IS THE INTERNET AN UNFINISHED DEMO? MEET RINA!

Eleni Trouva[1], Eduard Grasa[1], John Day[2], Ibrahim Matta[2], Lubomir T. Chitkushev[2], Patrick Phelan[3], Miguel Ponce de Leon[3] and Steve Bunch[4]

1
i2CAT Foundation, Jordi Girona, Barcelona, Spain
eleni.trouva@i2cat.net, eduard.grasa@i2cat.net

2
Computer Science, Boston University, Massachusetts, USA
day@bu.edu, matta@bu.edu, ltc@bu.edu

3
TSSG, Waterford Institute of Technology, Ireland
pphelan@tssg.org, miguelpdl@tssg.org

4
TRIA Network Systems, LLC, Illinois, USA
steve.bunch@ieee.org

Paper type: Conceptual paper

Abstract
The aim of this paper is to look at the deficiencies of the current Internet architecture, consider a deeper understanding on why the current architecture is failing to provide solutions and contrast the traditional beliefs on networking with new ones, coming from a network architecture based on the fundamentals. First, we briefly introduce the early history of packet-switched networking to provide the reader with background for the discussion that follows. We highlight the main issues that the current Internet faces and expose the architectural decisions that lead to these problems. Next, we present RINA (Recursive InterNetwork Architecture), a network architecture based on fundamentals among which is that networking is interprocess communication and only IPC. We show the fundamental principles from which RINA is derived, the core elements of the architecture and give a simple example of communication. The adoption of RINA as the architecture for the future networks would enable enhanced security, inherent support for quality of service, mobility, multi-homing, offer new market opportunities and decrease the complexity of the current technology by an order of magnitude.

Keywords
Network Architecture, Recursive InterNetwork Architecture (RINA), Future Internet, naming and addressing

1. Introduction
Although the ARPANET created the foundation upon which today's Internet is built, many of the Internet's problems today have their roots in the ARPANET's limited design. ARPANET's goal was to develop the first

distributed packet-switching network technology. This initial effort had one purpose: to demonstrate that communication between applications hosted on different machines was feasible. The focus was not in making the correct architectural decisions, but in developing something that worked. Despite several indications that revealed deficiencies in the architecture, the TCP/IP protocol suite adopted the same principles. Today's Internet faces well-known problems, which prove that the Internet suffers from its inheritance. In this paper we first explore the origins of the current Internet architecture by briefly examining the history of packet-switching networks. Then, we review the shortcomings and the problems we are currently facing. We argue that the inability to provide answers to these challenges derives from architectural decisions in the primary design of the Internet that follow the structure to our days. Next, we present RINA (Recursive InterNetwork Architecture), an architecture based on the fundamentals. We introduce the principles behind RINA, review RINA's response to the hot networking issues and highlight the advantages that make it a strong candidate architecture for the future Internet.

2. Origins

The story is well-known: In 1969 the first computer was connected to the ARPANET's IMP (Interface Message Processor) at the University of California at Los Angeles (UCLA) and on October 29th sent the first packets on the ARPANET to Stanford Research Institute, putting in operation the world's first packet-switched network. In the ARPANET the hosts were connected to the network through IMPs, the first generation of today's routers. The IMPs provided a reliable packet delivery service to the hosts attached to the network. A host's address was the number of the IMP it was connected to, concatenated with the IMP port number that attached the host to the network [8]. The software residing in the hosts was the NCP (Network Control Program), which was the first host-to-host communications protocol. NCP provided a standard method to establish reliable, flow-controlled, bi-directional communication links among applications running in different hosts. The applications supported were three, Telnet, FTP and RJE.

In 1972 the French research network, CYCLADES, the world's first datagram network [9, 10], was created under the direction of Louis Pouzin. CYCLADES was built in IRIA (Institut de Recherche d' lnformatique et d'Automatique), today's INRIA French research institute. The purpose of the CYCLADES project was to create a testbed for further experimentation on packet switching and routing. The CYCLADES network embodied for the first time concepts such as best effort service with unreliable out of order delivery service and end-to-end error recovery in the hosts. In addition, there was clear splitting of the transport and network functions, internetwork communication, hierarchical addressing, and the use of a dynamic sliding window mechanism. It also included early conceptualization of layers, ideas for a congestion control scheme and architectural support for multi-homing and mobility. In the meantime, as ARPANET expanded, NCP proved to be incapable of keeping up with the growing network traffic load. In 1974, a new, more robust suite of communications protocols was proposed and implemented throughout the ARPANET, based upon the Transmission Control Protocol (TCP) for end-to-end network communication. In 1978 a new design split responsibilities between a pair of protocols; the new Internet Protocol (IP) for routing packets and device-to-device communication and TCP for reliable, end-to-end host communication. The migration of the ARPANET to TCP/IP was officially completed on January 1, 1983. The core idea behind TCP/IP was to develop a protocol that allowed the interconnection of several packet switched networks of different technologies. The design of the network included the recognition that it should provide only the functions of efficiently transmitting and routing traffic between end nodes and that all other intelligence should be located at the edge of the network, in the host nodes. Although undoubtedly CYCLADES was an innovative and groundbreaking project, it was forced to shut down. A virtual circuit service is more directly marketable, not requiring substantial modifications to customers' host computers. For this reason, the French PTTs decided to promote Reseau a Commutation par Paquets (RCP), an experimental packet switched network based on virtual circuit connection [17, 18].

