directions. In this case the rule was extended to have four tariffs *volTariffUpLink*, *volTariffDownLink*, *timeTariffUpLink*, *timeTariffDownLink* as opposed to the current *volTariff*, *timeTariff*. The charging function then looks

```
inp(timeChargePeriod,volChargeAmount, neId, timeTariffUpLink, timeTariffDownLink,
    timeFilter, volTariffUpLink, volTariffDownLink, balance, meter);
.....
......
inp(upLinkVolume, downLinkVolume);
    charge=charge+((downLinkVolume/volChargeAmount)*volTariffUpLink)+
              ((downLinkVolume/volChargeAmount)*volTariffDownLink)+
            (timeChargePeriod*timeTariffUpLink) + (imeChargePeriod*timeTariffDownLink));
```

6.2.3.3 Split Charging

It was somewhat more complicated to implement split charging. A number of modules and rules were changed. To enable split charging the sender and receiver part must both have the same service type. The identification of the receiver is provided by the sender to the bandwidth broker. The bandwidth broker can then retrieve the SLA for the server party using a combination of customer identity and service type.

The charging system calculated the charges due but delegated to the bandwidth broker to update the actual charges. The reason for this is to isolate the charging system from the payment model as a variety of payment models and protocols can be assumed to exist in the NGN.

The charging rules and modules presented previously were modified to deal with split charging and are shown in the following discussion. A new rule was introduced, IpQ_getChargedPartyandDirection, to determine the charged party and direction. This is shown below in Figure 6-4.

```
Rule IpQ_getChargedPartyandDirection{

   inp(originForCharging);
   switch(originForCharging)
   {

      between(31,78) :
         chargedParty="aParty";
         chargeDirection="up";

      between(79,100) :
         chargeParty="aParty";
         chargeDirection="both";

      between(101,120) :
         chargeParty="aParty";
         chargeDirection="down";

      between(121,130) :
         chargeParty="bParty";
         chargeDirection="down";

      between(121,130) :
         chargeParty="bothParty";
         chargeDirection="both";

       default :
         chargingScheme="none"
         chargeDirection="both";

   }

   pack(destUrl, chargedParty, chargedDirection);

} //end rule
```

Figure 6-4 Rule to determine charged party and direction

The other major change was to the CEP module IpQ_Eftandv. Part of the code of the modified module is shown in

```
application="IpAccounting";

// wait for next event
inputEvent {
  timeoutRep->
  {
    inp(upLinkVolume, downLinkVlume);
    upCharge=upcharge+((upLinkVolume/volChargeAmount)*volTariff)
          +(timeChargePeriod*timeTariff);

    downCharge=upcharge+((downLinkVolume/volChargeAmount)*volTariff)
          +(timeChargePeriod*timeTariff);
    if (chargedDirection=="up") then
     charge=upCharge;
    else {
       if (chargedDirection=="down") then
          charge=downCharge;
       else
         charge=upCharge+downCharge;
    }

    message="updateCharge";
    send(ccpUrl, application, message,charge, upCharge, downCharge, chargedParty);

  }
```

Figure 6-5 Module IpQ_EFtandv for split charging

Of course the CCP module also needed to be adapted to deal with separate charges for up and down links and the bandwidth broker modified to update the charge to the database and to the client part terminal if appropriate.

6.2.3.4 Reverse Charging

Reverse charging is simply a subset of split charging and was catered for by the IpQ_getChargedPartandDirection policy.

6.2.4 Distribution Scenarios

The charging architecture presented in the thesis is based on the three main nodes, the CAP, CCP and CEP. These are logical nodes and the architecture does not mandate how they are distributed. In this section we analysed a number of distribution and implementation scenarios in order to:

1. Get a feel for the basic performance of PEACH, and

2. Compare the relative performance of different distribution scenarios.

The lessons learned are applicable to all case studies. We have not looked at how PEACH system performs when handling many concurrent requests.

The distribution scenarios are described in Figure 6-6 below. The experiments were conducted on three machines, Boyne, Erne and Shannon.

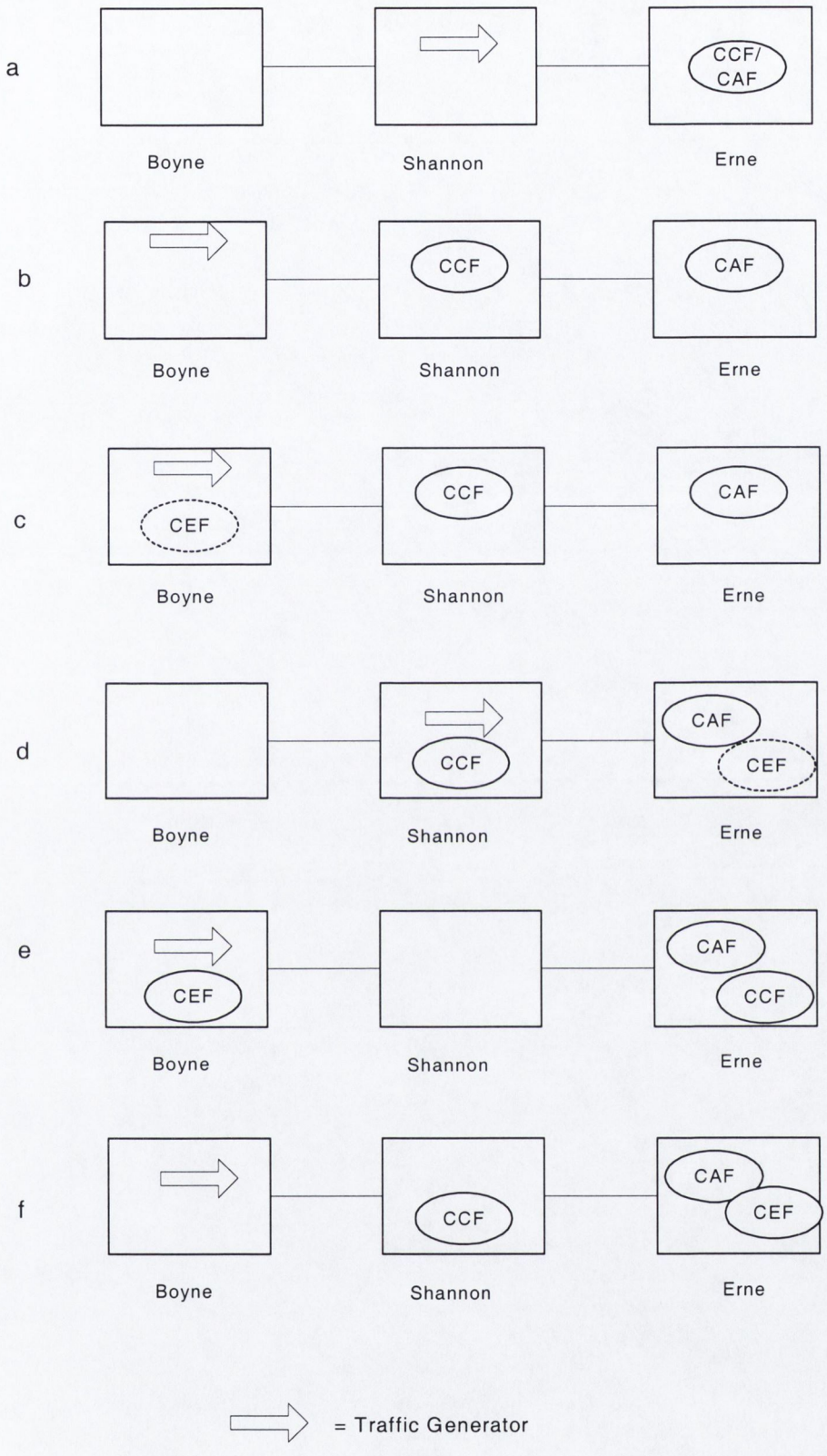


Figure 6-6 PEACH Distribution Scenarios

Erne is a 866 MHz Pentium III with 256 K Ram with Windows 2000

Boyne is a 600 Mhz AMD K6 with 128 K Ram with Linux 2.4.18

Shannon is a 1.4 GHz Celeron with 256 K Ram with Linux 2.4.18

Charging requests were generated from a Java test program, shown as an arrow in the diagram above. A number of configurations were tested in order to assess the performance of the system with different degrees of distribution. For all configurations shown above the basic scenario of is shown in Figure 6-7 , though it should be clear from the configuration descriptions that not all nodes are invoked on each occasion.

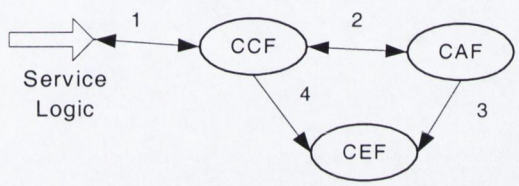


Figure 6-7 PEACH Node Interworking

The arrow on the left-hand side represents an arbitrary service session controller which wishes to perform charging analysis for a service session. The order of interaction between the nodes is indicted by the numbering. For each configuration the measurements taken represent the time for the combined charging nodes to respond to a request for charging from the service logic. In the diagram above this is the time between the request portion of step 1 and the reply portion of step 1. Normally this means that he charging system has completed analysis and set up metering i.e. charging for the service is ready to proceed.

The configurations were chosen to examine a number of issues such as

- Is there a performance cost incurred distributing all elements on different machines versus collocation of some elements on a single machine ?

- Is there any performance gain from merging logical nodes?

- What is the relationship between performance and degree of distribution ?
- What are the relative costs of migrating a module (installing and starting) to another machine ?

The following configurations were used:

(a) Traffic is generated on Shannon, merged CAF+CCF on Erne. There was no interaction with the CEF. This is the minimum configuration with just one charging node and two "message hops " in the message sequence.

(b) Traffic generation on Boyne. CCF running on Shannon, CAF running on Erne. There was no interaction with the CEF. This is the "traditional" configuration. There are two charging nodes and four message hops.

(c) This is similar to (b) but a CEF module is installed on Boyne by the CAF but not started. There are two three charging nodes involved and five message hops.

(d) Traffic generated on Shannon, CCF on Shannon, CAF on Erne, CEF module installed but not started. This is similar to (c) but a number of the nodes have been collocated to see if any performance improvement occurs. The number of nodes and message hops remains the same.

(e) Traffic generated on Boyne, CAF and CCF running on Erne, CEF module installed and started on Boyne. This is similar to (d) but the CEF is started.

(f) Traffic generated Boyne, CCF on Shannon, CAF on Erne, CEF module installed and started on Erne. This is similar to (e) but the CEF has been collocated with the CAF and the CCF distributed to a third machine.

The configurations above are chosen to incrementally expand the 'degree of distribution' and the scope of module migration. This allows for a comparative analysis of results.

The measured results are presented in the following table

Table 6-3 IPQoS Performance Measurements

Response Times (ms)	Configurations					
	a	b	c	d	e	f
Average	212	620	796	538	1001	1065
Min	165	497	563	287	634	762
Max	429	845	2079	1636	2370	2106
Number of test runs	30	30	30	30	30	30

Thirty tests were run for each configuration. This figure was deemed be sufficient to calculate an average delay time for each configuration. A lot of variation was noted between repetitions of the tests on each configuration cf. min and max above. As the test environment, e.g. processor load, was similar for all test repetitions it is difficult to explain the load variation with reference to the PEACH system. A similar spread of readings was noted while working with the Diego MP3 streamer in case study 3. We infer from this that the most likely cause of the variation of the readings relates to the Java Virtual Machine (JVM) and in particular the use of "Hotspot" compiler by the JVM. Hotspot compiler are intended to spot which areas of code need special treatment to boost performance. It takes some time to 'train' the compiler and so variations may be observed until the JVM has optimised code execution.

The main conclusions to be drawn from the above data are:

1. As might be expected the time taken to service a charging request increases with the number of charging logical nodes and the degree of distribution of these nodes. The rate of increase seems to be approximately linear.

2. The increase in response time depends on both logical and physical distribution i.e. on whether the nodes are distributed on different machines or collocated. In general collocation results in a lower cost even though RMI is used for all inter-node communication.

3. The time taken in a "normal" configuration (b or c) is up to three times as lengthy as the most optimised configuration (a). Since the charging logic is largely the same in all three scenarios we can infer that networking i.e. RMI, accounts for a large part of the overhead of charging message processing. In

configuration (a) three RMI messages are sent while in configuration (b) that number rises to six and to seven in configuration (c). The handling of charging logic in separate PEACH nodes also adds to the extra load. Notwithstanding such charging logic handling the processing of RMI messages constitutes a very significant part of the response time delay, and for highly distributed scenarios is the most significant delay cost.

4. The time taken to install and start a module on a remote node (mobile agent scenario) is comparatively expensive. The incremental cost of installing the module can be seen in configurations (c) and (d). The response times increases by approximately 150 ms between (b) and (c) due to handling the installation of the module on Boyne. This figure implies that the cost of installing a module is of the same order as dealing with an 'ordinary' analysis request. In (d) the response time is significantly lower than (c) even though both carry out the same functions. The obvious conclusion is that the improvement is due to the collocation of the nodes.

5. This cost of starting up the module once it is installed is expensive. (e) extends (d) by the action of starting up the installed module. The performance cost almost doubles. There are two factors however which have to be taken into account however when considering these figures. Firstly the nodes are distributed on different machines. In (e) the CEF and traffic generator are running on Boyne which is a much less powerful machine than Shannon which is used in (d). Also the CAF and CCF are collocated in (e) while the CAF and CEF are collocated in (d). As the install/start messages are between CAF and CEF then a collocated CAF/CEF will give better response than a collocated CAF/CCF for this scenario. This is borne out in (f) where CAF and CEF are collocated. The response time decreases compared to (e) even though the degree of physical distribution increases. A better estimate of the cost of starting a charging module can be obtained by comparing (f) and (d). the response time increases from 787 ms to 1012 ms, an increase of about 25%. The key points here seems to be

    a. Module migration (mobile agent) is expensive compared to 'normal' request processing

b. That the best performance is achieved for the module migration scenario when the CEF is collocated with either the CAF or CCF.

It is impossible, from the above data, to draw conclusions about the ultimate scalability of PEACH in its current form. The test-bed used consisted of relatively low specification machines and PEACH implementation is not at all optimised. We believe that with attention to the implementation these response times could be greatly improved.

PEACH has been designed as a multi-threaded system. The platform uses a thread pool that is configurable at run time. However we have not attempted to measure performance of processing multiple parallel requests though we have observed such operation. The reason we have refrained from presenting such measurements is that the current PEACH system has not been implemented with performance considerations in mind and. This consequently makes it difficult to draw any meaningful conclusions about the overall performance of the platform.

We have observed that the average response time for configuration (a) is considerably less than the Diego MP3 Web server used in Case study 3 which, we believe, empirically places handling a PEACH request in the same category as handling a web request.

6.2.5 Analysis and Results

We outlined at the beginning of this chapter, in

, a number of attributes of the PEACH system which we believe are demonstrated by this case study. A number of these attributes are demonstrated for the PEACH system by the combination of case studies and we defer discussion on these attributes until the end of the chapter. Three features from Table 6-1 are explored only in this case study and we discuss these here.

6.2.5.1 Charge Allocation

Charges may be allocated to either party in a communication session or the charges may be split between the parties in some proportion. Any charging system must be able to flexibly allocate charges to any party. We have shown in the course of this

case study that PEACH easily deals with this issue. Charges can be allocated for uplink and downlink separately or together and different tariffs can be applied to both links simultaneously. APPLE policies are easy to create, change and deploy and result in the rapid creation of charge allocation scenarios. This enables service providers to better tune their service offerings and be more competitive in the marketplace.

