# An Architectural Framework for Supporting Distributed Object Based Routing

Dhavy Gantsou
University of Valenciennes
F - 59313 Valenciennes Cedex 9
Phone : +33 327 511 944
Dhavy.gantsou@univ-valenciennes.fr

## ABSTRACT
TCP/IP routing protocols essentially implement distributed algorithms. Traditionally, the C and C++ programming languages have been used for implementing software supporting these protocols. Since the semantics of these languages do not provide adequate supports to cover concurrency, real-time and intrinsic properties of networking systems, protocols software are designed according to classical schemes where operating system features are extensively used to overcome C or C++ limitations. These models have served the Internet extremely well .nowadays However, as the current Internet routing system is evolving to address new requirements, so many software design methodologies. This paper shows an Ada95 technology based approach. It describes a distributed object based prototype of routing protocols.

## Categories and Subject Descriptors
C.2.1 [**Network Architecture and Design**]. C.2.2 [**Network Protocols**]. C.2.4 [**Distributed Systems**]. D.1 [**Programming Techniques**]. D.3.3 [**Language Constructs and Features**]

## General Terms
Languages, Design .

## Keywords
Ada95, Distributed System Annex, Glade, TCP/IP, Routing protocols, Distributed objects

## 1. INTRODUCTION
Routing protocols consist of many algorithms that execute concurrently to address the path computing and the forwarding problems. Although distributed applications are ubiquitous, implementing reliable routing protocols software remains challenging. Most fundamentally, concurrency introduces inter-process communication, race condition, shared memory management, and synchronization. Additional difficulties arise from intrinsic network's properties: asynchronous and real-time communication, loses of message, error, complicated message formats that may consists of large number of bit, or byte fields of varying length, heterogeneous data representation, and so on.

As new applications are emerging, and high-speed network technology is being widely used, the Internet architecture is called to evolve towards a new ones [5], [6]. This implies to address two major issues. First, new protocols have to be deployed. Second, new mechanisms have to be developed to adapt current protocols to new network technology. Deploying new protocols or adapting existing ones to a new architecture introduces a wide range of problems. This paper presents a distributed object based prototype in order to achieve this goal. It has been implemented using Ada95 technology [7], [8].

The rest of the paper is organised as follows. A brief overview of the model of the routing protocol software's architecture and the factors affecting its design are presented in section 3. The prototype implementation is explain in section 4, while section 5 concludes the paper.

## 2. ARCHITECTURE OVERVIEW
Current Internet protocols were guided by an architecture's network developed in the 1980's. At that time, C was considered as the most suitable language for underlying software's implementing, whereas it was written with plenty old legacy systems in mind leaving many aspects of the language dependent from the implementation. In fact, software developed using C have two major drawbacks that are imputable to the weakness of the C semantics. First is the use specific code for handling error. Second is the extensive use of native operating system mechanisms in dealing with interrupt processing, buffer manipulation, memory management, inter-process communication, and synchronization [1], [2].

The combined use of C++ and the object oriented technology was considered as an alternative to the design model based on C. To deal with memory management, concurrency, communication, and synchronization, corresponding native operating system mechanisms are abstracted in terms of classes or objects [3], [4]. This makes it easy to manipulate operating system constructs, but does not enhance protocol performance. Although it offers features for modern software engineering such as object-oriented


*SIGAda'02*, December 8–12, 2002, Houston, Texas, USA.

programming, exception handling, and templates, C++ lacks strong typing, concurrency, real-time, and distributed programming supports. Thus, concurrency and related critical aspects like timer management are handled using low-level operating system mechanisms. As result, software components involving concurrency are less efficient than those implemented using Ada95.

Another difficulty that is common to networking software design and has strong influence on the implementation of routing protocols is the complicated data formats. Routing protocol entities exchange data that consist of a number of bit fields of varying length. In C/C++ protocol implementation these fields are declared using simple types. Specific rules are needed in order to provide mapping between the logical view and the realistic packets [1], [2]. This is time consuming and prone to errors. The use of standard Ada95 typing, namely record and representation clause, avoids dealing with irrelevant complexities due to conversion needed for mapping, while providing a realistic packet formats.

