Cooperative computing in the control plane

Application to NGN services and control

Claude Rigault

Département informatique et réseaux,
ENST, 46 rue Barrault, 75 013 Paris, France
GET-Télécom Paris ; LTCI-UMR 5141 CNRS
claude.rigault@enst.fr

Rony Chahine

Département informatique et réseaux,
ENST, 46 rue Barrault, 75 013 Paris, France,
GET-Télécom Paris ;
CoreBridge, 3 rue Saint Philippe du Roule, 75 008 Paris
rony.chahine@enst.fr,

ABSTRACT: *This paper analyses some control requirements for the NGN and proposes original solutions. The special nature of control plane software is underlined and some of the research challenges raised by this type of software are pointed out. Special attention is devoted to the unbundling of the switching architecture and its consequences. The problem of signalling is then analysed and new solutions are proposed to design services in the NGN.*

KEY WORDS: *NGN control and service planes. Cooperative computing. Signalling.*

1. Introduction: the special nature of network and service control software

In this paper, we analyse network and service control activities and the special nature of the software that executes these activities. We outline some of the computer science research problems raised by control software and we indicate new approaches for these research problems. We finally show how these approaches may be used in the case of telecommunication services over the NGN [1].

In the part 2 of this paper we explain that services may be designed using a variety of communication paradigms. Not all communication paradigms require control activities. We give a formal definition of control, and we characterize which kind of communication paradigm requires control activities. At this point we underline the special nature of the software that is required for the execution of control activities. Throughout this entire part we try to revisit many network concepts for which the vocabulary is often used in contradictory manners.

In the part 3 of this paper, we show that control activities may be partitioned into several control domains that may be operated independently. This leads us to a generalized definition of the unbundling concept, to a general model for the unbundling of network functions and to a classification of the many signalling protocols.

In the part 4 of this paper, we explain some important mechanisms required by control activities, we give a new and formal definition for signalling and we apply this new definition to propose new signalling protocols.

2. Communication paradigms, control activities and cooperative computing

2.1. *Communication paradigms*

A **service** is a coordinated set of functions that a system brings to people or to software applications. A service **instance** is a single execution of a service for some particular actors. For example, when two windows of a same web browser are opened, two instances of one service are initiated. Communication services may use one of several communication paradigms (or ways of communicating). We talk of **synchronous communication paradigms** when the emitter of a message cannot proceed further in its communication activities while waiting for the answer. On the contrary, we talk of **asynchronous communication paradigms** when the emitter of a message may carry on its activities while waiting for the answer. The so-called Message Oriented Middleware (MOM) are examples of communication services using an asynchronous communication paradigm. At this point in time, five communication paradigms have been identified. Two of them are synchronous paradigms: the "request and answer" paradigm and the "conversional" paradigm. Three communications paradigms are asynchronous: the "Message passing" paradigm, the "Message queuing" paradigm and the "Publication/Subscription" paradigm. In the scope of this paper our attention is mainly focused on the synchronous paradigms: "request and answer" and "conversational".

In the **"request and answer"** paradigm the communication session lasts only during the time that is necessary to build up and send the answer to the request. There is no memorization of the exchange, no persistence of any resource. It is a connectionless mode of operation according to OSI terminology and it is a stateless

paradigm. In the general case of "request and answer" communication both parties may originate the request and the other party replies with the answer. However this general case is rarely used. A particular case of "request and answer" communication called the "client-server" communication paradigm is mostly used (see Figure 1). It is a single mono-directional type of "request and answer" where one party only issues request and the other party only issues answers: A client always originates the communication session. A server is "always on" and only sends answers to a client. A server never originates a request to a client. A client only **"pulls"** information from a server; a server never **"pushes"** information in a client. It should be noted that the Internet is a network that has been optimized for the client server communication paradigm, although some other types of communication may take place over it, in a non-optimised manner.

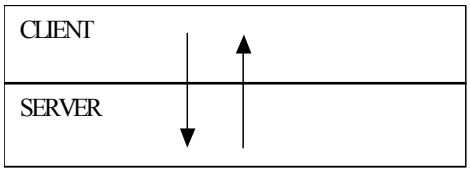

Figure 1. *"Client-server" as a particular case of "request and answer"*

In the **"conversational"** paradigm, a **communication environment** is explicitly set-up before the users start exchanging media and this environment remains established even in the absence of user activity. We can say that this communication environment is **"persistent"**, meaning that it remains set-up as long as an explicit release is not issued. Therefore, this environment is memorized for the duration of the communication session. It is a connection-oriented mode of operation according to OSI terminology and it is a state-full paradigm. The emblematic example of service requiring a conversational communication paradigm is the telephone service.

What do we mean by "communication environment"? First it is memory. Because the communication environment is memorized, a memory page has to be opened in each partner of the communication session. We call this memory page a **"local context"**. Each local context memorizes the appropriate session parameters for the session duration. We call these Session parameters **"Session Instance Data"**. However, the local contexts of each participating partner put together are to be considered as a **"global context"** for the service session. The "Global Context" is the union of all local contexts that give a global view of the session. When the communication session is terminated, an explicit release is issued and all the memory pages are freed, deleting in the same time all the Session Instance Data.

