Engineering of a broadband connectivity service: the TINA approach

N. Charton, Y. Hervé, N. Mercouroff

Alcatel Telecom Research Division

Route de Nozay

F-91460 Marcoussis, France

Tel: +33 (0)1 69 63 12 70

Fax: +33 (0)1 69 63 17 89

mercouroff@aar.alcatel-alsthom.fr

Abstract

The Telecommunications Information Networking Architecture Consortium, TINA-C, has defined a business model identifying domains for the future telecom industry. One of these domains is the connectivity provider domain, in which stakeholders will support the connectivity requirements of the services within the retailer domain.

The network provider offers the connectivity service to the retailer, that is, through the network to the service, via an interface. The interface supports the specifications of a reference point called the Connectivity Service Reference Point, ConS-RP. The ConS-RP allows for the request of connectivity in the form of a connection graph, independently of the implemented underlying infrastructure (ATM, PSTN, IP, etc.).

The ReTINA ACTS project aims at developing an industrial-quality distributed processing environment (DPE) for telecommunication applications. The ReTINA DPE thus offers a number of services targeted for this type of telecommunication application. Whereas some of these services offer extensions to traditional CORBA services, such as trading and notification services, others support

specific requirements from TINA applications, connectivity requirements in particular.

In 1996, a TINA connectivity service implementation was provided for the ReTINA DPE, based on connection management applied to ATM networks. The ReTINA activities include several trials (BVPN service, information service) experimenting the ReTINA DPE implementation, in particular its DPE connectivity service. Feedback from this experimentation allows preliminary observations to be drawn on the service requirements. In particular, it has been shown that performance is a key component for a usage not limited to the sole support of network planning and configuration.

Several solutions for exhibiting sufficient performance of the connectivity service are envisioned. Among these solutions, careful engineering of distributing the connectivity service over CORBA is a key issue. One of the engineering angles covers the definition of the API through which telecommunication services request connectivity. The API takes the form of IDL specifications, and needs to follow the TINA specifications for the ConS-RP. Building a connectivity graph potentially demands several interactions between telecommunication services and a connectivity service. To reduce the number of interactions, which may be related to operation invocations across international networks, it is proposed to factor repetitive invocations into a single, multi-parameter operation invocation.

Regarding the implementation of the connection management itself, it is proposed to tune its engineering to allow for parallelism and concurrency. With parallelism, simultaneous routing can be established in subnetworks belonging to a same network. Concurrency allows for the treatment of simultaneous requests to the connectivity service. While the former decreases the overall response time in connectivity establishment, the latter minimizes the blocking of requests to the connectivity service.

A final aspect of connection management engineering is related to the general deployment of the service over the network. Implementing connection management implies inherent distribution of intelligence. One part of this intelligence is dedicated to the support of connectivity provision to telecommunication services through the ConS-RP interface. Cautious engineering practices advise putting this intelligence as close as possible to the telecommunication service components requesting the connectivity. Other parts of this intelligence are dedicated to the control and management of resources belonging to the network, such as subnetworks and network elements. Once again, a good engineering practice entails locating these parts as close as possible to the resources of concern. This implies contradicting requirements on the implementation which should be handled with care.

The article concludes with the need to support the approaches presented by current trials to find an efficient balance between intelligence distribution over the network and overall performance.

Keywords

TINA, ATM, connectivity service, interfaces, engineering, performance, distribution, reference points

1 TINA CONNECTIVITY SERVICE

TINA provides a common telecommunication software architecture to be used in a multi-stakeholder, multi-domain environment. It provides TINA system implementors with a consistent specification of the prescriptive parts of the architecture that meet the conformance requirements of TINA. These specifications are defined in the form of a set of architectural principles, a business model, and specifications of Reference Points (RPs). A reference point comprises a set of interfaces describing the interactions that take place between the various TINA entities identified in the TINA business model, responsible for performing a given task.

