Programmable Accounting Management for Virtual Private Networks *

A. Evlogimenou
University of Toronto
annae@cs.toronto.edu

R. Boutaba
University of Waterloo
rboutaba@bbcr.uwaterloo.ca

Abstract

Until now, proposals for IP accounting management have not exploited the power of programmable networks. Thus, real-time charging and pricing schemes are still regarded as impractical, because the accounting management tasks are executed only at the edges. In this article, we introduce the notion of programmable accounting management which is more flexible, efficient and scalable. Specifically, we propose real-time charging and pricing mechanisms using programmable networks. Moreover, we propose a novel programmable accounting architecture for QoS-enabled Virtual Private Networks (VPNs). This architecture gives the flexibility to the providers to employ real-time accounting, and offers an open accounting interface inside the network nodes. This entails that also the VPN subscribers can execute accounting tasks inside the network nodes. For example, they can keep their providers under surveillance, or they can apply their own accounting policies for their users.

Keywords

Accounting management, programmable networks, virtual private networks, real-time charging and pricing, metering, Quality of Services (QoSs), Differentiated Services (DiffServ), auctioning

1. Introduction

The Internet provides a best-effort service. However, the demand for deploying QoSs implies also the need for accounting management in the Internet. QoSs and accounting management encourage the users to choose the service that is most appropriate for them, thereby discouraging over-allocation and abuse of network resources.

In the literature, the terminology used for Internet accounting is quite confusing [1]. In this article, the term *accounting management* is used to denote the four processes: *metering*, *charging*, *pricing*, and *accounting applications*.

*This work was supported by a research grant from the Canadian Institute for Telecommunications Research (CITR)

The first process of accounting management (*metering*) involves monitoring and measuring the network resources. Metering granularity varies depending on the network layer at which the meter is employed. An issue of traditional meter design is that it does not consider processing traffic data inside the node, but rather it transmits raw data to a central accounting server at the cost of network overhead. Metering is required only for usage-based charging schemes and not for flat-rate schemes. During the second process (*charging*), the tariffs and the accounting data from the first process, are combined to charge the users. Therefore, through this process the accounting records are transformed into monetary units for users per some period of time. Tariffs are determined by the *pricing* process. Setting prices is often treated as an optimization problem with several possible objective functions. We classify these objective functions into two categories: maximize the social welfare and maximize the producer surplus. At the final process, *accounting applications* such as billing, auditing, capacity planning and trend analysis are performed. Applications do not have the same security and reliability requirements. As a consequence, the goal of IETF [2] is to provide a set of tools that can be used to meet the requirements of each application, rather than designing a single accounting protocol and a set of security services that will meet all the needs.

Real-time accounting management is still an open issue. Several schemes have been proposed, but few of them have been implemented and tested in real environment. All of these schemes assume that they can be executed either at the end-points of the network (i.e, hosts), or at the accounting server. However, some input parameters of these models exist inside the network. Thus, they require feedback signals from the network nodes in order to perform their operations.

Following the philosophy of programmable networks, we can push the accounting management tasks in the core of the network. The benefits will be: (a) real-time metering, charging and pricing, (b) distributed management which increases scalability and performance, and (c) integration of new mechanisms for charging and pricing without the need to introduce new protocols.

Accounting management is particularly important at the service management layer. In this paper, emphasis lies on the virtual private network (VPN). A VPN provided over a public infrastructure, like the Internet, is both cheaper and more flexible compared to leased lines and to less extent to ATM or frame relay virtual circuits. A major concern of providing VPN services over the Internet is to segregate VPN's traffic from the other traffics which share the same network infrastructure. Nowadays, there are Internet protocols which can satisfy the need of data integrity and privacy. Another concern is that VPN subscribers require to control and manage "their" network resources, in order to run service specific control software. For example, VPN subscriber wants to deploy low-level monitoring of his traffic, or he wants to control his own routing and addressing mechanisms. Finally, VPN subscribers require service guarantees. The research area of Programmable Virtual Private Networks investigates the above issues.