In addition to reserving memory some application require resource reservation in order to guarantee an upper bound to the transfer delay. We define as a **"connection"** the assignment of resources (other than memory) to a particular

session of conversational communication. These resources may be physical resources like bandwidth in circuit switching or in INTSERV [2]. They may be more virtual as a route reservation in connection-oriented packet switching or even as a scheduling priority like in DIFFSERV [3] or also a traffic aggregation label like in MPLS [4]. According to our definition **QoS** mechanisms in IP networks are indeed connection mechanisms. Connection services are also named **"Bearer Services"**. The most important example of services requiring connection mechanisms is voice services between human end users.

In the following we will therefore define as a "communication environment" either memory alone: the global context, or memory and resources. We may remark at this point that, while the Internet was optimized for the request and answer communication paradigm, X25 networks were designed according to the conversational communication paradigm.

2.2. *Control activities, local and global context, associations*

A service using the conversational communication paradigm should entail special functions dedicated to setting-up and releasing the communication environment. We define as **"Control functions"** the functions executed by all the partners of a conversational communication instance to **set-up, modify**, and finally **release** the communication environment for this communication instance. Therefore, in each partner of a conversational communication session there is a control process activated. The circles on Figure 2 show these control processes. Each control process, in its turn, opens a local context shown by a rectangle. The control process uses the local context to store its state and its Session-Instance-Data for the session duration. When the communication session is terminated, an explicit release is issued and all the memory pages are freed, deleting in the same time all the Session Instance Data. This limitation of control activities to the session duration makes a fundamental difference between control and management activities. Management, in general, is the adjustment of service parameters. The effects of control finish with the session while the effects of management persist beyond the sessions. Control acts on Session Instance Data, while management acts on **Service Support Data (SSD)**. Every type of service has to be managed regardless of the communication paradigm they are using. On the other hand only services requiring a conversational communication paradigm need control functions.

On Figure 2 we represent a conversational communication session between machine (or human) Alice "A" and machine (or human) Bob "B". We have control processes running in Alice, at the Originating Local Exchange (OLEX), at the Transit Exchange (TEX) and at the Terminating Local Exchange (TLEX).

We call **“Control plane”** the connected set of all processes or entities executing control functions either in the terminals or within the network. Because machines are multitasks they usually have many simultaneous sessions of conversational

communication and therefore many local context open simultaneously, belonging to different communication sessions.

If control processes need to communicate with distant control processes for a given communication session, they have to give the reference of the distant context.

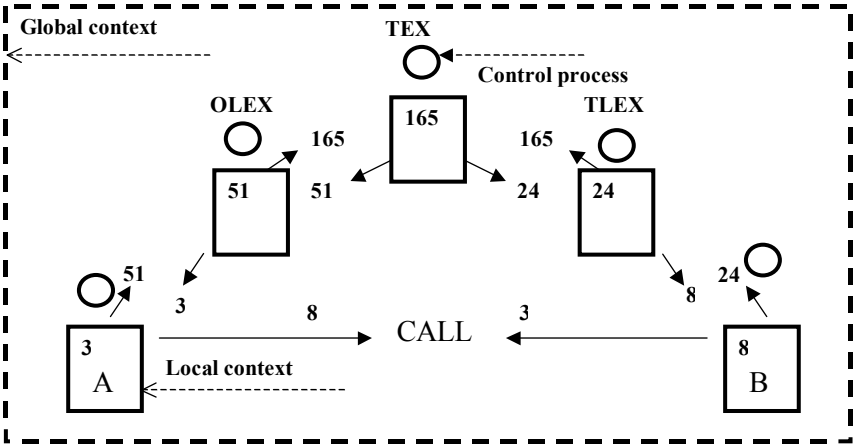

Figure 2. *Control processes, local and global context, associations*

We define that Local control processes are **associated** if they can mutually address each other among multiple control instances within multitask machines. These associations are achieved by the cross-referencing of contexts: each participating control process must maintain a table of the context references of the other processes with which it communicates. By the association mechanism, each local context has a pointer to the others as shown on Figure 2. Because the global view of the communication session, i.e. the complete information about it, is spread in all the local contexts, the global context is therefore made of a link list of associated local contexts in the same manner as sectors of a disk are linked together to form a file. Today there are many protocols that allow such persistent cross-referencing like the TCP protocol, the dialogs and transactions identifications in TCAP [5], and the CORBA [6] associations. The disadvantage of these protocols is that they are specific to some particular networks and do not allow a cross-network operation.

An association of special interest is the end points association. For example, on Figure 2 Alice knows her conversation 3 is the conversation 8 of Bob and Bob knows his conversation 8 is the conversation 3 of Alice. Several research groups have found convenient to name *"Call"* this particular association.