The evolution of Internet from a homogeneous group of academic researchers to a broad set of participants reflects contradictory interests and objectives. This situation leads to a variety of issues. Implementing software for addressing these issues requires techniques and mechanisms that are exceeding the ability of design approaches based on C or C++. Among these issues are:

•the desire of many edge networks to be connected to several Internet Service Provider (ISP). For a routing entity, being "multihomed" means the ability to resolve physical, logical, and policy constraints.

• user empowerment : the ability for each internet user to have control over routing. Achieving this goal requires to answer following questions:

(i)How to design inter-domain routing system if, in contrast to today's Internet, the routing system is called on to resolve physical, logical, and policy constraints ?

(ii)how to design protocols if routing and management are under the control of the user ?

(iii) which methodological approach to use ?

Parts of issues related to (i) and (ii) can be addressed if we answer the latest question which is concerned with the structure and the behaviour of the main routing system.

"Multihoming" and user "empowerment" mean the placement of user controllable functionality in the routing node of a network. In contrast to current architecture, this highlights the fact that the entities of the routing system are active parts and so, they behave as communicating entities of a distributed system. In turn, this leads to conclude that related issues have to be addressed like those encountered in distributed computing environment. To take advantage of object-oriented techniques, we model the routing system in terms of distributed objects.

Another difficulty that a routing protocol implementation has to resolve is the network byte-ordering problem. These problem results on different data representation formats due to different hardware. For instance, on RISC processors that are built into Unix server the highest significant byte of a 32 bits integer is at the end of a four-byte sequence. The same integer has its lowest-significant byte at the end of the sequence on an Intel-based Windows NT machine.

This co-existence of two alternative data representation requires to encode/decode data before/after they are exchanged among hosts that may possess heterogeneous processor byte-orders. To enable network byte-order, sequential C/C++ protocol implementation have to manually do a considerable amount of byte and word swapping [1]. This is time consuming and prone to errors. In distributed object-based software, designer have not to worry about resolving data heterogeneity since network byte-order is carried out by the target middleware, provided it performs the presentation layer functionality. Unlike the Ada95 reference manual, which does not require a distributed system annex (DSA) implementation to support system heterogeneity, but like other implementation [9], [10], [11], [12], Glade [8] offers a set of XDR-like operations performing network byte-order.

Above all, software efficiency depends on the semantics of programming language used for the implementation. Although it is not commonly used in for implementing networking software, we have chose the Ada95 technology for several reasons. Besides strong typing, Ada95 combines object orientation, concurrency, real-time, exception handling, generic, and language interfacing facilities within a single language framework. Thanks the strength of its semantics, a lot of issues related to C/C++ implementation of routing software are easily addressed on the language level. In addition to offering a variety of principles, methods and tools that help to alleviate much of the complexity associated with network programming in C or C++, Ada95 provides the Distributed System Annex (DSA) for implementing distributed application. For this end, we have used :

• Glade, a freely available, open-source middleware allowing the implementation of both distributed applications using remote procedure calls or distributed object-based applications. Glade is built on top of

• gnat which also is a freely available, open-source Ada95 compiler. On of the salient aspect of Glade is that it allows the user to develop his application the same way whether the application is going to be executed on a network of PC and/or workstations, or on a single processor.

For distributed execution, the model uses

•Remote_call_interface (RCI)**,**

•Remote_types (RT), and

•Shared_passive (SP) categorized Glade partition as objects [10]. Communication between objects may be synchronous or asynchronous. It is supplied by :

•Remote Procedure Calls for RCI partitions.

•Remote objects invocation for RT partitions.

# 3. PROPOSED ARCHITECTURE

This section shows how a part of the proposed architecture can be used to implement distributed object-based routing protocols. We have chosen the OSPF [13] routing protocol since its

implementation highlights a wide range of issues that are common to distributed routing protocols. The prototype software is a collection of following classes:

- the class ***Router_Class***: specifies the behavior of distributed objects implementing routers. Router_Class is a virtual class defining three abstract methods. Subclasses derived from Router_Class must implement these methods that perform basic routing services. The first basic routing service is to compute the best path that a packet should take for reaching its destination. The second service is to actually forward packets received on an input interface to the appropriate interface for transmission across the network. The third major routing service is to temporarily store packets to absorb the bursts and temporary congestion.