6.2.5.2 Computing Performance

The architectural approach in PEACH allows for a variety of distribution scenarios. A number of these were examined in the course of the case study. The response time of the charging system to a request depends on the number of distinct logical nodes and the manner in which they have been distributed to physical nodes. In general the response time increases with the number of logical nodes and increases further the more the logical nodes are distributed. Collocating nodes can reduce response time significantly. The rate of increase is approximately linear. The response time does seem to increase in a non-linear fashion for module migration ("mobile agent ") scenarios.

We have not analysed real-world performance of PEACH under heavy loads. The system is not designed to be performance optimised. Also the test machines in use are relatively low performance. We are encouraged by the results achieved, which in the optimum case are less than the cost of servicing a WWW request on the same test machines.

6.2.5.3 Mobility Support

We have demonstrated that a charging module, along with its state, may be migrated from node to node. This is done in the case study for the CEF module in a number of instances. In scenarios where mobility of the user triggers a need to migrate the charging context then this can be done by the APPLE module migration mechanisms. Migrating modules may give flexibility but it can be seen from the measurements in this case study that migration can be costly and service providers will need to be aware of the cost benefit trade-off before implementing module mobility.

6.3 SIP VoIP services

6.3.1 Motivation

This case study considers how charging for SIP Voice over IP telephony, or related services, which use Diffserv QoS for media delivery could be implemented using PEACH. This is an example of a bundled service where a service in one layer uses the uses the service of a lower layer to deliver a combined end to end service. The case study has been chosen to complement the other case studies so that the combination of all demonstrates how PEACH meets the NGN requirements.

This is a paper-based case study and has not been implemented. A possible implementation is shown in the next section.

6.3.2 Scenario

This case study relies on the ability of SIP network elements to reserve QoS resources and to interact with the charging system as depicted in Figure 6-8

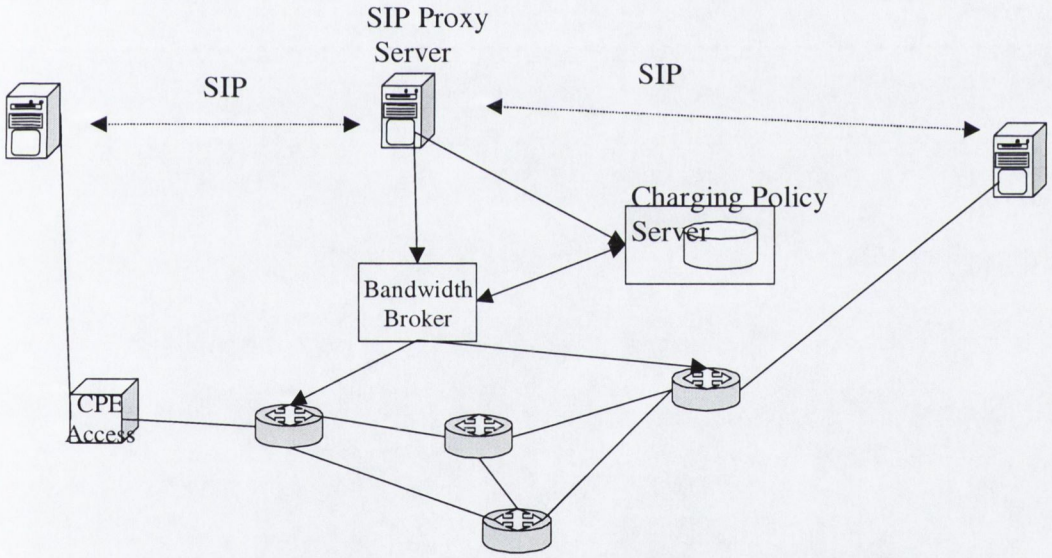


Figure 6-8 VoIP Charging

A high level view of the sequence of events for QIP charging is as follows

1. A SIP client requests a session to be set up by signalling a SIP proxy server in the service provider network

2. The proxy server instantiates a SIP charge session coordination agent which in turn invokes charging analysis in a SIP charge analysis point. If the analysis determines that charging for media transmission is to be carried out the SIP CCP informs the SIP proxy server and returns an originForCharging parameter to be used in the lower layer charging.

3. The SIP proxy server then initiates media reservation by contacting the Diffserv bandwidth broker. It passes details of the media end points, the type of QoS required, amount of bandwidth etc.

4. The bandwidth broker instantiates a Diffserv CCP to invoke charging analysis for the Diffserv connection. Subsequent processing depends on the charging outcome and will be as described in the previous case study.

5. When charging has been set up and resources allocated the bandwidth broker informs the SIP proxy server which then authorises the bandwidth broker and the SIP user client to begin the session.

6. Charges accumulated by the media session are passed to the bandwidth broker which, in turn, passes them to the proxy server. These charges are combined with any charges for the SIP session and, depending on the payment method, are posted to the users account or are displayed on the user terminal.

The standard version of SIP does not have support for resource reservation or charging analysis handling. Salsano and Veltri [Sals02] describe an extension of SIP called QSIP which has been modified to handle resource reservation for Diffserv using COPS-DRA,(COPS for Diffserv Resource Allocation) ,[Sals02]. An overview of their QSIP/Diffserv proposal is shown in Figure 6-9

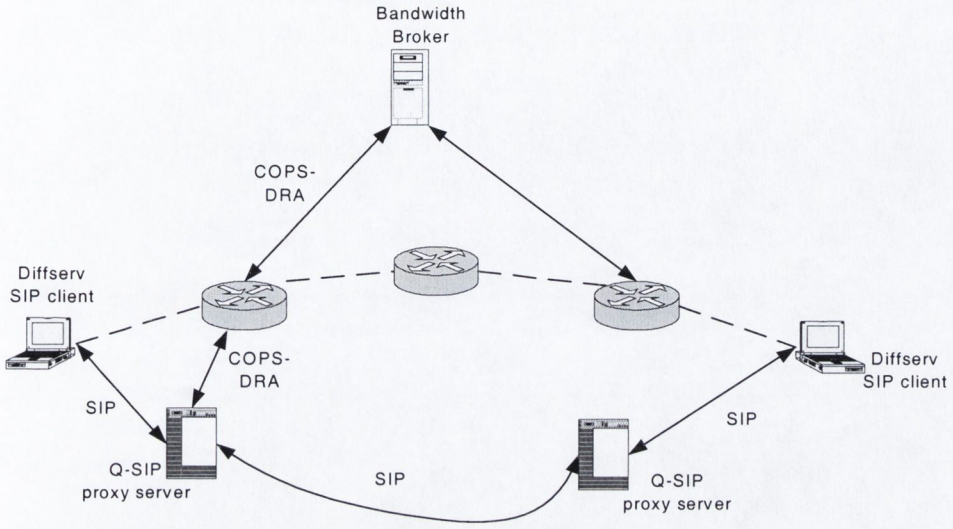


Figure 6-9 Q-SIP /Diffserv overview

Figure 6-10 shows how this architecture would be modified to cater for QSIP charging with PEACH. COPS-DRA is replaced by BBTP. The QSIP BBTP client is implemented in Java while the remaining BBTP components are implemented in C.

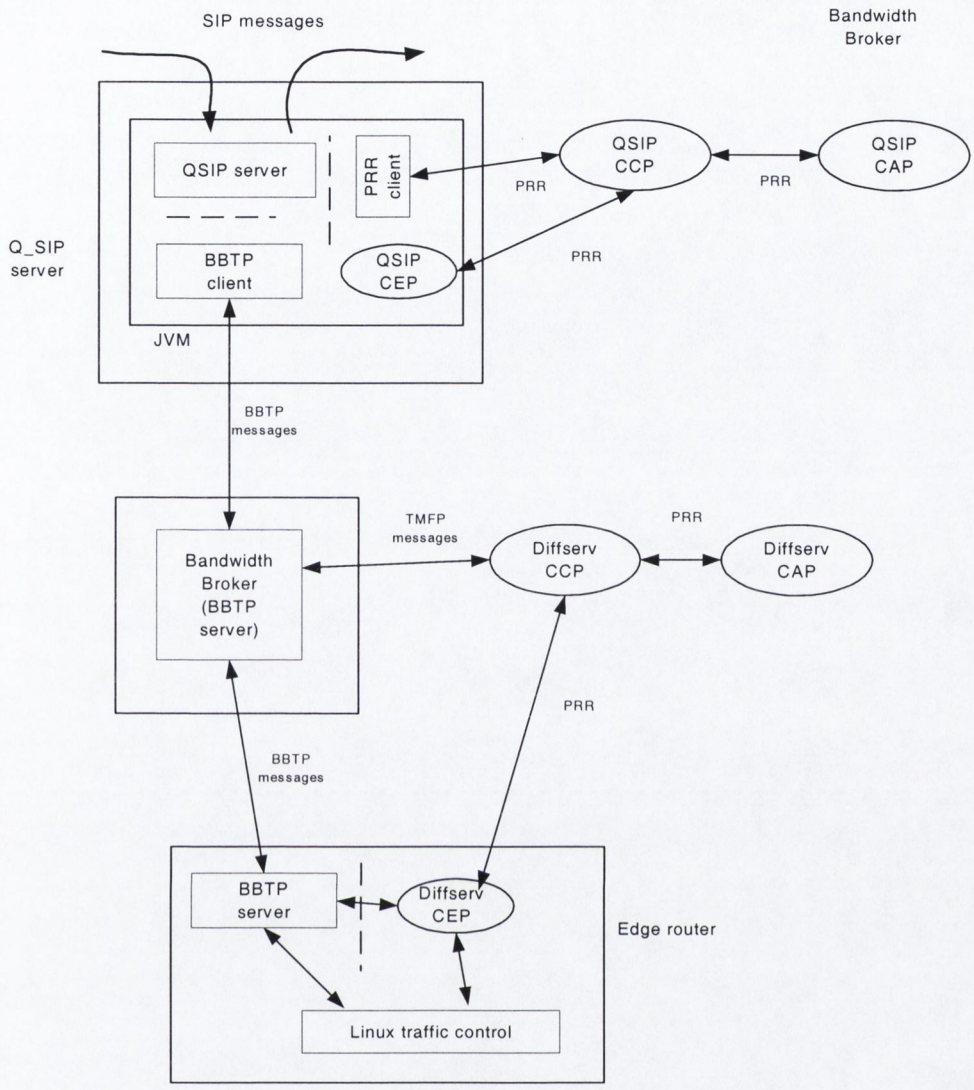


Figure 6-10 Q-SIP charging architecture

The QSIP BBTP client converses with the bandwidth broker server to reserve flow resources. The type of information transferred includes

- the *scope* and *amount* of the reservation i.e. where the reservation applies and how much bandwidth
- the *type* of requested service i.e. possibly including a set of QoS parameters
- the *flow* identification

The reserved flows are used to carry RTP media sessions, though the network layer is unaware of this. The bandwidth broker server uses this information to reserve resources in the edge router (cf. case study 1).

The QSIP server is extended to have a PRR client. PRR (PEACH Request-Reply) is the internal protocol used in PEACH. BBTP is extended to transfer charge details to the QSIP server. This server is extended to update charging information in the customer database or to transfer charge information to the user terminal. This latter extension would require a new SIP message, UPDATE_CHARGE if implemented. If charging is required for the SIP service itself a QSIP_CEP is installed on the Q-SIP server JVM.

6.3.3 Charging

Charging takes place at two layers here, the session layer for the QSIP service and the network layer for the QoS resources. There are two sets of charging nodes instantiated- cf. Figure 6-10. There is no direct interaction between these two layers, following the PEACH principle of "loose coupling". The relationship between the charging at each layer is implemented by the "originForCharging" parameter. This allows very flexible charging scenarios to be implemented.

To see how charging might be applied we will consider an example (take from RFC2543) where caller Alice wishes to contact callee Bob. Assume that Alice has included the following description in her INVITE request. It includes an audio stream and two bi-directional video streams, using H.261 (payload type 31) and MPEG (payload type 32). Bob responds indicating he can support the requested streams.

Salsano does not specify how the RTP encoding e.g. MPEG is mapped to a particular QoS but we can assume that the BB has some internal mapping table. In practise for a BBTP based implementation we would perform this mapping in the QSIP server and map the information to existing BBTP parameters that are expressed more directly in Diffserv terminology.

Different charging schemes can apply for different SIP based services e.g. videotelephony, cf. Figure 6-11 which contains an excerpt from the Sip_getChsh policy rule of Appendix B.

```
inp(serviceType);

 switch (serviceType)
{
   eq (1) :  // POTS
     audioCharging = 21 ;  // Diffserv originForCharging - no
charge
     pack(audioCharging);

   eq (2) :  // enhanced POTS
     audioCharging = 5      // flat rate
     pack(audioCharging);

   eq (3) :  // video telephony
     audioCharging = 80;    // time metered
     videoCharging = 180;   // volume metered
     pack (audioCharging, videoCharging);

   eq (9) : // arbitrary streams
     audioCharging = 80;    // time
     videoCharging = 120 ;  // time and volume
     dataCharging  = 120;   // time and volume
     pack (audioCharging, videoCharging, dataCharging);
```

Figure 6-11 Charging schemes for SIP based QoS

In this case we assume that Alice is transferring a number of arbitrary media streams rather than subscribed to a formal SIP service. This corresponds to a "serviceType=9" in the above rule. This particular charging scheme determines that audio services be charged by time while video and data channels be time and volume charged. These values are used as the "originForCharging" in the subsequent analysis in the Diffserv charging. The Diffserv policies are not shown. The QSIP server will communicate the necessary specification and charging information for each flow to the bandwidth broker server.

For the whole session the following charging policies are taken to apply:

- The SIP session is charged for on a time duration basis
- The video sessions are charged on a volume basis while the audio session is charged for on a volume.

- Each sender pays for his own media and does not pay for receiving.

Further we consider only charging of Alice. As for the previous case study Alice is assumed to have an SLA for her particular SIP service,.

6.3.4 Analysis and Results

Discussion is deferred to end of chapter.

6.4 Charging for MP3 Streaming

6.4.1 Motivation

This case study involves charging for download of MP3 files to a Web client from an MP3 server. The case study illustrates how PEACH can be used for charging for content services. It also illustrates how service providers charging systems may co-operate to facilitate bundled service offerings i.e. as identified in Figure 6-2 and shown below

Attribute	MP3 Streaming
Marketplace Support	
Service Provider Interworking	✔

The case study has two scenarios:

1. The MP3 service has no interaction with network charging
2. Network charges are bundled with the MP3 charges. This scenario is similar to the VoIP case study i.e. interaction between service layers

6.4.2 Scenario 1

The basic network scenario is illustrated in Figure 6-12.

This scenario was implemented on the Erne, Boyne and Shannon PC's. The MP3 server is implemented on Shannon, the charging server on Boyne and the Mp3Client on Erne. The MP3 server is accessed across the Internet using HTTP. The user is presented with a list of MP3 songs. He selects individual songs to stream or can create playlists and stream the entire playlist. This case study was implemented using an open source Java MP3 streamer called Diego, [Dieg02]. Diego is a stateless, "thread per request" server which creates a new thread for each streamed file. Diego was extended to retain state for each client session. It was also extended to interface to PEACH. The PEACH CCF and CEP functions have been combined in this example.