According to this definition, the "Call' is the association (of context references) between network end-points. More generally, in the case of multi-party calls, a ***"call"*** *is an association graph between network end-points*. This definition has very useful consequences and has been adopted by several ITU-T recommendations for

B-ISDN and for IMT2000. [7]. We will use this definition in the rest of our paper. While the call is an association between network end points, "*a **connection** is an assignment of resources to a given call"*. It follows that the *Call function is an "end to end" process*, while the *connection is a link-by-link process.*

End-to-end call services include, in addition to the fundamental association service, presentation functions and bearer negotiation functions. Once the call is accepted, the agreed bearer service has to be setup by Bearer Control. What technicians, and many standards like the Intelligent Networks (IN) [8] standards usually name a "Call Control Function" is indeed a "Bearer Control Function".

As these concepts are now defined we derive that a Public Switched Telephone Network (PSTN) does not process calls but processes connections. The calls (cross referencing) are done by the human users. The small conversation: "Hello, I am Alice, I would like to talk to Bob…Hi Alice, Bob speaking!" is actually a protocol by which Alice and Bob associate their references. It is a call protocol. Therefore, the task of the PSTN is not to do calls but to make connections (setup bearer services) for calls.

If Alice and Bob are machines they have to use a similar call protocol to associate their local contexts. In a GSM network, when a VLR calls an HLR over the SCCP network in a connectionless mode, the call protocol used is TCAP. This protocol exchanges originating and terminating dialog identifications.

Multimedia networks must necessarily include end-to-end call control functions and protocols because multimedia functionalities have to be negotiated and agreed before the communication session starts. Examples of call protocols used for Multimedia networks are the IP telephony "Session Initiation Protocol" SIP [9], the H323 [10] protocol suite where the call protocol is the H225-Q931 protocol, and also the "Bearer Independent Call Control" BICC [11] used in B-ISDN.

2.3. *The "Cooperative Computing" nature of control activities*

So far control activities have required huge amounts of programming effort, certainly classifying them among the largest programs ever developed. The programming of the classical call control of present day's digital telephone exchanges has required thousands of man-years of programming effort. *The origin of this difficulty may be traced into the special cooperative nature of control software.* To understand this point we should underline that it is possible to classify computer science into 3 main branches: centralized computing, distributed computing and cooperative computing.

Centralized Computing was the original state of the art of computer science. A very powerful mainframe would master all the processes in a company. All terminals, machines, tools, would be intelligence-less slaves executing orders of the central Master computer.

Later on, new companies pushed forward a new type of computer science called **Distributed Computing**. In distributed computing, many smaller computers, called

minis, work together, specializing on given types of tasks and providing some amount of department or activity independence. This new computing organization required communication, and therefore networks, between the computers. The general solution developed by computer science for distributed computing is the "Client-Server" architecture, based on the "request and answer" communication paradigm. However, the client server architecture should be considered more like an adaptation of the former centralized scheme to the distribution problem than like a radically new solution. The client is mostly concerned by customization and interface problems and the essential service data and service logic are located in the server central position.

A radically new solution to the distribution of intelligence on many smaller computers would be a new kind of computer science called **"Cooperative computing"**. In cooperative computing, there is no central position, all the computers are equal and no one is in a *permanent* position to give orders to the others. While many different efforts are taking place towards the development of a theoretical solution for cooperative computing, (grid computing, peer to peer processing, agents…), no generally accepted theoretical base has been yet proposed.

Nevertheless some examples of working cooperative processes, successfully developed, do exist. The main one, for our concern, is the so-called "call control" of telephone switches. Indeed control functions work in a cooperative manner. In the telephone network all switches are equal, there is no centralized platform controlling the setup of a call or its release. Each switch works on a peer-to-peer basis to achieve a global service. It is because of this special cooperative nature of control activities, and of the lack of a general theoretical base for this new type of computing, that "Call control" was developed as an ad hoc solution through a huge effort and many trial and errors.

So far, the main efforts attempted by the computer science research community towards the handling of control activities are directed towards some adaptations of the "Client-server" architectures. This is the case of the, now generally accepted, Session Initiation Protocol SIP. A remarkable exception to this statement is the "active network" research and its connections to agent programming. However this type of solution still remains in a very early stage.

We advocate at this point that the special cooperative nature of control activities for services requiring a conversational paradigm justifies a serious attention to fundamental research in cooperative computing.

For this purpose it is possible to identify some key subjects for research in cooperative computing:

- **Cooperative computing requires information sharing**. Information sharing between control and service plane partners is called "signalling". It derives that Signalling research is not merely a research problem for telephony; it is a fundamental computer science research problem for cooperative computing. Signalling is certainly one of the foundations of cooperative computing

- Cooperative computing requires policies for the distribution of decision authorities. It is not because every participating entity is equal that decisions should not be taken. For a given problem, at a given moment, who takes the decision for the whole cooperation? This is a general and very difficult problem (also experienced in other areas than computer science). This point also shows that the ad hoc solution of the telephone networks is not a general solution. Telephone networks use a round robin policy: First Alice takes a decision, then the originating exchange, then the transit exchange, then the terminating exchange, then Bob. It is clear that this policy, well adapted to the link-by-link operating mode of the connection process cannot be generalized to any kind of cooperative computing problem.

