Recursive InterNetwork Architecture (RINA)

From Wikipedia, the free encyclopedia

The **Recursive InterNetwork Architecture (RINA)** is a computer network architecture that unifies distributed computing and telecommunications. RINA's fundamental principle is that computer networking is just Inter-Process Communication or IPC. RINA reconstructs the overall structure of the Internet, forming a model that comprises a single repeating layer, the DIF (Distributed IPC Facility), which is the minimal set of components required to allow distributed IPC between application processes. RINA inherently supports mobility, multi-homing and Quality of Service without the need for extra mechanisms, provides a secure and programmable environment, motivates for a more competitive marketplace, and allows for a seamless adoption.

Contents

History and Motivation

The principles behind RINA were first presented by John Day in his book “Patterns in Network Architecture: A return to Fundamentals”...[1] This work is a start afresh, taking into account lessons learned in the 35 years of TCP/IP’s existence, as well as the lessons of OSI’s failure and the lessons of other network technologies of the past few decades, such as CYCLADES, DECnet, and Xerox Network Systems.

From the early days of telephony to the present, the telecommunications and computing industries have evolved significantly. However, they have been following separate paths, without achieving full integration that can optimally support distributed computing; the paradigm shift from telephony to distributed applications is still not complete. Telecoms have been focusing on connecting devices, perpetuating the telephony model where devices and applications are the same. A look at the current Internet protocol suite shows many symptoms of this thinking:[2]

- The network routes data between interfaces of computers, as the public switched telephone network switched calls between phone terminals. However, it is not the source and destination *interfaces* that wish to communicate, but the distributed *applications*.
- Applications have no way of expressing their desired service characteristics to the network, other than choosing a reliable (TCP) or unreliable (UDP) type of transport. The network assumes that applications are homogeneous by providing only a single quality of service.
- The network has no notion of application names, and has to use a combination of the interface address and transport layer port number to identify different applications. In other words, the network uses information on “*where*” an application is located to identify “*which*” application this is. Every time the application changes its point of attachment, it seems different to the network, greatly complicating multi-homing, mobility, and security.

Several attempts have been made to propose architectures that overcome the current Internet limitations, under the umbrella of the Future Internet research efforts. However, most proposals argue that requirements have changed, and therefore the Internet is no longer capable to cope with them. While it is true that the environment in which the technologies that support the Internet today live is very different from when they where conceived in the late 1970s, changing requirements are not the only reason behind the Internet's problems related to multihoming, mobility, security or QoS to name a few. The root of the problems may be attributed to the fact the current Internet is based on a tradition focused on keeping the original ARPANET demo working and fundamentally unchanged, as illustrated by the following paragraphs.

1972. Multi-homing not supported by the ARPANET. In 1972 the Tinker Air Force Base wanted connections to two different IMPs (Interface Message Processors, the predecessors of today's routers) for redundancy. ARPANET designers realized that they couldn't support this feature because host addresses were the addresses of the IMP port number the host was connected to (borrowing from telephony). To the ARPANET, two interfaces of the same host had different addresses, therefore it had no way of knowing that they belong to the same host. The solution was obvious: as in operating systems, a logical address space naming the nodes (hosts and routers) was required on top of the physical interface address space. However, the implementation of this solution was left for future work, and it is still not done today: “*IP addresses of all types are assigned to interfaces, not to nodes*”.[3] As a consequence, routing table sizes are orders of magnitude bigger than they would need to be, and multi-homing and mobility are complex to achieve, requiring both special protocols and point solutions.

1978. Transmission Control Protocol (TCP) split from the Internet Protocol (IP). Initial TCP versions performed the error and flow control (current TCP) and relaying and multiplexing (IP) functions in the same protocol. In 1978 TCP was split from IP, although the two layers had the same scope. This would not be a problem if: i) the two layers were independent and ii) the two layers didn't contain repeated functions. However none of both is right: in order to operate effectively IP needs to know what TCP is doing. IP fragmentation and the workaround of MTU discovery that TCP does in order to avoid it to happen is a clear example of this issue. In fact, as early as in 1987 the networking community was well aware of the IP fragmentation problems, to the point of considering it harmful.[4] However, it was not understood as a symptom that TCP and IP were interdependent and therefore splitting it into two layers of the same scope was not a good decision.