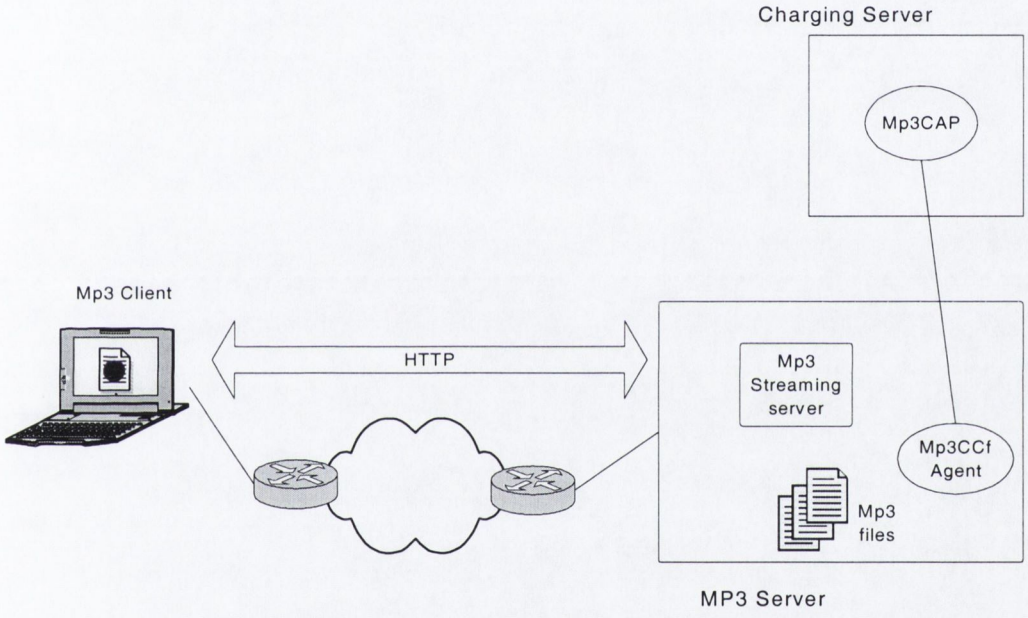


Figure 6-12 MP3 Streaming Testbed

The sequence of events is

1. A user accesses the MP3 service website.

2. The user authenticates himself to the service provider. A number of possibilities exist – the user may have a prepaid subscription or may be a trusted user based on a previous relationship. Alternatively the user may have pre-authorised payment for this session to a specific limit (e.g. by using a credit card) or the user may be

authenticated by the network and the cost of the MP3 session deducted from his prepaid network service balance. For the purpose of this implementation we assume one of the above mechanisms has been used prior to charging being invoked.

3. When a track or playlist is chosen the MP3 server generates am M3U file(i.e. playlist) to send to the client browser. At this stage the server invokes the charging server to analyse the cost based on the charging model described below.

4. When the browser receives the playlist it then requests each track in turn. The MP3 server then requests the charge for the track from the charging server.

5. The charge is computed and returned to the MP3 server which forwards it to the browser where the session price is displayed via an applet.

6.4.3 Charging for Scenario 1

In the charging model applied in this scenario each track is assigned a particular tariff. Charges are computed based on this tariff but are weighted by the size of the file and the number of tracks selected as part of a playlist selection. A discount is applied to each track cost based on the number of number of tracks selected. This charging scheme is implemented in the charging policies Mp3InitAnalysis and Mp3GetFilePrice. Excerpts from these charging rules are shown below. The complete set of modules and rules is contained in Appendix B – Part 3.

```
switch (numberOfTracks)
{
   in (2,3,4) :
        discountRate = 0.9;

   between (5,10) :
        discountRate = 0.7;

   gt (10) :
        discountRate=0.5;

   default :
       discountRate=1.0;
 }


switch (mp3chargeCategory)
{
        eq (1) :
                trackCharge = 8;
        eq (2) :
                trackCharge = 10;
        eq (3) :
                trackCharge = 13;
        default :
                trackCharge = 0;
}

if (fileSize > 6) then
        sizeRate = 1.1;
else
 if (fileSize > 12) then
        sizeRate = 1.25;
 else
        sizeRate = 1;
 trackCharge= trackCharge*sizeRate;
 outp( trackCharge);
```

A screen shot of the client browser is shown below in Figure 6-13.

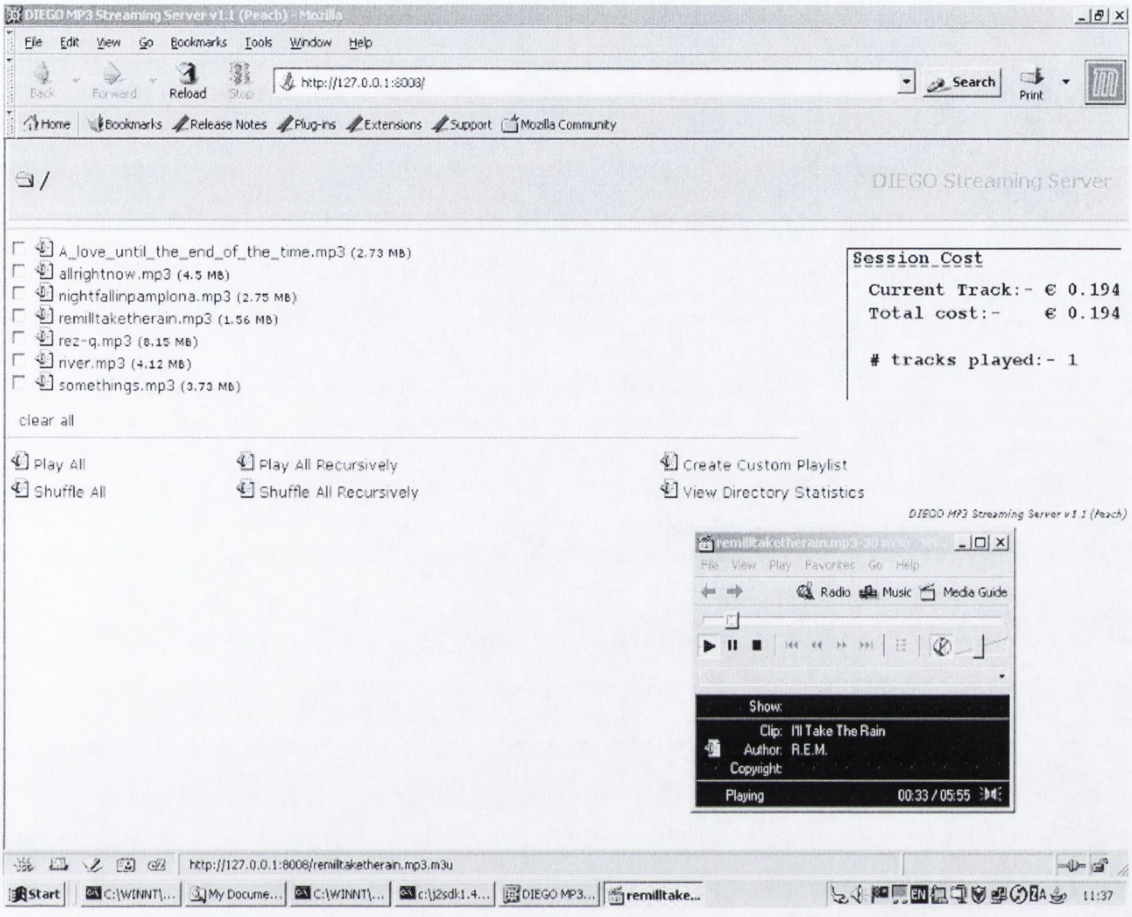


Figure 6-13 MP3 client showing session cost

6.4.4 Charging for Scenario 2

In this scenario the MP3 files are streamed via a high QoS flow/channel. The user has chosen this option from the MP3 provider web site. The scenario is shown in Figure 6-14. The MP3 server negotiates a QoS reservation with the BB for the downlink only. The BB invokes the network CAF/CCF to set up the charging in the appropriate edge router. QoS charging is handled as outlined in case study 1. The network usage charge and the MP3 charge are calculated independently. Both charges are routed to the user browser via the Mp3CcfAgent which forwards the information to the display applet.

An enhanced version of this scenario would allow for payment of the MP3 tracks via the network service providers prepayment system (This version was not implemented). In this case the MP3 CCF/CAF would receive all calculated charges and would compare this to the user's prepayment limit. The MP3 CCF would then issue a warning to the user when the limit was reached and would take what ever actions were decreed should the limit be exceeded.

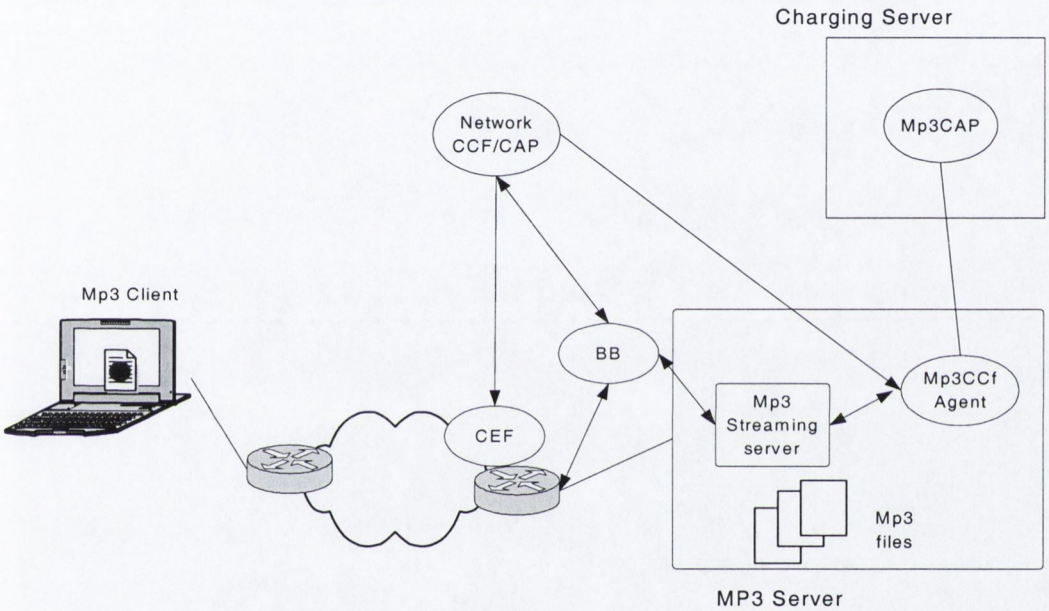


Figure 6-14 Network charging included for MP3 session

Although the focus of PEACH is on charging scheme determination and charge calculation the system can easily be extended to handle related issue such as payment limits being exceeded.

6.4.5 Analysis and Results

As with the other case studies discussion of system global features is deferred until the end of the chapter. The case study is used however to demonstrate a particular feature of PEACH i.e. service provider interworking.

Scenario 2 here extends the points about charging for inter layer, bundled, services which were introduced in the previous case study. In the previous case the service were assumed to be provided by a single service provider. Here there are two different providers. The interaction is much the same i.e. loosely coupled charging, but there is one difference. The charge calculations are routed via the MP3 charge coordination function as the MP3 service provider has chosen to display the charge in real time. In general the interaction between PEACH and payment systems is fluid and can be done either by PEACH or otherwise.

We have indicated how PEACH can be used to extend this service scenario even further in the case of use of the network provider's prepayment system. Another extension to this would be to have the charge calculation for the MP3 carried out in the network operator's CEP. This could be done by downloading and executing an MP3 charging function module to the CEP in the network. There are some issues that would need to be solved e.g. security etc. The PEACH system is however flexible enough to handle this scenario with no extension.

6.5 Overall Analysis

At the beginning of this chapter we indicated a set of requirements that an NGN charging system must adhere to. In this section we point out the aspects of the case studies which we believe demonstrates PEACH's abilities. We have already discussed a number of these features/platform characteristics where these are demonstrated in the context of individual case studies. In this section we discuss the remaining, system global, features. These are shown in Table 6-4 below.

Table 6-4 Global Feature Set

Attribute	Case Study		
	Diffserv QoS	SIP VoIP	MP3 Streaming
Marketplace Support			
Pricing model diversity	✔	✔	✔
Service diversity	✔	✔	✔
Service bundling		✔	✔
Real time charging	✔	✔	✔
Platform Attributes			
Platform extensibility	✔	✔	✔
Architectural flexibility	✔	✔	✔

6.5.1.1 Pricing Model Diversity

Pricing models are represented in PEACH by charging schemes. We have seen that a variety of charging schemes is applied in the case studies. In the case of the connectivity case studies i.e. Diffserv and QoS, charging schemes have been based parameters such as time and volume. For example Diffserv EF can be charged on the time of the session (unlikely to be a very common form of charging for this service) or the volume of data transmitted or some combination of both. Furthermore the tariffs to be applied can easily be changed. For the application layer service, (MP3 streaming), pricing has been based on the number of songs which are streamed. It is also easy to implement variations on the charging schemes as e.g. in the Diffserv case study different tariffs are applied to uplink and downlink channels while for the MP3 streaming discounts can be applied based on the number of tracks.

In PEACH the selection of a particular policy or parameter value is governed by parameters input from the service session environment. There are many examples in the case studies modules and rules. The 'originForCharging' parameter in particular has been widely used. It's purpose has been to relate external policies to particular charging policies. In essence it provides the bridge between the pricing model and the charging scheme. In the QoS case study it associates the charging scheme to the customer SLA. For the bundled services case studies it has been used to show how the pricing policy of the network layer can be driven by the pricing policy for the upper layer service and can be easily changed from service to service. For example in MP3 the 'qosService' chosen can be either silver or bronze depending on the originForCharging. This will in turn determine the charging scheme for the network layer service. By varying the origin for charging one can easily associate a different charging scheme, and hence pricing model, with a service. The case studies thus establish that PEACH allows very flexible management of services and pricing models. Furthermore in a real charging system we can expect that the charging scheme determination could depend on a number of such input parameters. As PEACH is entirely programmable it can easily be extended to deal with these systems, though the rule sets would become more complex.

Taken as a whole the set of case studies demonstrates that PEACH allows great flexibility to define diverse pricing models for services.

6.5.1.2 Service Diversity

PEACH is predicated on a couple of fundamental assumptions about service usage. These are

1. Service usage is encapsulated by service sessions. Services are dynamic and within a session different service related events may occur.

2. Each service has its own specific *vocabulary* i.e. set of data and events.

Service charging is related to service usage and so PEACH provides mechanisms that allow services sessions to be charged for. These include the module mechanism that captures the session abstraction and the *context* mechanism that defines a set of service specific charging parameters and events. These events and data can then be

combined in service specific rules and modules to implement the required pricing models. This enables PEACH to be adapted to any service environment.

The use of these mechanisms is evident in all of the case studies. The case studies demonstrate that PEACH can be used to charge for services at every layer of the NGN. Three different contexts are defined in the course of the case studies in this chapter IpAccounting, SipServices and Mp3Streaming. Each has its own parameters and events. Service specific event examples include "SipAnalysisResult" (SIP), nextTrack & noMoreTracks (MP3), chargeResult (IpAccounting). Context parameters include numberOfTracks (MP3) , dscode, burstSize (IpAccounting), and audioCharging and videoCharging (SIP).