Figure 1 illustrates a VPN subscriber consisting of Site 1 and 2, the Internet Service Providers (ISP 1, ISP 2, ..., ISP n) which provide VPN services, and the potential users/customers of the VPN subscriber. The arrows represent the flow of money, e.g., both users at Site 1 and 2 pay the VPN subscriber, then the VPN subscriber pays the ISP1 and ISP n, then ISP 1 pays the ISP 2, etc. The VPN subscribers can retail IP services to their users only if they are able to monitor and control their users' traffic, and apply their own accounting management schemes, for charging their users. In addition, a VPN subscriber has to deal with the heterogeneous policies (charging and pricing) adopted by his ISPs. In order to handle this heterogeneity, the VPN subscriber has to coordinate his resource requirement among the ISPs, and also he may have to map the services provided by the one ISP into the services provided by other ISPs. This entails the need of developing an open accounting interface to allow subscribers to enforce their accounting policies in critical network nodes.

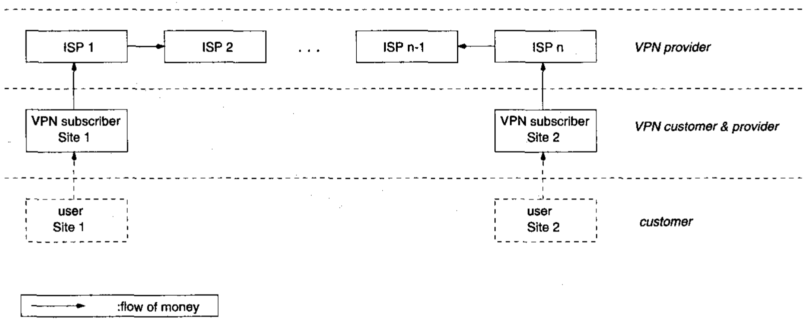

Figure 1: Accounting management for VPN: an example scenario

Although it is emphasized that accounting management is of great importance in programmable VPNs, no previous work has tackled this problem. In addition, to our knowledge, the only work that used the programmable networks for processing the metering data at the points where they are derived from (i.e., network nodes) is [3]. However, in this article we do not only process the metering data inside the network nodes, but we also execute the pricing and charging tasks in these nodes.

In this article, (1) we examine how the programmable network can enhance accounting management processes, and then (2) we propose a programmable accounting management architecture for VPNs considering the accounting management processes of metering, charging, pricing and billing. A detailed version of this work can be found in [4].

The article starts with Section 2, which discusses how programmable networks can enhance charging and pricing models. In Section 3, we present our accounting management architecture. Finally, Section 4 describes related works and Section 5 concludes this article.

2. How charging and pricing can be enhanced by programmable networks

We propose here a new charging and pricing mechanism which executes inside the network nodes. In this way, common problems can be overcome at the expense of requiring more CPU cycles. Before presenting our mechanism, let us explain what we mean by charging granularity.

Charging granularity is determined by the pricing policies. The pricing policies can be either dynamic or static [5]. If dynamic pricing is applied, then the prices fluctuate according to the network load, and the charging granularity is on per-packet basis. An example of dynamic pricing is auctioning, which has been advocated as an optimal pricing model. On the other hand, if pricing is static, the prices are independent of the network conditions. For instance, when the costs of SLAs are determined according to the time of the day, then the pricing is constant.

The major drawback of dynamic pricing is that it is considered impractical, because the prices change on per-packet basis. For every packet that comes into the network, the provider has to decide the price of transmitting it through the network. On the other hand, the disadvantage of static pricing is that it is unfair. Specifically, users who do not fully utilize their reserved resources are charged the same price as the users who utilize all their resources.

We believe that charging on packet granularity can be better supported with programmable networks. In the following paragraphs, we describe how this is possible. We do not consider charging on coarser granularity, since bargaining long-term SLAs is an application level job, and obviously it cannot be executed in the network nodes. Since we refer to programmable networks, each customer is able to execute a code at network nodes (see Figure 2). This code controls all customer's flows by selecting the quality of service and affecting the charging on per packet granularity. We will examine the behaviour of this basic idea in two cases: Differentiated Services and Auctioning mechanism. We will also discuss the benefits and the drawbacks of the proposed schemes.

In a differentiated services (DiffServ) IP network, in each border router of ISP, which is a programmable router, customers of this ISP run a code whose responsibility is to decide which class of service to buy for each packet. The input parameters of customer's code may be ISP's tariffs (which may change dynamically), maximum cost that customer can afford, the current network characteristics (bandwidth, delay, etc.) and the traffic characteristics

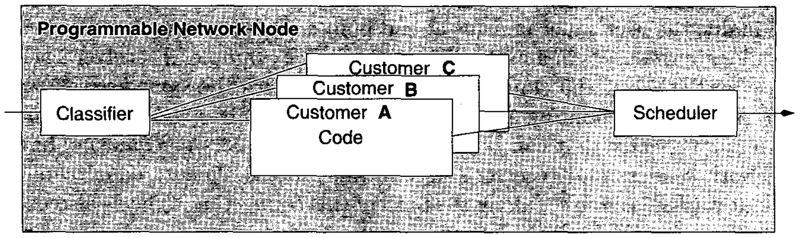

Figure 2: Customer's processes (i.e., charging mechanism) inside the network node

of customer's flows. Then the customer's process marks each packet with the class of service which he decided and finally forwards it to the scheduler of the node.

