An Architecture for Managing QoS-enabled VPNs over the Internet

Manuel Günter, Torsten Braun, Ibrahim Khalil
Institute of Computer Science and Applied Mathematics, University of Berne
http://www.iam.unibe.ch/~rvs/
Email:[mguenter|braun|ibrahim]@iam.unibe.ch

Abstract

This paper describes an architecture for the management of QoS-enabled virtual private networks (VPNs) over the Internet. The architecture focuses on two important issues of VPNs: security and Quality-of-Service (QoS). The security achieved in VPNs is based on IPSec tunnels, while QoS can be supported by mechanisms as proposed by the Differentiated Services currently being defined by the IETF. We describe an architecture that is based on the concept of service brokers. These service brokers are used for communication between different domains (such as ISP and customer networks) as well as within domains. The architecture described in the paper is currently being implemented as part of the CATI project funded by the Swiss National Science Foundation (SNF).

1. Introduction

Virtual Private Networks are becoming a popular technology for interconnecting private subnetworks of a single logical institution over the Internet. The main purpose of a VPN is security, i.e. only a limited set of hosts shall get access to the VPN and the traffic of a VPN traversing the Internet shall not be visible to non-members of the VPN. VPNs seem therefore a promising approach to implement enterprise networks covering several distributed locations or branches of a company. Those networks are today often built with corporate network technology using leased lines in the wide area to interconnect the subnetworks of a company. These leased lines usually guarantee QoS. Since business users are accustomed to QoS and have high demands for QoS guarantees, IP-VPNs need to provide similar features in order to replace leased-line based corporate networks.

The basic technologies for secure VPNs and for QoS support are introduced in the second chapter of this paper. The third chapter describes our vision of a QoS-enabled VPN service over the Internet. It also discusses in detail the required components and their interactions of an appropriate architecture. Chapter 4 briefly describes implementation work based on the architecture, as performed in the CATI project [1][2][3]. Chapter 5 concludes the paper.

2. VPNs for the Internet (IP-VPN)

A VPN is a private network constructed within a public network infrastructure, such as the global Internet [4]. In this paper we focus on the case where the public network is the Internet. For this case the definition above is missing one key concept of VPNs namely tunneling. VPN solutions, more often than not, set up tunnels to treat the Internet as one hop between two friendly parties. The endpoints of a tunnel encapsulate a data packet into another one (possibly using a different protocol). Tunneling is a powerful technique used in many different areas, e.g. to route packets of one protocol through a network using another one or for mobile IP. The following VPN properties we want to focus on:

- The VPN uses the Internet as a public communication infrastructure.
- The VPN ensures privacy at the network layer. This means privacy is ensured per packet. The technical way to do so is by encapsulating IP packets into other IP packets (tunneling) and using cryptographic mechanisms to authenticate and encrypt the payload.
- VPNs partition the Internet by allowing the internal use of private addressing schemes (e.g. for Intranets).
- In contrast to current VPN implementations the VPN solution proposed in this paper will support quality-of-service (e.g. assured bandwidth), thus eliminating the only real disadvantage of VPNs compared to real private networks using leased lines. (see also 2.4)

2.1 VPN business scenarios

Most vendors of VPN solutions identified three usage scenarios that are all placed around a corporate Intranet that is securely connected to friendly „entities“ over the Internet. The scenarios differ in the entities connected: remote users, branch office networks or partners/suppliers. Note that only network layer VPNs are general enough to handle all three scenarios.

- **Remote access network.** A remote user at home or on the road needs access to his/her company's electronic resources. An ideal VPN enables the remote user to work as if (s)he was at a workstation in the office. Authentication, transparency and ease of use for the remote user are crucial factors for this scenario.
- **Branch office connection network.** Here, two or more trusted Intranets are interconnected. Usually, the Intranets are protected by firewalls which are the ideal location to deploy VPN software. Thus, the client workstations do not have to worry about the VPN and the network manager can be sure that all Internet traffic exchange between the two Intranets is secured. Its broad and transparent security and the relative simplicity makes this a popular VPN scenario.
- **Business partner/supplier networks.** This scenario (also called Extranet) represents the most recent trend for VPN usage and also the least mature field. Companies can grant their partners temporal and limited access to their Intranet. The wide availability of the Internet and its relatively small costs together with mature IP-VPN technology shall allow fully functional electronic business applications including initial contact of customer, sales negotiation, order fulfillment and on-going support. Furthermore, such a solution allows to automate the supply chain and facilitate collaborative projects with partners.

2.2 A VPN enabling technology: IPSec

IP Security (IPSec) [5] evolved from the IPv6 development of the IETF. It is an open architecture for IP-packet encryption and authentication, thus it is located in the network layer. IPSec adds additional headers/trailers to an IP packet and can encapsulate (tunnel) IP packets into new ones. There are three main functionalities of IPSec separated in three protocols. One is the authentication through an Authentication Header (AH), the other is the encryption through an Encapsulating Security Payload (ESP), and finally automated key management through the Internet Key Exchange (IKE) protocol. We will refer to these three mechanisms as IPSec protocols. IKE (formerly called ISAKMP/Oakley) is the most complex IPSec protocol and is still under discussion. Nevertheless, IPSec provides an architecture for key management, encryption, authentication and tunneling. Therefore, all of the previously defined VPN business scenarios can be implemented with IPSec.