The TCP/IP protocol suite came later to satisfy the growing needs of a developing network. It adopted some ideas found in ARPANET and CYCLADES but also inherited some design principles that proved to be problematic with time. The inability to provide answers to major problems today stems from these principles.

3. Problems looking for an answer

The main issues of today's Internet are the inability to provide security, multi-homing, and Quality of Service, the address space exhaustion, the complexity in providing mobility, the increasing size of the router tables and the poor utilization of available resources.

Network security is one of the most active research areas today and a great deal of research effort and investment has gone into improving the security of the Internet, however being only partially successful. The list of threats is long; DoS attacks, unwanted information (spamming), trojans, viruses, impersonation, frauds, port-scanning, phishing and other malicious behaviours exist, while new vulnerabilities get discovered day by day, despite of the numerous security mechanisms developed.

An increasing number of nodes in the network need to be multi-homed, i.e., to be connected with more than one connection to the network at the same time. With the strong dependence on the Internet by even modest sized businesses, the turn to multi-homing in order to increase the reliability level of their businesses by avoiding the single point of failure network connectivity is reasonable. There are also cases in which multi-homing serves as a load-balancing solution, balancing the traffic across the multiple connections. Whatever the approach, multi-homing is one of the desired characteristics of the network that the current architecture provides with added complexity. The need of today's business and commercial make this requirement more critical. However, multi-homing had been a requirement from the first days of the ARPANET.

Modern applications such as real-time streaming multimedia, Voice over IP, safety-critical applications and others have unique requirements and restrictions in parameters like bandwidth, delay, loss, jitter and error probability. The existence of mechanisms to support QoS and provide service guarantees in today's network becomes vital. Shifting from a small research network to a business tool, the current architecture does not provide mechanisms to support QoS in a large scale but only in small, dedicated networks, since there is no inherent mechanism in the architecture that can be used to deliver service under specific guarantees throughout.

The IANA Central IPv4 Registry was exhausted in January 2011 [11]. That was the first milestone of the IPv4 address pool depletion. Now the regional registries will start to run out and sometime later in 2011, according to the estimates by the end of the summer, there will be no more IPv4 public addresses to allocate. IPv6, by changing the address length from 32-bits to 128-bits, is bringing the solution to the inevitable depletion of the pool of unallocated IPv4 addresses. However, for most of us the migration process to IPv6 has not even started, as there is no adequate support to make the transition. Another issue is that no one is willing to pay the IPv6 upgrade cost.

The proliferation of mobile phones, laptops and other mobile devices undoubtedly makes mobility one of the most important requirements of the future Internet. Mobility can be seen as a dynamic version of multi-homing with expected failures, which means that any architecture that is able to provide multi-homing, will be also able to provide mobility without the need of any extra protocols. Currently mobility of applications is supported for specific hand-off scenarios (e.g. the mobile node connects to a new access point (AP) that is under the same domain as the old AP), while in most of the cases it is not possible to provide seamless mobility; low performance, long delay and frequent disconnections result in degradation of the service perceived by the user. IP comes with Mobile IP protocol that achieves mobility with some extra cost and overhead.

The exponential growth of the Internet has created a new problem, corresponding growth in size of routing tables, Internet backbone routers have to maintain complete routing information for the Internet. In recent years, routing tables have experienced an exponential growth, as an increasing number of businesses, organizations and companies joined the Internet [4]. In 1993 the size of the router tables started to outgrow the capacity of the routers. Even though hardware performance has been increasing following Moore's law over the years, installing more router memory and increasing the size of the routing tables cannot be the solution to the problem forever; not to mention that this is not the most cost effective approach. As the router table size grows, the need for CPU power, required to compute routing table/topology changes grows. The new generation Internet protocol, IPv6, can solve the current address space exhaustion problem by changing the number of bits in the IP address to 128. However, IPv6 poses a serious requirement to the router's control planes: an IPv6 prefix consumes four times more memory in a router than an IPv4 prefix and requires more computing power for convergence of routes. This fact, compounded with the multi-homing issue and the advent of the Internet of Things billions of sensors, smart-meters and devices directly connected to the Internet may lead to routers not being able to converge on the calculations of the BGP routing tables, causing routing instabilities and ultimately an Internet far less reliable than today.

Another problem is that there are cases in which the current technology is failing to take advantage in the best way possible of the available resources. An example is the behaviour of the congestion control mechanisms in TCP over heterogeneous networks. Wireless links exhibit characteristics much more different from the wired

ones; in wireless channels high error and loss rates are the norm. TCP interprets the losses that happen due to the nature of the transmission medium as losses coming from congestion in the network. As a result, the congestion control mechanisms are triggered, forcing a reduction in the TCP window and degradation in performance.

4. Architectural choices that lead to problems

It is our belief that the difficulty to find solutions for the afore-mentioned problems stems from specific architectural principles that the Internet carries from the 1970's. These are:

Incomplete naming schema The current Internet architecture does not provide separate names for the basic entities in the architecture: nodes, points of attachment to the network (PoAs) and applications. The only names provided are PoA names (IP addresses), which are usually confused to be node names. This defect originates in the ARPANET's design where host addresses were named after the IMP port number (interface). The result is that the network has no means to understand that two or more IP addresses of a multi-homed node belong to the same node, making multi-homing hard to realize. The same choice, naming the interface and not the node, forces the Internet to perform routing on the interface level instead of the node level, resulting in having much bigger routing tables than they really need to be. While the problem became apparent very soon, in 1972, when Tinker Air Force Base in Oklahoma joined the Net and wanted two connections for reliability, it was never changed. Mobility, which can be seen as dynamic multi-homing, is the next feature that suffers from having an incomplete naming schema. Numerous other problems arise as well when more sophisticated distributed applications are attempted beyond the trivial client-server schemes. It should be noted that every other network architecture (XNS, CYCLADES, DECNET, OSI, etc) developed in the 1970s and 80s avoided this problem.