1.1 TINA business model and reference points (RPs)

The TINA Business Model identifies the types of “businesses” in which TINA stakeholders are involved (TINA-C, 1996). It includes the definition of the consumer, the retailer, the broker, the third-party (3Pty) service provider, and the connectivity provider (see Figure 1). In particular, a stakeholder in the business of connectivity provider is the owner (manager) of a transport network (switches, cross-connects, bridges, routers, and trunks), controlled by the TINA connectivity service.

- A Contract Profile Management (CPM) service: It retrieves and modifies the contract profile associated with each client.

TINA Network Resource Architecture (TINA-C, 1997-a) defines a session as the temporary relation among a group of resources that are assigned to collectively fulfill a task for a period of time. Thus, a connectivity session serves as an environment for establishing and managing network flow connections. TINA handles these connections through the concept of Physical Connection Graph, defined as a set of network flow connections transporting information across the network between flow endpoints.

A network flow connection is characterized by a number of parameters such as an identifier, its topology (either point-to-point bi-/uni-directional or point-to-multipoint uni-directional), the list of endpoints (source and leaves), the traffic type (Constant Bit Rate, Variable Bit Rate, ...), routing constraints, etc. The endpoints of a network flow connection can also be associated to characteristic information, such as the endpoint type (root or leaf), a name, the maximum or average transmission rates, and so on.

1.3 TINA connection management architecture

For implementing a connectivity service over an ATM network, TINA has developed specifications, derived from existing standards such as ITU-T Recommendation M.3100 (ITU-T, 1992-a) and G.803 (ITU-T, 1992-b), under the name of *TINA Connection Management Architecture* (CMA). From a computational viewpoint, CMA is defined by a set of operational interfaces, which specify the operations supported by the Connection Management (CM) objects deployed and interacting over the DPE. Operations are grouped in interfaces according to functional considerations, and independently of distribution aspects of the objects that will support the interfaces (TINA-C, 1995). Interfaces are supported by the following objects (see Figure 2):

- The Connection Coordinator (CC) which coordinates various networks. It offers its clients an interface in compliance with the ConS-RP for specifying end-to-end connectivity in the form of a connection graph.
- The Layer Network Coordinator (LNC) provides connectivity in a layer network by offering an interface to clients to create and manipulate trails.
- The Network Management Layer-Connection Performer (NML-CP) and Element Management Layer-Connection Performer (EML-CP) provide their clients (usually LNC objects or an NML-CP) with a usage interface for the control of Subnetwork Connections. The interfaces defined for the CP is expressed in terms of Subnetwork Connections.

Figure 2 Global connection management architecture.

2 RETINA ACTS PROJECT

The success of the TINA approach to large-scale information networking is predicated, among other things, upon the availability of flexible and scalable DPE technology, and on its seamless deployment in different, heterogeneous network equipment. The ReTINA project is a three-year R&D project launched in September 1995 under the auspices of the European ACTS program. Its main goal is the development of DPE specifications and associated software technology that meet the above requirements. The project has three related objectives:

1. Develop an industrially sound, TINA-compliant Distributed Processing Environment, together with its specifications and an associated set of development tools.
2. Demonstrate and experiment with information network services implemented using the project's DPE.
3. Use the project's DPE specifications as a basis for information networking standards.

The ReTINA project implements:

- The ReTINA TORB (Telecom Object Request Broker) as a combination of software tools and libraries providing abstractions for the management and control of resources made available by the underlying operating systems.
- The ReTINA real-time profile, consisting of an implementation of the ReTINA TORB on a distributed real-time operating system microkernel (CHORUS) with full support for real-time programming, resource management, and temporal quality of service guarantees.
- The ReTINA general-purpose profile, consisting of an implementation of the ReTINA TORB on standard operating systems (UNIX).
- Generic computing and telecommunications services, including the implementation, based on the ReTINA TORB, of object-based data management services (with transactions, queries, and persistence) and generic connection-management services.
- A set of associated application development tools, targeted at the ReTINA TORB, and providing support for information and computational notations used within TINA.
- A set of service demonstrators, built on the ReTINA TORB, using the ReTINA profiles and generic services.

