Object-oriented design of a VPN bandwidth management system

T. Saydam[a], *J.-P. Gaspoz*[b], *P.-A. Etique*[b], *J.-P. Hubaux*[b]

[a] *University of Delaware, Newark, DE. 19716, USA, tel. (1) 302 831 27 16, fax (1) 302 831 84 58, e-mail: saydam@cis.udel.edu*

[b] *Swiss Federal Institute of Technology (EPFL), Telecommunications Laboratory, 1015 Lausanne, Switzerland, tel. (41) 21 693 5258, fax (41) 21 693 2683, e-mail: gaspoz@tcom.epfl.ch*

Abstract

This paper describes the application of a general purpose object-oriented software engineering method to the design of a bandwidth management system for ATM-based virtual private networks (VPNs). Such a system allows a VPN customer to dynamically modify the bandwidth allocated to VPN connections. The design process has focused on the service management information model and interfaces required to provide that service to the customer. Object interaction graphs have been designed and class descriptions have been derived. Finally the VPN customer, value added service provider and network providers service management system interfaces have been designed and corresponding primitives are given.[1]

Keywords

VPN, ATM, TMN, object-oriented design, service management, bandwidth management

1 INTRODUCTION

One of the major trends in the evolution of current business information networking is an increasing need for high performance data communications, especially in the wide area. Provided as an alternative to dedicated leased lines networks, virtual private networks (VPNs) are gaining more and more acceptance among customers and network providers. VPNs permit to connect physically separated business sites without using dedicated resources.

The principal applications to be supported by future VPNs, that is, LAN interconnection and emerging multimedia applications, require the use of a flexible networking technology supporting a variety of services with very different quality of service requirements, in other

[1] Part of this work has been performed in the framework of the RACE project R2041 PRISM and thus has been funded by the 'Office Fédéral de l'Education et de la Science' (OFES, Switzerland)

words ATM (Asynchronous Transfer Mode). This paper will thus focus on ATM-based VPNs and more precisely on an open and very important issue in such an environment, namely bandwidth management. Indeed, multimedia applications have very different and often unpredictable bandwidth requirements which may vary over time. Moreover, ATM networks require, in general, resources to be reserved for each connection established over the network. Therefore, bandwidth management mechanisms would be very useful for the customer subscribing to the VPN service over ATM as a way to optimize resources usage and cost.

The main goal of this paper is to design a bandwidth management service, provided as an enhancement to the basic VPN service, and that allows the customer to dynamically modify the bandwidth allocated to VPN connections. A second generation object-oriented method called Fusion (Coleman, 1994) has been chosen for design purposes in order to provide a consistent approach, promoting reusability and scalability along the system design process. This design is based on the corresponding object-oriented analysis presented in (Gaspoz, 1994).

2 ATM-BASED VPN

A VPN allows to build a logical private network by using the physical public network infrastructure instead of dedicated network resources (e.g. leased lines). The service is offered as an extension and/or an alternative to a company's own network and aims at offering economic advantages as well as meeting ever changing customer needs and requirements.

ATM is a packet oriented transfer mode based on fixed length cells. It provides a non-hierarchical structure in which, cells belonging to different applications are transported commonly, independent of bit rate and burstiness. Multiplexing and switching may be performed at two levels: the virtual channel (VC) level and the virtual path (VP) level. As ATM is intrinsically a connection oriented service, communications between VPN users will be realized by Virtual Channel Connections (VCCs). This includes in general the allocation of the required resources on the user access and within the network.

The concept of virtual path allows the grouping of a set of virtual channels into a 'pipe'. VP cross-connects systems treat such bundled channels as an entity, regardless of the constituting virtual channels. In these systems virtual path connections (VPCs) are semi-permanently allocated between endpoints, thus allowing a simple and efficient management of network resources. When the cross-connected network handles connections between end nodes belonging to the same customer, it offers a virtual private network service.

The provision of VPN services over Virtual Path networks is mentioned several times in the literature (Wernik, 1992), (Verbeeck, 1992), (Gaspoz, 1994). Most of these papers refer to VPNs based on semi-permanent VPCs. In the same way, the broadband multimedia VPN considered in this paper is built by connecting each customer premise network (CPN) to every other, with the help of one or several semi-permanent virtual path connections, thus forming a logically fully meshed virtual private network, based on one or more physical networks.

3 VPN MANAGEMENT ARCHITECTURE

To support the provision of the bandwidth management service and more generally of VPN related customer network management services in an heterogeneous environment, an open and standardized management architecture based on the TMN layering principles has been considered (Figure 1). Figure 1 shows how different management systems interact with each other and with the underlying networks and network elements, to monitor and control the network resources, as well as to provide, enhance and offer network services.

According to (M.3010, 1992) the element management layer manages subsets of network elements on an individual basis and supports an abstraction of the functions provided by the network element layer. The network management layer has the responsibility for the management of all the network elements both individually and as a set. Service management is responsible for the implementation of the contractual aspects of services that are being provided to customers. Management services are provided to the customer in a client/server way. The VASP-SMS acts as a server with regards to the customer NMS (client) and as a client to the services provided by the network providers NMSs.