2.3 Differentiated services for QoS support

Because of the absence of a scalable resource reservation mechanism for the Internet core networks, the IETF developed the Differentiated Services concept [6]. DiffServ is a light-weight and scalable QoS mechanism. A single byte (DS Code Point, formerly called TOS) in the IP header is used to code different per-hop behaviors (PHB) that an IP packet can experience. Inside of a network, all IP traffic using the same code point is called 'DiffServ behavior aggregate' and is treated the same way. Since there is only a handful of PHBs, the DiffServ architecture scales also to large core networks.

Up to now, the IETF proposed two PHBs. One is called Expedited Forwarding (EF) [7] and the other is called Assured Forwarding (AF)[8]. The EF service allows packets to be forwarded at a constant bit-rate. As long as a traffic source sends its traffic in-profile, the traffic is guaranteed to be delivered within a well-defined delay. Traffic is in-profile if the packet rate does not exceed a rate that was previously agreed upon with the service provider. For its constant delivery rate, and its predictable delay characteristics, the EF service is also referred to as 'virtual leased line' service.

The Assured Forwarding (AF) service is more flexible. It allows for the composition of a variety of different services. AF allows for a service which can handle bursts by using buffering mechanisms. It contains up to four (but at least two) service classes. Each service class is logically separated from each other, which must be ensured by buffering and scheduling mechanisms. The classes are ordered by priority; a different amount of resources is allocated for each class. AF traffic always stays within the same class, so that the packets of a flow using one AF class are not accidentally delivered out of order. The reclassification of AF traffic in case of congestion is done by the use of three different drop precedences. Packets with the lowest drop-precedence are discarded first. Out-of-profile packets from lower drop-precedences are reclassified to higher drop-precedence.

In order to enable end-to-end QoS properties, the DiffServ architecture needs not only the standardization of per-hop behaviors, but also administrative policy mechanisms. This holds especially true in a multi-ISP scenario. ***Service level agreements*** (SLA) are contracts negotiated between an ISP and its customers and also between peer ISPs. The SLA specifies a traffic profile, network behavior and payment/billing scenario. Note that the SLA can refer to aggregated flow identifiers such as address prefixes. The SLA can be negotiated in a number of ways, for example via a phone call, fax or preferably using bandwidth brokers (BBs). To deliver the user data and to provide QoS through the entire network such as the Internet that consists of multiple administrative domains, a SLA would be required not only between the customer and the ISP, but also between ISPs. Thus, each domain negotiates with its adjacent domain a bilateral SLA specifying the volume and the type of traffic. The agreements can be pre-negotiated and static or they can be established dynamically. The static agreements are defined with the initiation of the service and they change quite infrequently. The negotiation of static agreements is done by human interaction. In this model the network resources can be provisioned based on the expected negotiated traffic. Negotiation of dynamic agreements requires an automated protocol between the BBs.

Bandwidth Brokers (BBs) [9] are agents that are responsible for resource allocation and traffic control of an administrative domain as well as for maintaining bilateral agreements between neighbor domains. BBs have their own policy databases that specify which users can use the resources and how much they are allowed to use. Figure 1 illustrates an end-to-end communication using BBs. Between each adjacent network a bilateral agreement is established specifying the traffic profile the peer network can send/receive. The traffic profile includes a service class, the rate, the maximum burst size and the time period when the service is required.

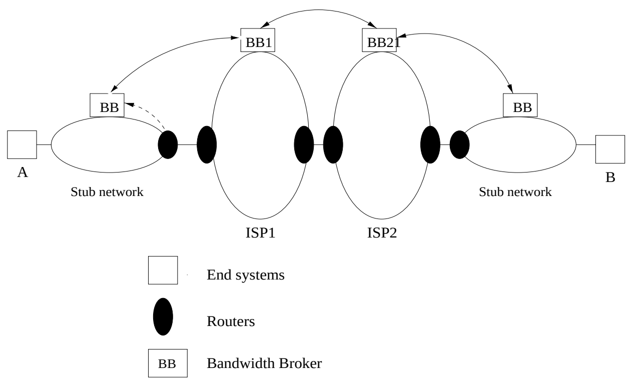

FIGURE 1. Networks with bandwidth brokers.

The edge router in the stub network processes individual signaling messages generated by the hosts (e.g., RSVP [10] or H.323 messages) or uses static policies and forwards the information to the local BB. In turn, the local BB sends the message with aggregate flow requirements to BB1. When BB1 receives the request, it first authenticates the requester and then verifies if the available resources to the next domain (e.g. ISP2) are sufficient to serve the request. If the request is approved, it will set up an appropriate traffic profile in the border routers. In turn, BB2 and the local BB of the stub network will exchange the messages in order to reserve the resources for the aggregate flow between the domains. An inter-domain message exchanged between BBs could contain aggregate flow information as described in [11] and solve several DiffServ problems such as dynamic SLA negotiation, and providing high service probabilities.