In addition, the problem of sender-pays or receiver-pays of differentiated services (introduced by [6] and [7]) can be solved. To be more specific, in DiffServ the sender marks the packets (i.e., selects the service of his flow), and also the sender pays his local ISP for his traffic. The problem with this scenario is that there are circumstances in which the receiver is the one who should pay for service, and so is the one who wants to select the required class of service. How can the receiver select the class of service? One solution in traditional networks is to utilize the control packets (like TCP acknowledgments packets) flowing in the reverse direction from the receiver. These packets can be enhanced to indicate the receiver's preference of class of service.

However, with our approach this problem can be solved without utilizing the control packets, and can be applied on top of the current Internet protocols. In more details, the code that executes in ingress network nodes is provided by the one who pays, i.e., either the sender or the receiver. If the receiver is the natural party who should pay, then he executes his code at the ingress network nodes of sender's ISP. The receiver's code is responsible to select the class of services of his traffic. On the other hand, if the sender is the natural party who should pay, then this is the one who executes a code at the network nodes.

The above pattern can also be extended to the multicast sessions where the flow somewhere inside the Internet cloud is replicated many times. There are cases where receivers want to choose the class of service of these flows, because those are the ones who pay. A scenario can be a media server which transmits a video stream to more than one receivers. One receiver may desire a poor quality of video, whereas another receiver may desire a high quality of video. In this case receivers should send their code to the network node where the replication takes place. Then, each receiver's code pays marks its flow, independently from the others.

Finally, the user's mechanisms at end points of a connection can cooperate and allocate resources in a more effective way. For example, if receiver's access link is bottlenecked and cannot forward the flow with the same arrival's rate, then receiver's accounting process can send explicitly a message with the maximum capacity that can be supported to the senders mechanism. In this way, both senders and receiver's accounting process allocate equal capacity without wasting resources along the path of this connection. This is also useful in DiffServ, in which class of services are provided in a connection-less way. Thus, when a provider in the core of network or the receiver's access link has insufficient capacity, either implicit or explicit congestion signals are received by the sender, although he has purchased enough capacity from his home-ISP. With the proposed scheme, along these congestion signals the receiver's mechanism can also inform the sender's mechanism of its free capacity. Then, sender may buy less capacity, equal to the amount of free capacity at the receiver's access link.

Our scheme it can also be applied to the auctioning mechanisms (called as smart market) proposed by economists Mackie-Mason and Varian [8, 9]. Specifically, the user associates a *bid* in each packet, which denotes the user's willingness to pay for transmission. The network node shorts all the bids; determines the threshold value, which is the marginal congestion cost; and finally the packets with bid greater than this marginal cost are transmitted and charged this cost. Roughly, all the transmitted packets are charged less than their bids, and consumer's surplus is always positive. Considering our scheme, user is able to submit bids to the auctioning system in the network node. More specifically, instead of executing the process which computes the bids at the end-points, we propose to execute it inside the network nodes.

Figure 3 depicts the smart market pricing method in traditional networks and in programmable networks. In the former case, users set their bids at the source point of their traffic, and at each congested hop these bids either win or lose. In our example, users A and B win and the user whose bid is 15 is defeated. Thus, users A and B transmit their packet with cost equal to 15 (i.e., equal to the highest defeated bid). In the proposed case, users set their bids in each congested hop. Note, that the bids offered by the users in the first case are greater than the bids offered by the users in the second case, since in the first case the users value the service from the origin to destination, whereas in the second case users value the service for accessing only one hop. In our case, the total cost of transmitting a packet from the source to the destination equals to the sum of marginal costs in each congested link. For example the winning users A and B transmit their packet with cost equal to $5 + 2 = 7$.