In the following chapters, the design efforts will focus on the management systems in the upper box, namely, the information model and the functionalities of the VASP-SMS as well as its interactions and interfaces with the CPN- and NP-NMSs in a bandwidth management perspective. To facilitate service layer information modeling, an abstract model of the VPN service under study has been established (Gaspoz, 1994). Some of its constitutive concepts are illustrated in Figure 1. For instance, a *virtual private line* is defined as a VPN end-to-end logical link connecting two CPNs and supporting the connections established between these CPNs. A *segment* is the part of a virtual private line belonging to one single management domain.

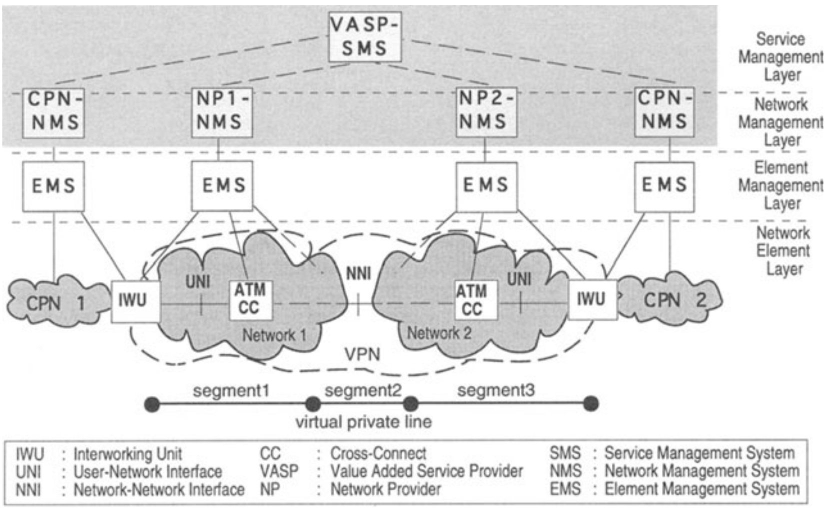


Figure 1 VPN Management Architecture.

4 VPN BANDWIDTH MANAGEMENT

4.1 Motivation

Our principal motivation in this paper is to specify and design a bandwidth management system to allow the end-users manage their bandwidth requirements. Bandwidth management plays a central role in ATM networks due to the great bandwidth access and transfer flexibility offered by this technology. From the network operator's point of view, this issue often refers to mechanisms used to protect the network against misbehaving users and avoid congestion.

Considered from the customer's point of view bandwidth management aims at optimizing bandwidth utilization. This is particularly true in an ATM context where resources have to be reserved for each connection to guarantee the required quality of service (QoS). A crucial issue in this context is to achieve the dual, yet often contradictory, goal of ensuring a high utilization of the reserved resources, while maintaining a sufficient QoS to the individual connections. The use of a bandwidth allocation scheme providing an optimal compromise between statistical multiplexing gain and loss rate is certainly of major importance in this respect. For this purpose, dynamic bandwidth management allows the user to specify the resources needed by a connection (VCC) as well as to renegotiate them during the lifetime of the connection.

4.2 Bandwidth allocation and enforcement

It results from the specification of the VPN and its related actors that bandwidth will be allocated and enforced at two different levels, the VPC level and the VCC level, in our example under the responsibility of the network providers and the service provider, respectively. Indeed, the network providers will sell VPC bandwidth to the service provider and will enforce that bandwidth to ensure the contract agreements and prevent network congestion. The service provider will in turn sell that bandwidth to the customer, but to ensure the QoS of the individual connections, bandwidth enforcement will have to be performed at the VCC level as well. Normally three traffic descriptors parameters are required for bandwidth allocation at that level, namely, peak rate, mean rate and maximum burst size.

5 OBJECT-ORIENTED BANDWIDTH MANAGEMENT DESIGN

The main focus of this study is to specify and primarily design the service management layer object classes required to provide a dynamic bandwidth management service to the customer. The interactions between the customer and the SMS (Service Management System) are only considered from a bandwidth management point of view. The object-oriented specification and design of the bandwidth management system follows the Fusion method (Coleman, 1994).

5.1 Requirements of bandwidth management

The bandwidth management system will allow the customer to monitor and dynamically control the bandwidth allocated to a VPN connection. In order for the service provider to satisfy most of the customer requests directly (i.e. without requiring from the network providers to update the virtual private line bandwidth, each time one of its connections is modified), as well as to limit the frequency of network resources reservation and release requests, some spare capacity is foreseen at the virtual private line level. Thus, when a connection is released or when the bandwidth of a connection is decreased, the amount of aggregate bandwidth that will actually be released will depend on the spare capacity available at that time.