In 1982, Jerry Saltzer in his work "On the Naming and Binding of Network Destinations" [7] documented what XNS and CYCLADES were doing, noting the entities and the relationships that make a complete naming and addressing schema in networks. According to Saltzer there are four elements that need to be identified: applications, nodes, points of attachment to the network (PoAs) and paths. A service can run to one or more nodes and should be able to move from one node to another without losing its identity in the network. However, Saltzer failed to notice that because nodes can have more than one path to the same next hop routing has to be on the node to prevent a combinatorial explosion. As illustrated in Figure 1, a directory maps an application name to a node address, routes are sequences of nodes addresses and paths result from mapping the node addresses to PoAs of nearest neighbors. In this way, routing is a two-step process: First, we have to calculate the route, which is a sequence of node addresses and then, for each hop choose the appropriate PoA to select the specific path to be traversed. Although, clearly CYCLADES and XNS had complete naming and addressing schema with different names for nodes, applications and PoAs, the TCP/IP protocol suite followed the ARPANET approach, causing many of the problems of today's Internet.

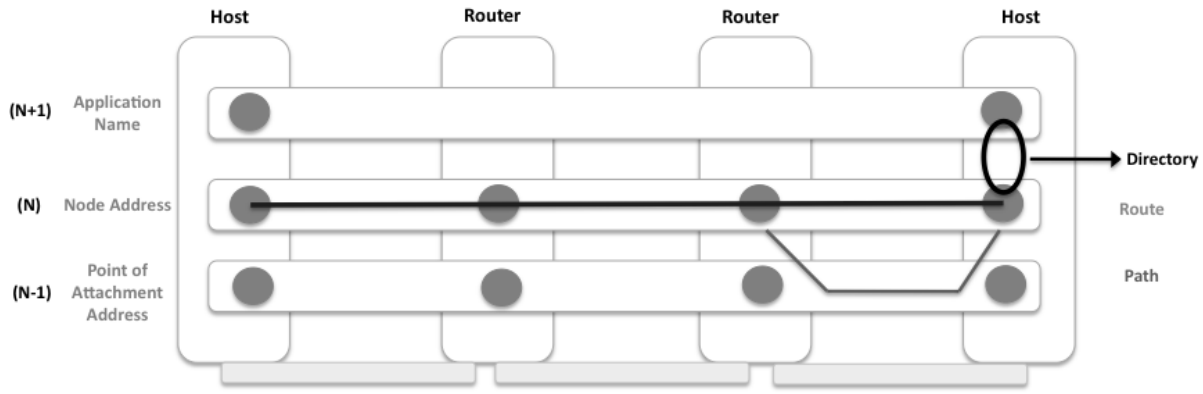

Figure 1 Saltzer's point of view for a complete naming and addressing schema.

Table 1 shows the components of the naming and addressing schema that the current Internet architecture has, compared to other Internetwork architectures that have existed over the years. As it becomes apparent, all the architectures except TCP/IP had already figured out what was the solution to support multi-homing and mobility, going back as far as 1973 in the case of CYCLADES. This is evidence that perhaps TCP/IP did not become the "de facto" standard for packet networking because it was the best technical solution, but due to other factors [5],

a choice that causes many of the problems of today's Internet.

	ARPANET	TCP/IP	CYCLADES	XNS	DECNET	OSI
Application names	No	No	Yes	Yes	Yes	Yes
Node addresses	No	No	Yes	Yes	Yes	Yes
PoA address	Yes	Yes, twice (IP address and MAC address)	Yes	Yes	Yes	Yes

Table 1 Different packet network architectures and their naming and addressing schema

Bad choices for names Saltzer [7] underlines the importance of choosing the right properties for the names of the elements of a network. Application names should be location-independent, while node names should be location-dependent. Location-independent names for applications would allow the applications to move inside the network. Location-dependent names for nodes would result in more efficient routing and smaller routing tables. Having in mind a network graph, the location-dependent property does not denote geographical proximity (spatial distance) but closeness in terms of reachability. Another property is choosing hierarchical names that can form a tree structure. The prefix in a name should give a clue where we can find the children names that share the same prefix, allowing searching for a name in a more efficient manner. In addition, a proper authentication schema that prevents everyone from connecting to everyone through configuration would make apparent that global names are unnecessary, resulting in smaller names. Under the same view, that there is no need for names of global scope, the address space exhaustion would not be considered an issue any more. The choices for naming and addressing are not the optimal ones. Choosing carefully the properties of the names for applications, nodes and PoAs could solve several problems.