The case studies are quite diverse and demonstrate that PEACH can be applied to services with different characteristics and different pricing models. They demonstrate that PEACH works equally well for counting time and bytes as well as music tracks. The model could be extended to charge for any service specific item e.g. transactions, downloads, bullets etc. Introducing new services is simply a matter of creating the appropriate context and writing the appropriate policies.

6.5.1.3 Service Bundling

Charging for bundled services can be achieved in PEACH through a combination of flexible association between policy sets and the flexibility of charging node allocation and distribution.

Flexible association between policy sets has been alluded to in the discussion on Pricing Model Diversity. Upper layer charging analysis indicates which lower layer services are to be charged for and associates a particular "originForCharging " to be applied to the lower layer. This results in the appropriate charging policy being applied at the lower layer and the correct charges being computed and returned to the charge accumulation point. The case studies show that PEACH is quite easily used to charge for composite services. This approach is demonstrated by the VoIP and MP3 case studies. In both cases the (lower layer) QoS service is charged separately and the price passed to the session accumulation point. In the VoIP case the tariff to be applied at the lower layer depends on the types of upper layer service cf. Figure 6-11.

So, for example, the QoS channel required to carry audio is not charged for a POTS call but *is* charged for videotelephony.

Depending on the model of co-operation between the service providers different charging and payment scenarios may apply. This is illustrated in both the VoIP and the MP3 study. In the former the charging is distributed in a node on each layer and the composite charge is presented though the SIP server, thereby isolating the payment mechanism from the charging. In the MP3 case two charging scenarios are applied. In the first case no network charging applies and the charge is calculated in the application layer node. In the second scenario network charging is applied and two the charging is similar to the SIP case i.e. carried out in separate layer nodes. There are a couple of differences between the case studies. In the MP3 study we calculate the charge for presentation to the user and we also show how the network providers real-time payment system could be used to pay for the MP3 in real time. We also show that all the charging could be carried out in the network layer if the MP3 charging algorithm is migrated to the network charging node.

The case studies demonstrate that PEACH can enable charging for composite services as easily as it can for unbundled services and can allow for flexible and scalable inter-provider run time solutions.

6.5.1.4 Real Time Charging

The need for real time processing of charging data is probably the single most important driver in NGN charging. PEACH has been designed explicitly to facilitate real time charging of service sessions. PEACH extends current approaches by pushing charge calculation into the network layer and by providing features to allow for real time charging of complex multi-layer composite services.

The primary PEACH feature which supports real time charging is the run time execution of the charge calculation, or rating, function in network nodes. This is implemented in PEACH by casting the charging function as a PEACH module that allows the charging function to be executed in parallel with the service usage. The case studies provide many examples of real time charge execution. They demonstrate the approach is feasible for different service types and at all service layers of the NGN

network. They also show that such a distributed approach is feasible for charging of composite services. The Diffserv case study also shows that charging functions may also be migrated to different nodes, thereby acting as mobile agents and as such providing a basis for complex mobility charging scenarios.

The charging functions easily implement varying charging schemes e.g. time and volume charging for connectivity services and song tracks for MP3 streaming. Charging for other applications services such as gaming is also seen to be easy to implement. The charging functions can also be extended to be more general purpose and deal with different charge allocation strategies (Diffserv QoS case study) or to enable real time payment by interaction with a payment system (MP3 streaming).

6.5.1.5 Platform Extensibility

The PEACH platform is extensible for new services and for new network technologies and protocols. The platform is designed as an *application framework*. An application provides a generalised set of functionality for a family of related applications. The framework provides hooks which allow particular applications to be embedded in the framework to provide total solution.

In PEACH extensibility is provided primarily through support of the context mechanism which, as we have already, seen allows service specific events and charging parameter vocabularies to be added. PEACH makes heavy use of the Java language *reflection* mechanism to implement context support. This allows new object classes to be referenced and executed at run time. Reflection greatly eases the implementation of context's but the context feature is not dependent on reflection and can be implemented in other languages also. Support for the context mechanism is also given by the automatic generation of the Java code to implement the charge parameters.

All of the case studies provide first hand evidence of the flexibility and adaptability of the context mechanism.

PEACH also provides support for interaction with different network policy protocols. The default *PRR* protocol is used in all three case studies to send/receive data and in

the Diffserv QoS case study the *Tmfi* is used for interaction between the bandwidth broker server and the CCF charging agent.

6.5.1.6 Architectural Flexibility

PEACH exhibits several examples of architectural flexibility.

The basic PEACH architecture of charge coordination, charge analysis and charge execution is shown, from the case studies, to be general enough to deal with a range of NGN services. The same entities are used in all case studies and we believe can be used to meet the needs of any NGN service.

PEACH also supports the policy based layered charging framework proposed by Hartano, [Hart99]. In particular PEACH implements the charging layer of this model. By adhering to this model and separating the charging and metering functions a PEACH charging function can be used to charge for a connectivity service independently of where and how the data is produced. Thus the Diffserv charging function can be used on a Linux device or on any other routing device.

The use of charging parameters such as originForCharging means that association between charging policies and other types of policy can easily be enabled and allows diverse pricing models to be easily and flexibly associated with a service and also facilitates decoupled charging between network layers. This allows for a more scalable approach to charging for services in general and for composite services in particular. This is evidenced in all of the case studies presented.

The active network approach underlying PEACH means that the charging logic can be distributed as needs be in the network. This allows a service provider to match his computation needs to the processor power available and thus greatly facilitates the scalability of the charging solution. PEACH logical nodes can be flexibly distributed on different machines (cf. Diffserv QoS, MP3) or can be collocated on the same machine (Diffserv QoS). Logical node functions can also be combined into a composite node for even greater efficiency (Diffserv QoS). Each distribution scenario has different costs associated as we have seen in the Diffserv case study and these will need to be taken into account by the service provider when dimensioning a service scenario.

Chapter 7

7 Conclusions and Future Work

7.1 Introduction

The communications network continues to grow. New services, technologies and business models are contributing to the creation of a diverse, complex and service rich next-generation network. This network will be characterised by a dynamic and competitive marketplace with a variety of service provider types and where "co-opetition" will be common. The NGN will in turn place new demands on network support systems. New architectures and technologies will be employed to create business support solutions. In the case of charging systems the variety of services to be managed and the volume of transactions to be handled will both increase tremendously. Next generation charging systems will need to be flexible and scalable to handle the demands of the NGN. This thesis investigates the use of programmable networking techniques to provide a solution for a next generation charging system.

7.1 Summary of Contributions

PEACH enables scalable, network embedded, real time charging

Moving the rating function to the network elements and combining this with a real-time payment function removes the need for CDR generation, transmission and storage. This has been demonstrated for a variety of services and marketplace scenarios. There is still network traffic between the PEACH nodes but this is greatly reduced compared to a more conventional scenario. The active network architectural approach means that PEACH nodes can be deployed where and when needed. This is a key feature in achieving scalability of the charging system in the network. Furthermore, logical nodes can be merged (e.g. CAF and CCF) if need be, to achieve greater efficiency. As with any network the actual deployment of PEACH nodes for a

particular application will have to balance a number of demands e.g. trade off between complexity and efficiency. The combination of network embedded rating and scalable node deployment clearly demonstrates the feasibility of the approach proposed in this dissertation.

The *performance* of the charging system for the evaluated case studies demonstrates that a scalable solution based on a programmable network approach is technically feasible. With some caution we conclude empirically that it costs about the same to process a request to the charging system server as it does to process a relatively simple request to a web server. The current PEACH implementation is somewhat naive and is not designed for performance and we believe that the performance of the system can be increased by an order of magnitude.

PEACH supports the rapid introduction of new services by providing the flexibility to easily create and deploy new charging schemes for real time charging.

The system exhibits flexibility at many levels. In the business context a wide variety of pricing models will apply. The system can be adapted to handle a wide variety of rating approaches and rule-sets can be quickly created for new services. APPLE has been specifically designed for NGN charging and the language facilitates data manipulation and sharing. It also has specific constructs for computing and communication and service specific control, execution and analysis models are easy to create. PEACH does not place limits on the complexity of the charging schemes created and deployed.

The system is also easy to adapt to a variety of payment models e.g. a-party charging, reverse charging, split charging and so on. The modular approach taken in the present work to limit the charging functionality allows the charging system to interwork with a variety of payment systems without impacting on the charging system itself.

PEACH facilitates change in the network by easily adapting to new service families and new network technologies and protocols.

The context mechanism allows the PEACH framework to be 'programmed' with new charging policy parameters for new service families. The PEACH SDK enables these policy parameters to be compiled into (Java) support objects to transfer and receive

data. When these support object are combined with the service composition parameters of APPLE new policies and charging modules can be very quickly created and deployed.

The platform is based on an object-oriented framework design approach. The framework approach combined with the use of Java language reflection mechanism clearly facilitates the addition of new services via the context mechanism. Although the reflection mechanism does give first hand support to dynamic service creation similar mechanisms can be adopted in other languages.

The architecture of the platform also allows new protocols to be added without impacting the charging logic and policy.

The suggested charging reference model based on the charge analysis (CAF), charge coordination (CCF) and charge execution functions (CEF) is shown to be applicable to a number of diverse service scenarios. It deals well for charging of services of different complexities.

PEACH supports bundled service business models

The NGN network will be a layered network and bundled service offerings will be common, whether offered by one or more providers. "Unbundled" charging will however be needed in order to manage charging systems efficiently. PEACH provides direct support for this by allowing the charging systems on each layer to evolve independently. This is achieved by means of the "origin for charging" concept. The active node approach means that charging nodes in each system can coordinate the charging in each layer in order to effect overall charging for the bundled service. We have seen this demonstrated in two case studies.

7.2 Further Work

The current PEACH system is incomplete in many ways and there are some areas where further study is needed to achieve an optimal system. These include

1.1.1 Policy Management and Deployment

The deployment and management of policies is key to the success of any policy

system. The issue is only partly addressed in PEACH and a significant amount of work remains to be done to devise a suitable management framework. A simple editor has been created as the basis of a policy software development kit (SDK). However in its present form it is not suitable for policy creation. Also the PEACH runtime engine has a defined RMI management interface to enable addition/removal of policies. However this has not been implemented. Incorporation of PEACH into a broader framework such as DEN-ng is an interesting challenge.

1.1.2 Policy Conflict Resolution

A policy management solution will normally consist of a number of policy rules. Individual rules may conflict or contradict each other and may consequently cause problems and errors in the deployed policy solution. Resolving potential conflicts is a major headache for all policy languages and a variety of approaches are taken. Policy rule conflict has not been addressed in PEACH and further studies are needed to devise a suitable conflict resolution system.

1.1.3 Deployment to handsets

Executing charging policy rules on mobile handsets is a compelling goal for the current research. Viewed from a policy execution perspective, handsets represent the ultimate distributed computing network.

Several problems remain to be solved. Not all services may be amenable to processing on handsets. The exact scenario and business need remains to be defined. Also the current implementation of APPLE is not amenable to processing on current technology handsets, not least as handsets Java virtual machines do not implement reflection. Nor is running a scripted language very feasible.

A simplified APPLE language with a much more efficient interpreter coupled with execution of the heavy duty primitives (action, subscribe, etc) on network based back-end server seems to be a promising approach to take but remains to be investigated.

1.1.4 Integration with Other Policy Work

PEACH can be viewed as a component in a larger policy based charging and billing solution and we have already referred to this fact earlier in the thesis. There are two

aspects of integration of PEACH with other policy approaches

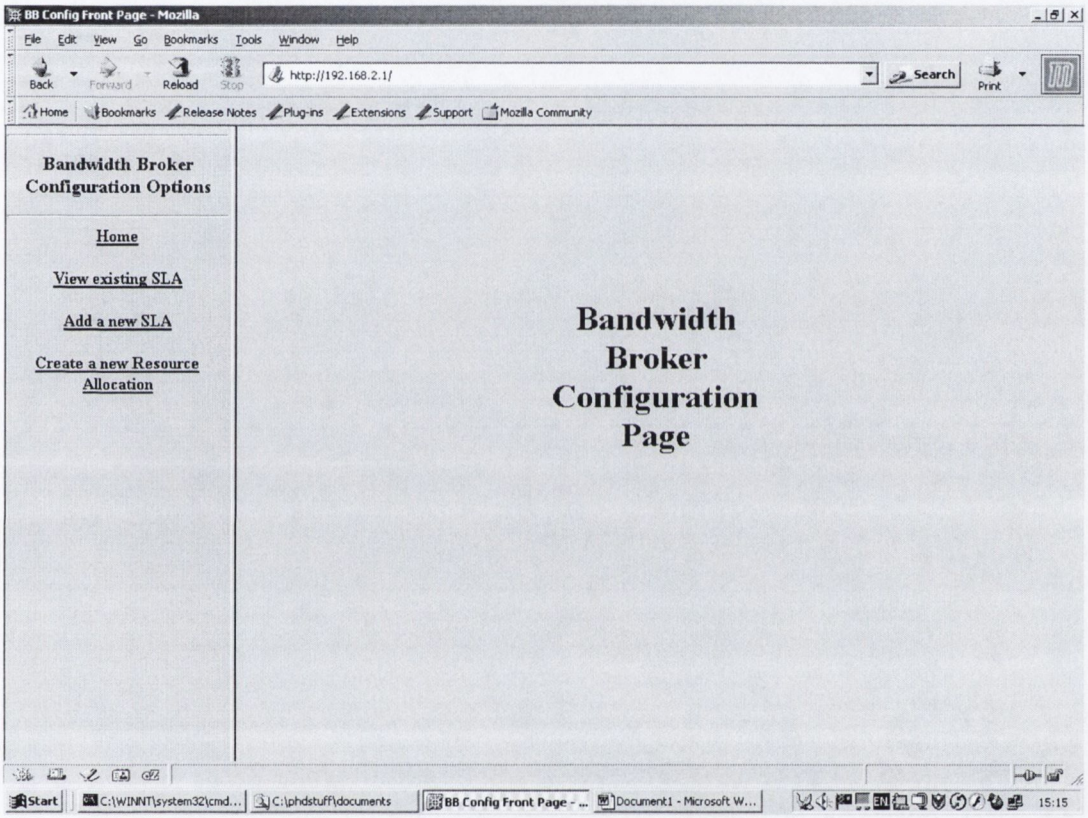
- PEACH can be viewed as component of a policy based billing and charging as described by Hartano, [Hart99] and which has been depicted in Figure 5.1

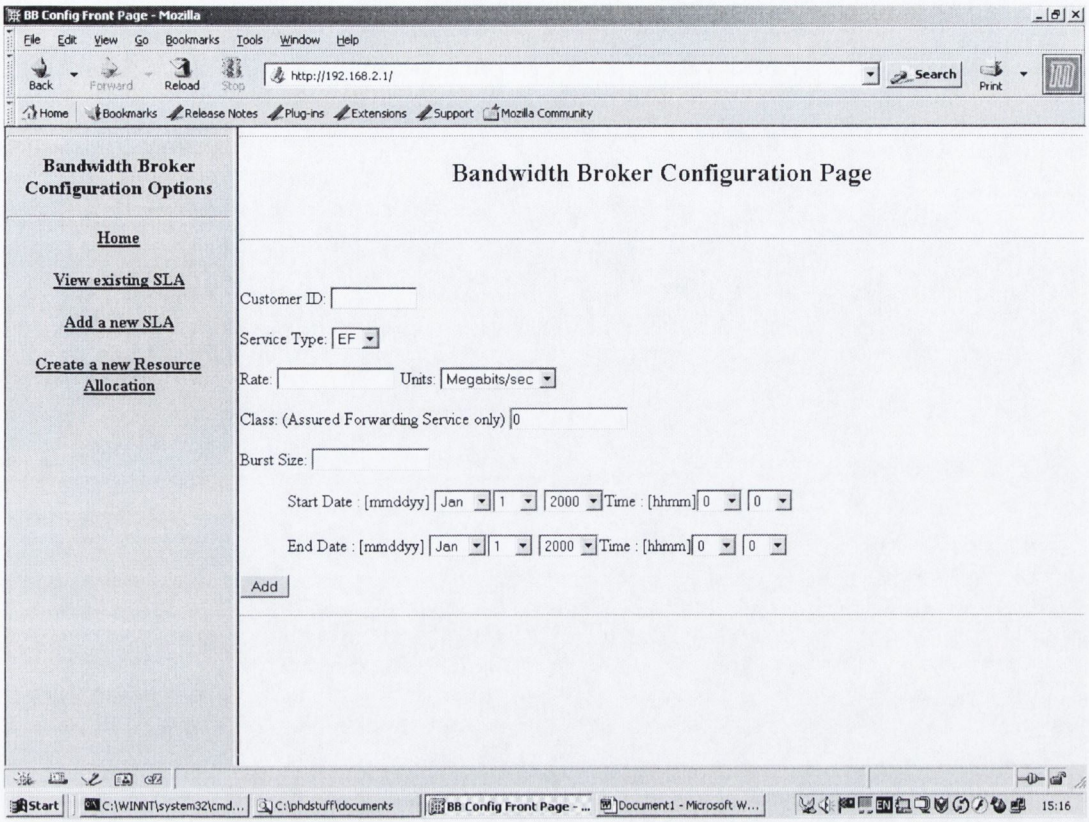
- PEACH can be viewed as a part of a DEN-ng, [Stras03], type policy continuum. In this case PEACH is part of the *network* and *system* levels of the policy continuum. A starting point for a DEN-ng approach would be to model pricing models, charging schemes, meters, tariffs etc as objects in a charging information model. Pricing schemes would be defined in the business layer of the continuum and mapped to charging schemes etc by translation rules.