2.4 A new valuable service: QoS enabled IP-VPN

A major competitor of IP-VPNs are private leased lines (Frame Relay, ISDN). While the leased lines are more expensive because the user has to pay even when not using the line, they usually come with guaranteed QoS. Enhancing todays VPN solutions with QoS will eliminate the VPNs only real disadvantage compared to the leased line solutions. DiffServ seems to be the technology that is simple enough to enhance VPNs without restricting the global reachability of the Internet. Another problem of VPNs is the relative high expertise necessary for managing a VPN. Mismanagement leads either to loss of privacy or loss of connectivity.

An Internet Service Provider (ISP) has control over a part of the Internet. Using emerging technologies it can offer QoS guarantees. It has the expertise and the financial resources to build up a VPN service, as well. Furthermore, in order to serve its customers and to lower costs the ISP must try to automate as much of its services as possible. Currently, the most promising technologies for such a service bundle are DiffServ and IPSec. In order to ensure interoperability between these technologies, the tunnel endpoints and the ISP ingress nodes must be located in the same machines. Otherwise, the tunneling and encryption of the VPN service may hide information necessary for DiffServ. But given that an ISP provides the VPN and DiffServ together, it can place the tunnel endpoints accordingly. Although DiffServ and VPNs are two different services, they have similar concepts and can enhance each other:

- DiffServ can provide QoS commitments for a VPN as a whole or it can be used to differentiate the treatment of traffic classes within a VPN.
- VPNs are traffic aggregations with known traffic destination points. DiffServ also operates on traffic aggregations. The known destination points can furthermore ease the specification of necessary service level agreements.
- DiffServ and VPNs both need enhanced functionality of border routers of the ISP but not of intermediate routers. Both share some similar functionality in the border routers, e.g. the traffic classification.
- The simplicity and the coarse grained traffic classification make DiffServ a scalable technology. DiffServ is therefore suitable for the QoS support

between different ISPs. On the other hand, a VPN tunnel that crosses intermediate ISPs is transparent to them and therefore does not allow fine grained QoS support.

3. Vision of an Internet with configurable services

This chapter focuses on a higher level of abstraction. VPNs and QoS support are presented as one of many services an Internet service provider (ISP) could offer. In order to do so effectively, the ISPs needs to collaborate with each other and automate their interaction processes.

The Internet is an interconnection of multiple autonomous networks (domains). Some of these networks are driven by business companies which we will call Internet Service Providers (ISPs). The ISPs are eager to not only sell best effort transport services, but also to sell new and value added services. Such services can include QoS guarantees or privacy guarantees such as VPNs. The planning, accounting and the management of these services are further „meta“ services an ISP can provide. We can assume that the future Internet's ISPs have the following features:

- A network which is fully remotely controllable (configurable) by the ISP and whose hardware is able to support the new service.
- Intra domain resource management able to provide service guarantees. This can be a fine grained mechanism (e.g. per flow) if the domain is small.
- Application level interfaces for service requests by customers.

In order to produce the complete revenue of these investments, the ISPs need to collaborate. Today the collaboration is done by human network administrators communicating with each other by phone or fax. However, with automatically configurable networks and appropriate communication protocols, an automated approach is much more favorable. We envision the following requirements:

- Electronic bilateral service level agreements (SLAs) are used between adjacent ISPs.
- Also, there is an inter domain resource management procedure which allows the new services to span multiple ISPs. In order to be scalable to large backbone networks, this management must handle aggregations of the intra domain resource management entities; it must be coarse grained.

The architecture we present in the following shall allow the automatic provision of new services spanning multiple ISPs based on the mentioned assumptions. The architecture shall support several design goals as follows:

- The new services we will focus on are the support of QoS with differentiated services (DiffServ) and virtual private networks (VPNs). However, the architecture shall be service independent.
- The architecture shall not limit the ISP to a given business model.
- The architecture shall consist of generic components that are specialized for the given purpose.
- The architecture shall be open and interoperable in the sense that component interfaces are declared.
- The components interact with each other. The interactions that will be described in this document can be implemented using different existing/new protocols.
- The architecture shall allow different implementations.
- The architecture shall focus on security issues (robustness and prevention of abuse).

3.1 Basic components: service brokers

A service broker accepts service specifications for the specific service it sells. It can query the cost of the service and negotiate its price with a customer. Upon agreement, the broker can setup the service. The broker keeps the knowledge about the services provided in the form of service level agreements (SLA). It can thus accept requests to change the specifications of a service for a given customer. Note that the service broker may synchronize and coordinate multiple service requests. Furthermore, a service broker is an autonomous entity. It can proactively change an SLA. For such decisions, it needs access to various policy databases.

3.1.1 Broker classifications

Services can be classified as follows:

(1) (a) The service can be provided by an ISP alone or
    (b) it needs collaboration with other ISPs.
(2) (a) The service can be implemented by hardware-configuration orthogonally to other services or
    (b) must be coordinated with configuration of other services.