The partners in the project are Alcatel Telecom, Chorus Systems, Siemens, Hewlett-Packard, CSELT, France Telecom, British Telecom, Telenor, O2 Technology, Broadcom, and Lancaster University. In Figure 3 below, the overall workpackage structure, as well as a rough estimate of the distribution of effort among the partners is shown. In the figure, v0, v1, and v2 refer to the main iterations of the basic cycle (requirements - specifications - prototyping) which are planned for the workpackages.

Figure 3 ReTINA DPE structure.

The focus here is on generic telecommunications services in the DPE. The telecommunications industry has expectations from ReTINA DPE to provide a solution to support large, distributed, interactive multimedia applications. Therefore, one of the activities of the project focuses on DPE telecommunications services with particular emphasis on TINA connectivity services (such as Connection Management functions necessary to set up, maintain, and release connections). The objective of this activity is to provide a **ReTINA Connectivity Service** implementation as a core component of the DPE.

An initial implementation of this DPE connectivity service, taking advantage of ongoing work and existing software packages from several partners and from TINA, has been developed with ReTINA. Some conclusions on its specifications can already be drawn.

3 LESSONS LEARNED FROM THE INITIAL IMPLEMENTATION

The prototype of the ReTINA Connectivity Service complies with the first release of TINA-C Connection Management Architecture (TINA-C, 1995), and hence follows the computational specifications described as interface IDL specifications. The prototype was implemented using a commercial Object

Request Broker (ORB) as DPE kernel. No specific effort was made to optimize performance and scalability.

An important issue at the design stage is the granularity of objects. CORBA Object model makes a clear distinction between:

- Objects which are only described by their interfaces and cannot be passed on the network as request parameters, and
- Data structure that flows over the network as request or reply parameters.

For these reasons people sometimes qualify CORBA as Object-Based instead of Object-Oriented.

Furthermore, to cope efficiently with marshalling problems, complex structures such as graphs cannot be directly and simply described as IDL structures (this problem is somewhat similar to the problem of representing complex data structure in relational databases).

Thus, although ORBs define interactions between distributed components in a language-neutral platform, independent of the transport layer, experiments have shown that interfaces must be written with great care: A "naive" design of IDL interfaces can lead to a functionally correct system yet with poor performance.

In the initial implementation of the ReTINA Connectivity Service prototype, clients interact with the connectivity service through the ConS-RP to define a connectivity graph representing the parties involved in point-to-point or point-to-multipoint connections. More precisely, a client requests some factory interface of the connectivity service to instantiate a connection graph. The reference of the graph object is returned to the client who then sends several requests to define the graph (see Figure 4). Several aspects have made this approach appealing:

- Only interactions with the graph objects are specified, which gives more flexibility for the implementation.
- The graph object itself never flows over the network. Instead, simple operations are invoked to define, modify, and control the graph. These operations have simple parameters (for example, adding a port).

Figure 4 The ReTINA connectivity service (first release).

A solution of this type is suited and performs well when the client and the server are co-localized. Unfortunately, in the general case (distributed remote objects), the solution has turned out to be inefficient.

Indeed in the ReTINA first release, nine remote operation invocations are required between the client and server objects to establish one end-to-end connection. Note that other interactions are required between objects (client and server) within the connectivity service itself. Experimental measures showed that each remote invocation was consuming an "incompressible" amount of time, and that several small requests between remote objects were more costly than a single complex request between them.

Note that better performance would be achieved if the client could instantiate and construct the graph object locally and eventually move it to the connectivity service. Two services are defined in CORBA specifications for this purpose: Externalization and Object LifeCycle. Externalization service enables objects to export their state on a stream, thus giving the opportunity to rebuild the object somewhere else. LifeCycle service permits objects to be migrated across the network. However, neither of these services is currently on the market.

In addition to the design of object interfaces, other implementation issues limit the efficiency of the ReTINA Connectivity Service. In particular, requests are processed sequentially which prevents the current release from being scalable. The main consequence is that the duration of a connection establishment increases with the number of involved network elements.

Finally, experimental results showed that two factors were limiting performance. The first factor deals with the design of computational object interfaces, while the second is related to implementation issues.