1981. Watson's fundamental results ignored. Richard Watson in 1981 provided a fundamental theory of reliable transport,[5] whereby connection management requires only timers bounded by a small factor of the Maximum Packet Lifetime (MPL). Based on this theory, Watson et al. developed the Delta-t protocol [6] in which the state of a connection at the sender and receiver can be safely removed once the connection-state timers expire without the need for explicit removal messages. And new connections are established without an explicit handshaking phase. On the other hand, TCP uses both explicit handshaking as well as more limited timer-based management of the connection’s state. Had TCP incorporated Watson's results it would be more efficient, robust and secure, eliminating the use of SYNs and FINs and therefore all the associated complexities and vulnerabilities to attack (such as SYN flood).

1983. Internetwork layer lost, the Internet ceases to be an Internet. Early in 1972 the International Network Working Group (INWG) was created to bring together the nascent network research community. One of the early tasks it accomplished was voting an international network transport protocol, which was approved in 1976.[2] A remarkable aspect is that the selected option, as well as all the other candidates, had an architecture composed of 3 layers of increasing scope: data link (to handle different types of physical medias), network (to handle different types of networks) and internetwork (to handle a network of networks), each layer with its own addresses. In fact when TCP/IP was introduced it run at the internetwork layer on top of the Network Control Program and other network technologies. But when NCP was shut down, TCP/IP took the network role and the internetwork layer was lost.[7] As a result, the Internet ceased to be an Internet and became a concatenation of IP networks with an end-to-end transport layer on top. A consequence of this decision is the complex routing system required today, with both intra-domain and inter-domain routing happening at the network layer [8] or the use of NAT, Network Address Translation, as a mechanism for allowing independent address spaces within a single network layer.

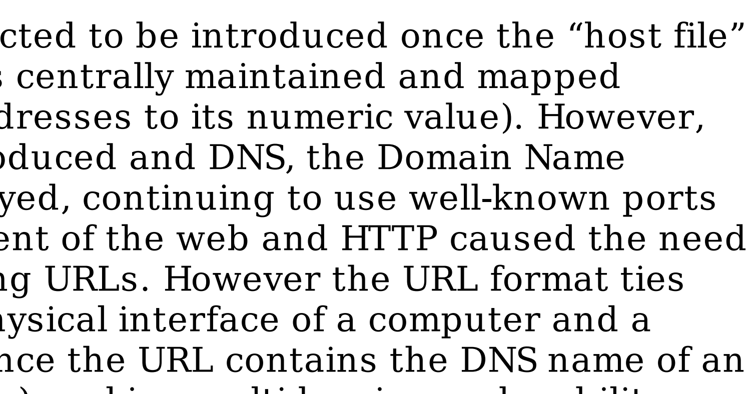

Figure1. The Internet architecture as seen by the INWG

1983. First opportunity to fix addressing missed. The need for application names and distributed directories that mapped application names to internetwork addresses was well understood since mid 1970s. They were not there at the beginning since it was a major effort and there were very few applications, but they were expected to be introduced once the “host file” was automated (the host file was centrally maintained and mapped human-readable synonyms of addresses to its numeric value). However, application names were not introduced and DNS, the Domain Name System, was designed and deployed, continuing to use well-known ports to identify applications. The advent of the web and HTTP caused the need for application names, introducing URLs. However the URL format ties each application instance to a physical interface of a computer and a specific transport connection (since the URL contains the DNS name of an IP interface and TCP port number), making multi-homing and mobility very hard to achieve.

1986. Congestion collapse takes the Internet by surprise. Despite the fact that the problem of congestion control in datagram networks had been known since the very beginning (in fact there had been several publications during the 70s and early 80s,[9][10]) the congestion collapse in 1986 caught the Internet by surprise. What is worse, it was decided to adopt the congestion avoidance scheme from Ethernet networks with a few modifications, but it was put in TCP. The effectiveness of a congestion control scheme is determined by the time-to-notify, i.e. reaction time. Putting congestion avoidance in TCP maximizes the value of the congestion notification delay and its variance, making it the worst place it could be. Moreover, congestion detection is implicit, causing several problems: i) congestion avoidance mechanisms are predatory: by definition they need to cause congestion to act; ii) congestion avoidance mechanisms may be triggered when the network is not congested, causing a downgrade in performance.