We believe PEACH provides a new and novel approach to charging in next generation networks. It enables the rapid creation of pricing models for real-time services and facilitates the deployment of real-time charging. While designed for standalone usage it can also form an integral part of a policy, (e.g. DEN-ng), based charging and billing architecture similar to that presented by Carle, [Carl02], and Hartano, [Har99].

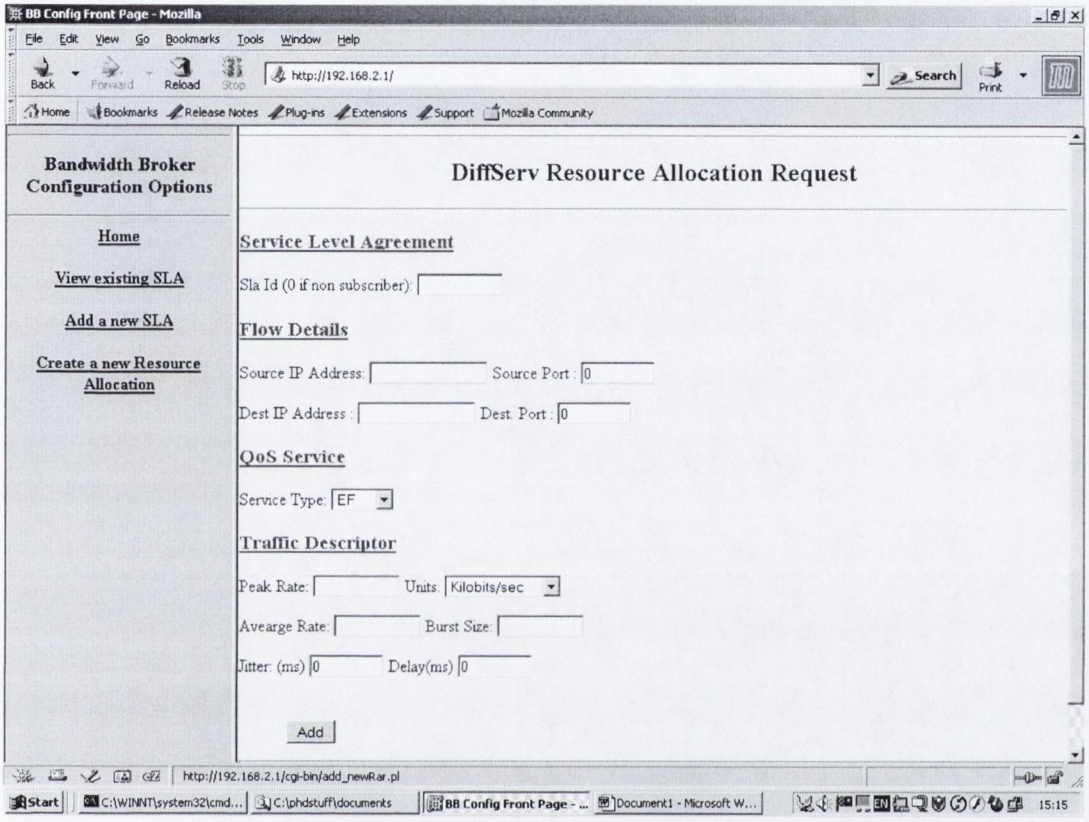
8 Appendix A – IP QoS Charging Screen Shots

A web based management system was implemented to allow Diffserv to be configured and allocated on the QoS test network of Chapter 6. The root page of that configuration system is shown below. Only a subset of the functions of a complete system are implemented. Users choose a particular task from the left-hand screen.

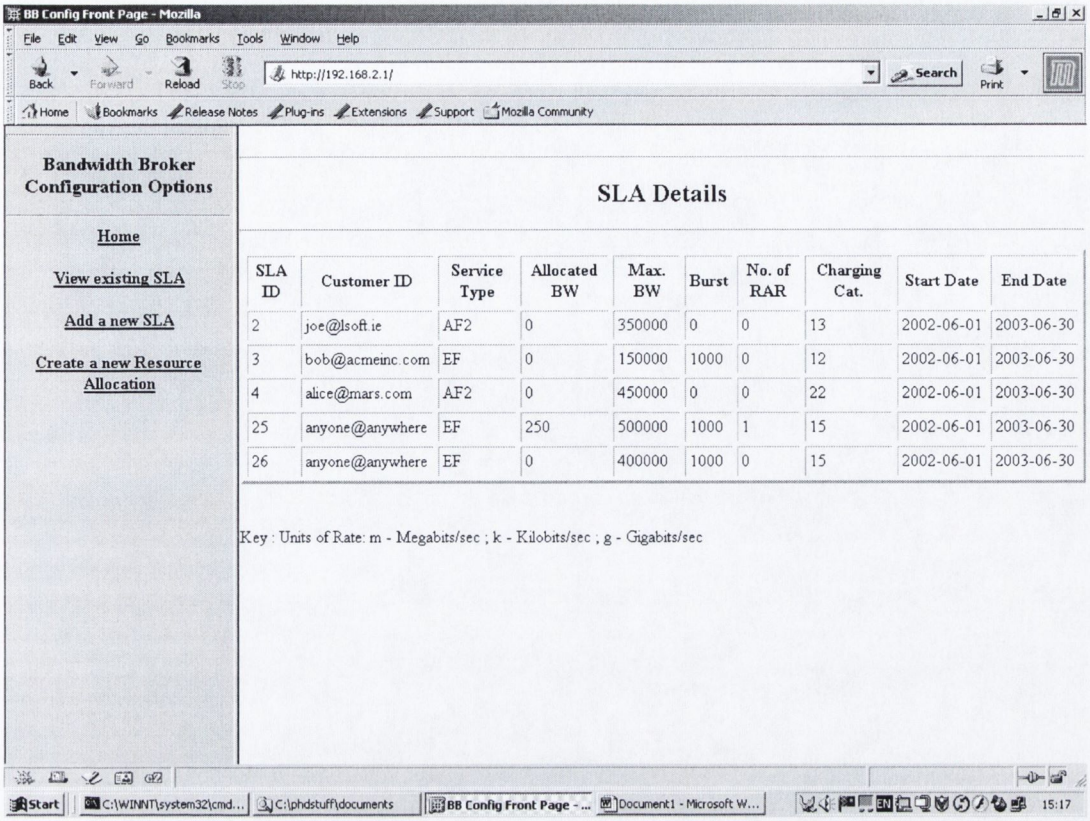




This screen is used to create Service Level Agreements for expedited forwarding or assured forwarding "per-hop-behaviours".



This screen is used to create a Resource Allocation Request. The normal response to this request is the allocation of the requested resources in the network.



This screen shows the list of provisioned SLA. For SLA 25 we can see that currently one RAR is in operation while there are none outstanding for the other SLA's

9 Appendix B – IP QoS Case Study

```
Module IpQ_Analysis {
   context is IpAccounting;
   ccfApplication="IpAccounting"; ccfMessage="chargeResult";

   // get network information
   inpk(neId, ccpUrl, cepUrl);

   destUrl=cepUrl; outp(destUrl);

   // carry out the actual charging analysis for QoS
   call IpQ_analysis;

   // get charging result and identity of CEP module
   //
   inp(analysisResult, chargingFunction);
   pack(destUrl, neId);

   // install the charging module in the CEP
   //
   move  (destUrl, chargingFunction);

   // send the charging result to the CCP
   send(ccpUrl, ccfApplication, ccfMessage, analysisResult, chargingFunction);

   // wait for next event ..

   inputEvent{
      closeAnalysis ->
      {
        exit();
          }

   }// end imputEvent

} // end Module IpQ_Analysis
```

Part 1 – Ip_QoS Case Study

```
Rule IpQ_analysis {

 inp (dscode);
 error=#no_error;
 switch(dscode) {
  in (1,2, 3)  : call IpQ_getAFcsch;
  eq (220)     : call IpQ_getEFcsch;
  default      : error=#charge_error;

 }

 switch (error)
 {
         eq(#no_error) :

       inp(chargingScheme);
          switch (chargingScheme) {

          eq (#no_charge) :
             { analysisResult="no_charge";
               outp(analysisResult);
             }

          eq (#flat_rate)        :
          analysisResult="no_charging";

          eq (#AF_volume)        :
             analysisResult="dynamic_charging";

          eq (#EF_time)          :
             analysisResult="dynamic_charging";

          eq (#EF_volume)        :
             analysisResult="dynamic_charging";

          eq (#EF_time_volume)   :
          {
             call IpQ_getEF_TV_tariffs;
             chargingFunction="IpQ_EFtandv";
             outp(chargingFunction);
             analysisResult="dynamic_charging";
             inp(snet, dnet, sport, dport, balance, meter);
             pack(destUrl,  snet, dnet, sport, dport, balance, meter);

          }
       } // end switch (chargingScheme)

    default :
       analysisResult="charge_error";

 } // end switch (error)

 outp(analysisResult);

} // end IpQ_AnalyseEF
```

```
Rule IpQ_getEFcsch{


   inp(originForCharging);
   switch(originForCharging)
   {
      between(1,30) :
         chargingScheme =#flat_rate;

      between(31,70) :
         chargingScheme=#no_charge;

      between(71,150) :
         chargingScheme=#EF_time_volume;

       default :
         chargingScheme=#EF_volume;
   }

   outp(chargingScheme);

} //end rule
```

```
Rule IpQ_getEF_TV_tariffs {

inp(peakRate, MTU, jitter);

// determine hte tariffs based on trafic profile parameters
//
switch (peakRate) {

 between (0, 10000)      :
   timeTariff=0.1;
   switch (MTU)  {
    between (0,500)    :
     switch (jitter) {
       between (0,5)  : volTariff = 0.05;
          between (5,10) : volTariff = 0.02;
             between (10,50): volTariff = 0.01;
             gt 50 : volTariff = 0.008;
    }

   between (500, 1000) :
     switch (jitter) {
       between (0,5)  : volTariff = 0.05;
       between (5,10) : volTariff = 0.03;
       between (10,50): volTariff = 0.02;
       gt 50 : volTariff = 0.01;
   }

   gt (1000 )          :
     switch (jitter) {
       between (0,5)  : volTariff = 0.07;
       between (5,10) : volTariff = 0.05;
       between (10,50): volTariff = 0.03;
       gt 50 : volTariff = 0.02;
     }
  }
 between (10001, 30000)   :
         timeTariff=0.1;
    switch (MTU)   {
     between (0,1000)   :
        switch (jitter) {
          between (0,5)  : volTariff = 0.05;
          between (5,10) : volTariff = 0.04;
          between (10,30): volTariff = 0.03;
          gt 30 : volTariff = 0.02;
        }
     between (1000,3000):
        switch (jitter) {
              between (0,5)  : volTariff = 0.06;
          between (5,10) : volTariff = 0.055;
          between (10,30): volTariff = 0.4;
          gt 30 : volTariff = 0.02;
        }
     gt 5000          :
        switch (jitter) {
          between (0,5)  : volTariff = 0.08;
          between (5,10) : volTariff = 0.06;
          between (10,30): volTariff = 0.04;
          gt 30 : volTariff = 0.03;
     }
  }
 between (30001, 100000)  :
  timeTariff = 0.15;
    switch (MTU){
     between (0,1000)  :
        switch (jitter) {
          between (0,5)  : volTariff = 0.05;
          between (5,10) : volTariff = 0.04;
          between (10,30): volTariff = 0.03;
          gt 30 : volTariff = 0.02;
     }
```

```
      between (1000,5000) :
        switch (jitter) {
          between (0,5)  : volTariff = 0.08;
          between (5,10) : volTariff = 0.07;
          between (10,30): volTariff = 0.06;
          gt 30 : volTariff = 0.03;
        }
      between (5000,15000) :
        switch (jitter) {
          between (0,5)  : volTariff = 0.1;
          between (5,10) : volTariff = 0.08;
          between (10,30): volTariff = 0.06;
          gt 30 : volTariff = 0.05;
        }
      between  (15000, 50000):
        switch (jitter) {
          between (0,5)  : volTariff = 0.12;
          between (5,10) : volTariff = 0.1;
          between (10,30): volTariff = 0.08;
          gt 30 : volTariff = 0.04;
      }

      gt 50000          :
        switch (jitter) {
        between (0,5)  : volTariff = 0.25;
        between (5,10) : volTariff = 0.2;
        between (10,30): volTariff = 0.18;
        gt 30 : volTariff = 0.16;
      }
    }
  between (100001,1000000) :
    timeTariff = 0.2;
    switch (MTU) {
      between (0, 50000) :
        switch (jitter) {
          between (0,5)  : volTariff = 0.1;
          between (5,10) : volTariff = 0.08;
          between (10,30): volTariff = 0.06;
          gt 30 : volTariff = 0.03;
        }

      between (50000, 100000):
        switch (jitter) {
          between (0,5)  : volTariff = 0.18;
          between (5,10) : volTariff = 0.15;
          between (10,30): volTariff = 0.1;
          gt 30 : volTariff = 0.08;
        }

      gt 100000          :
        switch (jitter) {
          between (0,5)  : volTariff = 0.3;
          between (5,10) : volTariff = 0.2;
          between (10,30): volTariff = 0.18;
          gt 30 : volTariff = 0.15;
        }

  gt 1000000          :
    timeTariff = 0.2;
      switch (MTU) {
        between (10000, 50000) :
          switch (jitter) {
            between (0,5)  : volTariff = 0.1;
            between (5,10) : volTariff = 0.08;
            between (10,30): volTariff = 0.06;
            gt 30 : volTariff = 0.03;
          }
        between (50000, 100000):
          switch (jitter) {
            between (0,5)  : volTariff = 0.18;
            between (5,10) : volTariff = 0.15;
            between (10,30): volTariff = 0.1;
            gt 30 : volTariff = 0.08;
          }
        gt 100000          :
          switch (jitter) {
            between (0,5)  : volTariff = 0.3;
            between (5,10) : volTariff = 0.2;
            between (10,30): volTariff = 0.18;
            gt 30 : volTariff = 0.15;
          }
      }
}
}
```

```
   inp (dayOfWeek) ;

   // determine quantities to be tariffed, depends on time of day
and day of week

         switch (dayOfWeek)
         {
      in(1,5) :
                  inp (startTime);
                  switch (startTime)
                  {
         between (8,18) :
            timeFilter = " secs==300";
            //volFilter = " volBytes==2000000 ";
                              timeChargePeriod=30;
            volChargeAmount=1000000;
                           default :
                                    timeFilter = " secs==300";
                                    //volFilter = "
volBytes==2000000 ";
            timeChargePeriod = 300;
            volChargeAmount = 2500000;
                  }

                  default :
                           timeFilter = " secs==30";
                           //volFilter  = " volBytes == 10000000";
         timeChargePeriod = 600;
         volChargeAmount= 10000000;
         }

   // store the data for transfer to CEP, in "move" command
   //
         pack(destUrl, timeTariff,timeFilter, volTariff,
timeChargePeriod, volChargeAmount);


  }
```