- Cooperative computing requires behavioural models for the partners. How to take a decision if the behaviour of the partners is not known (i.e. they are not predictable)? Each partner should have a behavioural model of the partners with whom it has working relations. This is the reason for the so important "Basic Call State Model" BSCM in intelligent network technology, and for the various call models in Computer Telephony Integration CTI, as well as the connection modelling in MEGACO signalling. This consideration is an argument in favour of state-full proxies in the new VoIP architectures rather than stateless proxies.

- Cooperative computing requires confidence in the partners. This obvious consideration is not an easier research problem than the preceding ones. In their monolithic model, the historical telephone operators would identify the "trusted domain" of their own closed network and the "un-trusted domain" of the third party service suppliers. The soft-switch architecture and its derivatives like the NGN IMS architecture are bound to terminate the integrated model of the telephone switch and to raise the confidence problem of cooperation among external partners. Authentication is required to ascertain the identity of the partner. Ciphering is required to avoid eaves dropping or information substitution.

This list of research problems for the establishment of a cooperative computing theory is certainly not exhaustive, but is already sufficient to understand the huge research area raised by control activities in the conversational communication paradigm. The rest of this paper will concentrate on the signalling problem.

3. Partitioning control activities and unbundling network services

3.1. *Functional domains for the control of conversational communication services, horizontal unbundling*

In legacy networks, control activities are bundled together in a single platform: the telephone switch control unit. However, the new soft-switch technologies based on an asynchronous, packet based, transport plane challenge this monolithic model. It is not any more necessary to keep all the functions of a telephone switch

integrated in a same platform like in synchronous technologies. It becomes then legitimate to raise the question of "who does what in switching?" and it appears clearly that the bearer control function was actually an integrated processing (by a master) of activities of a different nature that could be advantageously processed by peer cooperative systems without subordination relations. Several research groups have attempted to identify the control activities that could be eligible for a separate processing in different cooperative platforms, eventually belonging to distinct business partners, and propose unbundled architectures. Therefore, the unbundling concept is not limited to the access function. Indeed, the unbundling concept may be extended to many more activities that we are now going to identify.

A first proposal from the TINA research effort [12] consists in separating Access services, Transport services and Intelligent Network services. **Originating Access** services are services required for the login function (Authentication, localization, Virtual Home Environment (VHE) services, location dependent services…) **Terminating Access** services are services required for the contact function (Name address Translation, Presence services, calling party record presentation…). Terminating access depends from the Originating access through the localization function and therefore both functions have to remain bundled as the **"Access services"**.

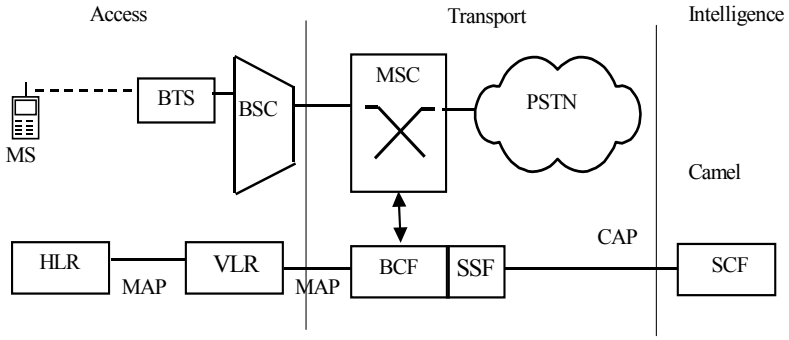

Figure 3. *Mapping of a mobile telephone network on the various functional domains*

Transport services include the Call functions and the Connection (or Bearer) functions. As already explained, legacy telephone network, designed for human communication, did not really include call functions as the bearer services could not be negotiated and were determined by the physical nature of the terminals. On the contrary multimedia networks must include end-to-end call control functions for the association of the terminating processes and the negotiation of the bearer facilities.

Bearer control works on a link-by-link basis and therefore the networks have imbedded bearer control functions. These imbedded bearer control functions are normally executed as a sequence of (bearer) "functional elements" or "bearer service

features" or "bearer components". There is a default sequence of these bearer components run by the network switches. In PSTN this default sequence is called the Plain Old Telephone Service **POTS**.

Intelligent Network services are services achieved by substituting a different sequence of bearer components to the default sequence. It derives from this definition that an intelligent network service is a service that can only be provided by the network. Intelligent network services are services using network functions or network databases containing operator information. Services that may be implemented entirely in a network terminal without resorting to any network function are not intelligent network services. We derive from this first list of separable functions that control activities may be classified in three functional domains: the **Access domain** providing access services, the **Transport domain** providing transport services and the **Intelligence domain** providing intelligent network services. Figure 3 shows how a mobile telephone network may be partitioned into these 3 domains.

Different stakeholders may operate these three functional domains, in an unbundled manner, provided that they work in a cooperative manner. We define as horizontal unbundling this first unbundling scheme.