1992. Second opportunity to fix addressing missed. In 1992 the Internet Architecture Board (IAB) produced a series of recommendations to resolve the scaling problems of the IPv4 based Internet: address space consumption and routing information explosion. Three types of solutions were proposed: introduce CIDR (Classless Inter-Domain Routing) to mitigate the problem, design the next version of IP (IPv7) based on CLNP (ConnectionLess Network Protocol) and continue the research into naming, addressing and routing.[11] CNLP was an OSI based protocol that addressed nodes instead of interfaces, solving the old multi-homing problem introduced by the ARPANET, and allowing for better routing information aggregation. CIDR was introduced but the IETF didn't accept an IPv7 based on CLNP. IAB reconsidered its decision and the IPng process started, culminating with IPv6. One of the rules for IPng was not to change the semantics of the IP address, which continues to name the interface perpetuating the multi-homing problem.[3]

There are still more wrong decisions that have resulted in long-term problems for the current Internet, such as:

- In 1988 IAB recommended using the Simple Network Management Protocol (SNMP) as the initial network management protocol for the Internet to later transition to the object-oriented approach of the Common Management Information Protocol (CMIP).[12] SNMP was a step backwards in network management, justified as a temporal measure while the required more sophisticated approaches were implemented, but the transition never happened.
- Since IPv6 didn’t solve the multi-homing problem and naming the node was not accepted, the major theory pursued by the field is that the IP address semantics are overloaded with both identity and location information, and therefore the solution is to separate the two, leading to the work on Locator/Identifier Separation Protocol (LISP). However all approaches based on LISP have scaling problems [13] because i) it is based on a false distinction (identity vs. location) and ii) it is not routing packets to the end destination (LISP is using the locator for routing, which is an interface address; therefore the multi-homing problem is still there).[14]
- The recent discovery of bufferbloat due to the use of large buffers in the network. Since the beginning of the 80s it was already known that the buffer size should be the minimal to damp out transient traffic bursts,[15] but no more since buffers increase the transit delay of packets within the network.
- The inability to provide efficient solutions to security problems such as authentication, access control, integrity and confidentiality, since they were not part of the initial design. As stated in [16] “*experience has shown that it is difficult to add security to a protocol suite unless it is built into the architecture from the beginning*”.

Terminology

- **Application Entity**. A task within a DAP directly involved with exchanging application information with other DAPs.
- **Common Distributed Application Process (CDAP)**. CDAP enables distributed applications to deal with communications at an object level, rather than forcing applications to explicitly deal with serialization and input/output operations. CDAP provides the application protocol component of a Distributed Application Facility (DAF) that can be used to construct arbitrary distributed applications, of which the DIF is an example. CDAP provides a straightforward and unifying approach to sharing data over a network without having to create specialized protocols.
- **Distributed Application Facility (DAF)**. A collection of two or more cooperating DAPs in one or more processing systems, which exchange information using IPC and maintain shared state. In some Distributed Applications, all members will be the same, i.e. a homogeneous DAF, or may be different, a heterogeneous DAF.
- **Distributed Application Process (DAP)**. The instantiation of a computer program executing in a processing system intended to accomplish some purpose. A Distributed Application Process contains one or more tasks or Application-Entities, as well as functions for managing

the resources (processor, storage, and IPC) allocated to this DAP.

- **Distributed IPC Facility (DIF), Layer**. A collection of two or more DAPs cooperating to provide Interprocess Communication (IPC). A DIF is a DAF that does IPC. The DIF provides IPC services to Applications via a set of API primitives that are used to exchange information with the Application's peer.
- **IPC Process (IPCP)**. An Application-Process, which is a member of a DIF and implements locally the functionality to support and manage IPC using multiple sub-tasks.
- **Processing System**. The hardware and software capable of executing programs instantiated as DAPs that can coordinate with the equivalent of a “test and set” instruction, i.e. the tasks can all atomically reference the same memory.
- **Protocol Data Unit (PDU)**. The string of octets exchanged among the Protocol Machines (PM). PDUs contain two parts: the PCI, which is understood and interpreted by the DIF, and User-Data, that is incomprehensible to this PM and is passed to its user.
- **Resource Information Base (RIB)**. For the DAF, the RIB is the logical representation of the local repository of the objects. Each member of the DAF maintains a RIB. A Distributed Application may define a RIB to be its local representation of its view of the distributed application. From the point of view of the OS model, this is storage.
- **Service Data Unit (SDU)**. The amount of data passed across the (N)-DIF interface to be transferred to the destination application process. The integrity of an SDU is maintained by the (N)-DIF. An SDU may be fragmented or combined with other SDUs for sending as one or more PDUs.