No indirection The current Internet does not have any form of directory that maps application names to nodes. DNS (Domain Name System) is the only directory service in the current Internet, and it only provides synonyms for IP addresses that, as seen before, are just Point of Attachment addresses. As a consequence, the most used structure today for naming applications, URIs and URLs (Universal Resource Identifiers and Universal Resource Locators), use the IP address (or the domain name, a synonym) and socket port number as part of the application name. This way of naming applications defines a static binding between layers: The application name tells you that the application is executing on a given interface of a given node, and using a given socket to connect to it. The problems that this causes are:

- Application developers (and application users) have to know where the instances of applications are executing.
- If any of the parameters embedded in the application name changes (IP address of the interface or domain name, socket port number), the application name changes, making multi-homing and mobility very cumbersome to support.

If the application name space was completely independent of the network addresses and the network provided a Directory function, application developers and users could communicate to any instance of an application by just knowing its name, not its location.

Security is missing The current architecture lacks an inherent mechanism that prevents any application (including the network protocol stack) from connecting to any other application without permission. A policy associated with this mechanism should determine the authentication algorithm used according to the security level required for the specific application. Over the years, several protocols have been introduced at various layers of the Internet in order to improve its security, but, as reported in [6], "experience has shown that it is difficult to add security to a protocol suite unless it is built into the architecture from the beginning". Although it was clearly an experimental network, the absence of such a mechanism from the ARPANET seems strange when you consider that it was originally designed for the Department of Defense of US. This defect, along with the exposed addresses that anyone can see and the use of the well-known ports impose security risks.

Only best effort Another shortcoming of the current Internet architecture is that there is no built-in mechanism that allows the network to provide specific QoS (Quality of Service) levels. No interface for applications to

request it, despite of the development of several real-time applications that require some minimal level of resources to operate effectively. The current architecture can provide only end-to-end best effort service, as it treats all packets the same way, providing a single level of service. The Integrated Services (IntServ) [12] model was the first attempt to enhance the Internet with QoS capabilities. The main problem with IntServ was scalability. IntServ requires from each router along the path to store the states of each flow (reservation) and several concerns about the scalability of the model arose. In practise, IntServ can work on a small scale but not for a system with the scale of the Internet. The Differentiated Services model (DiffServ) [13] was another proposal, which is based on traffic classification into classes of different QoS requirements. DiffServ provides a framework to allow the classification and the differentiated treatment of the traffic. However, how the individual routers treat the different types of service is not specified, making the end-to-end behaviour impossible to predict.

Perception of layers A layered approach is required to design a network architecture, as it provides the ability to hide information when it is not needed and abstract the details. The current Internet architecture uses layers, with every layer having the responsibility of a different function. A consequence of this perception of layers is that the current architecture lacks a mechanism to provide different configurations over different physical media according to the requirements of each application. As we mentioned before, TCP acts on global scope on heterogeneous paths and congestion control mechanisms are triggered regardless of the nature of the physical media. This example shows the need for further configuration and points to an architecture of layers that perform the same functions, but configured differently, over different scopes. Under this new definition of a layer, we could manage heterogeneous paths by having a layer to manage a wired network, another layer could manage a wireless network, and a layer on top of them managing the end to end inter-network. In addition, the same architectural decision of the current Internet resulted in a large number of protocols that provide similar functions under different configuration in different layers of the network stack. The result is the so-called layer violations inside the “protocol hourglass”.

5. RINA

5.1 Introducing RINA

Reviewing the deficiencies of the current Internet, it becomes apparent that the inability to provide answers to these challenges stems architectural decisions in the primary design of the TCP/IP protocol suite. In the following paragraphs, we present RINA (Recursive InterNetwork Architecture), an architecture proposed by John Day in his book “Patterns in Network Architecture: A return to fundamentals” [1, 3]. RINA leverages many of the lessons learned by previous network architectures such as CYCLADES, and brings them one step further by identifying that networking can be seen as a set of recursive layers that provide distributed inter-process communication services over different scopes. The resulting architecture is surprisingly simple compared to today’s protocol complexity and provides a structure that allows network designers to solve the problems identified in the current Internet.

5.2 Main principles

RINA is based on the view that networking is only inter-process communication (IPC). The core element of the architecture is a Distributed IPC Facility (DIF), which is simply used to group cooperating application processes into manageable sub-networks, which are configured through policies. Any two application processes in different systems are able to communicate using the services provided by a DIF and DIFs of higher levels are able to use the services provided by DIFs of lower levels, forming a recursive structure. A DIF can be seen as a layer, but not in the same sense as in the TCP/IP architecture. In RINA all layers use the same protocols to perform a coordinated set of policy managed mechanisms and achieve the desired IPC service, matching in the best way possible the requirements of the user application.

5.3 An example of the RINA architecture

Figure 2 depicts an example of the RINA architecture with three levels of DIFs. Each layer (illustrated in a different pattern in the figure) provides IPC services over a limited scope. The first level of DIFs operates on top of the physical media and its policies are optimized to deal with the characteristics of the physical medium. This first level of DIFs provides IPC services to the second level of DIFs, the second to the third and so on. In RINA only three types of systems exist: hosts, border routers and internal routers. There is no need for the middle boxes of today such as firewalls, NATs and others.