At the following paragraphs we discuss the benefits of our schemes. First, makes practical the auctioning mechanism. Auctioning [7] has been criticized as being impractical for many reasons. One problem is that the auctioning process takes place on a hop-by-hop manner; whereas the bids are set by the

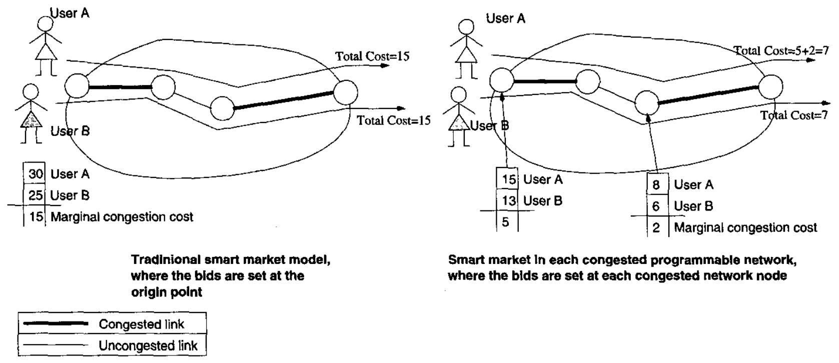
User A
User B
Total Cost=15
Total Cost=15
30 User A
25 User B
15 Marginal congestion cost
Tradinional smart market model, where the bids are set at the origin point
User A
User B
Total Cost=5+2=7
Total Cost=7
15 User A
13 User B
5
8 User A
6 User B
2 Marginal congestion cost
Smart market in each congested programmable network, where the bids are set at each congested network node
Congested link
Uncongested link

Figure 3: Traditional smart market scheme vs. Smart-market in programmable networks

customer only at the origin of the traffic. However, the bid submitted by the customer values the service from the origin to the destination, though it participates in an auctioning system for getting access in one hop and not in an auctioning system for getting access from the origin to destination point. The correct approach is that customer submits a bid for each hop along the path. How can this be implemented? An obvious solution is that the packet carries the bid of each hop which is not an efficient solution. Instead, we propose to exploit the programmable networks, where each user can execute a code in each congested network node. This code is responsible for submitting the bids to the auctioning system of this network node.

Second, it allows customers to select class of service on per packet granularity. In other words, customers can treat each packet differently, and independently on which flow it belongs to. Traditionally the classification was based only on static parameters, such as the fields of the packet header, whereas our mechanism allows customers to mark their packets according to dynamic parameters. For example, during periods of peak traffic, a customer buys a better class of service for the packets of a real-time application, than for the packets which belong to elastic applications. However, during the off-peak periods customer buys a lower quality class of service for both real-time applications and non real-time applications without experiencing any performance penalty.

Third, customers react more quickly to the network state (e.g., traffic load and tariffs), because they do not have to wait feedback in order to make their decisions. More specifically, they show their willingness to pay inside the network node and they do not have to encode it in a field of the packet header. Likewise, providers do not have to send pricing information or any other information which is necessary for the customers to select class of service for their traffic. In few words, everything operates where the parameters are being captured. This also entails, shorter delays which in turn means that the pricing information can be updated at finer time granularity.

The major disadvantage of the proposed scheme is the CPU overhead. Indeed the proposed scheme requires extra-processing inside the network compared to traditional accounting management schemes. However in the auctioning pricing mechanism, we expect negligible performance degradation, because the auctioning is applied only when there is congestion. Usually, congestion does not occur on each network node. This means that if there is a congestion in two network nodes along the path, the auctioning is executed at most twice along the path. A final limitation is that this mechanism is not suitable for mobile networks. The reason is that, if either the sender or the receiver is a roaming user, then the path from the sender to the receiver changes dynamically, in short time scales. This requires the customer's process to migrate from one network node to another network node (involved in the new path), which includes an additional overhead.

3. A generic architecture for Programmable Accounting Management of VPNs

In this section, we describe the components of a programmable Accounting Management Architecture for VPNs. We have adopted the MIBlet architecture proposed in [10]. Briefly, a MIBlet is a logical partition of the MIB which is executed in the network node. The VPN provider installs one MIBlet for each VPN subscriber. Through the MIBlets, we segregate the accounting information among the VPN customers. Furthermore, we provide an open accounting interface through which any customer can control and manage his resources. Finally, network nodes perform accounting tasks which traditionally were executed into centralized accounting servers.

The overall accounting management architecture in a single VPN is depicted in Figure 4. There are three major components which are involved in our architecture: *Accounting Management Controllers* (AMCs), *MIBlet Controllers*, and *Accounting Manager* (AM). The first two components are in the resource management layer and the last one is in the network management layer. In the Network Management Layer, the VPN provider manages and controls its resources through the MIB, while the customer manages and control his resources through the MIBlet. In the Resource Management Layer, there are the network nodes which are programmable.