Part 2 – VoIP case study.

```
Module SIP_Analysis {

   context is SipServices;

   // get network information
   inp(sipServer, slaId );
   inpk(sipServer, sccpUrl);
   inpk(slaId, originForCharging, serviceType);

   destUrl=sccpUrl; outp(destUrl, originForCharging, serviceType);

   // carry out the actual charging analysis for QoS
   call Sip_getCsch;
   inp(analysisResult);

   message="SipAnalysisResult"; application="SipServices";

   send( sccpUrl, application,message, analysisResult);

   // wait for next event ..
   inputEvent{
      closeAnalysis ->
      {
        exit();
       }

   }// end imputEvent


} // end module
```

```
Rule Sip_getCsch{

// check for SIP session charging
  inp(originForCharging);
  switch(originForCharging)
  {
     between(1,25) :
        chargingScheme =#no_charge;
        analysisResult = "no_charging";

     between(26,95) :
        chargingScheme=#flat_rate;
        analysisResult = "no_charging";


     between(96,200) :
        chargingScheme=#sip_timed;
        analysisResult = "dynamic_charging";

      default :
        chargingScheme=#flat_rate;
        analysisResult = "no_charging";

  }

// get tariffs, if any

  inp (timeOfDay, destUrl);

  switch (chargingScheme)
  {
     eq(#sip_timed) :
        switch (timeOfDay)
        {
           between (8,18):
              timeTariff=1;
              timeFilter="secs=180";
           default :
              timeTariff=0.1;
              timeFilter="secs=180";

         }
        pack(destUrl, timeTariff, timeFilter, chargingScheme);
  }


// check if media streams are to be charged.
// set the "originForCharging" accordingly

inp(serviceType);

switch (serviceType)
{
   eq (1) :  // POTS
      audioCharging = 21 ;  // Diffserv originForCharging - no charge
      pack(audioCharging);

   eq (2) :  // enhanced POTS
      audioCharging = 5      // flat rate
      pack(audioCharging);

   eq (3) :  // video telephony
      audioCharging = 80;    // time metered
      videoCharging = 180;   // volume metered
      pack (audioCharging, videoCharging);

   eq (9) : // arbitrary streams
      audioCharging = 80;    // time
      videoCharging = 120 ;  // time and volume
      dataCharging  = 120;   // time and volume
      pack (audioCharging, videoCharging, dataCharging);


   default :
      audioCharging = 21;    // time
      videoCharging = 21 ;  // time and volume
      dataCharging  = 21;   // time and volume
      pack (audioCharging, videoCharging, dataCharging);

}  // end switch (serviceType)
```

```
Module SIP_SessionAgent {

   context is SipServices;

   // get network information
   inp(sipServer, slaId, numberOfSessions );
   inpk(sipServer, scapUrl);
   inp(sourceUrl);
   sipUrl=sourceUrl;

   application="SipServices"; message="analyseSip";

   send( application, message, slaId, sipServer, numberOfSessions);

   // wait for next event ..
   inputEvent{

      SipAnalysisResult ->
      {
         inp(analysisResult);
         inp(serviceType); destUrl=sipUrl;

         switch (serviceType)
         {
            eq (1) : // POTS
              inp( audioCharging) ;pack(audioCharging);

            eq (2) : // enhanced POTS
               inp( audioCharging);pack(audioCharging);

            eq (3) : // video telephony
              inp (audioCharging, videoCharging);
              pack (audioCharging, videoCharging);

           default :
              pack (audioCharging, videoCharging, dataCharging);

         }

         application="SipServer"; message = "analysisResult";
         send (sipUrl, application, message, analysisResult);

         if (analysisResult=="dynamic_charging") then
         {
            inp(chargingScheme);
            if (chargingScheme==#sip_timed) then
            {
               inp( timeTariff, timeFilter, timeChargePeriod);
                // start time out for charge calculation
               timeoutRep=120 ; // ever two mins
               subscribe(PTimer, timeoutRep, timeFilter );
            }
         }

      } // end

      //
      timeOutRep ->
      {
         charge = charge+timeChargePeriod*timeTariff;

      }

      //
      closeAnalysis ->
      {
         message="chargeResult"; application="SipServer";
         send (sipUrl, application, charge);
         exit();
       }

   }// end imputEvent
} // end module
```

Part 3 Mp3 Streaming

```
Module Mp3Charge {

  inp( snet);
       inpk (snet, ccfUrl);

       call Mp3InitAnalysis;

  Mp3application="mp3"; costMessage="trackCost";

   inp(networkCharging);
       if (networkCharging == "yes") then
       {
              inp (originForCharging, snet, sport, dport, qosService);

              application ="IpAccounting" ; message ="checkAspQos";
              send(ccfUrl, application, message, snet, dnet, sport, dport, qosService);
       }

       inputEvent {

              nextTrack ->
              {
                 inp(sourceURL);
                     call Mp3GetFilePrice;
                     inp(trackCharge, discountRate);
                     charge=(charge+(trackCharge * discountRate));
                     send(sourceURL,Mp3application, costMessage, charge);
                     charge=0; networkCharge=0;

              }

              noMoreTracks ->
              {
                     application="IpAccounting"; message="close";
                     send(ccfUrl, application, message);
                     exit();
              }

              networkCharge ->
              {
                     inp( networkCharge);
                     charge=charge+networkCharge;

              }

       } // end inputEvent
```

```
Rule Mp3GetFilePrice{

 inpf(mp3chargeCategory, fileSize);
 //outp(mp3chargeCategory, fileSize);

        switch (mp3chargeCategory)
        {

                eq (1) :
                        trackCharge = 8;
                eq (2) :
                        trackCharge = 10;
                eq (3) :
                        trackCharge = 13;
                default :
                        trackCharge = 0;
        }

        if (fileSize > 6) then
                sizeRate = 1.1;
        else
         if (fileSize > 12) then
                sizeRate = 1.25;
         else
                sizeRate = 1;

         trackCharge= trackCharge*sizeRate;
         outp( trackCharge);


} //end rule
```

```
Mp3InitAnalysis {

        inp (originForCharging, numberOfTracks );

    switch( originForCharging)
    {
                between (800, 900) :
                        networkCharging="yes";
                        qosService="silver";
                        outp(qosService);

                between (900, 1000) ;
                        networkCharging="yes";
                        qosService="bronze";
                        outp(qosService);

                default :
                        networkCharging="no";
        }


        switch (numberOfTracks)
        {
           in (2,3,4) :
                discountRate = 0.9;

           between (5,10) :
                discountRate = 0.7;

           gt (10) :
                discountRate=0.5;

           default :
               discountRate=1.0;
         }

        outp(discountRate,networkCharging, numberOfTracks);

} // end Mp3InitAnalysis
```

10 Appendix C – PEACH Implementation

10.1 Package Structure

The Java package structure for the complete system is shown in below.

There are three main subgroups under the PEACH root

1. **ee** (execution environment) contains classes for the runtime platform
2. **interpreter –** contains classes for the APPLE interpreter and the event filtering interpreter
3. **gui –** contains classes for viewing and editing context and protocol objects.

EE Package Group

This group contains the following subgroups/packages

PEACH.ee.contextBuilder –This package contains classes for parsing and loading XML files. It also contains Java base classes that are used to create policy parameter classes and related helper classes.

PEACH.ee.contexts. – This subgroup contains all contexts that are used within a node. There will be a separate package for each context used. These have the form *PEACH.ee.contexts.xyz,*.for context xyz and *PEACH.ee.contexts.xyz.protocol* where *protocol* is the policy protocol in use. The first of these two contains context specific policy parameter components, configuration files and other policy components. The second package contains all policy parameters which are derived from the policy protocol *protocol.* The exact contents of the PEACH.ee.contexts. subgroup will depend of course on how PEACH is used. The diagram shows the IpAccounting context as an example and the Prr protocol.

PEACH.ee.engine – This package contains much of the runtime subsystem including the policy space object and the PEACH main controller.

PEACH.ee.event – This package contains base classes which are used to derive context event handling classes and it also contains run time support for event forwarding.

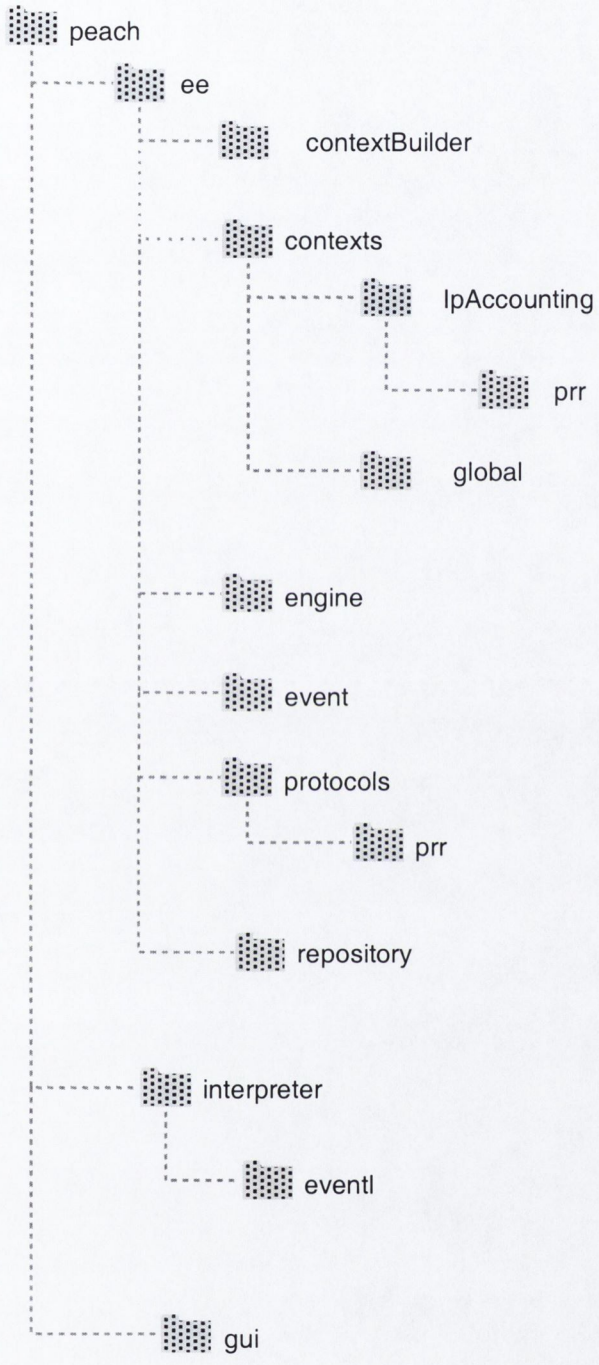


Figure 10-1- Java Package Structure

PEACH.ee.protocols – This subgroup/package contains a package for every protocol used in a PEACH node. The exact content may therefore vary from node to node. The protocol package will contain policy protocol servers and protocol object classes as well as helper classes of various sorts. The diagram above is shown for the case of the Prr protocol.

PEACH.ee.repository – This package contains classes that are used to store context information in a runtime system.

Interpreter Package Group

PEACH.interpreter – contains all the classes that are used to implement a parser and interpreter for APPLE.

PEACH.interpreter.eventl – This package contains the interpreter for the event filtering language.

PEACH Gui Package

PEACH.gui.- This package contains classes for manipulating and viewing context information.

We can map the packages to the various subsystems Figure 5-8 as follows

Subsystem	Package
PEACH Runtime Subsystem	PEACH.ee.engine , PEACH.interpreter
Communication Subsystem	PEACH.ee.protocols, PEACH.ee.protocols.*xyz* , PEACH.ee.event
Context Management Subsystem	PEACH.ee.contextBuilder, PEACH.ee.repository
PEACH Management	PEACH.gui

Table 10-1PEACH Subsystem Java Packages

10.2 Main Classes

The main classes used in the design are shown below in Figure 10-2.