The connection bandwidth may be increased directly if there is enough spare capacity on the virtual private line supporting that connection. If the spare capacity is smaller than the requested amount, the system transparently attempts to reserve more network resources for each segment composing the virtual private line. A request will thus be issued to each corresponding network provider to increase the bandwidth of the virtual path connection represented by each segment. The virtual private line will only be updated if all the reservation requests have been accepted.

5.2 Object model

The object model defines the static structure of the information in the system, i.e., the classes and their relationships specified to accomplish a certain task. The object model in Figure 2 is centered on a connection bandwidth update request by the customer. Each class is represented by a box with the name of the class at the top and the attributes in the lower part of the box. Relationships are represented as diamonds joined to the participating classes by arcs.

Aggregation is represented by nesting the component class into the box of the aggregate class. A number, a range, zero or more ('*'), one or more ('+') are allowed cardinality constraints.

As illustrated in Figure 2, both the *VirtualPrivateLine* and the *Connection* have a *VplBw* and a *ConnectionBw,* respectively. This 'has a' relationship is modeled as an aggregation representing a logical rather than a physical containment. For a complete treatment of object models and other specification details please refer to (Gaspoz, 1994).

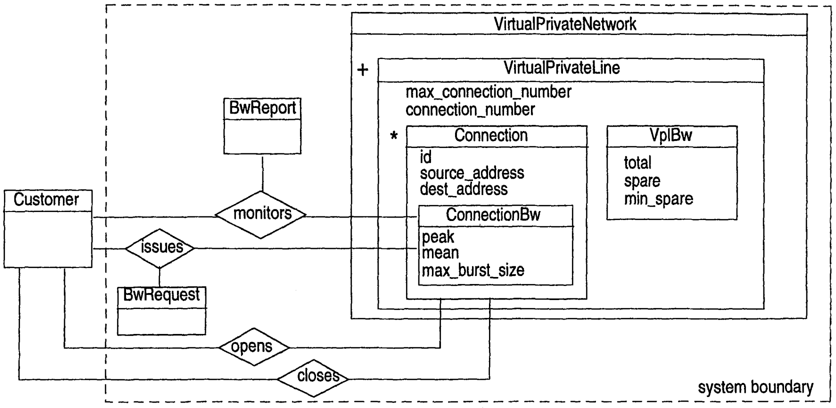


Figure 2 System Object Model for Bandwidth Update.

5.3 System interface

The object models take into account both the system under study and its environment. The next phase in the Fusion analysis process is to determine the boundary between the two, that is to say, the system interface. A useful technique for that purpose is to consider scenarios of usage.

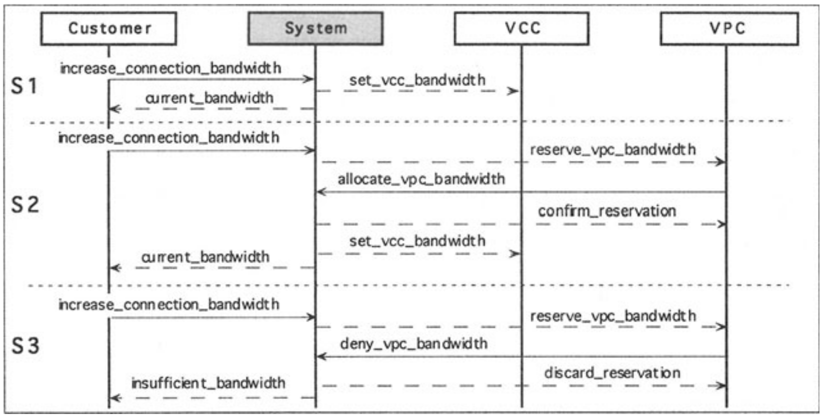


Figure 3 Scenario for Connection Bandwidth Allocation.

The Figure 3 shows a scenario for a connection bandwidth increase represented in timeline diagrams. This scenario considers three different alternatives involving three external agents. In the first one, S1, the system has enough spare capacity to satisfy the request of the customer. The other two alternatives deal with the case where the system tries to reserve additional resources from the network, either successfully (S2) or not (S3). Similar scenarios can be defined for bandwidth monitoring, bandwidth decrease, etc. One of the main benefits of these scenarios is that they allow to draw the boundary of the system, by considering the classes modeling the agents in these scenarios as external to the system (see Figure 2).

These scenarios may be generalized and formalized into life-cycle expressions, that is, regular expressions allowing to express sequences, repetition, alternatives as well as optionality and whose complete set constitutes the life-cycle model. This model specifies the allowable sequence of system operations (i.e. the input events and the effects they can have) and output events. The life-cycle model of the system under study has been developed in (Gaspoz, 1994).