4 SPECIFICATIONS OF TINA CONNECTIVITY SERVICE INTERFACES

One of the solutions for exhibiting sufficient performance of the connectivity service is the re-design of object interfaces. The main idea is to reduce the number of required invocations between remote objects. Indeed, as described in the previous section, a CORBA application spends an incompressible part of its processing time on the kernel transport network (DPE network), transporting messages between remote objects. Therefore, interfaces of remote objects should be designed to allow minimal interactions while providing the required service.

It is proposed to replace sequential and repetitive, simple-operation invocations (those with one parameter) by a single, multi-parameter operation invocation.

Telecommunication applications (or Retailers) request the establishment of a connection graph, that is, they use the connectivity service, interacting with a Connectivity Provider. From the computational viewpoint, such interactions correspond to operation invocations from a client object (in the Retailer domain) on server objects (in the Connectivity Provider domain), through interfaces. TINA proposes that the clients interact with the CC (Connection Coordinator) computational object through its Conn_Session_Control interface.

Although building a connection graph might potentially demand several interactions between clients and the connectivity service, TINA proposes a single, rather complex operation, namely the Setup_Flow_Connections operation. This operation is used by the clients to establish a connection graph (one or more flow connections) passing the required information attached to each flow connection as input parameters of the operation. Then the reference of the graph object is returned to the client who may access it to modify or control it (see Figure 5).

Figure 5 The connectivity service as defined by ConS-RP.

The graph is built by the client object and is described using data structures that flatten the graph. This solution may seem less elegant than the one applied in the first release of ReTINA Connectivity Service. However, it is more pragmatic since a connection graph can be built with a single invocation (instead of nine) between the client and the connectivity service. This approach should eventually improve the connectivity service performance.

5 PARALLELISM AND CONCURRENCY OF TINA CONNECTIVITY SERVICE

The choices of the first prototype were to keep non-functional features as simple as possible. However, several aspects related to performance could be improved at the expense of a more complex implementation. More precisely, performance could be improved by:

- processing multiple requests concurrently;
- processing the establishment of subconnection in parallel.

The first aspect is expected to increase the throughput of the whole connectivity service, while the second would both decrease the perceived latency of each individual client and increase the scalability of the service. Indeed when subconnections are processed in parallel, the expected time needed to set up the whole connection is about equal to the time needed to set up the slowest subconnection, instead of being equal to the sum of the subconnection times (as in the current implementation).

The next paragraph briefly presents the connection establishment algorithm. Then technical issues and possible trade-offs to implement concurrency and parallelism inside a CORBA architecture are discussed.

Figure 6 illustrates the connection establishment algorithm. A connection is basically created in three steps:

- Step 1 consists of a routing algorithm (a variant of Dijkstra) that finds a route for the connection. Routing decisions are made according to resource status (for example, available bandwidth) locally cached for efficiency. Consistency of cached information is ensured by the hierarchical nature of the architecture. During this step no I/O operations are performed and only shared resources status can be read.
- Step 2 consists of actual reservation of the resources. Write locks must be acquired for all resources along the route and a local copy of resources be updated.

Note that separating route computation (Step 1) from resource reservation (Step 2) may simplify the backtrack of the routing algorithm, since only resources are read during the first step.

- Step 3 consists of sending CORBA requests over the network to the inferior connectivity layer. This scheme is recursive until single network elements are reached. The thread processing the current request is blocked until every subconnection is established.

On reception of a "create connection" request:

1. Find an admissible route based on available resources.
2. Reserve necessary resources for the establishment of this route.
3. Ask each subnetwork chosen by the routing algorithm to create its own subconnection.

Figure 6 Connection establishment algorithm.

Concurrency

Processing several requests simultaneously requires a multi-threaded Object Request Broker (ORB). This feature is not prescribed by CORBA specifications, but several ORB providers do offer this functionality. Some systems even offer the choice between different threading models (thread per object, thread per request, pool of threads) allowing fine tuning according to the application's needs. Note that the threading model considered here is preemptive.

Locking issues

Locking is not only fundamental for correctness of the program but also for achieving acceptable performance. A misconception of locks can even result in a degradation of performance compared to the same single-threaded application. Basically there is a subtle trade-off between the overhead imposed by fine-grain locks and expected benefit from concurrency.