3.1 Functional Blocks of Resource Management Layer

Accounting Management Controllers (AMCs) are the components which realize the accounting tasks inside the network nodes. They are installed at

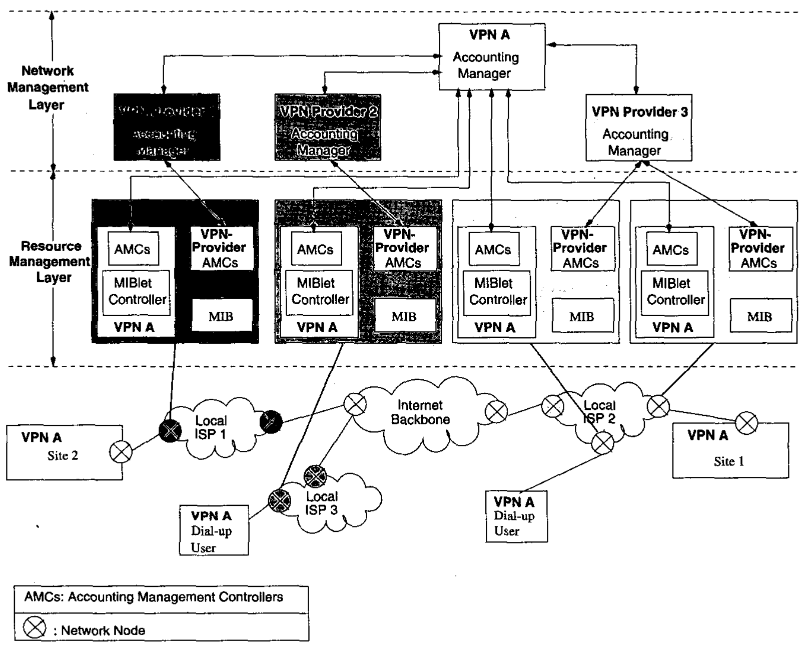

Figure 4: Overall Accounting Management architecture in a Customer VPN, called VPN A

specific network nodes. Figure 4 illustrates AMCs being executed at the border routers of local-ISPs of the two sites ('Site 1' and 'Site 2'). However AMCs can be executed either at the borders of local ISPs of the VPN or at any network node in the Internet core. This depends on how the VPN service is provided by the ISPs and also on the QoSs. If for example, DiffServ are supported, then accounting takes place only at the border network routers of local ISPs; whereas if auctioning is applied, then the accounting tasks are executed at the congested network nodes. Both the provider and the customer have their own AMCs.

The Accounting Management Controllers can be classified according to their functionality:

- *Charging and Pricing Controllers*: These controllers are executed when charging granularity is in terms of packets. They realize the mechanism we have proposed in Section 2. Some of their parameters can be received by the AM and some other parameters may be maintained in the MIBlet controllers (such as the current tariffs of the class of services).
- *Accounting Group Controllers*: These controllers are mainly responsible

for (1) processing of accounting data (such as compressing the accounting data before transmitting them to the AM), (2) configuring the Accounting Group in the MIB, (3) updating the MIB. It is this controller that configures the access of the MIB objects. A MIBlet Controller of a customer can access only the rows that refer to its sessions. That means that different MIBlet Controllers are permitted to read different rows/tables of MIB.

MIBlet Controller not only has access to specific rows of the MIB, but also can maintain its own MIB objects. For example, a customer can request a basic accounting summary from his provider, and maintain his own MIB data. Thus, customer can configure the Accounting Group of his MIB, as desired. For example, he can classify his packets according to which applications they belong to, and also he can maintain some statistics about his traffic.

In the context of hierarchical VPNs (see Figure 1), an advantage of the MIBlet controllers and the AMCs, is that the VPN customer can retail the network resources to its users, i.e., the VPN customer can act as a third-party provider. This is feasible, because the MIBlet controllers let this third-party provider maintain its own accounting group independently of the accounting group of VPN provider. Thus, this MIBlet holds accounting information of the users of this third-party provider. Moreover, the third-party provider can execute any charging, pricing and accounting mechanism, without being restricted by its provider.

