# A Virtual Signalling Protocol for Transparently Embedding Advanced Traffic Control and Resource Management Functionality in ATM Core Networks

K. Katzourakis[1, 2], G. Kormentzas[1, 3], K. Kontovasilis[1], and C. Efstathiou[2]

[1] National Center for Scientific Research "DEMOKRITOS",
Institute for Informatics & Telecommunications,
GR-15310 AG. PARASKEVI, POB 60228, GREECE
{gkorm; kkont}@iit.demokritos.gr
[2] Technological Education Institute of Athens,
Dept. of Informatics,
GR-12210, Egaleo, GREECE
ckatzoura@cs.teiath.gr; cefsta@teiath.gr
[3] University of the Aegean,
Dept. of Information and Communication Systems Engineering,
GR-83200, Karlovassi, GREECE
gkorm@aegean.gr

**Abstract.** Following recent proposals for open and programmable network infrastructures, the paper discusses an out of band Virtual Signalling Protocol (VSP), which bypasses the conventional control information channels and supports externally defined traffic control and resource management functionality. The examination of VSP messages that run through the L and U interfaces (defined according to the IEEE P1520 reference model) constitutes one of the major objectives of the paper. The functionality of VSP is tested by means of an implemented prototype platform, which enables the portable deployment of advanced traffic control and resource management algorithms in heterogeneous ATM networks.

## 1 Introduction

Traffic control and resource management are important mechanisms for any Quality of Service (QoS) providing network environment and this importance will continue to grow towards broadband networking environments of the future, as more complex services and stringent and more diverse QoS demands will emerge. In this setting, signalling systems supporting advanced traffic control and resource management functionality have to play a key role towards the assurance of requested QoS and the simultaneous maximization of network availability and minimization of network operational costs.

The native ATM signalling protocols (e.g., ATM Forum UNI 3.1 [1], ATM Forum UNI 4.1 [2], ITU-T Q.2931 [3], ATM Forum PNNI 1.1 [4], etc.) support only internally defined traffic control and resource management functionality through in-band signalling channels. This fact contradicts the current trend of open signalling,

A. Marshall and N. Agoulmine: MMNS 2003, LNCS 2839, pp. 259–271, 2003.

which enables external functionality through out-of-band generic/abstract control channels. The main idea of OpenSig (Open Signalling) [5,6] concerns the use of programmable interfaces, which provide open control and management access to various network devices (e.g., ATM switches, IP and MPLS routers, etc.). According to OpenSig, the control/management and data paths are strictly separated and the defined open programmable interfaces can be seen as channels for flexible and transparent deployment of new advanced control and management algorithms.

Based on the OpenSig concept, various related works appear in the literature. Among them, we can discriminate: An attempt for standardisation of open programmable interfaces for control and management of various types of networks (e.g., ATM, SS7, IP, etc.) by the IEEE P1520 project [7], a virtual out-of-band signalling protocol by the Xbind project [8], and the Tempest open programmable framework [9]. For the implementation of such out-of-band open signalling protocols, a variety of architectures (e.g., FIPA [10], OMG MASIF [11], etc.), communication technologies (e.g., CORBA [12], DCOM [13], Java [14], etc.) and agent platforms (e.g., Grasshopper [15], Voyager [16], etc.) can be used.

Major issues for the selection of the appropriate open signalling protocol are the availability and the simplicity of the employed programmable interfaces, the maturity of the architectural framework that is going to be followed and the performance overheads, which enforces in the underlying network and the required implementation time [5]. Satisfying in an adequate level these issues, the paper presents a Virtual Signalling Protocol (VSP), which bypasses the conventional control information channels and supports externally defined traffic control and resource management functionality. The presented VSP builds upon earlier work [17]; in introducing constructs allowing the interaction of traffic control with resource management, towards increasing the network's potential for acceptance of calls in a dynamic way.

Besides the signalling protocol itself, the paper outlines a generic framework for the employment of several IEEE P1520 constructs, such as L and U interfaces, as well as NGSL (Network Generic Services Layer), in a standardised and systematic way for the purposes of device independent traffic control and resource management. The functionality of the VSP is tested through an implemented Java-based prototype.

The organization of the rest of the paper is as follows: Section 2 gives the overall architectural framework, which incorporates the proposed virtual signalling protocol. Section 3 presents the protocol, while Section 4 outlines the prototype implementation and comments on the protocol's performance. Finally, Section 6 concludes the paper.

## 2 Software Switch Extensions Enabling Virtual Signalling

The incorporation of an open signalling protocol supporting externally defined traffic control and resource management functionality into a heterogeneous networking platform requires out-of-band control paths/channels, which can bypass the ones established by fixed standards signalling protocols. Addressing this issue, Figure 1 presents a generic architectural framework, which enables enhanced traffic control and resource management functionality to be communicated between the nodes of a heterogeneous network. A detailed description of the framework may be found in [17]. The key-idea is that the network nodes export appropriate well defined programmable in-

terfaces to distributed software entities, thus providing a virtual environment for the deployment of traffic control and resource management functionality on top of the nodes. The deployed functionality is then exchanged between the software entities through an out-of-band signalling channel.