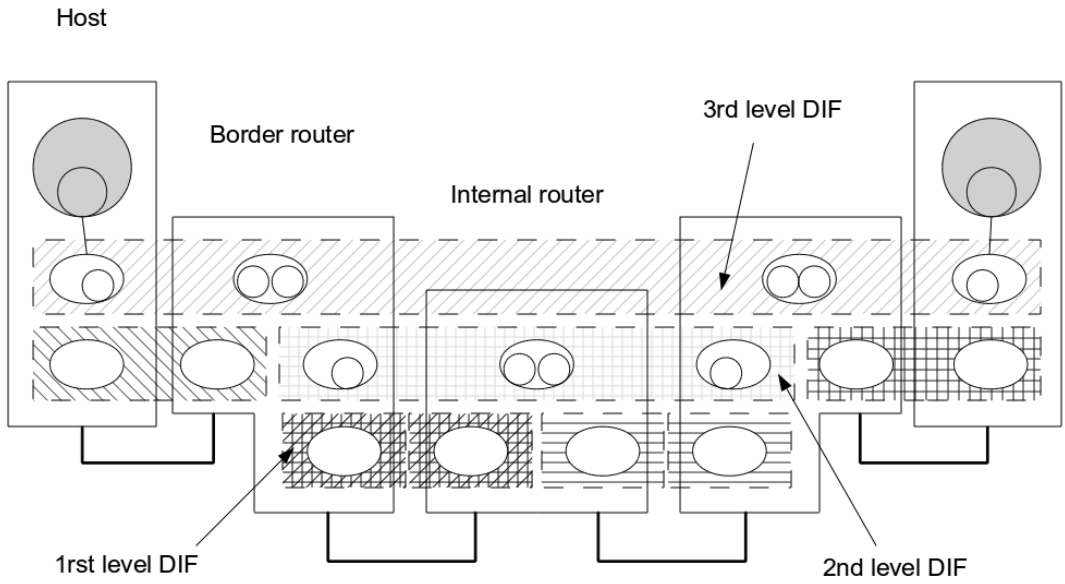

Figure 2 An example of RINA architecture with 3-levels of DIFs.

5.4 Zooming into an IPC application process

Each IPC process executes routing, transport, security/authentication and management functions. The components of an IPC process responsible for providing these functions can be categorized under three main categories, IPC API, IPC Data Transfer and IPC Management. The behaviour of each component of the IPC process can be configured via policy. A graphical representation of the components of an IPC application process can be seen in Figure 3.

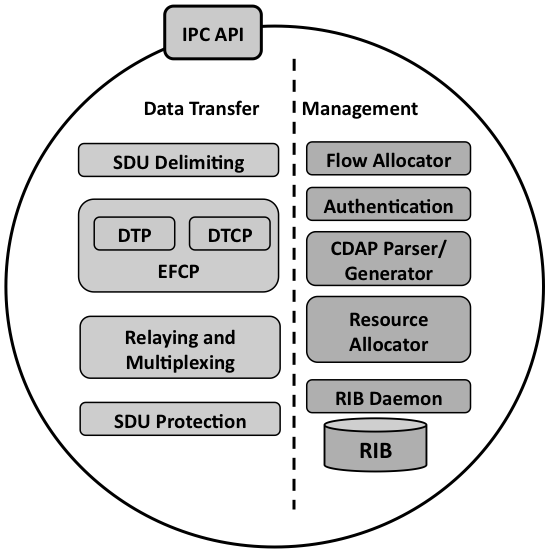

Figure 3 Components of an IPC process.

IPC API

Each IPC application provides an API to the client layer that uses the IPC services provided. The service provided by a DIF to the applications above is referred to as a flow. The IPC API provides four operations:

- **allocate** - Allocates resources to support a flow between source and destination application processes with a certain quality of service. A port-id is returned as a handle for the allocation.
- **send** - Sends an amount of data to the destination application process on the specified port. The amount of data passed across two layers to be transferred to the destination is referred to as a Service Data Unit (SDU).

An SDU may be fragmented or combined with other SDUs.

- **receive** - Receives an SDU from the destination application process on the specified port.
- **deallocate** - Terminates the flow and frees all the communication resources allocated to it.

IPC Data Transfer

The IPC Data Transfer consists of the SDU Delimiting task, the Error and Flow Control Protocol (EFCP), the relaying and multiplexing tasks and the SDU protection task:

- **SDU Delimiting** - Mechanisms used to indicate the beginning and the end of SDUs. Delimiting can be done using a special bit pattern to denote the start and the end of the SDU or using a length field in the Protocol Control Information (PCI/header) that can be used to calculate the end of the SDU.
- **Error and Flow Control Protocol (EFCP)** - A data transfer protocol based on Delta-t [14]. EFCP is split into two independent protocol machines (DTP and DTCP), loosely coupled through a state vector. The Data Transfer Protocol (DTP) machine implements the mechanisms that are tightly coupled to the transported SDU, such as fragmentation, reassembly, sequencing, concatenation and separation. The Data Transfer Control Protocol (DTCP) machine implements the mechanisms that are loosely coupled to the transported SDU, such as transmission control, retransmission control and flow control. The string of octets exchanged between two protocol machines is referred to as Protocol Data Unit (PDU). PDUs comprise of two parts, Protocol Control Information (PCI) and user data. PCI is the part understood by the DIF, while the user data is incomprehensible to the DIF and is passed to its user.
- **Relaying Task** - The role of this task is to forward the PDUs passing through this IPC process to the destination Protocol Machine (PM) by checking the destination address in the PCI.
- **Multiplexing Task** - Mapping of the flows of PMs belonging to a higher layer onto flows of PMs belonging to a lower layer.
- **SDU Protection** - Mechanisms to protect the SDU from byte errors and to provide confidentiality and integrity. Mechanisms such as Data Corruption Protection, checksums, hop count, Time To Live and encryption are included.