Several approaches are proposed to control the access to shared resources (for example, endpoints with a given amount of available bandwidth) in the connectivity service. Among them are the transaction-like two-phase locking and the optimistic locking approaches.

▽ *Transaction-like two-phase locking*

A solution is to enclose Step 1 and Step 2 (see Figure 6) into a single subtransaction. Subtransaction here means that the Atomicity, Consistency and Isolation properties—from the usual ACID properties—are respected (Gray, 1993) whatever the actual implementation. Read locks are acquired during Step 1 and Write locks during Step 2. All locks are released after Step 2 has been completed. Granularity of the locks can differ, as follows:

- Coarse grain. All resources are protected by a single lock. This approach is expected to perform well in a single-processor host, because there is at most one thread (for the purposes of this article, called *T*) at a time executing the code of Step 1 and Step 2. The other clients are blocked until the subtransaction has completed for *T*. Doing this limits the number of unnecessary thread-context swappings. Only threads in Step 3 that have received all their replies can preempt *T* to terminate their job.
- Fine grain. Single resources have their own locks. Doing this increases concurrency and allows several processors to calculate routes simultaneously. This solution has the drawback of potentially giving rise to deadlock situations.

▽ *Optimistic locking*

Traditional locking policies are sometimes called pessimistic. This means that the application carefully acquires locks before manipulating shared resources. Optimistic locking makes the assumption that conflicts are rare, and that it is more efficient to execute the operations without acquiring locks and verifying at commit-time that no conflict has occurred. In case of conflict, the transaction must be undone and can be restarted.

The same kind of reasoning can be applied for the connectivity service: No locks are acquired in Step 1, and the resource status is potentially stale. For instance, an endpoint may have less bandwidth left than indicated by its status. When a route has been determined, the thread enters Step 2 and acquires exclusive Write lock, checks that the up-to-date amount of resources is sufficient for establishing the route, and reserves the resources. If the actual capacity of the resources is not sufficient, the thread would have to execute Step 1 once again to find a new admissible route.

6 TINA CONNECTIVITY SERVICE DEPLOYMENT OVER THE NETWORK

A final aspect of the connectivity service engineering is related to the deployment of the service over the network. Indeed the connectivity service controls (with respect to the setup, modification, and release of connections) the whole network, with its set of subnetworks and network elements (for example, ATM switches). Therefore it is by essence a distributed application.

Telecommunication applications uses the connectivity service through the ConS Reference Point, interacting with connectivity service objects. Therefore cautious engineering practices advise putting these objects (namely the CC object) as close as possible to their client.

Simultaneously, other connectivity service objects (namely the EML-CP objects) directly control network elements. The interactions between the EML-CP object and the controlled switch usually requests adaptation functions (since switches do not provide IDL interfaces) and specific communication technology (such as the SNMP protocol). Again, good engineering practice entails locating these objects as close as possible to the resources of concern.

Then engineering decisions should consider the above contradicting requirements: the need to co-localize client and server objects for better performance; the need to localize some connectivity service objects near the applications and other objects near the switches. Other elements may also be considered, such as the characteristics and capacity of the available physical equipment (such as the CPU, microprocessor type of computers and switches). Note that if a switch features an efficient control station, it would be interesting to implement the EML-CP object on it. Furthermore, different administrative domains may co-exist implying a given distribution of objects according to their controlling authority. Finally, engineering decisions should also follow the global objective of the designers who may choose to focus on performance aspects or on other aspects such as security. Thus implementing a connectivity service implies inherent distribution of intelligence. However, to handle this distribution and engineer the connectivity service, there is no strict rule, only a set of requirements, constraints, and objectives.

7 CONCLUSION

After one year of development and trials within ReTINA, it is already possible to gain some experience on TINA connectivity service implementation, and through it, on CORBA application design. Indeed, as an application requiring distribution of intelligence over the network, the TINA Connectivity Service proves to be a good case study for requirements over CORBA-based DPEs. Potential pitfalls, needs for careful design, and possible approaches have been highlighted, which could be applied to many CORBA applications. Such experience is currently under analysis within ReTINA for improving DPE kernel, services, and support tools to telecommunication services.