Introduction to RINA

RINA is the result of an effort that tries to work out the general principles in computer networking that applies to everything. RINA is the specific architecture, implementation, testing platform and ultimately deployment of the theory. This theory is informally known as the Inter-Process Communication “IPC model”[17] although it also deals with concepts and results that are generic for any distributed application and not just for networking.

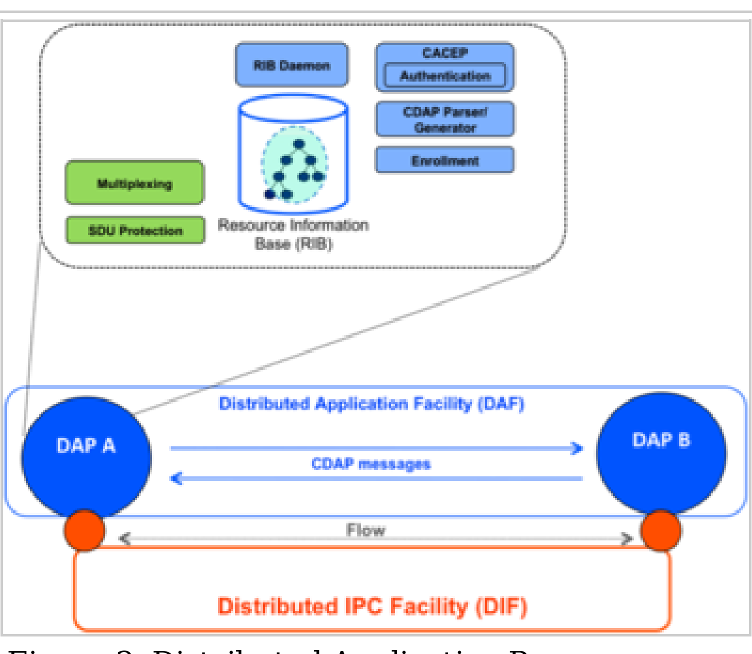

Figure 2. Distributed Application Processes (DAPs) and their components

The IPC model captures the common elements of distributed applications, called DAFs (Distributed Application Facilities), as illustrated in the Figure to the right. A DAF is composed by two or more Distributed Application Processes or DAPs, which collaborate to perform a task. These DAPs communicate using a single application protocol called CDAP (Common Distributed Application Protocol), which enables two DAPs to exchange structured data in the form of objects. All of the DAP's externally visible information is represented by objects and structured in a Resource Information Base (RIB), which provides a naming schema and a logical organization to the objects known by the DAP (for example a naming tree). CDAP allows the DAPs to perform six remote operations on the peer's objects (create, delete, read, write, start and stop).

In order to exchange information, DAPs need an underlying facility that provides communication services to them. This facility is another DAF whose task is to provide and manage Inter Process Communication services over a certain scope ; hence this DAF is called DIF: Distributed IPC Facility - the DIF can be thought of as a layer. A DIF enables a DAP to allocate flows to one or more DAPs, by just providing the names of the targeted DAPs and the characteristics required for the flow (bounds on data loss and delay, in-order delivery of data, reliability, etc.). DAPs may not trust the DIF they are using, therefore may decide to protect their data before writing it to the flow - for example using encryption - via the SDU (Service Data Unit) Protection module.