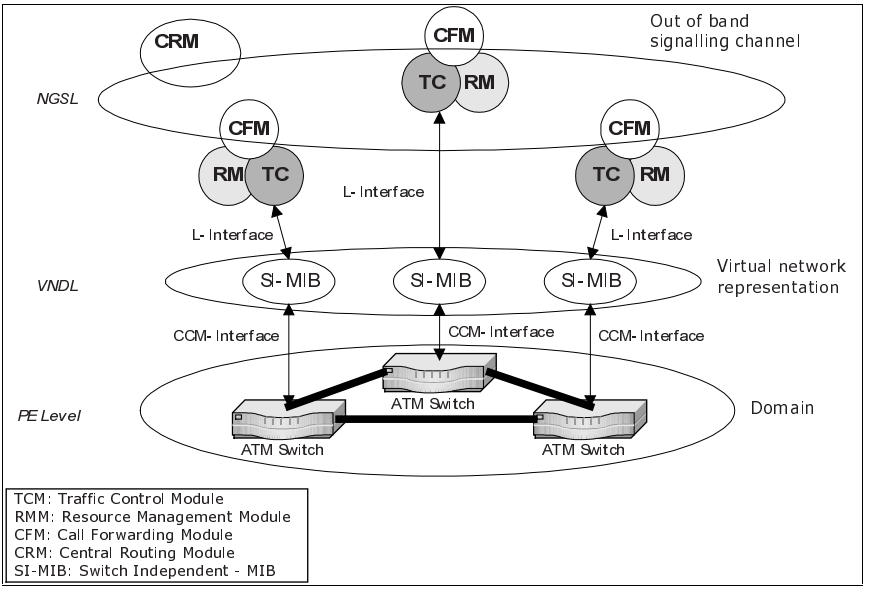


**Fig. 1.** Software switch extensions enabling virtual signalling

In terms of P1520, each node in Figure 1 exports a generic and switch-independent L interface, enabling the information flow between traffic control and resource management algorithmic components and the so-called Virtual Network Device Layers (VNDLs). A VNDL constitutes a virtual (software) representation of a node, containing all information relevant to the control and management algorithms operating on top of the node. In the presented architecture, the VNDL (i.e., the collection of appropriate managed/controlled objects) is built around a switch-independent MIB, called SI-MIB, which provides a portable virtual representation of the resources and traffic load conditions within the corresponding network node. For an in-depth description of the SI-MIB see [18]. The communication between SI-MIB and its corresponding underlying network node is based on a switch dependent CCM-interface.

The distributed software entities implementing the control and management algorithms (i.e., TCMs and RMMs) are uncoupled from equipment details by referring to (and manipulating) only switch-independent information contained in the SI-MIB. These entities belong to NGSL; their communication coordinated by CFMs through an out-of-band signalling channel. CFMs take instructions by a centralised (per domain-level) routing module, which is responsible for intra-domain routing in a heterogeneous ATM core network.

The following section presents an appropriate out-of-band Virtual Signalling Protocol (VSP) to support the control information exchange of just described architectural framework.

## 3 The Virtual Signalling System

As indicated in Figure 1, the exchange of control information between switches occurs at the virtual level out-of-band, i.e., outside the control paths/channels established between the inter-connected switches, bypassing ATM signalling. This section describes the exchange of VSP messages for handling a call request/call release.

The VSP's traffic control functionality concerns CAC, while the corresponding resource management functionality refers to bandwidth redistribution between Virtual Multiplexing Units (VMUs). A VMU corresponds to a group of Virtual Paths (VPs) sharing common resources (part of the output port's buffer space and link capacity) and serving a traffic mix that consists of connections with the same QoS requirements. Heterogeneous QoS may be supported by appropriately distributing traffic into different VMUs. In this context, the generic distributed software entities TCM and RMM of architectural framework of Figure 1 shift to CAC modules and Bandwidth Redistributor (BR) modules correspondingly.

### 3.1 Call Set-up

In the process of handling a new call request, one of the following two things may happen:

1. The CAC modules of all nodes across a path between the source and destination terminal accept the call request.
2. One of the involved CAC modules rejects the new call request. In that case, the currently checked route is generally rejected and an alternative (if it exists) is examined. However if the route under examination is the last possible one, instead of directly rejected, the offending CAC module triggers its corresponding BR to redistribute the bandwidth between VMUs to make room for placing the new call. If the redistribution is successful in all offending CAC modules, the call request is accepted, otherwise it is rejected.