Similarly, the development of design practices for tuning the computational and the engineering specifications of a key TINA component, the connectivity service, has been initiated. In the domain of computational models, results have already been fed back to TINA-C, and improvement of some of its specifications have already been taken into account (TINA-C, 1997-b). The on-going experimentation within ReTINA now focuses on developing and implementing solutions to tackle the challenge of finding the right balance between intelligence distribution over the network, and application performance and scalability.

8 REFERENCES

- Gray, Jim and Reuter, Andreas (1993) Transaction Processing: Concepts and Techniques. Morgan Kaufmann Publishers.
- ITU-T (1992-a) Generic Network Information Model. ITU-T Recommendation M.3100.
- ITU-T (1992-b) Architectures of Transport Networks Based on the Synchronous Digital Hierarchy (SDH). ITU-T Recommendation G.803.
- TINA-C (1995) Connection Management Architecture. TN_JJB.005_1.0_94.
- TINA-C (1996) TINA Business Model and Reference Points. EN_RCJ.030_3.1_96.
- TINA-C (1997-a) Network Resource Architecture v3.0. NRA_v3.0_97_02_10.
- TINA-C (1997-b) The ConS Reference Point v1.0. Draft document.

9 ABBREVIATIONS USED IN THIS ARTICLE

ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programmer Interface
ATM	Asynchronous Transfer Mode
BVPN	Broadband Virtual Private Network
CC	Connection Coordinator
CCS	Connectivity Control Service
CM	Connection Management
CMA	Connection Management Architecture
CO	Computational Object
ConS-RP	Connectivity Service Reference Point
CORBA	Common Object Request Broker Architecture
CP	Connection Performer
CPM	Contract Profile Management
CPU	Central Processing Unit
DPE	Distributed Processing Environment
EML	Element Management Layer
IDL	Interface Definition Language
IP	Internet Protocol
LNC	Layer Network Coordinator
NML	Network Management Layer
NRA	Network Resource Architecture
ORB	Object Request Broker
RP	Reference Point
SNMP	Simple Network Management Protocol
PSTN	Public Switched Telephone Network
TINA	Telecommunications Information Networking Architecture
TINA-C	Telecommunications Information Networking Architecture Consortium
TORB	Telecom Object Request Broker

10 BIOGRAPHIES

N. Charton

Nathalie Charton is a 1992 graduate of the Ecole Nationale Supérieure des Télécommunications (ENST), Paris, France. From 1992 to 1995, she worked as a researcher in the ENST networks department; in 1995, she completed her PhD entitled "Definition of manageable components of services".

In 1996, she joined Alcatel Telecom Research Division in France. She is currently working on the definition and assessment of architectures for future advanced networks, in particular for TINA-based networks. She is also active in the European ACTS project ReTINA, for which she is defining and developing connectivity services.

Y. Hervé

Yann Hervé is a 1994 graduate of the Université Pierre et Marie Curie (Université de Paris VI), France.

He worked for one year at the Institut National de Recherche en Informatique et Automatique (INRIA), the French national institute for research in computer and control sciences, in the area of distributed shared memories applied to distributed domestic-waste collection.

In 1995, he joined Protectic Ingénierie, a French company specializing in CORBA and Distributed Object Technology. He works there today as a technical assistant for Alcatel Telecom Research Division.

N. Mercouroff

In 1986, Nicolas Mercouroff graduated from the Ecole Polytechnique, France. From 1986 to 1991, he worked at the Laboratoire d'Informatique de l'Ecole Polytechnique (LIX), France, and at Brandeis University, Massachusetts, U.S.A. on program validation. He completed his PhD there in 1990.

In 1991, Nicolas Mercouroff joined Alcatel Telecom Research Division. Since then, he has been involved in many projects related to the fields of information technology and telecommunications integration. In 1994 and 1995, as member of the core team of the TINA Consortium, he was in charge of the group working on computing environment specifications. Today he is responsible for TINA and multimedia within Alcatel Telecom Research Division in France, and is the work-package leader in the European ACTS project ReTINA.