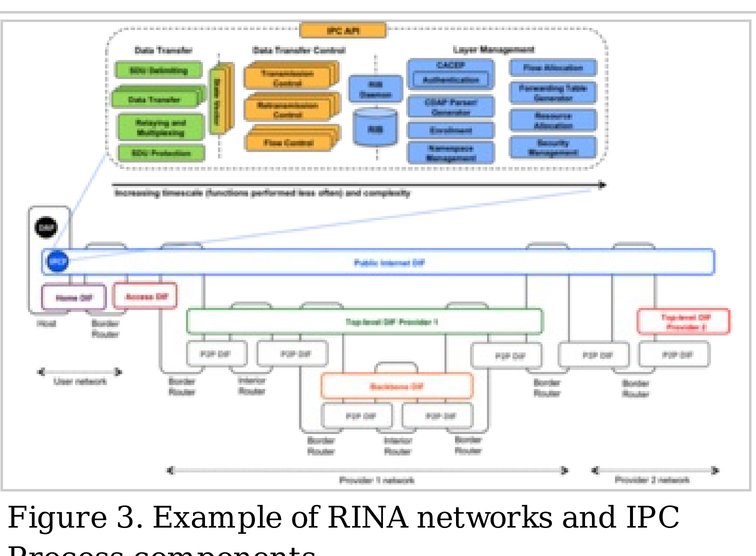
Figure 3. Example of RINA networks and IPC Process components

DIFs can also be the users of other underlying DIFs, creating in this way the recursive structure of the RINA architecture. The DAPs that are members of a DIF are called IPC Processes or IPCPs. They have the same generic DAP structure shown in Figure 2, plus some specific tasks to provide and manage IPC. These tasks, as shown in Figure 3, can be divided into three categories: data transfer, data transfer control and layer management. The elements are ordered in increasing complexity and frequency of use, with elements at the far left being used the most (per packet processing) but the least complex, and elements to the right being not often used, but very complex. All the layers provide the same functions and have the same structure and components, however these components are configured via policies in order to adapt to different operating environments.

As depicted in Figure 2 RINA networks are usually structured in DIFs of increasing scope, starting from the so-called lower layers and going up closer to the applications. A provider network can be formed by a hierarchy of DIFs multiplexing and aggregating traffic from upper layers into the provider's backbone. None of the provider internal layers need to be externally visible. Multi-provider DIFs (such as the public Internet or others) float on top of the ISP layers. Only three types of systems are required: hosts (which contain applications), interior routers (systems that are internal to a layer) and border routers (systems at the edges of a layer, which go one layer up or down). In short, RINA has the following features:

- It builds on a very basic premise, yet fresh perspective that networking is not a layered set of different functions but rather a single layer of distributed Inter-Process Communication (IPC) that repeats over different scopes. Each instance of this repeating IPC layer implements the same functions/mechanisms but policies are tuned to operate over different ranges of the performance space (e.g. capacity, delay, loss).

- It is based on a comprehensive theory of networking; it does not represent another patch, or point-solution to a piece of the problem. RINA does not propose to simply add a new “session layer” to perform some extra functionality for bridging ISP networks. Instead it takes a clean slate approach and begins with a comprehensive general theory of IPC where the number of IPC layers (DIFs) may vary at different parts of the Internet depending on the range of the resource allocation problem that must be addressed. The greater the operating ranges in a network, the more IPC layers it may have. Thus configuring the appropriate number of IPC layers allows for more predictable services to be delivered to users.

- This repeating structure scales indefinitely, thus it avoids current problems of growing routing tables, and supports features such as multi-homing and mobility, with little or no cost. By indefinitely we mean that the nature of RINA does not impose any limits. There may be physical limits and other constraints.

- An application process using a DIF only knows the name of the destination application process. It has no knowledge of addresses and there are no so-called “well-known ports”. Joining a DIF requires that the new member must be authenticated according to the policies of this particular facility. This yields a far more secure architecture.

- Stacking DIFs on top of each other allows networks to be built from smaller and more manageable layers of limited scope. This divide-and-conquer strategy gives providers more resource management options than just over-provisioning. It also provides the basis for operating subnetworks at much higher utilization than in the current Internet.

- RINA leverages the well-known concept of separating mechanism from policy in operating systems.[18] Applying this separation to network protocols allows a DIF to provide a common minimal set of mechanisms that once instantiated with the appropriate policies allows any transport solution to be realised.[19] Not only the transport functions of a DIF benefit from this approach, but also other ones such as management, authentication or access control; making the DIF a fully configurable container capable of effectively operating on top of heterogeneous physical medias and to provide differentiated levels of QoS to different types of applications.