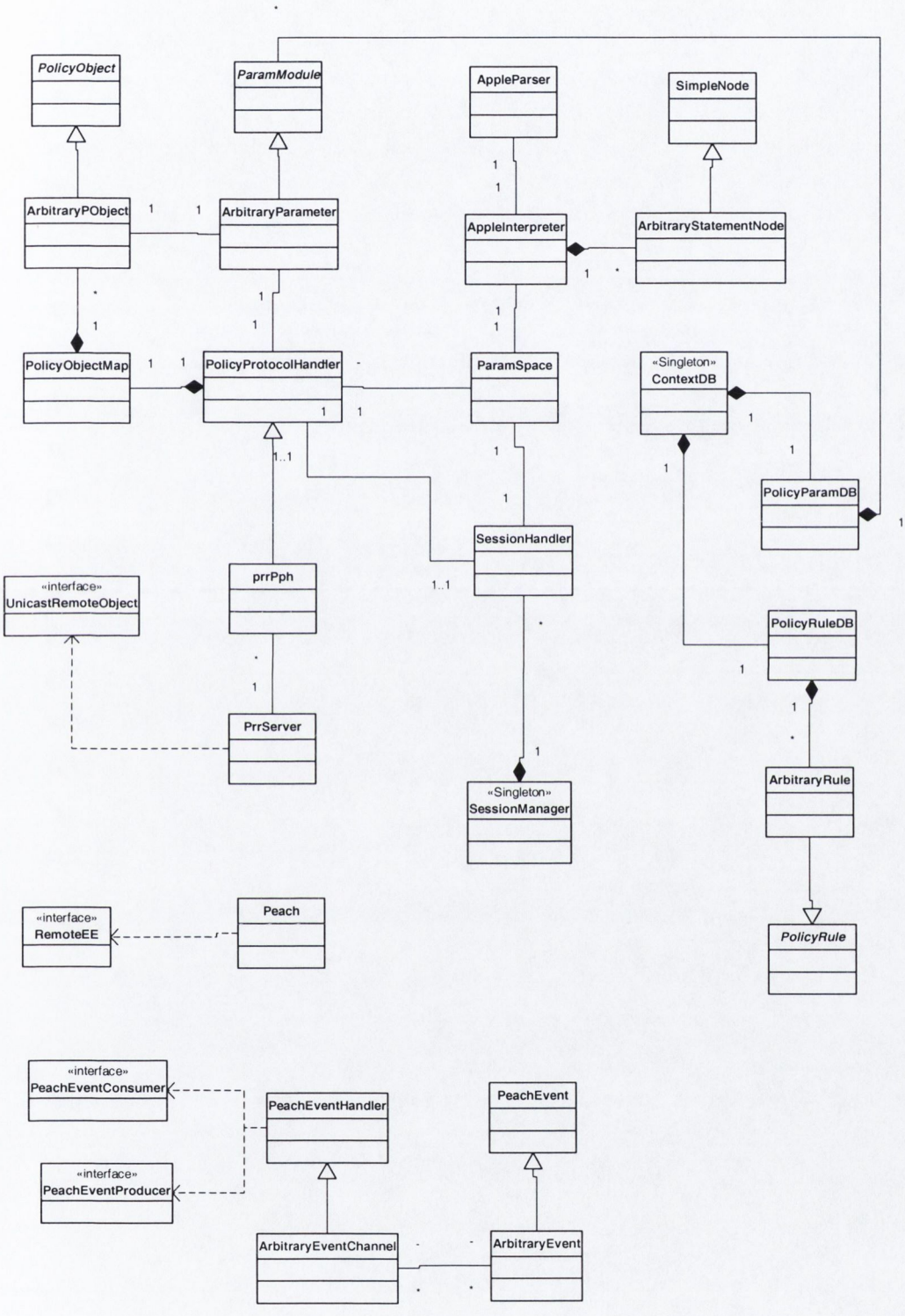


Figure 10-2 PEACH Main Classes

10.2.1 Interpreter

The APPLE interpreter is based on a JavaCC, a Java compiler-compiler .

10.2.2 Run-Time Core

The run-time core deals with service component invocation and session handling. It is based on interaction between the following main classes ParamSpace, ParamModule, PolicyObject, PolicyProtocolHandler, PEACH and in some cases subclasses of some of these classes. Certain other auxiliary classes are involved.

The ParamModule and PolicyObject classes are used as base classes to define policy parameters and policy objects. These are depicted in Figure 10-2 as *ArbitraryParameter* and *ArbitraryPObject* and act as place-holders for all such subclasses. At compilation or development time, a Java class is generated for each policy parameter and policy object which are defined in the respective context XML configuration files. The binding between policy parameter and policy object which is specified in the context protocol mapping XML file is written into the policy parameter class at compilation time. Also accessor methods for both parameter and policy object classes are written into the respective Java class files at compilation time. Accessor methods are defined as a '*get_parameter*' and *set_parameter*' for each parameter and policy object class. This is depicted in the diagrams below for the 'sustainableRate' policy parameter . The example is taken from the IpAccounting context referred to earlier. The entry for jitter in the IpAccounting context file is:-

```
<policy_param name="sustainableRate" type = "5017" keyed = "no" keyEntity =""> 
      <field name = "sustainableRate" type="int" defval="100000" />
    </policy_param>
```

while the corresponding entry from the mapping file is :-

```
    <policy_param name="sustainableRate" >
          <policy_object name = "dsFlowSpec" />
               <field name = "averageRate" />
    </policy_param>
```

This shows that the sustainableRate parameter is derived from a policy object call dsFlowSpec and is based on a field in that object called averageRate. The entry for dsFlowSpec in the Prr protocol file is

```
<policy_object name="dsFlowSpec"  typeId = "1006">
  <field name = "mtu" type = "int" defval="600" />
  <field name = "jitter" type = "int" defval="90" />
  <field name = "peakRate"  type = "int" defval="500200"/>
  <field name = "averageRate" type = "int" defval="90000" />
  <field name = "lossSensitivity" type = "int" defval="1" />
  <field name = "burstSize" type = "int" defval= "150000"/>
</policy_object>
```

The corresponding classes are shown in Figure 10-3 and Figure 10-4

These classes are invoked each time a reference is made to sustainableRate in either the *inp* or *outp* APPLE statements. ParamSpace serves as the main service composition device in PEACH. Every statement in the language which can invoke a service component, including parameter modules, has a related method defined in ParamSpace. This includes the *inp, outp, action, send, pack* and *subscribe* statements. For all of these statements the policy parameter or policy component name is used to find the appropriate Java class by the use of Java's reflection mechanism. Reflection is central to the working of PEACH.

When the statement *inp(sustanableRate)* is encountered the inp method in ParamSpace is invoked and using reflection a new instance of the sustainableRate class is created and the method get_sustainableRate() is called on this object. This in turn creates a new instance of dsFlowSpec and invokes method get_avarageRate() on this instance. The data is then returned to the interpreter.

A similar mechanism is used for setting data. For policy parameters which are not based on protocol objects the developer must hand code the accessor methods in the corresponding Java class.

For statements other than inp and outp ParamSpace creates an instance of the appropriate object and invokes the appropriate method.

```
//
//        PEACH - policy exection engine
//       (C) Brian Lee - 2000-2003
//
//

package  peach.ee.contexts.IpAccounting.prr;
import peach.ee.contextBuilder.*;
import  peach.ee.protocols.prr.*;
import  peach.ee.contexts.IpAccounting.*;
import java.lang.reflect.*;
import peach.ee.protocols.*;
import peach.ee.engine.*;
import peach.common.*;
import java.util.Vector;
//
public  class  sustainableRate extends ParamModule  {

//
//   Data Members
//

PolicyProtocolHandler  pph;
SessionHandler  sh;
int  sustainableRate;

//    Constructor
//
public  sustainableRate(PolicyProtocolHandler ph, SessionHandler ssh) {
//
  pph = ph;
  sh = ssh;
 Class cl = getClass();
 name = cl.getName();
}
//
//   Method Members
//
public int  get_sustainableRate()  {

  dsFlowSpec  po = new dsFlowSpec();
  sustainableRate = ((dsFlowSpec)pph.getPoData(po)).get_averageRate();

     return sustainableRate;
 }
//

public  void  set_sustainableRate( Integer obj)  {
  boolean newObj = false;
  dsFlowSpec po = (dsFlowSpec)pph.checkObjExist("dsFlowSpec");
  if (po == null) {
    po = new dsFlowSpec();
    newObj = true;
  }
  po.set_averageRate(obj);
  pph.storePoData(po, "dsFlowSpec", newObj);
 }
}
```

Figure 10-3 Java class for sustainableRates

```
package peach.ee.protocols.prr;
import peach.ee.contextBuilder.*;
import peach.ee.engine.*;
import peach.ee.protocols.*;
public class dsFlowSpec extends PolicyProtocolObject {
//
//    Constructor
//
public dsFlowSpec() {
//
        Class cl = getClass();
        name = cl.getName();
}
//
//   Data Members
//
int mtu;
int jitter;
int peakRate;
int averageRate;
int lossSensitivity;
int burstSize;
//   Method Members
..
..
public int get_averageRate() {
     return averageRate;
 }
public void set_averageRate(Integer aVar ) {
   averageRate = aVar.intValue() ;
 }

..
..
} // end class
```

Figure 10-4 Java class for dsFlowSpec

Other run time classes include PEACH which is the main management or control entity for the whole system and which provides an interface for ongoing management during the operation of the system.

10.2.3 Event Handling

PEACH provide a number of classes and interfaces to allow developers to define specific event channels and events. An object which wishes to receive events implements the PEACHEventConsumer while an object which supplies events implements the interface PEACHEventSupplier. These interfaces are loosely based on

elements of the CORBA event service and allow events to be either pushed to the consumer by the event channel or to be pulled from the event channel by the event consumer.

The PEACHEventHandler class provides support for storage, retrieval and forwarding of events. It supports both of the above interfaces and this allows the possibility of chaining event channels.

Other classes include PEACHEvent which serves as a base class for events, EventFilter which is used to create and evaluate filters, EventSubscription which is used to create a subscription for events.

10.2.4 Repository

The ContextDB and associated classes are used to store context parameters and modules during run time.

11 Appendix D – APPLE Grammar

APPLE-program -> Policy-Block

Policy-Block -> (PolicyRule | Module) +

PolicyRule -> 'Rule' identifier '{' ContextStatement* RuleStatement* | CommonStatement* '}'

Module -> 'Module' Identifier '{' ContextStatement* ModuleStatement | CommonStatement)* '}'

RuleStatement- > SwitchStatement

SwitchStatement -> 'switch' '(' expression ')' '{' SwitchLabel SwitchBlock '}'

SwitchLabel -> SwitchExpression ':' | 'default'

SwitchBlock -> CommonStatement* SwitchStatement*

SwitchExpression -> BetweenExpression | InExpression | EqExpression | GtExpression

BetweenExpression -> 'between' '(' SimpleExpression ',' SimpleExpression ')'

InExpression -> 'in' '(' (Literal [',])* ')'

GtExpression -> 'gt' Expression

Factor -> Literal | Identifier | '(' Expression ')'

Term -> Factor ('*' Factor | '/' Factor | '&&' Factor) *

SimpleExpression -> Term ('+' Term | '-' Term | '||' Term)*

Expression -> SimpleExpression ('==' SimpleExpression
| '<=' SimpleExpression
| '>=' SimpleExpression
| '<' SimpleExpression

| '>' SimpleExpression
| '!=' SimpleExpression)*

CommonStatement -> AssignmentStatement | IfStatement | IOStatement |
BlockStatement | CallStatement | PackStatement |
MoveStatement | SendStatement | ExitStatement |
StartStatement | ';'

ModuleStatement -> InpEventStatement | SubscribeStatement | ActionStatement

AssignStatement -> Identifier '=' Expression

IfStatement -> 'if' '(' Expression ')' 'then' CommonStatement
['else' CommonStatement]

IOStatement -> InpStatement | InpfStatement | InpkStatement
| OutpStatement

BlockStatement -> '{' CommonStatement* '}'

CallStatement -> 'call' Identifier

PackStatement -> 'pack' '(' Identifier ',' (Identifier [','])* ')'

MoveStatement -> 'move' '(' Identifier ',' Identifier ')'

StartStatement -> 'start' '(' Identifier ',' identifier ')'

ExitStatement -> 'exit' '(' ')'

SendStatement -> 'send' '(' Identifier ',' Identifier ',' Identifier ([','] Identifier
)* ')'

InpStatement -> 'inp' '(' (Identifier [','])* ')'

InpfStatement -> 'inpf' '(' (Identifier [','])* ')'

InpkStatement -> 'inpk' '(' Identifier ',' (Identifier [','])* ')'

OutpStatement -> 'outp' '(' (Identifier [','])* ')'

InpEventStatement -> 'inpEvent' '{' (Identifier '->' '{' (CommonStatement | ActionStatement)* '}')+'}'

SubscribeStatement -> 'subscribe' '(' Identifier ',' Identifier ',' Identifier ')'

ActionStatement -> 'action' '(' Identifier '.' Identifier ([',] Identifier)* ')'

ContextStatement -> 'context' 'is' Identifier

Identifier -> Letter (Letter | Digit)*

Letter -> ['a'-'z', 'A'-'Z', '_']

Digit -> ['0'-'9']

Literal -> IntegerLiteral | FloatingPointLiteral | StringLiteral | SymbolicLiteral

IntegerLiteral -> Digit Digit*

FloatingPointLiteral -> Digit+ '.' Digit+

StringLiteral -> "" Identifier ""

SymbolicLiteral ->'#'Identifier

This is the grammar for the actual implementation of APPLE used in the development of the thesis case studies. This version of the language is not safe however as it is possible to introduce some Module statements into Rules. The problem arises due to the shared usage of the 'if' statement in modules and rules. To make the language more safe it necessary to introduce a rule specific 'if' statement to prevent module statements from occurring in rules. It is also necessary to modify the related productions. These modifications are shown below.

RuleIfStatement -> 'ifp' '(' Expression ')' 'then' CommonStatement | SwitchStatement ['else' CommonStatement | SwitchStatement]

CommonStatement -> AssignmentStatement | IOStatement | BlockStatement

CallStatement | PackStatement | ';'

ModuleStatement -> InpEventStatement | SubscribeStatement | ActionStatement | StartStatement | MoveStatement | SendStatement | ExitStatement | IfStatement

IfStatement -> 'if' '(' Expression ')' 'then' CommonStatement | ModuleStatement ['else' CommonStatement | ModuleStatement]

RuleStatement -> SwitchStatement | RuleIfStatement

12 References

[Alae99] Alaettinoglu C. et al., "Routing Policy Specification Language", IETF RFC2622 June 1999.

[Alex98] Alexander D.S. et al., "A Secure Active Network Environment Architecture: Realization in SwitchWare" IEEE *Network* May 1998

[Anco98] Composable Services for Active Networks Active Network Composable Services Working group Sept. 1998.

[Banj01] Banarjee A. et al., "Generalised Multiprotocol Switching: An Overview of Management and Routing Enhancements", *IEEE Communications,* Jan 2001.

[BB98] "Bandwidth Broker High Level Design" Revision 0.4, British Columbia Institute of Technology, Group for Advanced Information Technology, Nov. 1998

[BBTP98] "Bandwidth Broker Transfer Protocol" Revision 0.3, British Columbia Institute of Technology, Group for Advanced Information Technology, Nov. 1998

[Bell02] Bellavista P, Corradi A and Vecchi S, "Mobile Agents for Usage-based Accounting in Wireless Ubiquitous Networks" WOA2002 University of Milan-Bicocca, November 2002.

[Bis98] J. Biswas et el., " The IEEE P1520 Standard Initiative for Programmable Network Interfaces" IEEE Communication Magazine, Special Issue in Programmable Networks, Oct. 1998.

[Breu98] Breugst M. and Magedanz T., "Mobile Agents – Enabling Technology for Active Network Implementation" IEEE *Network* May 1998

[Bris00] Briscoe B. et al., "Lightweight Policing and Charging for Packet Networks,". , *Proc. IEEE 3rd Conference on Open Architectures and Network Programming, OPENARCH 2000.* Tel Aviv, Israel, March 2000.

[Brow95] Brownlee N. "Internet Pricing in Practise" in [McKB97]

[Calv98] Calvert K.L. ed., "Architectural Framework for Active Networks – Version 1.0" Active Networks Research Group University of Kentucky Sept.1999

[Calv99] Calvert K.L. et al. "Directions in Active Network". *IEEE Network* Oct. 1998

[Clav01] Clavenna S. and Heywood P., "Optical Taxonomy", lightreading.com. http://www.lightreading.com/document.asp?site=lightreading&doc_id=3780&page_number=1

[Camp99] Campbell A. *et al.*, "A Survey of Programmable Networks", ACM SIGCOMM Computer Communications Review Vol. 29 issue 2, April 1999.