Operation model

The operation model determines the system functionality as expected by the user. The behavior of each system operation is specified in a declarative way, in particular by using preconditions and postconditions. The preconditions express the conditions that must be satisfied whenever a system operation is invoked. The postconditions describe how the state of the system (i.e. the set of objects that participate in relationships as defined in the system object models) is changed by an operation and which events are sent to the agents. The operation model consists of a set of schemata, one for each system operation. The schema for the system operation 'increase_connection_bandwidth' is shown in Figure 4. The preconditions and postconditions are expressed in the 'Assumes' and 'Result' clauses, respectively.

Operation :	increase_connection_bandwidth
Description :	Request the connection bandwidth to be increased by a given amount
Reads :	supplied *peak_amount*, *mean_amount* : *BitRate* supplied *max_burst* : *BurstSize*, supplied *conn_id* : *ConnectionId*
Changes :	*Connection* with *connection.id* equal to *conn_id*, *connectionbw*, *vplbw* *new bwrequest*, *new bwreport*, *new reservation*

Sends :	virtual_channel_connection : {*set_vcc_bandwidth*}
	virtual_path_connection : {*reserve_vpc_bandwidth*}
	customer : {*current_bandwidth*}

Assumes :	conn_id is a valid connection identification
Result :	a *bwrequest* has been created and initialized with the supplied values, *peak_amount*, *mean_amount* and *max_burst* If (initial *vplbw.spare*) is sufficient to support the requested bandwidth increase then *vplbw.spare* has been decreased by a value computed from *peak_amount*, *mean_amount* and *max_burst* *set_vcc_bandwidth* has been sent to *virtual_channel_connection* *connectionbw.peak* has been increased by *peak_amount* *connectionbw.mean* has been increased by *mean_amount* *connectionbw.max_burst_size* has been set to *max_burst* a *bwreport* has been created and initialized with the final values of *connectionbw* *current_bandwidth(bwreport)* has been sent to *customer* Otherwise /* not enough bandwidth on the vpl *reserve_vpc_bandwidth* has been sent to *virtual_path_connection* a *reservation* has been created *reservation.status* has been set to *pending* *reservation.pending_responses* has been set to *virtualprivateline.nb_of_segments*

Figure 4 Operation Schema for 'increase_connection_bandwidth' system operation

The communication between the system and its environment is asynchronous, that is, the sender does not wait for the event to be received (Coleman, 1994). This assumption has a significant influence on the way system operations are specified, as, for instance, the response to an output event has to be described in a different operation schema. Moreover, behavior conditional on output events (e.g., the fact that each 'reserve_vpc_bw' should be followed by either 'allocate_bandwidth' or 'deny_bandwidth') is difficult to express in these schemata.

5.4 Designing object interaction graphs

All the models described so far are part of the Fusion analysis process. Once this step completed, the goal of the object-oriented design consists in defining how objects interact to provide the system functionality specified in the operation model. The main scope of the design phase is then to collect abstract definitions into concrete software structures, especially with respect to implementation and distribution of functionality. This distribution is captured in an object interaction graph. Each graph allows to define the sequences of messages exchanged between a set of objects to realize a given operation. The system software architecture starts then appearing as each system operation is designed. There is no unique way to design this functional distribution. Certain assumptions, design tradeoffs and choices as well as the larger system issues all influence the design process.

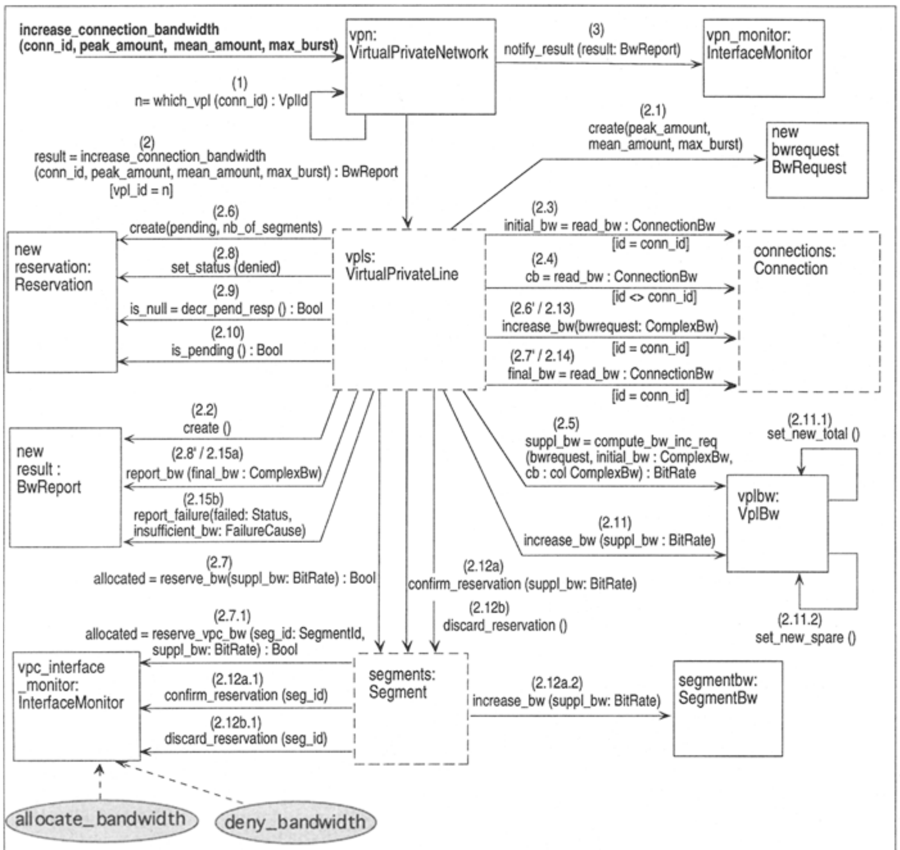
Object interactions are defined as procedural types of interactions. Indeed, when a message is sent to a server object, the corresponding method of its interface is invoked. This method is executed before the control is returned to the client. In other words, although the data flow may be bi-directional or unidirectional depending if a value is returned as the result of the method call, the control flow associated with such method calls is always bi-directional.