(Note that for both cases, we consider whenever it is applicable that destination accepts the call request.)

For the first case (i.e., an accepted call request without bandwidth redistribution), the interactions between the end terminals (Source and Destination respectively) and the routing module, CAC modules, call forwarding modules and SI-MIBs that are placed on top of the switches are depicted in Figure 2. (The number associated with each message in Figure 2 indicates the message's relevant position in a chronological sequence.)

As shown in Figure 2, the source terminal sends a call request message (`call_req`) to the routing module. Parameters of `call_req` are the source and destination terminal identification strings (parameters `source_id` and `dest_id`

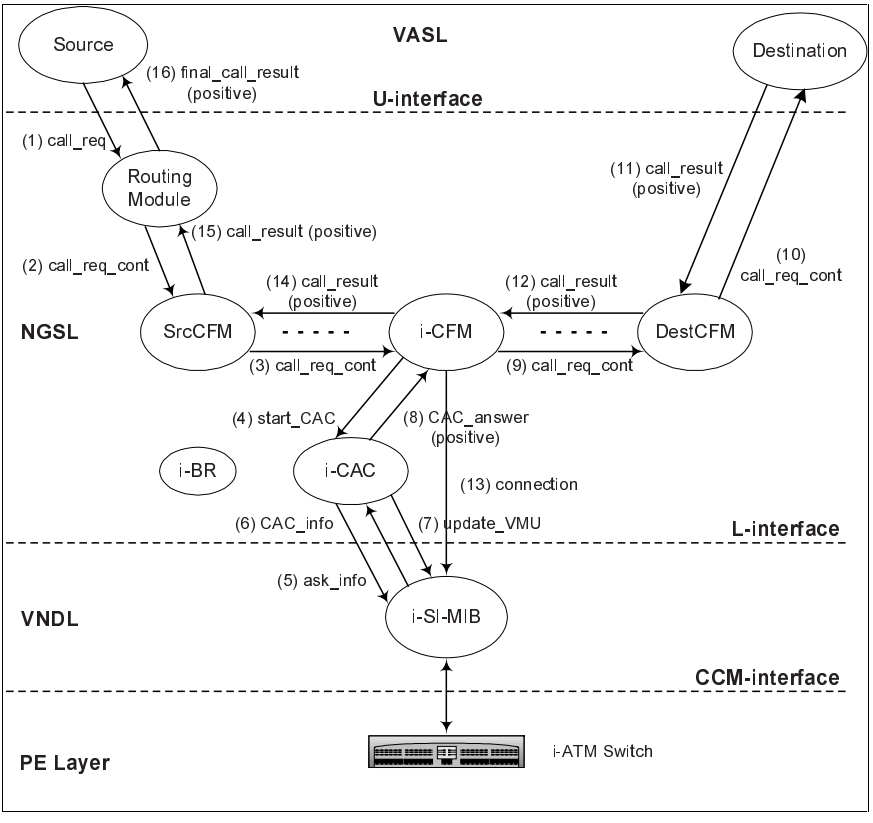


**Fig. 2.** Call acceptance without bandwidth redistribution

respectively), the requested level of service quality (parameter `QoS`), the expected traffic profile of the requested call (parameter `TrProf`) and a unique identification number (parameter `RefNum`), used to distinguish the call.

Receiving `call_req`, the routing module defines the possible routes that connect the source terminal to the destination one and selects to examine one of them. The routing vector of the selected route (object `RVector`) is wrapped together with the information of message `call_req` into a new message `call_req_cont`, which is forwarded to the call forwarding module of the first node participating in the chosen route (Source Call Forwarding Module - SrcCFM). The object `Rvector` contains the pairs of input port – input VPI (parameters `inputPortID` and `input-VPI` respectively) and output port – output VPI (parameters `outPortID` and `out-VPI respectively`) for every hop contained in the selected route.

Moving now in the i-node of the routing path between source and destination, i-CFM sends to its corresponding CAC module (i-CAC) a message `start_CAC` in order to commence the CAC process. The message `start_CAC` contains the parameters `outPortID`, `QoS`, `TrProf` and `RefNum`. i-CAC uses the first two of these parameters in a message ask_info in order to retrieve from the i-SI-MIB (i-SI-MIB), through a message `CAC_info`, appropriate information for the application of

a CAC scheme. This information consists in available resources (parameter res) and the load conditions (parameter BackgroundTraffic) within the (specified by the parameters outPortID and QoS) Virtual Multiplexing Unit (VMU) of the source switch (SrcSwitch).

Using the information of the parameters QoS, TrProf, res and BackgroundTraffic, i-CAC performs a CAC scheme in order to accept or reject the new call request. Given that i-CAC accepts the call (see Figure 2), it proceeds to compute the new value updBackgroundTraffic of the parameter BackgroundTraffic. Then, it firstly issues a message update_VMU to i-SI-MIB in order to update the load conditions of the examined VMU (preventive bandwidth reservation) and secondly it informs i-CFM, through a message CAC_answer, for the success of the CAC process. CAC_answer includes a boolean parameter answer, which actually contains the positive CAC answer for the examined scenario.