For the case of (2a) (orthogonal) we propose two kind of brokers. The Internal Service Broker (ISB) can be used to manage an ISP's service that is solely supported local to the ISP. The ISB can configure the ISP's network via secured connections to configuration daemons of each relevant network equipment (e.g. the border routers). The ISB manifests the fine grained resource management mentioned before. If the (orthogonal) service needs the collaboration of other ISPs we propose the External Service Broker component (ESB). The ESB has knowledge of the adjacent ISPs and can negotiate the necessary services of the adjacent ISP through its peer ESB broker. Therefore, an ESB can control the corresponding ISBs and thus the network configurations. The ESB manifests the coarse grained resource management mentioned before. An example for service brokers are bandwidth brokers (BB) for differentiated services. Clearly, this is an external service broker. It is often neglected how the BB interacts with its network. In our case the BB would not only interact with other domains BBs but also with an internal service broker that would for example map DiffServ to ATM configurations. Note that ESB and ISB represent a two class hierarchy which allows for a scalable solution. The separation matches the topology found in todays Internet. Figure 2 depicts the architecture for a single, orthogonal service established by two providers.

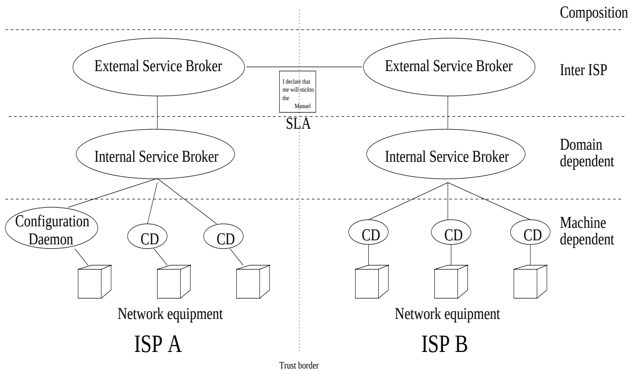

FIGURE 2. Broker hierarchy.

Class (2b) of services (non-orthogonal) must be handled by servers that are specially designed to offer a service combination. Such composite service servers (CoSS) can use and coordinate several ISBs and ESBs (of the different services influenced by this special service). Note that the management of such services is very complex. In general such a service can only be automated if the different ‘sub’-services only interfere in very few specific areas.

An example of such a service combination is the provision of VPNs with QoS guarantees with DiffServ. An example of such a service can be situated in the branch office scenario described before. The headquarter of a company is connected to its trusted ISP, the branch office to another ISP. The headquarter requests a VPN with QoS from its ISP using the QoS -VPN composite service server of the ISP. Figure 3 depicts the situation. The QoS-VPN server negotiates with the local ESBs that in their term negotiate with the next domain and so forth. Finally, the ISBs on the route between the headquarter and the branch office will configure their network to support the requested QoS, and the ISBs of the border ISPs will configure their border routers for the VPN tunnel.

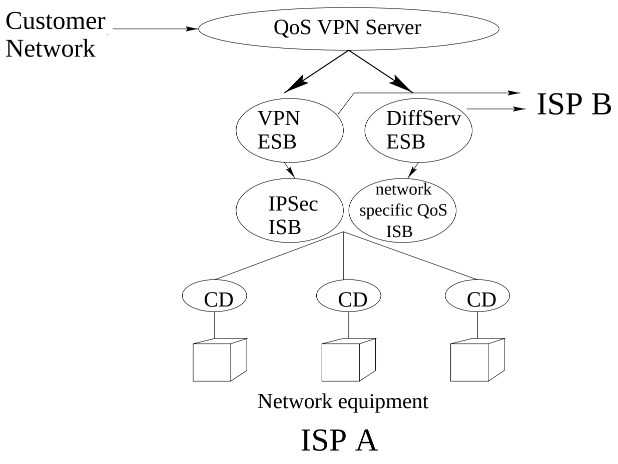

FIGURE 3. A QoS enhanced VPN service.

3.1.2 Customer service server

Conceptionally, there are only external service brokers interacting between two ISPs. At least in the case of a customer network it is desirable, that also a human can interact with a foreign service broker. We propose server that provides a graphical user interface and that translates between a human and an ESB. We will explain this in more detail in the next section.

3.2 Interaction of components

The interaction of components is implemented with communication protocols. Note that there is no complete protocol suite for this service broker architecture available today. We will describe the purpose of the different interactions between components by examples.

3.2.1 Configuration daemon - network equipment interaction

The configuration daemon (CD) is a configuration server specialized for a certain kind of network equipment such as commercial routers from a specific vendor. It accepts a machine independent, abstract configuration request and then starts to interact with the network equipment through a secured channel. It uses whatever authorization and configuration mechanism the network equipment requires to satisfy the configuration request. The CD notifies when its network equipment has installed the new configuration. It can keep a log of configurations done or even a complete backup configuration.

3.2.2 Configuration daemon - ISB interaction

The Internal Service Broker (ISB) accepts requests concerning a service provided by the ISP's network. It uses an abstract configuration language to send configuration requests to CDs and waits for the correct establishment of the service at different places in the network. The configuration requests must be authenticated by the ISB. The ISB is responsible to guarantee that no two service requests that the ISB is handling interfere with each other. This can be complicated if an ISB is implemented in a distributed manner.