The Figure 5 shows the object interaction graph corresponding to the three system operations 'increase_connection_bandwidth', 'allocate_bandwidth' and ' deny_bandwidth'. Boxes and dashed boxes represent (design) objects and collections of objects, respectively. The arrows represent the invocation of the corresponding method on an object. A selection predicate (in square brackets) may be defined to send a message to one particular object in a collection. By default, the message is sent to all objects in the collection. Numbers define the sequencing of invocation. Method invocations labeled with the same sequence label occur in an unspecified order. Letters appended to a sequence number define alternatives.

The *vpn* has been selected as controller object, that is, the object which takes the responsibility for the given system operation. Its main role is to find out, among all the virtual private lines it contains, the *vpl* on which the given connection has been established. The central role played by the *vpl* in this design arises quite naturally from the data structures and relationships defined during the analysis. Indeed, according to the system object models specified previously (see Figure 3 and (Gaspoz, 1994)) a *vpl* object has relationships to both the active *connections* it contains and the *segments* that constitute it.

The decision as to whether the increase request may be satisfied directly or requires further resources from the network providers, is taken by the *vplbw*. For this purpose, this object has to perform a statistical computing taking into account not only the current request but also the bandwidth parameters of all the existing connections as well as the admissible loss probability. As the goal of this paper is not to elaborate on such issues, the method 'compute_bw_inc_req' is supposed to encompass this statistical computation and will not be developed further.

Careful readers have certainly noted that three operation schemata have been designed into one single object interaction graph, which is clearly not in-line with the approach recommended by Fusion. The reason why there is not a one-to-one mapping comes from the different ways objects are supposed to communicate within the system and with external agents. Indeed objects exchange messages within the system in a synchronous request/response style of interaction called sometimes interrogation (ODP, 1993). On the other hand, the system -and thus the objects that constitute it- communicates with its environment in a fire-and-forget style of interaction called announcement (ODP, 1993).


increase_connection_bandwidth
(conn_id, peak_amount, mean_amount, max_burst)
vpn: VirtualPrivateNetwork
(3) notify_result (result: BwReport)
vpn_monitor: InterfaceMonitor
(1) n= which_vpl (conn_id) : VplId
(2) result = increase_connection_bandwidth
(conn_id, peak_amount, mean_amount, max_burst) : BwReport
[vpl_id = n]
(2.1) create(peak_amount, mean_amount, max_burst)
new bwrequest BwRequest
(2.6) create(pending, nb_of_segments)
new reservation: Reservation
(2.8) set_status (denied)
(2.9) is_null = decr_pend_resp () : Bool
(2.10) is_pending () : Bool
vpls: VirtualPrivateLine
(2.3) initial_bw = read_bw : ConnectionBw
[id = conn_id]
(2.4) cb = read_bw : ConnectionBw
[id <> conn_id]
(2.6' / 2.13) increase_bw(bwrequest: ComplexBw)
[id = conn_id]
(2.7' / 2.14) final_bw = read_bw : ConnectionBw
[id = conn_id]
connections: Connection
(2.2) create ()
new result : BwReport
(2.8' / 2.15a) report_bw (final_bw : ComplexBw)
(2.15b) report_failure(failed: Status, insufficient_bw: FailureCause)
(2.5) suppl_bw = compute_bw_inc_req
(bwrequest, initial_bw : ComplexBw,
cb : col ComplexBw) : BitRate
(2.11.1) set_new_total ()
vplbw: VplBw
(2.11) increase_bw (suppl_bw : BitRate)
(2.11.2) set_new_spare ()
(2.7) allocated = reserve_bw(suppl_bw: BitRate) : Bool
(2.12a) confirm_reservation (suppl_bw: BitRate)
(2.12b) discard_reservation ()
(2.7.1) allocated = reserve_vpc_bw (seg_id: SegmentId,
suppl_bw: BitRate) : Bool
vpc_interface _monitor: InterfaceMonitor
(2.12a.1) confirm_reservation (seg_id)
(2.12b.1) discard_reservation (seg_id)
segments: Segment
(2.12a.2) increase_bw (suppl_bw: BitRate)
segmentbw: SegmentBw
allocate_bandwidth
deny_bandwidth