In this scenario, the users are charged by the third-party provider, and they do not need to have specific knowledge of the charging and pricing schemes adopted by the local ISPs of their end-points, which most probably will not be the same (each ISP will adopt and will execute its own techniques). Instead, users have a general view of the Internet infrastructure, and they exchange pricing information, such as bargaining SLAs, only with the third-party provider. In the contrary, third-party provider has to deal with the heterogeneous charging schemes of several ISPs, and has to map these schemes into a uniform accounting system for its users. The AMCs realize this mapping. For example, if a user asks for premium service from his third-party provider, then the provider corresponds this premium service with the similar service supported by the local ISPs.

3.2 Functional Blocks of Network Management Layer

The Network Management Layer consists of the Accounting Manager (AM). The VPN Provider initializes one AM per customer. This gives the flexibility for provider of applying different policies between the customers. For example one customer could request real-time accounting, whereas another customer could request a monthly accounting. Each AM in the Provider interacts with the corresponding AM in the customer's side. The AM in Provider acts as a server and the AM in the Customer acts as a client. The accounting tasks in each AM are (see Figure 5): Billing, Pricing, maintaining an accounting database, accounting data aggregation.

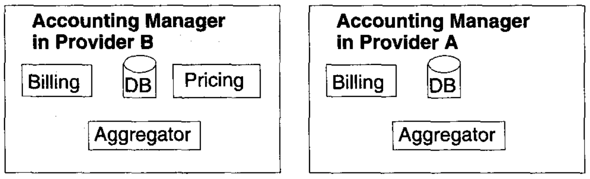
Accounting Manager in Provider B
Billing
DB
Pricing
Aggregator
Accounting Manager in Provider A
Billing
DB
Aggregator

Figure 5: Functional Blocks in the Accounting Manager. Figure 5.a (left) Pricing/charging performed at the network management layer. Figure 5.b (right) Pricing/charging performed at the resource management layer

The *Aggregator* collects the accounting data of a specific customer from many routers. Note that there can be more than one network node which maintain accounting data for a specific customer. For example, if a local ISP provides DiffServs, then there are at least two network nodes which meter the traffic of a customer. Accounting data are aggregated at the Network Management Layer and are stored at the accounting database.

The accounting records in the *accounting database* can be very detailed (e.g., the resources consumed by each application of the customer) or can be coarsed (e.g., the resources consumed by each site of the customer). Both customer and provider are the ones who select how much detailed the accounting base is. It is similar to the accounting summaries we get from the telephone companies. The basic fields of an accounting record are: (1) the amount of the resource that was consumed, (2) the period that was consumed, (3) in which network node it was consumed, (4) the price per unit of the resource that was consumed.

In case a provider applies non real-time charging and pricing processes, these processes are deployed in the Network Management Layer. In this case, there is also a *Pricing* task at the Network Management Layer (see Figure 5.a). On the other hand, if real-time processes are executed then these are executed in the Resource Management Layer (see Figure 5.b) as it will be explained later.

Finally, the task which interacts with the customer is the *Billing*. Provider sells SLAs through this task, or sends the accounting status of customer at specific time periods.

At customer's side, the Billing process is just a client application, which displays the account summary. Specifically, interaction between the customer and provider is done through a web browser and Java applets. The customer downloads an applet from the provider that communicates with the web server of the provider, and displays the current status of customer's account.

3.3 Example application scenarios & advantages

In this subsection we show the flexibility and efficiency of our architecture by providing some examples:

- Customer wants to execute a "Smart Marker" in the network nodes of a provider. "Smart Marker" monitors the network load, and whenever there are free resources in the non real-time class of service, it marks real-time traffic as non real-time. Note, that customer will not experience any performance penalty, because the non real-time class of service is under-utilized. In this way, customer saves money, since the non real-time is cheaper than the real-time class of service.
- Customer wants to check whether there are errors in the bills or not. Therefore, customer meters his traffic using his own MIBlet controller, periodically. Then he computes his expenses and compares them with the bills received by the provider.
- A provider wants to charge the customers for the CPU cycles consumed by their controllers, in order to discourage them to execute heavyweight controllers. Thus, provider just downloads a controller to the network node, which monitors the CPU and maintains a MIB. This MIB contains a table with the CPU cycles that have been wasted by each customer. Finally, provider includes into the bill of each customer the cost of utilizing the CPU.
- A provider decides to make the accounting protocol more efficient by reducing the traffic overhead at the expenses of requiring more CPU cycles in the network node. Thus, he executes a controller which compresses the accounting data before transmitting them to the Accounting Managers.
- Consider an ISP that provides a VPN service and deploys the auctioning pricing mechanism. Each customer has a bidding controller which computes the bids for his packet. The VPN provider has an auction-based scheduler which sends requests to the bidding controllers whenever there is a congestion. Then these bidding controllers compute the bids and respond to the auction-based scheduler, which finally decides which packets will be transmitted considering the customers' bids.