Subsequently, i-CFM forwards the message call_req_cont to the next node along the route being examined and the aforementioned process is repeated on all nodes along the selected route until the destination switch (DestSwitch). After the receipt of CAC_answer by the destination CF module (DestCFM), the message call_req_cont is forwarded to the destination terminal. The destination terminal (according to the presented scenario) accepts the call request and returns a positive message call_result to DestCFM (the boolean parameter result of the message has the value true). The latter transfers this message to the previous call forwarding module, which passes it to its own previous call forwarding module, etc. Eventually, the message call_result reaches at SrcCFM and from there at the routing module, which creates the message final_call_result and passes it to the source terminal informing it about the call acceptance.

Besides passing a message call_result to the previous call forwarding module, each call forwarding module sends a message connection to its SI-MIB (with parameters inPortID, inVPI, inVCI, outPortID, outVPI, outVCI) directing the SI-MIB to create the appropriate VC cross-connections into the underlying ATM switch. At the final stage, the successful set-up of a new call is the result of the application of messages connection from all the involved to the new call ATM switches.

If i-CAC does not accept the call request and provided that the examined route is not the last one, it delivers a negative message CAC_answer (the parameter answer takes the value false) to i-CFM. Subsequently, i-CFM sends to CFM of the preceding node along the current route a negative message call_result (the boolean parameter result of the message has the value false). In addition, each CFM informs its corresponding SI-MIB to release the pre-reserved resources through a message reverse_VMU. Eventually, the negative message call_result reaches at SrcCFM and from there at the routing module. Subsequently, the routing module starts to check an alternative routing path, which connects the source and destination terminals.

The second examined case of call set-up concerns an accepted call request after bandwidth redistribution. This scenario (see Figure 3) starts at the point where the call request has been rejected across all the possible routes except the last one, which is under check.

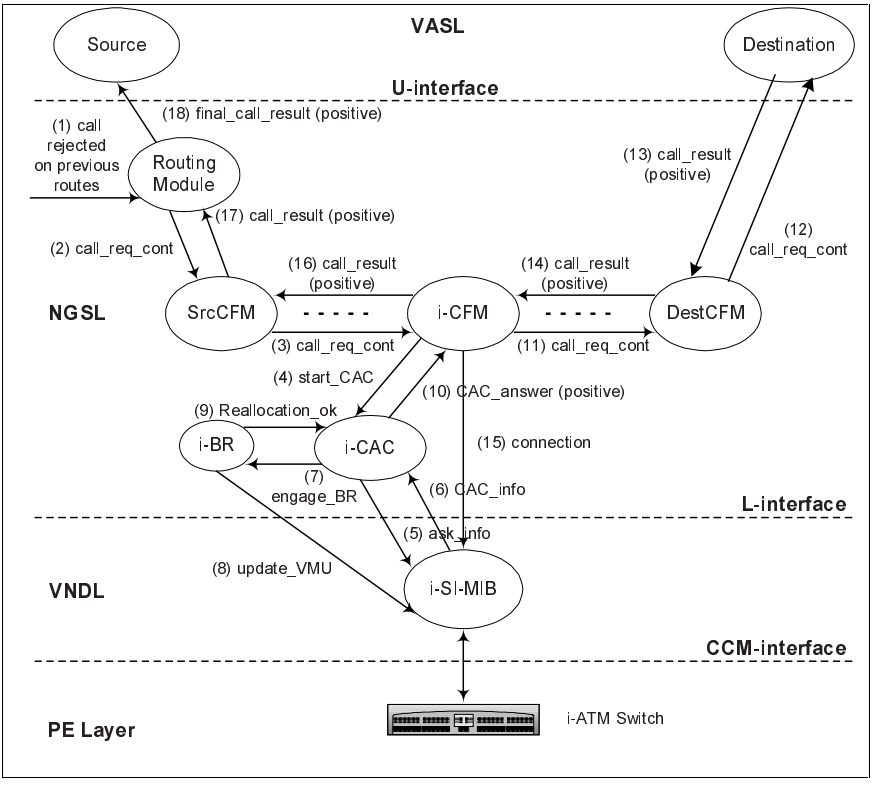


**Fig. 3.** Call acceptance after bandwidth redistribution

According to Figure 3, the signalling path for this case is similar with the path of the previous case until the point where i-CAC performing a CAC scheme does not accept the requested call due to the fact that the examined VMU does not have adequate resources to serve the request. Then, i-CAC sends a message `engage_BR` to the respective Bandwidth Redistributor (i-BR) in order to initiate the bandwidth redistribution process. The latter scans the port (indicated by the parameter `outPortID` of the message `engage_BR`) in which the examined congested VMU is attached and finds another VMU with bandwidth surplus. The amount of bandwidth needed by the incoming call is released from the VMU with unused residual bandwidth and appended to the congested VMU, thus allowing it to serve the new call request. i-BR notifies i-CAC about the successful bandwidth redistribution through the message `redistribution_ok`. Receiving this message, i-CAC continues its operation as in the case of accepted call request without bandwidth redistribution. The difference from the previous case is that it is the BR module, which updates the load conditions of examined VMU (adding the new call request) and not the CAC one.