Description :

operation VirtualPrivateNetwork : increase_connection_bandwidth (conn_id : ConnectionId, peak_amount, mean_amount: BitRate, max_burst : BurstSize)

```
  checks that the connection exists by retrieving the virtual
      private line to which the given connection belongs            (1)
  If it does then
      get this vpl to increase the bw of the given connection       (2)
      notify the vpn monitor about the bandwidth report             (3)
```

method VirtualPrivateLine : increase_connection_bandwidth (conn_id : ConnectionId, peak_amount, mean_amount: BitRate, max_burst : BurstSize) : BwReport

```
  create a new bw request initialized with the supplied values      (2.1)
  create a bandwidth report                                         (2.2)
  retrieve the bandwidth of the given connection                    (2.3)
  retrieve the bw of the remaining connections on the vpl           (2.4)
  get the vpl bandwidth to compute the supplementary bw
      needed to fulfill the request, if any                         (2.5)
If the spare capacity on the vpl is sufficient then /* suppl_bw = 0
  get the connection to update its bandwidth with the
      values of the bandwidth request                               (2.6')
  get the new connection bandwidth                                  (2.7')
  initialise the bw report with the new connection bw               (2.8')
Otherwise
  create a new reservation initialized with status pending
      and with the number of segments composing the vpl             (2.6)
  reserve supplementary bw for each segment of the vpl              (2.7)
  If the reservation has been denied then
      set the reservation status to denied                          (2.8)
  check if all responses have been received                         (2.9)
  If they have then              /* pending_responses =0
      check if the reservation status is still set to pending       (2.10)
      If it is then
          update the vpl bw with the allocated amount               (2.11)
          get each segment to confirm the reservation               (2.12a)
          get the given connection to update its bandwidth          (2.13)
          get the new connection bandwidth                          (2.14)
          initialize the bw report with the new connection bw       (2.15a)
      Otherwise
          get each segment to discard the reservation               (2.12b)
          initialize the bw report with failure and cause           (2.15b)
```

Figure 5 Object Interaction Graph for 'increase_connection_bandwidth' operation.

To keep analysis and design consistent, as well as to preserve the semantics of the object interaction graphs, this duality has been maintained. The mapping between these two types of interactions has thus to be performed at the boundary of the system by the so-called *InterfaceMonitors*. These objects get thus a more active role than initially described in the Fusion method. Concretely, they have to map each interrogation invoked on their interface into an announcement to the corresponding agent. The asynchronous response to this announcement, if any, is on its turn converted back into the result part of the initial interrogation. For instance, the two system operations 'allocate_bandwidth' and 'deny_bandwidth' are encapsulated in the boolean result of the method 'reserve_vpc_bw' invoked at the *vpc_interface_monitor*. A special notation has been introduced in Figure 5 to illustrate this situation. Thanks to these mappings, the *Interface_monitors* hide to the system objects the announcement-based style of communication of the system with its environment. Consequently, objects may communicate transparently with other objects inside or outside the system in a consistent interrogation-based way.

The previously mentioned design choices are trade-offs between simplicity and efficiency. The choice of a sequential approach which, by waiting for the network providers responses, prevents the system to process a new customer's request before the previous one is completed -according on this point to the life-cycle developed in the analysis (Gaspoz, 1994)- is certainly not the most efficient. However, it offers great advantages with respect to error handling and concurrency issues, thus leading to a much simpler design. For instance, missing responses or error messages may be considered as implementation issues of the *Interface_monitors*, i.e., dealt with by some kind of transaction processing mechanism, and need not be considered further. In the same way, two consecutive customer's requests addressing the same virtual private line will not give rise to any conflict.

On the other hand, a good improvement that is consistent with the life-cycle, would be to allow parallelism to the bandwidth requests going to the different network providers. This issue is left for further study.

5.5 Designing visibility graphs

In the previous design phase, all objects were assumed to be mutually visible. The goal of this second design step is then to determine for each class which objects the instances of the class need to reference as well as the type of reference required (Coleman, 1994).

The visibility graph for the *VirtualPrivateLine* class is shown in Figure 6. All server objects -or collections of server objects (in dashed boxes)- whose lifetime is bound to that of the *virtualprivateline* client are shown inside the class box. A dashed arrow and a double border box denote a dynamic reference (e.g. *bwrequest*) and an exclusive reference (e.g. *segments*), respectively. Constant mutability (i.e. the reference is not reassignable after initialization) is explicitly shown by prefixing the server object with the keyword 'constant'.