3.2.3 ISB - ESB interaction

In order to handle external requests and agreeing on SLAs, an ESB must interact with corresponding ISBs. It must be able to query the state of current service configurations in the ISP's network. Furthermore, the ISB can signal changes to these configurations. The ESB must wait for ISBs to react, and coordinate their effort to provide given services before it can complete its negotiation with peer ESBs of adjacent ISPs.

3.2.4 ESB - ESB interaction

This is the most complex type of the interactions we are going to discuss. Furthermore, this interaction takes place between different business parties, thus it needs standardization. Like in the previously described interactions we need an authenticated and possibly private connection between the ESB peers. This is more complicated to establish since the peers are not in the same domain. Therefore, the peers do not trust each other. In the interaction between two brokers, one broker plays the role of a customer. The ESBs store and manipulate SLAs. This happens upon a service level negotiation between two ESBs. Along with the negotiation procedure the ESB-ESB protocol needs to include ways for payment. Note that the ESB also communicates with other entities such as a network administrator and various policy databases (e.g. for pricing, authentication, action triggering thresholds etc.).

3.2.5 Composite service server interactions

The composite service servers (CoSS) can be seen as an extension of ESBs. Their structure and interaction patterns very much depend on which services they combine.

3.2.6 Customer - customer service server

The customer service server (CSS) must be accessible by a well known protocol such as http. A customer contacting the CSS can choose a stub ESB for the desired service. With this graphically enhanced stub ESB the customer can negotiate a service level agreement. Note that the stub ESB has only reduced ESB functionalities, it cannot signal local ISBs or accepting another ESB as its customer.

3.2.7 An interaction example for provisioning a new service

A user learns about a service provided by an ISP through the WWW. She loads a GUI enhanced stub ESB (e.g. a Java Applet) which can invoke the appropriate ESB. The stub ESB lets the user negotiate with the ESB and visualizes the results. In general, the interaction takes place between the stub ESB (customer, controlled by a human) and the ISP's ESB (broker, automated) as follows:

1. Customer: Authenticated service request.
2. Broker: Answers a list of service forms. This list may include exemplars and partially prefilled forms.
3. Customer: chooses and requests a form.
4. Broker: sends the form.
5. Customer: sends back the form now containing the parameters of the service requested.
6. Broker: checks the cost of the service (may also turn the request down). Broker answers the cost of the request. Along with the message the broker may suggest a payment method.
7. Customer: accepts or rejects the price of the service. May also start renegotiation. Here, a variety of negotiation schemes can be implemented.
8. Customer: triggers the payment method and signals acknowledgment.
9. Broker: provides the service and acknowledges the customer upon establishment.

Note that the forms contain a unique identifier. This identifier must be kept by the customer in order to cancel the service contract later. The cancellation can be done with a single authenticated message to the broker.

This description of the interaction between components allows us to derive a more detailed picture of the components of the architecture.

3.3 Refinement of the components

3.3.1 External service broker

The External Service Broker (ESB, see Figure 4) component bundles the most functionalities of all presented components. It is controllable by the network administrator via a master interface and it controls ISBs via a slave interface. The peer interface must implement the ESB-ESB protocol including form exchange, negotiations and electronic payment. The current state of the service collaborations between adjacent ISPs is stored in an SLA repository. The ESB can be equipped with a quite complex autonomous behavior because it should be able to automatically detect necessary updates to SLAs. Furthermore it should be able to react to changes in the other ISPs and it can vary its behavior depending on the network state and the daytime. A central module of the ESB must coordinate its own activities, since there may be a lot of requests being processed at a given time.

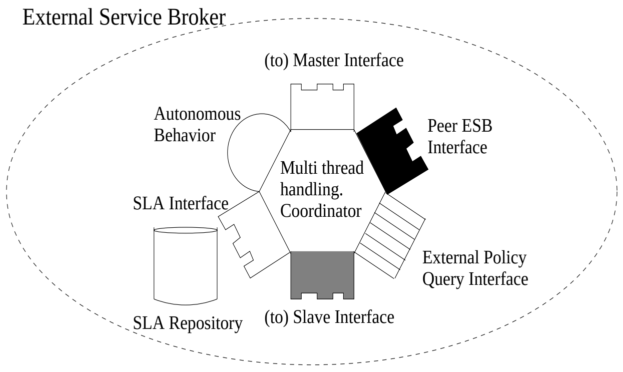

FIGURE 4. External service broker.

3.3.2 Internal service broker

The internal service broker's architecture is a simplified version of the one of the ESB (see Figure 5).The master interface includes an interface for the network administrator as well as for local ESBs. The internal service broker (ISB) coordinates the configuration of a service across its network. Therefore it needs an interface to control configuration daemons and a repository of the current configurations. An autonomous behavior module can be attached to an ISB that e.g. monitors a service relevant subset of the network equipment and triggers actions under certain conditions (e.g. alarms). The ISB needs coordination facilities as well.