3.2. *Vertical unbundling*

Standardization bodies use the concept of plane. A plane is a set of *communicating* entities linked by a specific network. Standardization bodies also agree that the NGN will be made of several planes, each of them dedicated to specific tasks in the provision of a global communication service. A plane being also a network, the NGN will include several networks, one per plane. Each plane works out-band from the others and may therefore have its own ciphering algorithms and keys. The generally agreed NGN model (see Figure 4), that we will name **"NGN plane model"**, considers three different planes: the Service plane, the Call Control plane and the Transport plane.

In the **Service plane** we have the service provision platforms operated by Service providers. In the **Control plane** (or session plane), we have the entities in charge of the end-to-end Call setup protocols and the entities in charge of the link-by-link setup of bearer services. In the **Transport plane** we have the transmission and switching (or routing) capabilities for the users media.

This NGN plane model is to be considered as a cooperative model. The entities in each plane work in a cooperative manner, either with other entities in the same plane or with entities in the other planes with which they are supposed to have open and standardized interfaces.

A very interesting aspect of this model is that each plane may be unbundled, i.e. operated by different stakeholders. We may therefore have Call Control Service Operators acting in the Control plane and Connectivity providers acting in the transport plane. This is made possible by the soft-switch architecture and already today we find, in many places, Call (or Session) services operators using Media

Gateway Controllers to provide Call services over the media gateways located in the transport plane. Media gateways may actually be imbedded in new IP telephone sets, now becoming available at a normal telephone set price, and compatible with the standard Media Gateway Control protocols MGCP or MEGACO [13]. Some other Call Control operators like the "Skype" company [14] use proprietary downloadable interfaces at the expense however of using a PC as a terminal.

This new separation between Control operators and transport operators consolidates in the NGN the breaking down of the monolithic model of the telephone switch that is currently taking place.

NGN plane model	ETSI Model	Simpson model
Service plane and service network	Service control	Client
		Provider
		Components
Control plane and control network	Call Control services	Call Control services
	Bearer control	Bearer control
	Media Control	Media Control
Transport plane and transport network	Transport services	
	Transport Control	
	Transport flows	

Figure 4. *Various models for service provision over the NGN*

While agreeing with the three planes of the NGN plane model, the ETSI [15] has further distinguished, within each plane, sub-services of a different nature that do not justify however a different communication network. The control plane includes Call Control Activities, Bearer (Connection) Control activities and Media control activities, dedicated to the communication between the participants various media coders and decoders. All these control plane sub-functions may however communicate through the same "control network". In the same way, the ETSI model distinguishes in the transport plane the transport services and the transport control functions from the transport user flows, although they all communicate via the transport network.

Another research group [16] has further refined the service plane sub-functions. In this model published under the name of **"Simpson model"** [21], where Simpson stands for "Signalling Model for Programmable Services over Networks", **"Multi-provider Services"** are taken in account. We define as "multi-provider service" a service built as a graph of independent service components of different nature and executed by different component providers. As an example we may consider a car manufacturer Virtual Private Network (VPN) service. In addition to VPN functions that may be supplied by an intelligent network service operator, financial components supplied by a banking company and inventory management components supplied by a specialised provider may be included. Such a service integrating service components from different suppliers in a single user interface is a

"multi-provider service". In order to take multi-provider services in account the service plane must include client sub-services, provider (integrator) sub-services and component sub-services. The Simpson model includes these sub-services in the NGN service plane as well as the ETSI sub-services of call control, bearer control and media control for the control plane and therefore the Simpson model is an unbundling model for the Service and the control planes.

The various Simpson levels constitute another unbundling scheme that we call a vertical unbundling. It should be noted however that for every horizontal service function that we have underlined in the horizontal unbundling scheme (Access, Transport, Intelligence) we have a vertical Simpson column.

Unbundling is therefore a 2-dimension problem as shown on the table of Figure 5. Each place in this table is actually an independent business opportunity. The condition for this however is that all parties shown on the table agree to work together in a cooperative manner. Benefits from this cooperation would be twofold: a) a richer service offer b) a controllable service complexity, each partner having a limited set of functions to develop.

Acces	**Transport**	**Intelligence**
Access Client	**Transport client**	**Intelligence Client**
Access Services provider	**Transport services provider**	**Intelligent network service provider**
Access component operator	**Transport Components**	**Intelligent network components**
Access session services	**Call and bearer control services**	**SSP, IVS**
Access transport network	**Transport network**	

Figure 5. *Horizontal and vertical unbundling dimensions*

3.3. *Signalling and APIs*

The SIMPSON model is a powerful model to identify and characterize functions, interfaces and mechanisms in the control and service planes. It is also a powerful modelling technique for many different service architectures. We will use this model here to identify two different types of interactions between control and service plane entities with the example of Parlay [17] services over a Soft-switch architecture [13].

Interacting entities may belong to the same service level and they operate in the peer-to-peer mode. We call **"horizontal signalling"** this type of horizontal communication within a single Simpson level. On the contrary, they may appear in adjacent service levels. We call **"vertical signalling"** this type of vertical

communication. Vertical signalling protocols are frequently referenced as **APIs** (Application Programming Interfaces). On the figure 6, the SIMPSON model shows the various vertical and horizontal signalling protocols required in the control and service planes architectures, identified by generic acronyms. In parenthesis we have given some examples of legacy protocols at the various levels.