- DIFs can be configured to not only provide the fundamental services of the traditional networking lower layers but also the services of application relaying (e.g. mail distribution and similar services), transaction processing, content distribution and peer-to-peer. 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.

- It turns out that private networks (with private addresses) are the norm. IPC processes are identified by addresses internal to the DIF and public networks are simply a degenerate case of a private network. This lays the foundation for major competition and innovation and avoids the rigidness of the current Internet structure. There's not just a single network where everybody has to be attached to; with RINA network operators, service providers and users have a choice of which networks to provide and which networks to join.

RINA compared to TCP/IP

Structure

While in operating systems layers are a convenience - a way to achieve modularity - in networks layers are a necessity, since in networks there is distributed shared state of different scopes. Layers are the tool for isolating distributed shared state of different scopes, nothing else is required. The current Internet architecture uses a layered architecture with a fixed number of layers, every layer having the responsibility of a different function, as depicted in Figure 4 (functional layering).

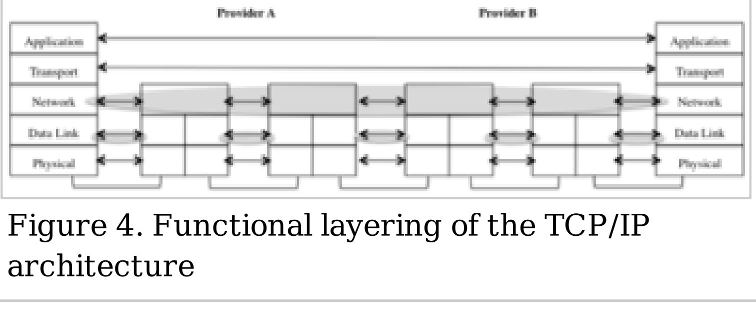
Figure 4. Functional layering of the TCP/IP architecture

The current architecture just provides two scopes: data link (scope of layers 1 and 2), and global (scope of layers 3 and 4). However, layer 4 is just implemented in the hosts, therefore the “network side” of the Internet ends at layer 3. This means that the current Internet is able to handle a network with heterogeneous physical links, but it is not designed to handle heterogeneous networks, although this is supposed to be its operation. To be able to do it, it would require an “Internetwork” scope, which is now missing.[7] As ironic as it may sound, the current Internet is not really an internetwork, but a concatenation of IP networks with an end-to-end transport layer on top of them. The consequences of this flaw are several: both inter-domain and intra-domain routing have to happen within the network layer, and its scope had to be artificially isolated through the introduction of the concept of Autonomous System (Internet) and an Exterior Gateway Protocol;[8] Network Address Translation (NATs) appeared as middleboxes in order to have a means of partitioning and reusing parts of the single IP address space.[20]

With an internetwork layer none of this would be necessary: inter-domain routing would happen at the internetwork layer, while intra-domain routing within each network would occur at each respective network layer. NATs would not be required since each network could have its own internal address space; only the addresses in the internetwork layer would have to be common. Moreover, congestion could be confined to individual networks, instead of having to deal with it at a global scope as it is done today. The internetwork layer was there in previous internetwork architectures, for example, the INWG architecture depicted in Figure 3, which was designed in 1976. It was somehow lost when the Network Control Program was phased out ant the Internet officially started in 1983.

The second issue in functional layering is that each layer is supposed to provide a different function, and that this function must not be repeated in the other layers of the stack. A quick analysis on today's protocols shows that there are repeated functions in different layers; what is more they tend to alternate: layer 1 provides multiplexing and relaying over a physical media, layer 2 provides error and flow control over a data link, layer 3 provides multiplexing and relaying over a network, layer 4 provides error and flow control end-to-end. Finally, the third issue is that layers have to be independent, in order to isolate the shared state of different scopes and allow scalability. Layer violations (layers that use other layer's information in order to achieve their job) are in the order of the day, starting with the TCP pseudo-header calculated with the information of IP source and destination addresses.

RINA goes beyond the static layering concepts and defines a layer as a distributed application that provides IPC services to applications (or other layers) that use it. In RINA layers are recursive; there is not a fixed number of layers in the stack. The number of layers in any given network is variable. There are simply as many as deemed necessary by the network designers. All layers use the same protocols that can be policy-configured to optimally support the operational requirements of each particular layer. The protocols running at each layer have the potential to provide all the functionality required to allow the layer to operate efficiently: data transport, data transport control, multiplexing, relaying, congestion control, routing, resource allocation, enrollment, authentication, access control, integrity and so forth. The number of these functions instantiated at each layer and how they behave can be configured on a layer-by-layer basis.