The last case in the call set-up area, which is depicted in Figure 4, concerns the call rejection by the network.

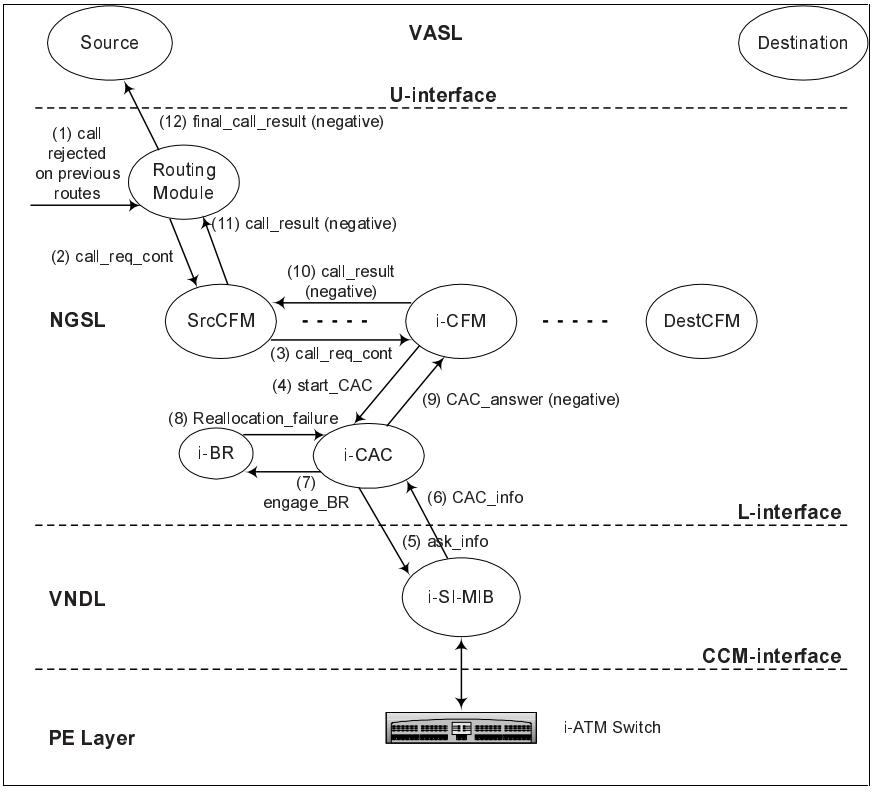


**Fig. 4.** Call rejection by the network

According to this case, the call request has been rejected in its all possible routes except the last one (as in Figure 3) where a i-CAC rejects the call request and its corresponding i-BR fails to perform bandwidth redistribution. i-BR announces to i-CAC its failure though the message `redistribution_failure`. Subsequently, i-CAC sends to i-CFM a message `CAC_answer` (where the boolean parameter `answer` of the message has the value `false`). i-CFM creates a negative message `call_result` (the parameter `result` has the value `false`) and forwards it to the previous call forwarding module along the route, which passes it to its own previous call forwarding module, etc. Eventually, the negative message `call_result` reaches at SrcCFM and from there at the routing module, which creates the negative message `final_call_result` (the parameter `result` has the value `false`) and passes it to the source terminal informing it about the call rejection.

Besides passing a negative message `call_result` to the previous call forwarding module, each call forwarding module (having performed bandwidth pre-reservation) sends a message `reverse_VMU` to its SI-MIB (with parameters `outPortID`, `QoS` and `BackgroundTraffic`) directing the SI-MIB to recall to the corresponding involved VMU the original load conditions.

### 3.2 Call Release

According to the call release scenario depicted in Figure 5, the source terminal initiates the call release function by sending a message `call_release` to the routing module. The message `call_release` contains a parameter `RefNum`, which is used by the routing module to locate the first node of the routing path under release. Then the routing module forwards the message `call_release` to the call forwarding module of the first node (SrcCFM), which passes it both to its corresponding CAC module (SrcCAC) and to the next call forwarding module along the route under release.