IPC Management

The IPC Management consists of the Flow Allocator component, authentication mechanisms, the Common Distributed Application Protocol (CDAP), the Resource Allocation, the Resource Information Base (RIB) and the RIB Daemon:

- **Flow Allocator** - A component responsible for responding to allocate/deallocate requests and for managing each new flow that passes through this IPC process.
- **Authentication** - Security related functions such as authentication, access control and protection against tampering and eavesdropping by the lower layers.
- **Common Distributed Application Protocol -** The application protocol, similar to an assembly language used to build all the distributed applications. CDAP is a descendant of common management information protocol (CMIP) [19, 20] and allows performing operations on objects such as read, write, create, delete, start and stop.
- **Resource Allocator** - A component that manages resource allocation and monitors the resources in the DIF by sharing information with other IPC processes and observing the performance of supporting DIFs.
- **Resource Information Base** - A database that stores all the information available to the IPC process represented as objects, such as mappings of addresses, resource allocation information, connectivity information, security credentials and others.
- **RIB Daemon** - A generalisation of event management and routing update. The RIB Daemon is responsible for responding to requests for information from other members of the DIF, notify them periodically or upon certain events on the current value of specific information, as well as reply to external requests for information. In addition, RIB Daemon ensures that the other management tasks have the information they require.

5.5 An example of communication

In this section, a simple example of inter-process communication between two application processes in different systems is shown. Figure 4 depicts the scenario of two application processes in different systems (system 1 and system 2) connected through an intermediate node. We denote S the source application process residing in system 1 and D the destination application process residing in system 2. A1, A2, A3, B1, B2, C1, C2 are the IPC application processes that form the DIFs A, B and C, illustrated with a dotted line.

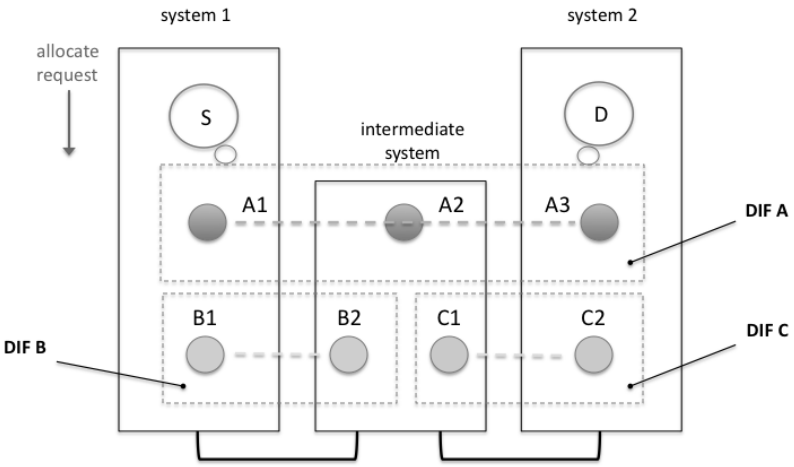

Figure 4 IPC between two application processes in different systems.

The steps that take place to achieve communication between the application processes S and D are the following:

- DIF A maps the source and the destination application processes names S and D to the IPC processes A1 and A3 respectively.
- Application process S using the IPC API does an allocate request to the underlying DIF A specifying the destination application process name D and the desired QoS parameters for the communication.
- The Flow Allocator in IPC application process A1 receives the request and validates it. If the request is well formed and the IPC process has enough resources to honour the request then it is accepted. The Flow Allocator in A1 creates the EFCP instances (DTP and if required DTCP) for the data transfer.
- Then, the Flow Allocator at A1 searches the local directory for the requested application process D, finds an entry that maps D to IPC process A3 and then sends a request to create a flow to A3. The request for the creation of the flow is a CDAP protocol exchange.
- The Flow Allocator at A3 receives the request to create flow and delivers the allocate request to the destination application process, D.
- Application process D submits an allocate response to the Flow Allocator in A3. The response depends on whether application S has the rights to access application D.
- The Flow Allocator in A3 creates the EFCP instances for data transfer (DTP and if required DTCP) and sends a response (CDAP) to the request for creating a flow it received to A1.
- If the response is positive, the two applications, S and D, can use the IPC API calls send and receive to send and receive data (SDUs) to/from each other.
- When the communication is over, both of them can invoke the de-allocate call to release the allocated resources.

5.6 Features and implications of the RINA architecture

A structure that is based on the fundamentals yields an architecture that has the following features:

A clean recursive architecture - Each layer is a distributed application, composed of a set of collaborating application processes that provide IPC services to the layer above. Therefore, each layer has the same functions; it just operates in (and configured for) a different scope.

A complete naming and addressing schema - RINA provides names for all the entities in the network that need to be named: Applications, nodes, points of attachment to the network. Applications request flows to other applications without having to know where the requested application is. Addresses are internal to a DIF, they are just synonyms of the names of the application processes that comprise the DIF, optimized for routing.

A single error and flow control protocol - By separating mechanism from policy, all the current data transfer protocols can be provided through proper configuration (management of policies) [15]. Therefore, only one transport protocol is required, decreasing the complexity of the network by orders of magnitude.

A single application protocol - RINA provides a single application protocol, CDAP (Common Distributed Application Protocol), which can be used to develop all the distributed applications. It is not a requirement to use it as any application protocol can be transported by RINA, but using it simplifies the development of distributed

applications.