4. Related Work

In this section we compare our architecture with other accounting architectures. In [3], the AIACE (Active IP Accounting Co-processor Environment) considers the active networks for accounting. It dynamically pushes the intelligence into network nodes and exploits the power of open and active network nodes for metering. Our architecture is fully compatible with the AIACE. However, we exploit the programmability of network nodes for charging and pricing mechanisms in addition to metering. Furthermore, we propose a specific accounting management architecture for programmable VPNs.

Briscoe *et.al.* in [11] proposes an architecture that follows the principle that pricing, charging, metering and accounting should be performed at the end-points. He introduces the term active tariffs, meaning that prices are estimated at the end-points according to the congestion signals from the network nodes, and also according to the tariffs of network nodes. In the literature, real-time pricing schemes which run at the end-points (such as [12]) have been proposed. As we have argued previously, the accounting management at the end points induces delays and traffic overhead, because the input parameters of accounting management tasks exist in the network core.

In [13], an architecture of QoS-enabled VPNs over the Internet is described. This work is a part of the CATI project funded by the Swiss National Science Foundation. IP Differentiated Services are supported by means of service brokers for communication between different domains as well as within domains. Customers agree on SLAs, which must be flexible enough to store a vast variety of payment schemes. These SLAs can be established dynamically. In order to provide QoS through the entire network, SLAs should be negotiated between customer and ISP, and also between ISPs across the communication path. The main differences with our architecture is that they do not deal with real-time pricing models and they do not consider the programmable networks for providing VPN services. Thus, their architecture is not as flexible and configurable as ours. Furthermore, they do not address the issue of setting the prices dynamically. Similar with our architecture, they have also adopted the edge-pricing model, in which the VPN customer is charged only by the local ISPs. Finally, they define specific SLAs, which can be also deployed in our architecture at the network management layer.

In the following paragraphs, we outline some architectures proposed for accounting in the Internet. These architectures are orthogonal to our architecture (i.e., policy-based vs. process-oriented), because open interfaces are not considered and specific policies are defined. Moreover, our architecture has been proposed in the context of programmable VPNs, whereas the following architectures have been proposed for providing general IP services. However, the Accounting Management Controllers can be deployed for charging IP services, and they are not restricted only to the VPN service.

In SUSIE [14] (an European ACTS project), a charging and accounting architecture for IntServ, DiffServ and ATM has been proposed. Charging schemes for ATM services are adapted and used for IntServ and DiffServ. Furthermore, the mapping of the services in the IntServ architecture into the ATM services has been examined from the charging perspective. Finally, the implementation of this architecture was based on the TINA framework. Several charging approaches have been implemented, but no result is reported on the effectiveness and fairness of these approaches.

To our knowledge, there are only two projects which investigated the efficiency of usage-based pricing through experiments. The Internet Demand Experiment (INDEX [15]) is a prototype alternative ISP model that offers differentiated-quality service on demand, with prices that reflect resource cost. The findings of INDEX project are that today's system of flat-rate pricing by ISPs is very inefficient, and also ISPs with differentiated quality service is technically feasible. Moreover, INDEX shows that customers pay less and suppliers increase their profits. Another implementation of network billing system was for the New Zealand Gateway [16], where users were charged by volume for their traffic, in order to cover the costs of this gateway. They have developed the NeTraMet [17].

In [18], a system for billing users instead of hosts for their traffic is described. This is accomplished by postponing the establishment of connections while the user is contacted, verifying in a secure way that he is prepared to pay. The results show that this billing system is feasible even for large campus networks. Furthermore, it is indicated that pricing schemes may be used to control network congestion.

Two different accounting and billing principles are discussed in [19]: (1) decentralized accounting where a backbone service provider charges its adjacent service providers and in turn these providers charge their adjacent providers etc. and (2) multiple service queues.

Another preliminary work is [20], in which the author introduces the principle of open accounting technology that can be used to build a proprietary billing scheme. This is achieved with private MIBs for each individual operator. The private MIBs proposed in this work, are similar to the MIBlet controllers used in our architecture. These MIBs also belong to the users of network node and they let users control their traffic characteristics and have access to their accounting information. Finally, he proposed the Temporal MIB which represents the past and the current accounting information. The main difference with our architecture is that this architecture did not consider programmable network nodes, and thus the private MIBs are as rigid as the traditional MIB.