- class ***Ospf_Router*** implements OSPF routing protocol functionality. In addition to methods performing basic routing function, the class Ospf_Router must provide operations implementing ospf-specific services such as managing simultaneous access to designated router, communication between routers, and maintenance of shared routing information. Thus, the class Ospf_Router must inherit methods from two classes: Router_Class and Shared_Object. To do so, we defined a generic package parametrized by a formal tagged type. Then, this type is used within Ospf_Router class for creating an instance of Shared_Object. Ospf_Router and Router_Class, from which it is derived, are implemented as RT objects.

- class ***Shared_Object*** : This class implements methods that are needed for performing query and/or modify operations related to the designated router concept. Given a set of OSPF routers that are connected on a broadcast network such as an Ethernet, a designated router is a single member of the set, which is elected by its neighbors to serve as central point of contact for routing information exchange. Since the DR concept requires to perform shared memory related services, it has been implemented as Shared passive categorized unit.

- class ***Service_Manager*** : allows registration of distributed objects. These objects are called on to communicate for requesting (client objects) or provide (server objects) data in order to perform protocol services. Client objects must identify objects from which they request services by an object reference. Since glade does not provide this function, manager also implements a set of operations that help to find and retrieve references to distributed objects or services.

# 4. CONCLUSION

Network protocols implementation is one of area where Ada technology is not commonly used despite its potential advantage of having a semantics which covers a wide range of internetworking problems. Given the availability of C and C++ TCP/IP application programs and utilities, one may well wonder if it is advisable to use Ada for implementing network protocols. This is right if Ada is used for writing code which is a simple translation of existing C/C++ software. We did not follow this approach. We prefer to implement protocols from scratch, while our goal is the development of protocols that outperform those written in C/C++. For example, instead of using data structures that are guided by the C/C++ semantics, we chose to use adequate Ada features for implementing networking entities. As result, some algorithms are different from those commonly used.

We ran parts of our model successfully on a network of workstations. We identified some challenging aspects of Ada technology that incite us to use it for implementing networking systems in other research areas such as Mobile Ad Hoc Networks (MANET) or next generation internet architecture.

# 5. REFERENCES


[1] Douglas E. Cormer, David L. Stevens Internetworking with TCP/IP, Volume 2, Design, Implementation, and Internals. Third edition. Prentice Hall, 1999.

[2] Gary R. Wright, Richard Stevens. TCP/IP Illustrated, Volume2. The Implementation. Addison-Wesley 1995.

[3] N.C. Hutchinson & al. Tools for Implementing Networks Protocols. Software Practice and Experience, Volume 19, 895-916.

[4] D.C. Schmidt, B. Stiller, T. Suda, A. Tantawy, M. Zitterbart. Language supports for flexible Application tailored Protocol Configuration. in proceedings of the 18th conference on Local Computer Networks, September 1993, 369-378.

[5] David D. Clark, Karen R. Sollins, John T. Wroclawski. Developing a Next-Generation Internet Architecture: New Arch. MIT Laboratory for Computer Science. March 2002. http://www.lcs.mit.edu/research/researchabstrac.php

[6] Bob Braden. Architectural Principles of Internet. IPAM Tutorial. March 12, 2002. http://www.isi.edu/newarch

[7] ISO Information Technology. Programming Language Ada ISO/IEC/ANSI 8652:1995.

[8] Laurent Pautet, Samuel Tardieu. Glade User's Guide. Glade version 3.14p, January 31, 2001. ftp://ftp.cs.nyu.edu

[9] Jens-Peter Redlich & al. Distributed Object Technology for Networking. IEEE Communication Magazine, October 1999, 100-111.

[10] Dhavy Gantsou Targeting Ada95/DSA for Distributed Simulation of Multiprotocol Communication Networks. In proceedings of ACM SIGAda 2001 Conf.. Bloomington, MN, USA. September 30- October 4, 2001, 91-96.

[11] Java Remote Method Invocation Specification. JavaSoft, revision 1.50, JDK1.

[12] Douglas C. Schmidt. ASX: An Object-Oriented Framework for Developing Distributed Applications. in proceedings of the 6th USENIX C++ Conference. Cambridge, MA,1994, April, 11-14.