Multi-homing, mobility and **multicast** - They are supported inherently without the need for any special protocols [2].

A more secure network - Applications need to authenticate with one another before being able to exchange information. Applications are not allowed to communicate with the elements inside a DIF unless they are members of that DIF, which requires authentication [16].

Quality of Service - Each DIF can support a set of QoS cubes (QoS classes with different restrictions on several QoS parameters such as bandwidth, delay, loss rate, ordered or not ordered delivery, jitter) and provide service guarantees to its clients. Mechanisms like error control, flow control, forward error correction, ensure the delivered service.

Scalability - Given its repeating nature, where each DIF has its own private internal addresses, and with the existence of policies that constrain the membership size of each DIF, RINA is expected to achieve much better address scalability compared to that of the current Internet. In addition, RINA comes with a complete naming and addressing schema. The narrower scope of topology changes and late binding from node name (address) to PoA address (interface) make RINA architecture much more scalable in terms of routing overhead. The choice to use hierarchical names/addresses contributes to the scalability.

A competitive marketplace - RINA creates the robust feedback needed for a healthy marketplace [21]. Each DIF can be configured to not only provide the traditional services of lower networking layers but also application-support (transport) services. This removes the barrier created by the Transport Layer in the current Internet, opening potential new markets for ISPs to provide IPC services directly to their customers, leveraging their expertise in resource management of lower layers and creating new competition with "host" providers.

6. Conclusions, on-going and future work

Having reviewed the shortcomings of the current Internet and exposed the fundamental causes of the problems, we can conclude that the current architecture, based on the TCP/IP protocol suite, is far from being perfect. It can be seen as an unfinished demo that has been living on band-aids and Moore's Law for almost thirty years. Several indications are showing that this approach is reaching its limits. Amongst them are the routing table scalability, the address space exhaustion, the security risks and the inability to support mobility, multi-homing and quality of service under a scalable solution. A brief review of the historical background of the Internet shows that other network architectures, designed and implemented in the 70's and 80's, managed to provide solutions to problems such as mobility and multi-homing that seem to have been forgotten. Under this understanding, we introduced RINA, a newly proposed network architecture. RINA utilizes the knowledge acquired through the past efforts on packet-switched networking such us CYCLADES, XNS and OSI and goes one step further by realizing that networking is distributed inter-process communication. It provides a simple, powerful framework for building the networks of the future and for evolving packet switched networks to a mature state.

Our current work involves the development of a prototype that implements the RINA architecture [22, 23, 24]. Future work includes experimentation with the developed prototype over networks of different physical media and further research, measurements and performance comparisons of RINA versus the current Internet architecture.

Acknowledgements

This work has been partially supported by the Spanish Government, MICINN, under research grant TIN2010-20136-C03 and by the US National Science Foundation under grant CNS-0963974.

References

[1] J. Day: "Patterns in Network Architecture: A Return to Fundamentals, Prentice Hall, 2008.
[2] V. Ishakian, I. Matta, J. Akinwumi: "On the cost of supporting mobility and multi-homing", GLOBECOM Workshops, IEEE, pp. 310–314, 2010.

[3] J. Day, I. Matta, K. Mattar: “Networking is IPC: A Guiding Principle to a Better Internet”, Proc. of ReArch08, Madrid, SPAIN, 2008.
[4] V. Fuller: “Scaling issues with routing and multihoming”, IRTF RRG meeting, 2007.
[5] R. Bennet: “Designed for Change: End to end arguments, Internet Innovation, and the Net Neutrality Debate”, Information Technology and Innovation Foundation Report. 2009. Available online at http://www.itif.org/files/2009-designed-for-change.pdf
[6] D. Clark, L. Chapin, V. Cerf, R. Braden, R. Hobby: “Towards de Future Internet Architecture”, RFC 1287, 1991.
[7] J. Saltzer: “On the Naming and Binding of network destinations”, RFC 1498.
[8] F. E. Heart, R. E. Kahn, S. M. Ornstein, W. R. Crowther, and D. C. Walden: “The interface message processor for the ARPA computer network”, Proceedings of the Spring joint computer conference (AFIPS), ACM, New York, NY, USA, 551-567, 1970.
[9] L. Pouzin: "Presentation and Major Design Aspects of the Cyclades Computer Network". Proceedings of the NATO Advanced Study Institute on Computer Communication Networks. Sussex, United Kingdom: Noordhoff International Publishing. pp. 415-434, 1973.
[10] L. Pouzin: "CIGALE, the packet switching machine of the CYCLADES computer network". Proc of IFIP, Stockholm, Sweden. pp. 155-159, 1974.
[11] Wikipedia - IPv4 address exhaustion: http://en.wikipedia.org/wiki/IPv4_address_exhaustion
[12] R. Braden, D. Clark and S. Shenker: "Integrated Services in the Internet Architecture: an Overview", RFC 1633, ISI, MIT, and PARC, June 1994.
[13] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss: "An Architecture for Differentiated Services", RFC 2475, December 1998.
[14] R. W. Watson: “Delta-t protocol specification”, tech. rep., Lawrence Livermore National Laboratory.
[15] K. Mattar, I. Matta, J. Day, V. Ishakian, and G. Gursun: “Declarative Transport: A Customizable Transport Service for the Future Internet”, Proceedings of the 5th International Workshop on Networking Meets Databases (NetDB 2009), co-located with SOSP 2009. Big Sky, MT, October 14, 2009.
[16] G. Boddapati, J. Day, I. Matta, L. Chitkushev: “Assessing the Security of a Clean-Slate Internet Architecture”, Technical Report BUCS-TR-2009-021, CS Department, Boston University, June 22, 2009.
[17] R. Lawrence: “The Evolution of Packet Switching”, Proceedings of the IEEE, volume 66, number 11, pp. 1307 - 1313, 1978.
[18] J. Pelkey: “Entrepreneurial Capitalism and Innovation: A History of Computer Communications 1968-1988”, Chapter 6 (Networking: Diffusion 1973-1979 Networking Protocols and Local Area Networks). Available online at http://www.historyofcomputercommunications.info/Book/6/6.3-CYCLADESNetworkLouisPouzin1-72.html
[19] U. Warrier, L. Besaw [Hewlett-Packard]: “Common Management Information Services and Protocol over TCP/IP (CMOT)”, RFC 1095, 1989
[20] U. Warrier, L. Besaw, L. LaBarre, B. Handspicker: “The Common Management Information Services and Protocols for the Internet (CMOT and CMIP)”, RFC 1189, 1990
[21] J. Day: “Creating a Viable Economic Model for a Viable Internet”, 2008. Available online at http://pouzin.pnanetworks.com/images/PNAEcon080518.pdf
[22] Pouzin Society website at http://www.pouzinsociety.org/
[23] RINA at Computer Science Department of Boston University at http://csr.bu.edu/rina/
[24] RINA prototyping project at i2CAT Foundation website at http://www.i2cat.net/en/projecte/rina-prototype-1