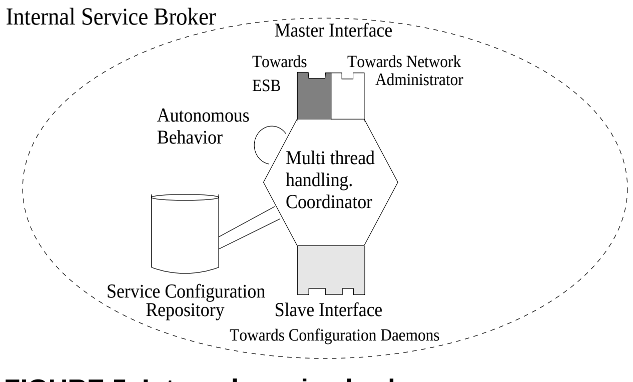
Internal Service Broker
Master Interface
Towards ESB
Towards Network Administrator
Autonomous Behavior
Multi thread handling. Coordinator
Service Configuration Repository
Slave Interface
Towards Configuration Daemons

FIGURE 5. Internal service broker.

3.3.3 Configuration daemon

The configuration daemon (see Figure 6) is controlled by its master interface (usually by an ISB). It can log its configuration actions and may use this log for roll-backs of configurations. Only a simple coordination module is necessary unless the underlying network equipment is hard to configure on-line (which is not the case for most of the commercial routers). The CD can support a limited autonomous behavior for monitoring and notification of the network equipments state. The CD includes several modules for the support of different services (e.g. a tunnel establishment module or a DiffServ classifier configuration module). The CD uses a machine dependent interface to configure the network equipment. If security would not be an issue here, the interface could use e.g. Telnet. Another possibility is to use the serial console port.

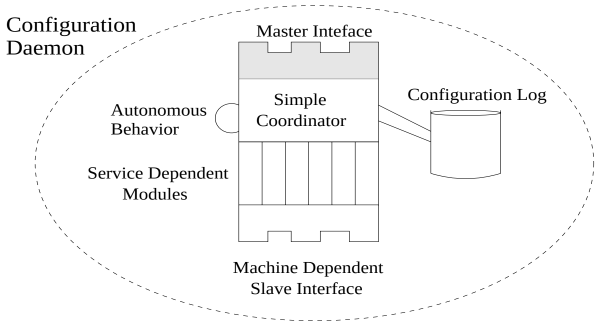

FIGURE 6. Configuration daemon.

3.3.4 Customer stub ESB

As mentioned before the customer stub ESB is delivered to a human customer who wants to configure a network service such as a VPN remotely. Therefore, the stub has to be reasonably small. However, it needs to include an ESB peer interface. Note that this will be a significant piece of code since this interface implements the customer part of the ESB-ESB negotiation protocol. However, policy and autonomous behavior modules of a normal ESB are not necessary since these functions are covered by the user. Additionally, the stub needs an interface towards a GUI. If the stub ESB is a Java applet, the GUI code will be provided by the web browser in use.

3.4 Refinement of the interactions

This section presents how the components exchange messages to accommodate service requests. The exact format of the messages is not specified in detail here. The messages are exchanged top down (ESB -> ISB -> CD) and bottom up (CD -> ISB -> ESB). There are six message types: queries, requests, commitments, and cancellations go top down, replies and signals go bottom up.

3.4.1 Message types

- **Queries**: A broker requests information from a peer or subordinate component about the service that the query target provides.
- **Request**: A broker announces that it will order a given service. Upon a request, resources for the service will be reserved, but the service will not yet be established. This is the first part of a two phase commitment. A request number will be assigned by which the announced service request can be identified.
- **Commitment**: The broker requests the establishment of the service under the given conditions. Commitments are only accepted after a successful request. The commitment may carry the payment for the service. The commitment carries the request number.
- **Cancellation**: A broker cancels an established service or a service request.
- **Reply**: This message type carries the reply of a peer or subordinate component, e.g. the results of a query or the acknowledgment of a request.
- **Signal**: A subordinate component sends information updates that were triggered by a third party (e.g. a network congestion).

The following two subsections present the message exchanges between ISB and CD and an ESB - ESB message exchange. Note, that the ESB-ISB message exchange is not presented, because in one possible implementation, the ESB and ISB of a service is running in the same process of a secured machine. In that case the interaction reduces to procedure calls.

3.4.2 ISB - CD message exchange

At any time, the ISB can be queried through its master interface by an administrator or an ESB. The configuration of a service consists of two phases. In the request phase, the ISB takes care that the service to be established is consistent with the already installed services. When the request phase terminates successfully, the service can be established by a commitment message.

3.4.3 ESB- ESB message exchange

The ESB-ESB message exchange is structured like the ISB - CD seen before. It consists of queries and request - commitment pairs. Here, each such message is relayed to the local ISB as well as to other ESBs if other ISPs must help to provide the service. Furthermore, reply messages can carry cost information, commitment messages can carry payment. Figure 7 shows the simplified message flow for a commitment message. Note that cancellation messages and request messages follows the same pattern as the commitment message.