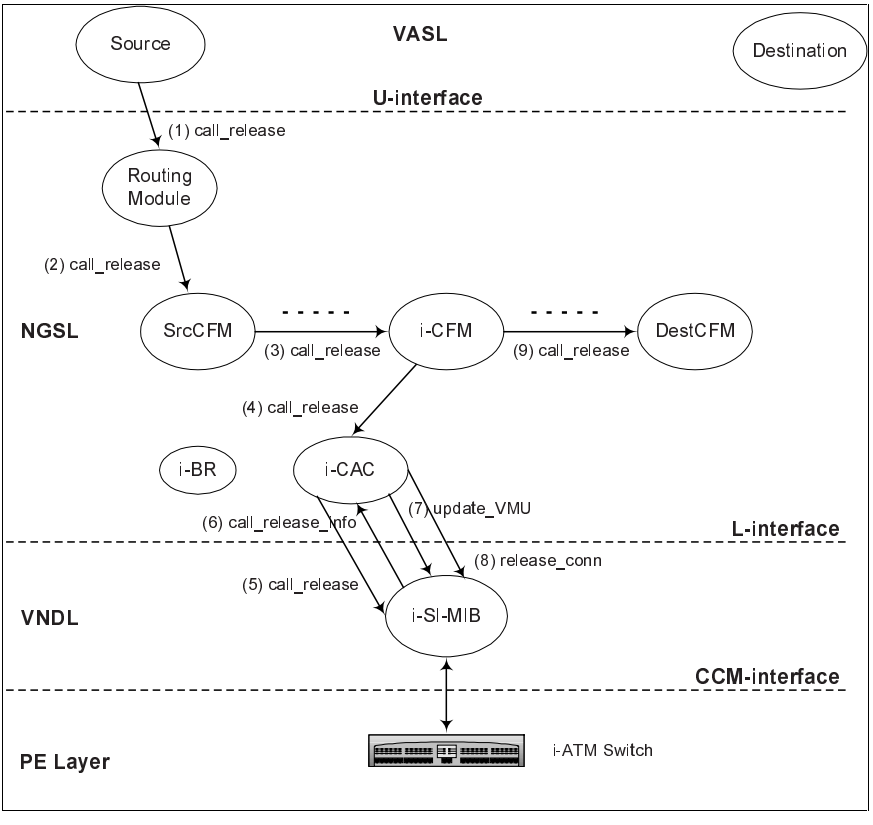


**Fig. 5.** Call release

Considering the i-node of the routing path between source and destination, i-CAC forwards the message to the i-SI-MIB and gets a reply in the form of a message `call_release_info`. The message `call_release_info` contains the parameters `QoS`, `TrProf`, `res` and `BackgroundTraffic`. These parameters are used for the calculation of the updated (the final amount after the call release) background traffic on the specified VMU (parameter `updBackgroundTraffic`). The VMU is updated with the new load conditions through the message `update_VMU`,

sent by i-CAC to i-SI-MIB. Besides the message `update_VMU`, i-CAC sends a message `release_conn` (with parameters `inPortID, inVPI, inVCI, outPortID, outVPI, outVCI`) to the i-SI-MIB, directing it to remove the VC cross-connections created to serve the released call. The aforementioned procedure is performed on every node along the route, leading, as requested by the source terminal, to a complete call release from the network.

As an overall observation on Figures 2,3,4 and 5, it is apparent that some of the messages run through the L and U interfaces (defined according to the IEEE P1520 reference model), while some others are internal messages within NGSL.

## 4 A VSP Prototype Implementation

A prototype implementation of VSP runs over a software platform, which can be installed over heterogeneous ATM network islands for providing advanced traffic control and resource management functionality. The platform includes various distributed software entities created by extending an intelligent agent platform, which was built using the BAT object library [19]. VSP is implemented through facilities based on the Java Remote Method Invocation (JRMI) programming and the powerful mechanism of signature matching. The invocation of methods belonging to the distributed software entities of the platform is based on the agent ontology specified by FIPA 10. The directory services of BAT are employed to locate and invoke the agents in the distributed environment. Every agent incorporates at least one worker module, which is usually implemented in a thread form, to carry out the agent's transactions.

Figure 6 outlines a portion of the platform, operating on top of a small ATM network consisting of one domain with three switches. As shown in this figure, the system consists of a number of distributed software entities that communicate through VSP. Five different types of entities may be identified:

1. Terminal Agents – TAs: Software agents representing the terminal devices. They are responsible for the message `call request` creation providing the user interface for the definition of the corresponding parameters (`source_id`, `dest_id`, `QoS`, and `TrProf`).
2. Routing Agent – RA: Single central agent responsible for intra-domain routing in a core ATM network. RA also has coordinating responsibilities.
3. CAC Agents – CAs: Distributed agents responsible for the performance of the CAC function on the ATM switches. CAs are also responsible for the forwarding of messages `call request` and `call result`. Each CA groups with its corresponding SwWA agent, leading to a "one-to-one" formation dedicated to the respective underlying ATM switch.
4. Switch Wrapper Agents - SwWAs: Delegated agents that wrap the ATM switches by abstracting their hardware resources into the SI-MIB. Switch wrapper agents sit on top of the ATM switches and provide the virtual environment for the deployment of traffic control and resource management operations on the switches.
5. Bandwidth Redistributors – BRs: The main purpose of a BR is the congestion relief of VMUs. It achieves this goal by transferring to a congested VMU a part of the surplus bandwidth of a VMU attached to the same port.