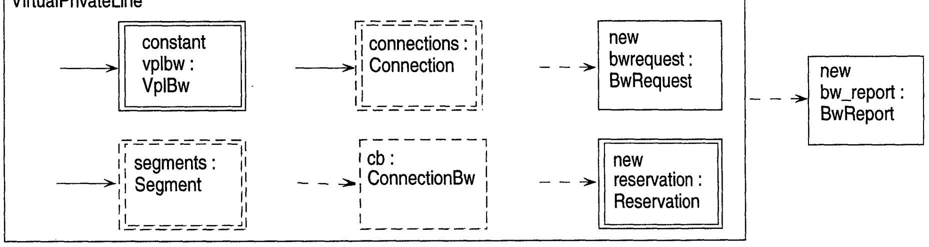


Figure 6 Visibility graph for the VirtualPrivateLine class.

5.6 Designing inheritance graphs

The inheritance considered in this document is a subtyping inheritance in the sense that objects of a subtype can only extend the properties of the supertype but not alter them. Unlike in programming languages, the focus is not on efficiency and code reuse but on simplicity of reasoning. A very useful consequence is that instances of a subclass may always be freely substituted for instances of a superclass.

A good starting point for deriving inheritance graphs is provided by the generalization and specialization relationships identified during the analysis. For instance, the classes *ConnectionBw, VplBw,* and *SegmentBw* have been identified as subclasses of an abstract class *Bandwidth.* During the design phase, it has been felt useful to partition the class *Bandwidth* into a *ComplexBw* (to deal with sustained rate allocation scheme) and a *SimpleBw* (for peak rate allocation). On the other hand, the two classes *BwReport* and *BwRequest* are used to encapsulate bandwidth related information. Using multiple inheritance, these classes may be defined as subtypes of the class *ComplexBw* and of the classes *Report* and *Request,* respectively. Thanks to this structure, it has been possible to reference objects of type *ComplexBw* and substitute them by instances of either of its subtypes (see Figure 5).

5.7 Deriving complete class descriptions

The individual design steps described in the previous sections are now all integrated into class descriptions. The complete class descriptions are final software design structures upon which implementation is based. They provide a specification of the class interface, i.e., the externally visible data attributes, object reference attributes and methods signatures, as well as of the inheritance relationship, if any. The description of the class *ConnectionBw* is presented below as an example.

```
class ConnectionBw is_a ComplexBw
 attribute peak : BitRate
 attribute mean : BitRate
 attribute max_burst_size : BurstSize
 method create ()
 method delete ()
 method add_bw (bw_chg : ComplexBw)
 method remove_bw (bw_chg : ComplexBw) : Bool
 method get_peak () : BitRate
 method get_mean () : BitRate
 method get_mbs () : BurstSize
endclass
```

5.8 Designing the system communication interfaces

The aim of this last step is to collect all the messages exchanged between the VASP-SMS and the NMSs that constitute its environment (see Figure 1). Mainly derived from the scenarios and the different object interaction graphs, this information allows to specify the different interfaces between these management systems in terms of the primitives exchanged (Figure 7).

On the CPN-NMS side, the interface specifies the management service offered to the customer in terms of the management functions he may invoke to perform bandwidth related management operations on his ATM-based VPN. The set of primitives that are part of the SMS - NP-NMS interface represent the service that the VASP has to request from the network providers NMS in order to provide the bandwidth management service to the customer.

A TMN conformant specification of these interfaces (X interfaces) would imply the mapping of these high level primitives into CMIP (Common Management Information Protocol) ones and would above all require a GDMO specification (X.722, 1991) of the information exchanged across the interfaces. These issues imply a level of design detail that goes beyond the scope of this paper and have not been considered further.

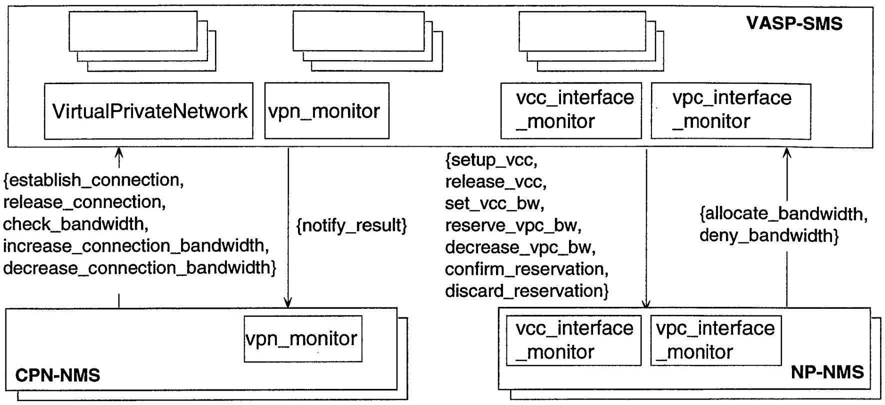

VASP-SMS
VirtualPrivateNetwork
vpn_monitor
vcc_interface _monitor
vpc_interface _monitor
{establish_connection, release_connection, check_bandwidth, increase_connection_bandwidth, decrease_connection_bandwidth}
{notify_result}
{setup_vcc, release_vcc, set_vcc_bw, reserve_vpc_bw, decrease_vpc_bw, confirm_reservation, discard_reservation}
{allocate_bandwidth, deny_bandwidth}
vpn_monitor
CPN-NMS
vcc_interface _monitor
vpc_interface _monitor
NP-NMS

Figure 7 SMS - NMSs communication interfaces.