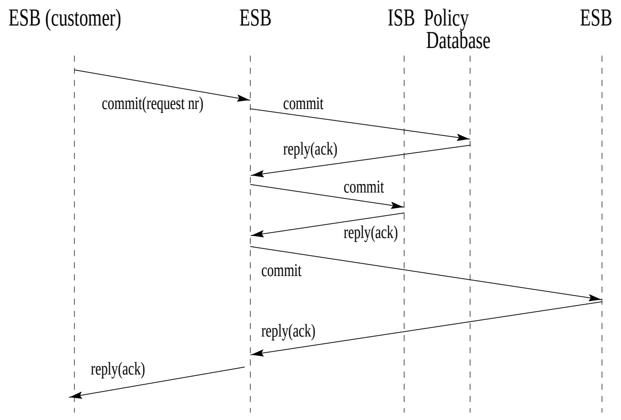

FIGURE 7. A successful service establishment.

As mentioned before, the composite service server (CoSS) is an extension of an ESB. The CoSS does not only communicate with adjacent ISP's ESBs but also with (at least two) local brokers. The interaction diagram for CoSS stays basically the same as for ESBs with the following difference: The Composite service server preprocesses (divides) queries, requests and commitments and relays them to the appropriate local brokers.

3.5 Charging and accounting of configurable services

Our architecture does not restrict the choice of a payment method or pricing model to a given solution. In this section we discuss the access points for ISP individual charging and accounting modules. From a high-level view revenue is generated as follows. A customer (administrator of a stub network) needs a configurable service (e.g. wants to set up a branch office VPN). The customer is willing to pay for that service. This will be the money source. The customer contacts his ISP on-line via the ESB. It negotiates the service and the payment (price specs including payment method). In general, the deployment of the service will cost the customer a one time fee. In order to propose a price for the service, the ESB needs to check if other ISPs are involved and ask their ESBs what they would charge. This leads to a query chain that involves all participating ISPs and results in the distribution of the revenue. Finally the customer agrees on a SLA. From now on, the customer pays a flat- or usage based fee as negotiated in the SLA. The revenue is shared by the involved ISPs as described implicitly in the SLAs between the ISPs. The following subsections describe the different charging aspects.

3.5.1 One time charging

As mentioned before, the ESB - ESB signaling may not be free of charge. The query messages can be served for free or for a small charge. The request and the commitment messages must contain a payment or a promise for a payment. Cancellation of services may be subject to charging if this is legal. An ISP in the role of a service seller must take into account, that it may have to use other ISPs services and pay for them, when it prices its service. Beside of that, it is free to define its pricing policy for ESB-ESB signaling. For example it might even reimburse costs to its customer when the customer decides to commit to the service. The ISP would then generate its revenues by continuous charging.

3.5.2 Continuous charging

The price continuously charged for the provided configurable service is specified in the SLA. Thus the SLA record structure must be flexible enough to store a vast variety of payment schemes. It contains a usage and time of usage based price specification. Attributes can be bandwidth, peak traffic, total traffic, daytime and many others. The payment method must also be specified. When an ISP classifies incoming traffic (e.g. for tunneling or DiffServ marking) it can also account it. Later, it can accumulate the continuous costs for a customer by using this accounting information and the appropriate SLA price specification.

3.5.3 Cost calculations

In the case of a customer in the stub (access) network, a human will probably decide whether a SLA is acceptable. On the ISP side, however, the ESB must be able to automatically handle most of the negotiations. It should contact a human administrator only in few special cases. This is one of the big challenges for ESBs. The ISP is supposed to have its pricing models and policies available for the ESB. Upon a service request the ESB will contact other ESBs and also check with the local ISBs how much work and cost the request generates. Based on that it should be able to calculate a competitive price specification for the service. Such an artificially intelligent behavior is out of the scope of the paper and subject of future research.

3.6 Security issues

The higher we climb in the component hierarchy the more critical is the security of a component. A corrupted CD can only directly impact one machine. Once the corruption is detected, the reestablishment of the correct service can be done relatively quickly. A corrupted ISB will affect the whole ISB network. A corrupted ESB will also directly affect neighbor ESBs. This case represents an extreme high threat potential, since the ESBs have the authority to handle payments automatically. The more 'intelligence' a component has the higher is the possible damage it can do in case of corruption. Malfunction of ESBs may only be detected in collaboration with other ISPs. It is difficult to reestablish correct service handling in that case.

While the interactions between components of the same ISP can be secured in a more statical manner with local key management, the ESB-ESB interaction needs to be secured using a trusted third party and/or public key cryptography. From what we said above we can draw the following conclusions:

- Not all components need the same level of protection. ESBs must be protected using the highest level of security available. The interactions could be secured with an encryption algorithm allowing for long keys (>128bits). The key material must be refreshed automatically and in short terms.
- ESB - ISP interactions need also strong protection mechanisms but shorter keys with frequent key refreshment can suffice.
- ISP - CD interactions need protection but shorter keys and a less frequent key refreshment can suffice.
- CD - network equipment interactions may be protected using a dedicated cable or application layer security such as secure shells (ssh).