Vitae

Eleni Trouva received her diploma in Computer Engineering from the University of Patras, Greece in 2006 and a Master in Science in Computer Science from the Department of Informatics, Athens University of Economics and Business, Greece in 2009. Currently, she is a PhD student at the Telematics Department of the UPC in Barcelona, Spain. Her current research interests are in Networks and Telecommunications. She joined the Fundació i2CAT in September 2009, where she initially contributed in the FP7 GEANT3 project. Eleni is today involved in the RINA initiative, working on a proof of concept implementation.

Eduard Grasa PhD (UPC, 2009) and Telecommunication Engineer (UPC, 2004). At the Fundació i2CAT he has participated in many international research projects (UCLPv1, UCLPv2, HULP, MANTICORE I and II) and FP7projects (Phosphorus, FEDERICA) related to bandwidth on demand technologies, network management automation and software resource management, and has 5 years of programming experience in Java. Since late 2009 he is involved with the RINA effort, contributing to prototype a network architecture that can provide a better infrastructure for building distributed applications.

John Day has been involved in research and development of computer networks since 1970 when he was involved in the design of transport and upper layer protocols for the ARPANet as well as the Internet. Mr. Day was in charge of the development of the OSI Reference Model, Naming and Addressing and a major contributor to the upper layer architecture and was a member of the Internet Research Task Forces Name Space Research Group. Recently he has been turning his attention to new network architectures that scale indefinitely and described in his book Patterns in Network Architecture.

Ibrahim Matta received his Ph.D. in computer science from the University of Maryland at College Park in 1995. He is an associate professor at the Computer Science Department of Boston University College of Arts and Sciences. His research is in transport and routing protocols for the Internet and wireless networks; feedback-based control design and analysis; architectures for protocol design and large-scale traffic management; modeling and performance evaluation.

Lubomir Chitkushev is Associate Professor and the Chairman of Computer Science Department at Boston University Metropolitan College and Director of Information Security and Biometrics Laboratories. He is co-founder and Associate Director of Boston University Center for Reliable Information Systems and Cyber Security (RISCS) which was established to promote and coordinate research on reliable and secure computation and information assurance education by developing ideas and tools to protect critical computational infrastructure and producing a growing number of highly-educated research professionals with expertise in information reliability and security.

Patrick Phelan has over 10 years software engineering experience with Openet Telecom, QNX Software Systems and Sun Microsystems. He is currently employed in the TSSG as a Lead Engineer. Patrick had previously worked in Openet Telecom as a senior engineer and technical consultant; designing, implementing and managing the deployment of convergent mediation solutions for tier one and two operators (Verizon, AT&T Wireless) that bridged both legacy telephony services and next generation services. At Openet Patrick was involved in proof of concept projects with various partners (Cisco, Ericsson,P-Cube) exploring the metering and charging of next generation services in operator test beds.

Miguel Ponce de Leon is currently a Research Manager at the TSSG. He participates in National and European funded research projects focusing on area of the Future Internet, where his research is concentrating on the "Architecture and Design for the Future Internet". Miguel manages a group of 50 researchers that are looking at the self-management of virtual resources, support service mobility, QoS and reliability. Miguel has worked on FP5 & FP6 projects and during the FP7 programme his research projects include ICT 4WARD, ICT EFIPSANS, ICT AutoI, ICT PanLabs II, ICT Perimeter, ICT ThinkTrust, ICT IncoTrust and ICT CoMiFin.

Steve Bunch started his career in the early 1970's at the University of Illinois doing protocol and network access system research on the Arpanet, and has since worked on secure message switches, UNIX operating systems, operating system security, high-performance microprocessors, standards, and cellphone software. He recently left Motorola, where he led software platforming activities, led software and hardware architecture teams, and was a lead architect on several cellphone software platforms. He is currently involved in a startup company in the computer networking area.