The bandwidth reallocation takes place when an incoming call cannot be served by a fully utilized VMU along its route.

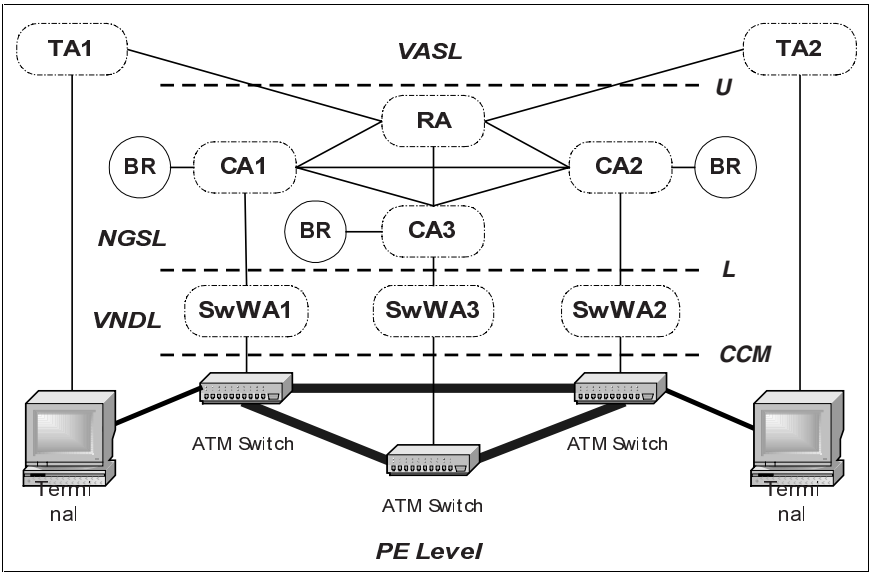


**Fig. 6.** A VSP prototype implementation

For details on the structure, functionality and implementation of the above entities, the reader is referred to [20]. Using the implemented prototype, the VSP call performance was benchmarked. Figure 7 depicts the VSP call set-up (without BR involvement) and call release delays. The involvement of BR in a call set-up process adds 5ms per hop. As Figure 7 shows, the average call set-up delay on a single Fore ASX-200BX ATM switch was measured at 76ms, increasing almost linearly with the number of nodes/hops per route. The corresponding average call release delay was measured at 33ms, increasing also linearly with the number of nodes/hops per route. The almost linear increment of VSP call set-up and release time constitutes a significant proof of protocol's robustness and scalability. Furthermore, concerning the fairness of VSP, it should be noted that all calls are treated equally.

An actual picture of VSP call performance is given through the comparison of VSP call set-up and release time with the native UNI 3.0 ones. According to [21], for a Fore ASX-200BX ATM switch (this type of switches are used in the implemented prototype) the call set-up and call release delays are 27ms and 6ms respectively. The variation of VSP's delays to UNI 3.0 ones can be both justified and accepted. It can be justified by the out-of-band nature of VSP and performance limitations of Java. It can be accepted considering that the goal of VSP is not to give call set-up/release times faster than the conventional standardized signaling protocols achieve, but to provide externally defined traffic control and resource management functionality.