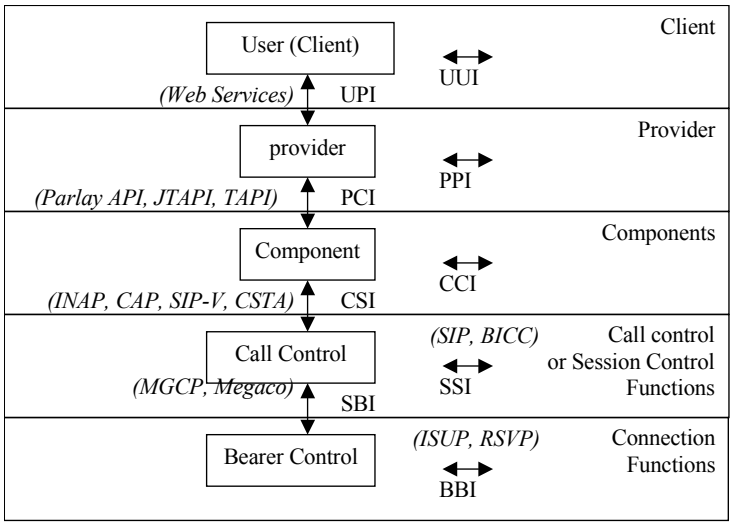

Figure 6. *Signalling protocols and APIs*

As APIs we find that the User to Provider Interface UPI may be implemented by Web services [18].

At the provider and component levels, the Parlay group proposes a new service architecture differing from the Intelligent Network IN architecture by a supplementary level of service customization. At the provider level a service integrates some component abstractions in a Parlay service platform. Some of these component abstractions are Call Control components. An example of the Provider to Component interface PCI is the Parlay API. By this API, a service provider invokes Call Control components in a Parlay gateway such as an Ericsson Jambala platform [19] belonging to a Call service operator.

At the Call service (or Session service) level and Bearer service level the soft-switch architecture implements a separation between call control services (performed by the Media Gateway Controllers MGCs) and bearer control functions (performed by the Media Gateways MGs). On one hand, IP connectivity operators (at the bearer level) provide customers with MGs and take care of the IP forwarding functions. On the other hand, Call Control operators (at the network level) operating MGCs outsource the Call Control functions or the IP-Centrex functions formerly performed by PABXs.

An example of Component to Session Interface CSI API could be the intelligent network INAP or CAP set of operation [20], by which a Parlay gateway may invoke the services of a Service Switching Point SSP control unit within a Soft-switch MGC.

Finally the Call control functions of the MGC request Bearer Services from Media Gateways MG by means of the Session to Bearer Interface SBI API. The MGCP or MEGACO protocols are examples of such SBI APIs.

As horizontal signalling protocols or peer to peer protocols we find in the client level the User to User Interface UUI type of signalling between Clients. SMS should be considered as medias and therefore do not enter in this category. User to user signalling, has been frequently mentioned, but has not been implemented so far.

We find in the provider level the Provider to Provider Interface PPI type of signalling between servers. Here again we cannot say that examples of such signalling protocols exist at the present time.

We find in the component level the CCI type of signalling between component providers. An example of CCI signalling is the MAP signalling between different mobile networks. Another example of CCI signalling is the SCP-to-SCP signalling of future IN capability sets, or the SCF to SDF signalling of IN-CS1.

We find at the Call service operator level the SSI Session to Session Interface types of signalling. The Session Initiation Protocol SIP signalling or the Bearer Independent Call Control BICC signalling are examples of SSI Call signalling protocols. Peer-to-peer services implement proprietary SSI Call signalling protocols to associate their users for file transfers.

Finally, in the bearer level, we find the Bearer to Bearer Interface BBI type of signalling. Examples of BBI signalling abound due to fact that the Plain Old Telephone Service POTS is indeed a Bearer control service. All Circuit Associated Signalling CAS protocols are actually BBI types of signalling protocols. The most important BBI signalling protocol at this point is the ISDN User Part ISUP signalling protocol widely used between telephone exchanges.

We may remark at this point that the main horizontal signalling protocols, implemented so far, belong to the lower levels of the SIMPSON level. This comes from the very centralized way in which services have been designed up to now. There is now a very sharp contrast between the cooperative computing of nowadays telephone exchanges and the centralized computing presently used in the service layers. An important direction for research is certainly to make the service layers more cooperative, which will require the development of horizontal signalling protocols in the upper layers of the SIMPSON model.