6 CONCLUSION

Bandwidth management is of critical importance in ATM-based networks due to the great bandwidth flexibility it offers to end-users. This paper has described the software structures that need to be implemented in the VASP-SMS to support the provision of a bandwidth management service to customers. In addition, the corresponding service required from the underlying network providers' NMSs for that purpose has also been brought into light. However, even if the work has focused on VPN bandwidth management based on cross-connected ATM networks, the model developed at the service management layer is quite abstract and general enough to be applied to other service management cases.

Although the VPN management architecture considered is based on TMN principles, the modeling approach selected in this paper provides an interesting alternative to the TMN methodology where both a functional and an object-oriented approach coexist (M.3020, 1992). Indeed, management services fulfilling the customer requirements are decomposed into management service components and management functions according to a top-down functional decomposition. Conversely, the modeling of the managed system is object-oriented, namely, all network resources are modeled as managed objects. Therefore, the mapping between the management functions and the managed objects is far from being straightforward.

On the other hand, the Fusion method retained in this paper models the entire problem domain in a consistent object-oriented way. The functionality of the system as expected by the customer is defined quite formally in the operation model, thanks to the use of pre- and post-conditions. System operations specified in this model, which are in fact similar to TMN management functions in our example, are implemented in the design phase as a set of interacting objects. The mapping of the functionality expected by the user into the object model representing the system is then realized in a very consistent and straightforward way.

The problem domain addressed in this paper involves several actors and different systems that work in parallel and interact to constitute a distributed bandwidth management system. Although this study has focused on one specific part of this distributed system, namely the VASP-SMS, the functionality needed to provide the final service is clearly distributed in the different management systems. As a software engineering method that has been developed for sequential and centralized systems, Fusion is not very well-suited to deal with the specification and design of distributed systems. Issues such as the conflict between system internal communications based on interrogation and external communications based on announcement could be dealt with in a more elegant way by using a distributed systems conformant approach all along the development process. However, the integration of some of the models advocated by Fusion into the ODP viewpoints could be a very interesting topic of further study.

7 REFERENCES

Coleman , D. et al. (1994) Object-Oriented Development: The Fusion Method, Prentice Hall.

Gaspoz, J.P., Saydam, T. and Hubaux J.P. (1994) Object-Oriented Specification of a Bandwidth Management System for ATM-based Virtual Private Networks, *proceedings of the third ICCCN conference.*

M.3010 (1992) Principles for a Telecommunications Management Network, *CCITT Recommendation M.3010.*

M.3020 (1992) TMN Interface Specification Methodology, *CCITT Recommendation M.3020.*

ODP (1993) Basic Reference Model of Open Distributed Processing (ODP), parts 1-3, *ISO/ITU-T Draft Recommendations X.901, X.902, X.903.*

Rumbaugh, J. et al., (1991) Object-Oriented Modeling and Design, Prentice Hall.

X.722 (1991) Guidelines for the Definition of Managed Objects, *ITU Recommendation X.722.*

Verbeeck, P. et al., (1992) Introduction Strategies Towards B-ISDN for Business and Residential Subscribers Based on ATM, *IEEE JSAC*, December edition.

Wernik, M. et al. (1992) Traffic Management for B-ISDN Services, *IEEE Network*, November edition.

8 BIOGRAPHY

Tuncay Saydam has been a professor of computer science at the University of Delaware since 1979. He has received his graduate degrees at Istanbul Technical University and The University of Texas at Austin. His current research interests include network management, network interconnections and object-oriented software design. Member of IEEE, Sigma XI and New-York Academy of Sciences, Dr. Saydam is author of over fifty technical articles.

Jean-Paul Gaspoz graduated in electrical engineering at the Swiss Federal Institute of Technology in Lausanne. He then worked three years at Ascom, a Swiss telecom company where he contributed to the development of an ISDN PABX. He is currently doing a Ph.D. at the Swiss Federal Institute of Technology in Lausanne and his research interests include virtual private networks, services management and distributed systems specification.

Pierre-Alain Etique graduated in computer science at the Swiss Federal Institute of Technology in Zurich. He then joined Ascom, a Swiss telecom company were he worked 3 1/2 years on the development of a PBX. He is currently with the Swiss Federal Institute of Technology in Lausanne where he is working on his Ph.D.

Jean-Pierre Hubaux graduated in computer science at the Institute of Technology of Milan. He then joined Alcatel where he worked ten years as development engineer, consultant and project manager. He has been a professor at the Swiss Federal Institute of Technology of Lausanne since 1990.