5. Summary

In this article, we have addressed the issue of accounting management in the context of programmable networks (more information can be found in [4]). We have shown that programmable networks can enhance most of the processes involved in accounting management including metering, charging and pricing. These processes have been discussed in the scope of a generic accounting management architecture. They can be customized easily to develop a specific accounting management application.

Common problems of packet-level charging and pricing schemes can be overcome, if we use programmable networks. We execute all the processes into the network nodes instead at the end-points. Hence, no pricing policies are transmitted, and the traffic overhead is reduced. Furthermore, there is no delay, because the processes retrieve their parameters in the same machine where they are performed. Another important benefit is that these mechanisms can be executed more than one time along the path. Specifically, they can be executed where there is congestion, and users are able to select a class of service according to the network load in the network core.

Finally, we proposed an accounting management architecture for programmable VPNs. We exploit the MIBlets [10] in order to isolate the accounting information among the VPNs which share the same network resources. Another feature of our architecture is that the accounting management tasks can be performed in the network nodes.

References

[1] Aiko Pras, Bert-Jan van Beijnum, Ron Sprenkels, and Robert Parhonyi. Internet Accounting. *IEEE Communications Magazine*, May 2001.

[2] Berbard Aboba, Jari Arkko, and David Harrington. Introduction to Accounting Management. *RFC 2975*, October 2000.

[3] Franco Travostino. Towards an Active IP Accounting Infrastructure. *IEEE OPENARCH*, 2000.

[4] Anna Evlogimenou. Programmable Accounting Management for Virtual Private Networks. Master's thesis, University of Toronto, 2001.

[5] Luiz A. DaSilva. Pricing for QoS-Enabled Networks: A Survey. *IEEE Communications Survey and Tutorials*, Second Quarter 2000.

[6] David D. Clark. A Model for Cost Allocation and Pricing in the Internet. *MIT Workshop in Internet Economics*, 1995.

[7] S. Shenker, D. Clark, D. Estrin, and S. Herzog. Pricing in Computer Networks: Reshaping the Research Agenda. *ACM Computer Communication Review, Vol. 26, No. 2*, 1996.

[8] J. Mackie-Mason and Hal Varian. Pricing the Internet. *Public Access to the Internet, Brian Kahin and James Keller, eds. Cambridge, MA: MIT Press*, 1995.

[9] Jeffrey K. Mackie-Mason. A Smart-Market for Resource Reservation in a Multiple Quality of Service Information Network. 1997.

[10] Walfrey Ng, Andrew Do-Sung Jun, HungKei Keith Chow, Raouf Boutaba, and Alberto Leon-Garcia. MIBlets: A Practical Approach to Virtual Network Management. *Sixth IFIP/IEEE International Symposium on Integrated Network Management*, 1999.

[11] Bob Briscoe, Mike Rizzo, Jerome Tassel, and Konstantinos Damianakis. Lightweight Policing and Charging for Packet Networks. *IEEE OPENARCH*, 2000.

[12] P. Marbach. Differentiated Services Networks: Pricing and Software Agents. *Technical Report CSRG-422*, 2001.

[13] Manuel Gunter, Torsten Braun, and Ibrahim Khalil. An Architecture for Managing QoS-enabled VPNs over the Internet. *IEEE, Local Computer Networks*, 1999.

[14] SUSIE project (ACTS). , 1999. http://www.teltec.dcu.ie/susie/.

[15] Richard Edell and Pravin Varaiya. Providing Internet Access: What We Learn From INDEX. *IEEE, Network v13, n5*, Sept-Oct 1999.

[16] Nevil Brownlee. New Zealand Experiences with Network Traffic Charging. *MIT Workshop in Internet Economics*, 1995.

[17] Nevil Brownlee. NeTraMet Version 4.3 Now Available.

[18] Richard J. Edell, Nick McKeown, and Pravin P. Varaiya. Billing Users and Pricing for TCP. *IEEE Journal On Selected Areas In Communication*, 13(7), September 1995.

[19] Hans-Werner Braun, Kimberly C. Claffy, and George C. Plyzos. A framework for flow-based accounting on the Internet. *IEEE, International Conference on Information Engineering*, 1993.

[20] Theodore K. Apostolopoulos. Accounting management in Communications Networks: Concepts and Architecture. *IEEE, Computers and Communications*, 1997.