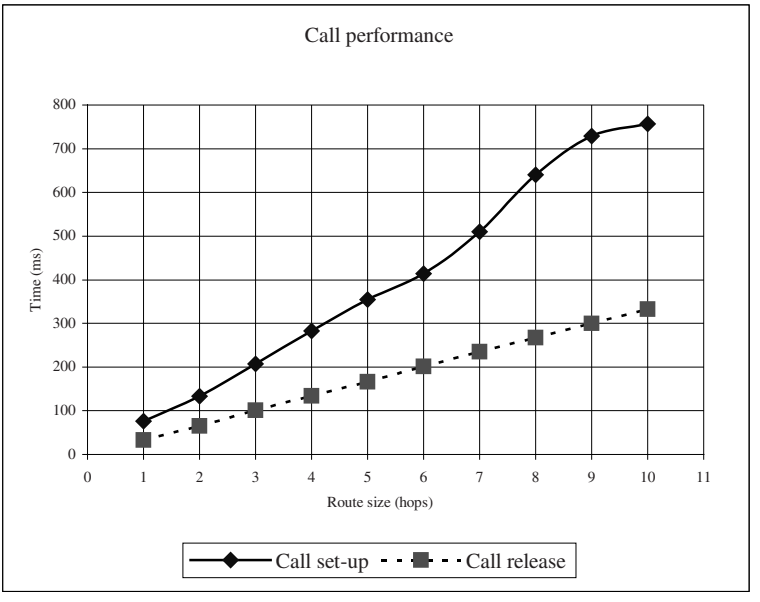


**Fig. 7.** VSP call performance

## 5 Conclusion

Following recent proposals for programmable network infrastructures, the paper presents a Virtual Signalling Protocol (VSP), which bypasses the conventional control information channels and supports externally defined traffic control and resource management functionality. Furthermore, the paper gives a generic framework for the employment of the IEEE P1520 L and U interfaces, as well as of NGSL in a standardized and systematic way for the purposes of device hardware independent traffic control and resource management. It should be noted that, although the protocol presented in the paper is tailored to ATM networking equipment, it is fairly general and can be extended for covering network nodes of a different technology (provided that this technology supports the potential for QoS concept and at least some notion of "connection" or, more generally, "traffic flow").

## References

1. ATM User Network Interface (UNI) Specification V3.1, Electronically available at http://www-mo.atmforum.com/ftp/atm/approved-specs/af-uni-0010.002/.
2. ATM User Network Interface (UNI) Signalling Specification version 4.1, Electronically available at ftp://ftp.atmforum.com/pub/approved-specs/af-sig-0061.002.pdf.

3. Recommendation Q.2931 – User-Network Interface layer 3 specification for basic call/connection control. Electronically available at http://www.itu.int/rec/recommendation.asp?type=items&lang=e&parent=T-REC-Q.2931-199502-I.
4. Private Network-Network Interface Specification V1.1, Electronically available at ftp://ftp.atmforum.com/pub/approved-specs/af-pnni-0055.001.pdf.
5. A.T. Campbell, H.G. De Meer, M.E. Kounavis, K. Miki, J.B. Vicente and D. Villela, "A survey of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 2, pp. 7–23, April 1999.
6. Open Signalling Working Group, http://comet.columbia.edu/opensig/
7. J. Biswas, "The IEEE P1520 Standards Initiative for Programmable Network Interfaces", *IEEE Communications Magazine, Special Issue on Programmable Networks*, October 1998.
8. M. Chan, J. Huard, A. Lazaar, K. Lim, "On realizing a Broadband Kernel for Multimedia Networks", 3rd COST 237 Workshop on Multimedia Telecommunications and Application, Barcelona, Spain, November 25–27, 1996.
9. J. Van der Merwe, S. Rooney, I. Leslie, S. Crosby, "The Tempest – A Practical Framework for Network Programmability", *IEEE Network*, November 1997.
10. Foundation for Intelligent Physical Agents, *FIPA00023 Agent Management Specification,* 2002, Electronically available from http://www.fipa.org/specs/fipa00023/SC00023J.html
11. Mobile Agent System Interoperability Specification, Electronically available at http://www.omg.org/cgi-bin/doc?orbos/97-10-05.pdf
12. S. Vinosky, "CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environment", *IEEE Communications Magazine*, Vol. 14, No. 2, February 1997.
13. Distributed Component Object Model http://www.microsoft.com/com/tech/DCOM.asp
14. Java Platform http://java.sun.com/
15. Grasshopper 2 Agent Platform http://www.grasshopper.de/
16. Voyager Agent Platform http://www.recursionsw.com/products/voyager/voyager.asp
17. G. Kormentzas, K. Kontovasilis, "An Open Architecture for Transparently Embedding Advanced Traffic Control Functionality in ATM switches", *Journal of Communications and Networks, Special Issue on Programmable Switches and Routers*, Vol. 3, No 1, March 2001.
18. G. Kormentzas, J. Soldatos, E. Vayias, K. Kontovasilis, and N. Mitrou, "An ATM Switch-Independent MIB for Portable Deployment of Traffic Control Algorithms", In Proc. *7th COMCON Conference*, July 1999, Athens, Greece.
19. J. Bigham, L.G. Cuthbert, A.L.G. Hayzelden, Z. Luo and H. Almiladi, "Agent Interaction for Network Resource Management," In Proc. *Intelligence in Services and Networks '99 (IS&N99) Conference*, Barcelona, April 1999.
20. K. Katzourakis, G. Kormentzas, K. Kontovasilis, C. Efstathiou, "A Software System for Providing Traffic Control and Resource Management Functionality in ATM Networks", Submitted to *INFOCOM 2004 Conference*, March 2004, Hong Kong.
21. R Pillai, KL Su, J Biswas and CK Tham, "Call Performance Studies on ATM Forum UNI Signalling Implementations", *Computer Communications,* Vol. 22, Issue 5, pp. 463–469*,* April 1999.