[Camp01] Campbell A and Gomez J., "IP Micro-Mobility Protocols", ACM SIGMOBILE Mobile and Computer Communication Review (MC2R) Vol. 4 No. 4 October 2001

[Carl02] Carle G. et al. "Policy Based Accounting" IRTF Internet Draft Feb 2002, <draft-irtf-aaaarch-pol-acct-04.txt>

[Chen99] Chen T, "Reliable Services in MPLS", *IEEE Communications* Dec. 1999

[Cisc03] Cisco Systems, NetFlow www.cisco.com

[Clar89] Clark D., "Policy Routing in Internet Protocols" IETF RFC 1102, May 1989. www.ietf.org

[Clark97] Clark D. "Internet Cost Allocation and Pricing" in [McKB97a]

[Cocc93] Cocchi R. et al., "Pricing in Computer Networks: Motiviation, Formulation and Example", *IEEE/ACM Transactions on Computer Networking*, 1(6) pp. 614-627 1993.

[Dam00] Daminaou N et al, "Ponder: A Language for Specifying Security and Management Policies for Distributed Systems". *Research Report DoC 2000/1*, Imperial College London, Apr. 2000.

[Das00] Da Silva L., "Pricing for QoS Enabled Networks : A Survey", *IEEE Communications Surveys*, No 2. 2000

[Dieg02] Diego MP3 Streaming Server http://diego.sourceforge.net

[DMTF04] Distributed Management Task Force, www.dmtf.org

[Edel95] Edell R. et al. "Billing and Pricing Users for TCP", *IEEE Journal on selected Areas in Communication*, Sept. 1995

[Edel99] Edell R and Varaiya P. "Pricing the Internet: What we Learn from INDEX "
Proceedings INFOCOM 99, March 1999, New York.

[Eric97] Ericsson Inc. "Routing Analysis Functional Specification", 1997. Internal publication

[Faul95] Faulhaber G. R. "Pricing the Net : What Economists do " Standford Mini-Conference on Data Network Pricing, Stanford University , March 17 1995

[Ferg98] Ferguson P. and Huston G. "*Quality of Service – Delivering QoS on the Internet and in Corporate Networks*", Wiley Computer Publishing, New York 1998

[Ghan99] Ghanwani A. et al., "Traffic Engineering Standards in IP networks using MPLS". *IEEE Communication*, December 1999

[Gong97] Gong J. and Srinagesh P., "The Economics of Layered Networks", in [MckB97a]

[GSM03] GSM Association, "Transferred Account Procedure (TAP3)" version 3.10, June 2003, www.gsmworld.com

[Hart99] Hartano F. And Carle G., "Policy Based Billing Architecture for Internet Differentiated Services"
Proceedings IFIP Fifth International Conference on Broadband Communications (BC'99) Hong Kong, 10-12 November 1999

[Hern02] E. Hernadez-Valencia ,"Hybrid Transport Solutions for TDM/Data Networking Services", *IEEE Communications,* May 2002

[Hous02] Houssos et al. "Advanced Business Models and Flexible Service provision for Service Reconfigurable Mobile Networks"
SDR 2002, San Diego California.

[IEEE11] IEEE 802.11 Working Group on Wireless Local Area Networks
http://grouper.iee.org/groups/802/11

[IEEE16] IEEE 802.16 Working Group on Broadband Wireless Access
http://grouper.iee.org/groups/802/16

[IETF91] Mills C. et al, "Internet Accounting Background"", RFC1272, 1991, www.ietf.org

[IETF94] Braden R. *et al.* " Integrated Services in the Internet Architecture: An Overview", RFC1633, 1994, www.ietf.org

[IETF96] Perkins C., "IP Mobility Support"

RFC2002 October 1996, www.ietf.org

[IETF98] Blake S. et al., "An Architecture for Differentiated Services", RFC2475, Dec 1998., www.ietf.org

[IETF99] Heinanen, J. "Assured Forwarding PHB Group" RFC2597 Jun. 1999, www.ietf.org

[IETF99a] Brownlee N. *et al.*, "Traffic Flow Measurement: Architecture" RFC2722, Oct.1999, www.ietf.org

[IETF99b] Brownlee N., "SRL: A Language for Describing Traffic Flows and Specifying Actions for Flow Groups"., RFC2723, Oct. 1999, www.ietf.org

[IETF00] Durham D et al., "The COPS (Common Open Policy Service) Protocol", IETF RFC2748 Jan. 2000

[IETF00a] Rigney C., "Radius Accounting" IETF RFC 2866 Jun. 2000, www.ietf.org

[IETF00a] Yavatkar R. et al., "A Framework for Policy based Admission Control", RFC 2753, Jan. 2000. www.ietf.org

[IETF03] Calhoun P et al., "Diameter Base Protocol" IETF RFC3588 Sept. 2003. www.ietf.org

[IETF04] Ernst T., "Network Mobility Support Goals and Requirements" <draft-ietf-nemo-requirements-02.txt> www.ietf.org

[IPDR02] IPDR Organization, "Network Data Management Usage Specification – Version 3.1.1", October 2002, www.ipdr.org

[Jain03] Sun Microsystems "Java API's for Integrated Networking" java.sun.com/products/jain

[Kaw00] Kawamura R and Stadler R., "Active Distributed management for IP Networks", *IEEE Communications* April 2000 pp114-120

[Kesh97] Keshav S., *"An Engineering Approach to Computer Networking"*, Addison-Wesley, Reading, Massachusetts, 1997.

[Koh00] Kohli M. and Lobo J., " Policy Based Management of Telecommunications Networks", Policy Workshop HP Labs Bristol UK Nov.1999

[Kout02] Koutsopolou M., Kaloxylos A. and Alonistioti A. , "Charging, Accounting and Billing as a Sophisticated and Reconfigurable Discrete Service for Next Generation Mobile Networks" IEEE Semiannual Vehicular Technology Conference (Fall VTC2002) September 2002 Vancouver Canada

[Kout04] Koutsopolou et al., "Charging, Accounting and Billing Management Schemes in Mobil Telecommunication networks and the Internet"
IEEE Communications Surveys and Tutorials, First Quarter 2004, pp 50-58

[Lach03] Lach H.Y. and Natarajan N., "Support of Mobile Networks in B3G Systems"
Proceedings of International Conference on Communication Technology ICCT2003, Beijing April 2003

[Laz97] Lazar A.A ., "Programming Telecommunication Networks"
IEEE Network vol.11, no.5, September 1997.

[Leida98] Leida. B.A," A Cost Model of Internet Service Providers: Implications for Internet Telephony and Yield Management"
, M. Sc. Massachusetts Institute of Technology, Feb 1998.

[Lein89] Leiner B, "Policy Issues in Interconnecting Networks",
IETF RFC 1124 1989 www.ietf.org

[Lin01] Lin Y.B. and Chlamtac I., "*Wireless and Mobile Network Architectures*", J. Wiley & Sons, New York, 2001

[Low93] Low S. and Varaiya P., "A new approach to service provisioning in ATM networks", *IEEE/ACM Transactions on Networks*, vol. 5 1993

[LuCo99] Lucas M. and Cohen O "Usage Collection and Analysis in an IP OSS", *Billing World*, March 1999

[LuLa99] Lucas M. and Labuda D., "Batch Systems for Internet Billing? Think Real-Time !", *Billing World*, Jan 99

[Lymb02] Lymberopoulos L., *et al* , "An Adaptive Policy Based Management Framework for Differentiated Service Networks"
Proc. 3rd Workshop on Policies for Distributed Systems and Networks, Monterey California, June 2002.

[Mack97] Mackie-Mason H, Murphy L and Murphy J, "Responsive Pricing in the Internet", in [McKB97a]

[Marr96] Marriot D. and Sloman M., "Implementation of a Management Agent for Interpreting Policy", *Proc. IEEE/IFIP 7th International Workshop on Distributed Systems Operations and Management, (DSOM96,)* Oct 1996

[McKB97] McKnight L.W. and Bailey J.P., "Internet Economics – When Constituencies Collide in Cyberspace", *IEEE Internet Computing* Nov./Dec. 1997

[McKB97a] McKnight L.W and Bailey J.P. , eds. *Internet Economics.* The MIT Press, Cambridge Mass. 1997

[Mess99] Messerschmitt D. and Hubaux J.P, "Opportunities for Electronic Commerce in Networking", *IEEE Communications*, Vol. 37, No. 9 Sept. 1999 pp 92-99

[Metz99] Metz C., "AAA Protocols: Authentication, Authorisation and Accounting for the Internet", *IEEE Internet Computing.* Nov./Dec. 1999

[MMV94] Mackie-Mason J.K and Varian H. R, "Pricing the Internet", Proc. "*Public Access to the Internet*", *JFK School of Government*, 1994.

[Mor99] Morris D. and Pronk V., "Charging for ATM Services", *IEEE Communications*, May 1999.

[MYSQ04] MySQL data base. www.mysql.org

[Murph95] Murphy L. and Murphy J., "Feedback and pricing in ATM Networks", Proc. IFIP TC^ Third Workshop on performance Modelling and Evaluation of ATM Networks, Ilkley UK Jul. 1995

[M3I03] Briscoe et al., "A Market Managed Multiservice Internet (M3I)", *Computer Communications* 26(4) pp 404-414, February 2003.

[OMah03] O'Mahony D. and Doyle L, "Beyond 3G: 4G IP-Based Mobile Networks" in *Wireless IP: Building the Mobile Internet* ed. Dixit S., Arctech House, Norwood, MA 2003 Chap. 6 pp 71-86.

[Odly98] Odlykzo A.M , "The Economics of the Internet: Utility, Utilization, Pricing and Quality of Service" , July 1998 available from http://www.research.att.com/~amo

[Odly98a] Odlykzo A.M, "Paris Metro Pricing for the Internet", May 1998 available from http://www.research.att.com/~amo

[OSA02] 3GPP – "Service Requirements for Open Services Access" TS22.127 www.3gpp.org

[Oust98] Ousterhout J K, "Scripting: Higher level programming for the 21st Century ". *IEEE Computer* March 1998

[Papa98] Papadopolous G. and Arbab F, " Coordination Models and Languages", in *Advances in Computers*, Academic Press, August 1998, vol. 46: The Engineering of Large Systems

[Parl03] Parlay 3.3 Specification

www.parlay.org

[Parr92] Parris C and Ferrari D., "A Resource based Pricing Policy for Real-Time Channels in a Packet-Switching Network" Technical Report TR-92-018 , International Computer Science Institute, Berkeley, California 1992.

[Pax00] PAX PDL http://www.alternic.net/drafts/drafts-n-o/draft-nossik-pax-pdl-00.html

[PCIM01] Moore B. et al., "Policy Core Information Model – Version 1 Specification", IETF RFC 3060 Feb. 2001, www.ietf.org

[PCLD01] Strassner J et al. "Policy Core LDAP Schema" http://www.ietf/org/internet-drafts/draft-ietf-policy-core-schema-16.txt

[Pcube04] "Encharge ", www.p-cube.com/products/encharge

[Peir99] Peirce M and O'Mahony D., "Flexible Real Time Payment Methods for Mobile Communications", *IEEE Personal Communications*, Dec. 1999

[Phan98] Phan V.U and Karmouch A , "Mobile Software Agents: An Overview". *IEEE Communications* vol.36 no.7 July 1998

[Poly99] Polyzois C. A. et al., "From POTS to PANS: A Commentary on the Evolution to Internet Telephony", *IEEE Communications* May/June 1999

[Pras01] Pras A., "Internet Accounting", *IEEE Communications*, May 2001

[QPIM01] Moore B. et al, "Policy QoS Information Model" http://www.ietf/org/internet-drafts/draft-ietf-policy-qos-info-model-05.txt

[Raj99] Rajan R.et al., "A Policy Framework for Integrated and Differentiated Services in the Internet", *IEEE Network,* Sept./Oct. 1999

[Rapp04] Rappa M.A., "The Utility Business Model and the Future of Computing Services" *IBM Systems Journal* Vol. 43 No. 1 2004

[Raz00] Raz D. and Shavitt Y, "Active Networks for Efficient Distributed Active Network Management", *IEEE Communications* March 2000

[Rose99] Rosenberg J. et al., "Programming Internet Telephony Services"

IEEE Network May 1999

[Sals02] Salsano S. and Vetriani L., "QoS and Policy Control by means of COPS to support SIP based Applications" *IEEE Network* Mar/Apr 2002

[Schu00] Schultzrinne H., "SIP for Mobile Applications", VON Developers Summer Conference 2000, Boston USA.

[Schn99] Schneider J-G, and Nierstrasz O., "Scripting: Higher Level Programming for Component based Systems", Tutorial at *European Conference on Object Oriented Programming (ECOOP)* 1999

[Shen95] Skenker S., "Service Models and Pricing Policies for an Integrated Services Internet " in *Public Access to the Internet* MIT Press 1995

[Shen96] Shenker S. et al., "Pricing in Computer Networks: Reshaping the Research Agenda", *ACM Computer Communication Review* vol. 26, April 1996

[Slo94] Sloman M., "Policy Driven Management for Distributed Systems" *Journal of Network and Systems Management*, 2(4):333-360, 1994

[Slo99] Sloman M. and Lupu E., "Policy Specification for Programmable Networks", *Proceedings First International Conference on Active Networks* IWAN99 Berlin June 1999

[Slo02] Sloman S. and Lupu E., "Security and Management Policy Specification", IEEE *Network* March 2002

[Song97] Songhurst D. and Kelly F., "Charging Schemes for Multiservice Networks", *Proceedings 15th International Teletraffic Conference,* Washington DC, June 1997

[Stil98] Stiller B et al., "Charging for Multimedia Flows in an Integrated Services Network", NOSSDAV 1998, Cambridge UK.

[Ston01] Stone, G.N., Lundy B., and Xie G.G, " Network Policy Languages: A Survey and a New Approach ", *IEEE Network*, Jan/Feb 2001

[Stras98] Strassner J. And Elleson E., "Terminology for Describing Network Policy and Services" internet draft draft-strassner-policy-terms-01.txt

[Stras03] Strassner J., "*Policy Based Network Management- Solutions for the Next Generation*", Morgan-Kaufmann, Amsterdam 2003

[Sund99] Sundaresan A. And Dhandapani G., "DiffSpec – A Differentiated Services tool", ITTC, Dept. of EE & CS, University of Kansas. 1999

[Sur01] Sur A, "Technical Tutorial: Mediation Device Requirements for GPRS", *Billing World*, vol. 7 no. 6 June 2001, pp 100-107

[Teit99] Teitelbaum B. et al., "Internet2 QBone: Building a Testbed for Differentiated Services", *IEEE Network*, Sept./Oct. 1999

[Trav00] Travostino F., "Towards an Active IP Accounting Infrastructure",., *Proc. 3rd IEEE Conference on Open Architectures and Network Programming, OPENARCH 2000*, Tel Aviv, Israel, March 2000

[Vari99] Varian H.R, *Intermediate Microeconomics – A Modern Approach* , fifth ed. , W.W. Norton, New York, London

[Vec98] De Veciana G. and Baldick R., "Resource Allocation in multi-service networks via pricing: statistical multiplexing", *Computer Networks and ISDN Systems* vol. 30 1998

[Weth98] Wetherall D. et al., "Introducing New Internet Services: Why and How", IEEE *Network* May 1998

[Wies94] Wies R , "Policies in Network and System Management – Formal Definition and Structure" .*Journal of Network and System Management*, 2(1):63-83, 1994

[Yem91] Yemini Y. et al., "Network Management by Delegation", *Proc. 2nd Intl. Symp. Int. Net. Man*. Washington D.C April 1991