4. New signalling paradigm: towards a global control plane

4.1. *Signalling and open networks*

As networks should be fast, networks should be open. This means that a service initiated by a party in one network could be terminated on a party in another type of network without excessive complication. This means also that innovative services may result from the composition of service components located in different networks. The transfer of Media from one network to the other is rather simple because media are *stateless*. The creation of media gateway is therefore a workable problem. However, the transfer of signalling from one network to the other is a very complicated problem because of the way signalling is done in present day's networks. Today, signalling is generally understood as **"the invocation of remote operations or exchange of notifications between local processes of a global control process"** These operations are network specific and the commands for their invocation are *state-full*, which means that the effect of a command will depend on the state of the remote entity, making signalling conversion between different state machines in signalling gateways, at least a very difficult problem, and probably an impossible problem in the general case.

This present conception of signalling is clearly a limitation to innovation in the field of communication services, making cross-network service termination and cross-network service composition unsolved problems in the general case at the present time. Up to now, this limitation was not perceived or understood as a serious problem as network operators did have to cooperate with networks of other technologies. However this situation has to change now because media and services are bound to travel or to be extended from one type of network to another, from IP network to TDM mobile networks for example. It has been advocated that the problem will be solved by the substitution of the various technologies by a single one (replacement rather than convergence). This idea is the contrary of the open network idea and does not seem very realistic in a short to middle term. On the contrary, it is generally thought that there will not be a single NGN technology for quite a while. It seems much more fruitful to really face the problem of cooperative computing between service components of different technologies and different operators and therefore to tackle the signalling problem. Our thesis is that the alleged excessive complexity of the control and service plane will be solved by theoretical improvements in cooperative computing and therefore in signalling protocol theory. In this objective, we present now some new ideas for the development of a new unified cross-network signalling protocol.

4.2. *Chaining local contexts: The CAT concept*

We have explained that the global view of a conversational communication session, i.e. the complete information about it, is spread in all the local contexts, the global context is therefore made of a link list of associated local contexts. We have proposed a new scheme [21] for the binding mechanism: Processes involved in a same global service session have to associate their local contexts to build a Global Context for sharing instance data. In the same manner as a File Allocation Table FAT [22] links together sectors of a same disk to build a file, **a Control Allocation Table CAT links together the various local contexts in different platforms to build a global context.**

When a process needs to share session instance data with a partner process, it gets the binding reference of its peer context from the CAT. The CAT is a binding reference graph, distributed on all the associated contexts (see Figure 7). The CAT is a data structure persisting during the whole session duration. It is created as the contexts are opened up and is erased at session termination.

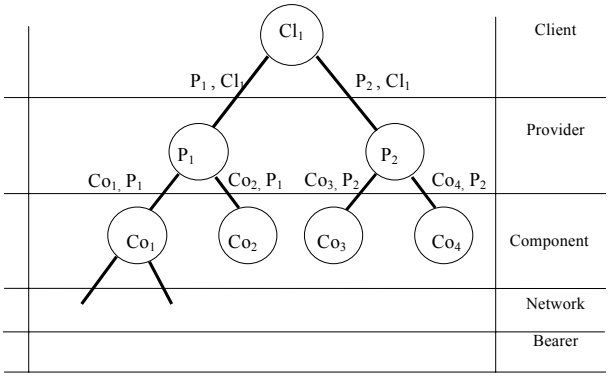

Figure 7. *SIMPSON view of a vertical CAT*

On the contrary of RMI references, the CAT pointers are dynamic and persist only during the service session. They are service instance dependent. In RMI, references to a remote object within a client are static and predefined by the programmer before the service execution. They are independent from the service session execution.

4.3. *A new signalling paradigm: The Cooperative Control Signalling Protocol (CCSP)*

Our concept of CAT allows a new definition of signalling. Considering signalling as the invocation of remote operations or exchange of notifications

between local processes of a global control process makes signalling state machine dependant and thus complicates the problem of signalling translation. To reduce this dependency, we propose a more general definition by which **"Signalling is the writing or reading of information by a local control process in remote parts of the global context"**. This new signalling paradigm is possible because contexts are linked by a CAT structure. In this view, we contend that a remote operation may be invoked by just performing **get/set/notify** operations on the value of a corresponding attribute in the remote context. With each attribute a signal field may block its modification because an other process is already working on this particular attribute.

We define the **Cooperative Control Signalling Protocol** (**CCSP**) as a new signalling protocol based on this new signalling paradigm. The CCSP has basically four methods: **"OPEN context"**, for the initiation of a new session in a remote partner, and as in the SNMP paradigm, simple **"GET"** and **"SET"** methods to read or modify a distant object. Unsolicited notifications are sent by the **"NOTIFY"** method.

This new concept assumes that a control process understands the syntax and the semantics of the information in the remote context. Contexts should therefore include a generic part, common to all services, followed by a service dependant part. The generic context structure is object oriented. Further work will contribute to the definition of the generic context and the UML Class diagram for the generic context class architecture.