3.6.1 Security of payment and pricing models

The trust model used here is the same as described in [12]. The ISPs do not trust each other. Therefore, as mentioned before, the proposed inter-component message exchange protocol must assure the integrity and authenticity of the messages exchanged. Then, customers can be held liable for the services they use. We did not specify a payment method, thus the security of the chosen payment method must be evaluated separately. The one time charge for the request messages between ESBs is a special case. This message must be charged (a small to medium amount). If the request message was for free, the architecture would allow a malicious customer to launch a denial-of-service attack. The attacker could send many requests and cancel them again. This would cause much overhead in the service providers networks since the networks would be prepared for the provision of services that are then not deployed and not payed for.

4. Implementation

We are currently implementing an instance of the generic broker architecture, namely for the management of a QoS VPN service. While the architecture distinguishes between external and internal brokers for VPN services and for QoS, our existing prototypical implementation realizes those components in one service broker. The functionality of this service broker is limited due to ongoing work. The main limitation concerns the ESB to ESB communication that is necessary in a multi provider environment. It is subject to current and future research. Thus the ESB to ESB interfaces do not exist as implementations at the moment. Combining some functionalities of external and internal brokers, the resulting internal broker looks as depicted in Figure 8.

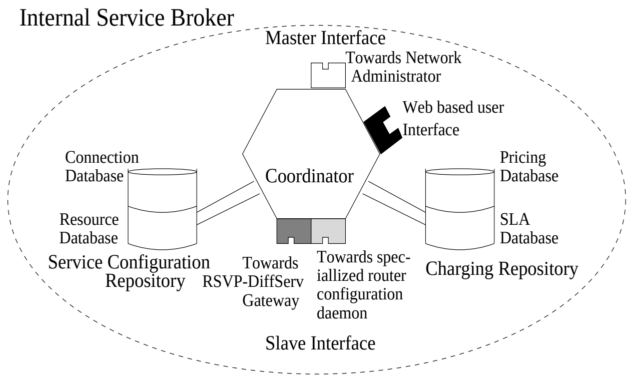

FIGURE 8. Realized ISB.

The broker allows a customer to configure DiffServ enabled IPSec tunnels on-line. There is a web based configuration interface for that purpose. Furthermore, the broker provides an interface for automated DiffServ reservations as sent by our RSVP-to-DiffServ gateway. The broker manages local network resources. QoS guarantees are ensured by CDs configuring the proprietary queuing system and shaping mechanism of our routers. A special version of the broker uses ATM QoS mechanisms. The broker calculates customer charges using SLAs.

5. Conclusion

This paper describes an architecture that allows to implement QoS-enabled VPNs. The architecture is based on an generalization of the bandwidth broker concept introduced in the DiffServ environment. The architectural framework includes a service broker hierarchy that allows for automated service configuration. An instantiation of the framework allows a user to set up, change, and modify VPNs including parameters such as security and QoS related parameters. This architecture forms the basis of the implementation of a demonstrator scenario currently being implemented.

6. Acknowledgments

The work described in this document is part of a deliverable to the „Charging and Accounting Technologies for the Internet“ (CATI) [1] project funded by the Swiss National Science Foundation. We like to thank all colleagues from the other CATI project partners for discussions about the presented architecture. This work has been funded by the Swiss National Science Foundation (SNF), project no. 5003-054559/1 and 5003-054560/1. The implementation platform has been funded by the SNF R'Equip project no. 2160-053299.98/1 and the foundation "Förderung der wissenschaftlichen Forschung an der Universität Bern".

7. References

[1] B. Stiller, T. Braun, M. Günter, and B. Plattner: *The CATI Project: Charging and Accounting Technology for the Internet;* ECMAST'99, Madrid, May 1999.
[2] T. Braun, B. Plattner, and B. Stiller: *The CATI Project: Charging and Accounting Technology for the Internet;* URL: http://www.tik.ee.ethz.ch/~cati/
[3] M. Günter et al.: *CATI WP2: VPN Configuration;* URL: http://www.iam.unibe.ch/~rvs/cati/
[4] P. Ferguson, G. Huston: *What is a VPN?*, The Internet Protocol Journal, Vol. 1, No. 1 and 2, 1998
[5] St. Kent, R. Atkinson: *Security Architecture for the Internet Protocol;* RFC 2401, November, 1998.
[6] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss: *An Architecture for Differentiated Services;* RFC 2475, December, 1998.
[7] V. Jacobson, K. Nichols, and K. Poduri: *An Expedited Forwarding PHB;* RFC 2598, June, 1999.
[8] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski: *Assured Forwarding PHB Group;* RFC 2597, June, 1999.
[9] F. Reichmeyer, L. Ong, A. Terzis, L. Zhang and R. Yavatka: A Two-Tier Resource Management Model for Differentiated Services Networks*;* IETF Draft, work in progress, November, 1998.
[10] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin: *Resource Reservation Protocol (RSVP);* RFC 2205, September, 1997.
[11] Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L. Zhang, K. Nichols and M. Speer: *A Framework for Use of RSVP with DiffServ Networks;* IETF Draft, work in progress, November, 1997.
[12] N. Weiler, G. Joller, B. Stiller, G. Fankhauser: *Trust Model*, CATI project deliverable, URL: http://www.tik.ee.ethz.ch/~cati/deliv/CATI-TIK-DN-P-010-1.1.pdf, March 20, 1999.