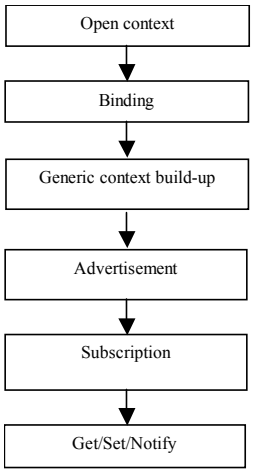

Figure 8. *Phases of a generic signalling mechanism (CCSP protocol)*

The Figure 8 summarizes the various phases of this new signalling paradigm. The implementation of the CAT graph is achieved when exchanging the initial signalling messages (OPEN context) method. The binding phase is followed by an advertisement phase where the remote context informs the local control process of

the content of its service dependant part. This advertisement phase indicates the syntax and the semantics of the information classified by known types.

After the advertisement, bound processes may subscribe for get/set/notification services. This subscription phase must, of course, be conditioned by standard security procedures. Signalling may then proceed as get/set/notify commands.

For performance and code universality the signalling protocol should be the least verbose as possible. Using object oriented programming; we can get the entire context, or a part of it, with one query. For example if we want to get the entire context we can use a GET method to download the Context object. The object is serialized and transferred over the network. The method *GET Context* will return an instance of the remote Context object. Downloading all the object is an option in some Object Oriented Middleware like RMI. In general the object is not downloaded and only remote method invocation is possible.

To modify an attribute value of a context, we first download the context (or a part of it), then do the modification locally by using *setMethodName(...)*. Once the modification is done, we send back (upload) the modified object to the remote process using the SET method (for *example SET Context*).

In the Web Services paradigm (and most of the Object Oriented Middleware) the object that a client is querying remains remote and is not downloaded to the client machine. The client has to remotely invoke its methods. On the contrary, in the new signalling paradigm implemented by CCSP we may download the full object (*GET Context*) and then read or modify its attributes. In case a modification has been done to the downloaded object we update it on the remote process by using a SET method (*SET Context*), with the result of the remote operation execution.

5. Conclusion and further work

In this work, we have given formal definitions for the control and service plane activities and we have underlined the cooperative nature of the control and service plane software. We have applied these concepts to the NGN architecture and to its Service, Control and Transport planes. We have identified the unbundling models for the Service and control planes and the cooperative software constraints deriving from these unbundled architectures. We have pointed out some of the problems requiring fundamental research for the progress of this field and we have focused on the signalling problem. We have shown that a new approach to signalling was necessary, and therefore we have proposed a new signalling paradigm and given the basic implementation principles for this new paradigm: the CCSP protocol.

Our further work is now to detail the generic context structure for all signalling domains: Access, Intelligence, Call and Bearer signalling domains. It is also to show its applicability to all the APIs and signalling paths identified by the SIMPSON model. The benefits expected from this approach should be to make cross-network services feasible, to allow a richer service offer, and to achieve a controllable service complexity, each partner having a limited set of functions to develop.

6. Bibliography/References

[1] "Principe général d'architecture d'un réseau NGN", Etude technique, économique et réglemntaire de l'évolution vers les réseaux de nouvelle génération, Etude réalisée par le cabinet Arcome pour le compte ART, Septembre 2003, http://www.art-telecom.fr/
[2] INTSERV: IETF RFC 1633; IETF RFC 2212 ; IETF RFC 2215
[3] DIFFSERV: IETF RFC 2474; IETF RFC 2475
[4] IETF RFC 3031: "Multi-protocol Label Switching Architecture", January 2001
[5] ITU-T Q.771_Q775 : TCAP : Transaction Capability
[6] CORBA : www.corba.org
[7] ITU-T recommendation Q 1701 Signalling requirements for IMT-2000 networks
[8] Global Functional Plane for Intelligent Network CS1. ITU-T Recommendation Q1213
[9] IETF RFC 3261: "Session Initiation Protocol (SIP)", June 2002
[10] ITU-T Recommendation H. 323
[11] ITU-T Recommendation Q1901: BICC: Bearer Independent Call Control
[12] TINA-C Consortium
(Telecommunication Information Networking Architecture) www.**tina**c.com/
[13] Soft-switch architecture : IETF RFC 2705: Media Gateway Control Protocol (MGCP) Version 1.0 Oct. 1999, IETF RFC 3015: Megaco Protocol Version 1.0, Nov. 2000
[14] Skype : http://www.skype.com
[15] TISPAN: http://www.item.ntnu.no/fag/ttm4130/NGN_Workshop_Protocol.ppt
[16] Astronefs: "Network and Telecommunication Global Service Convergence: White paper", http://www.infres.enst.fr/~rigault/white-paper.pdf
[17] Parlay : http://www.parlay.org/specs/index.asp
[18] W3C : "Web Services Description Language (WSDL) 1.1", 15 march 2001
http://www.w3.org/TR/wsdl
[19] Ericsson's Service Capability Server Parlay Gateway
http://www.ericsson.com/mobilityworld/sub/open/technologies/parlay/about/parlay_about_gs1
[20] CAMEL: ETSI recommendation TS 101 046 (V7.0.0)
[21] Softcom 2004 Claude Rigault, Rony Chahine:: "New signaling mechanisms for multi-provider and cross-network services"
[22] Silberschatz and Galvin: "Operating System Concepts", 2004:
http://www.cs.biu.ac.il/~wiseman/os/os/os12.pdf