Naming, addressing, routing, mobility and multi-homing

The current Internet architecture has an incomplete naming and addressing schema, which the reason why mobility and multi-homing require ad-hoc solutions and protocols tailored to different operational environments. The only names provided are Point of Attachment (PoA) names (IP addresses), which are usually confused to be node names. The

result is that the network has no way to understand that the two or more IP addresses of a multi-homed node belong to the same node, making multi-homing hard. 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. Mobility, which can be seen as dynamic multi-homing, is the next feature that suffers from having an incomplete naming schema.

In 1982, Jerry Saltzer in his work "On the Naming and Binding of network destinations" [21] described the entities and the relationships that make a complete naming and addressing schema in networks. According to Saltzer four are the elements that need to be identified: applications, nodes, points of attachment to the network (PoA) and paths. An application can run in one or more nodes and should be able to move from one node to another without losing its identity in the network. A node can be connected to a pair of PoAs and should be able to move between them without losing its identity in the network. A directory maps an application name to a node address, and routes are sequences of nodes addresses and point of attachments.

Saltzer took his model from operating systems, but it was not completely correct for internetworks, since there may be more than one path between the same pair of nodes (even whole networks!). The obvious solution is that routes are sequences of nodes, and at each hop each node chooses the most appropriate PoA (path) to forward the packet to the next node. Therefore, routing is a two-step process: First the route as a sequence of node addresses is calculated, and then, for each hop, an appropriate PoA to select the specific path to be traversed is chosen. These are the steps to generate the forwarding table, forwarding is still performed with a single lookup. Moreover, the last step can be performed more frequently in order to exploit multi-homing for load-balancing.

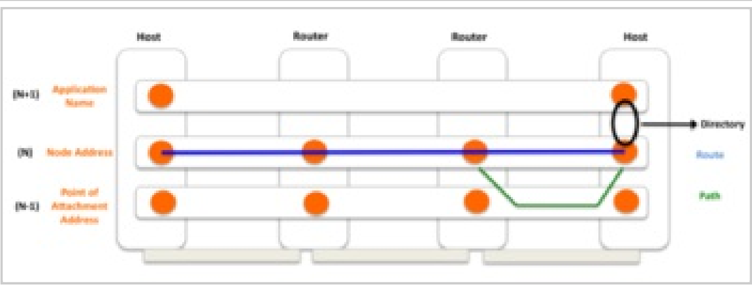
Figure 5. Saltzer's point of view on naming and addressing in computer networks.

With this naming structure, support for mobility and multi-homing is inherent, if the properties for the names are chosen with care: application names are location-independent to allow an application to move around, node addresses are location-dependent but route-independent. PoA addresses are by nature route-dependent. Applying this naming scheme to RINA, with recursive layers, an interesting observation can be made: mapping application names to node addresses is the same mapping than mapping node addresses to PoAs. In other words, for any layer N, nodes at the layer N+1 are applications and nodes at the layer N-1 are points of attachment, making this relationship relative.

The Locator/Identifier Separation Protocol (LISP or Loc/ID split) [22] has been proposed by IETF as a solution to issues as the scalability of the routing system. LISP main argument is that the semantics of the IP address are overloaded being to be both locator and identifier of an endpoint. LISP proposes to address this issue by separating the IP address into a locator part, which is hierarchical, and an identifier, which is flat. However this is a false distinction: in Computer Science it is impossible to locate something without identifying it and to identify something without locating it, since all the names are using for locating an object within a context.[21] Moreover, LISP continues to use the locator for routing, therefore routes are computed between locators (inter-faces). However, a path does not end on a locator but on an identifier, in other words, the locator is not the ultimate destination of the packet but a point on the path to the ultimate destination.[14] This issue leads to path-discovery problems, as documented by [13] whose solution is known not to scale.

RINA adopts and extends Saltzer's model by supporting internetworks, and making it recursive. It has a complete naming and addressing schema, providing names for the basic entities of the network (nodes, PoAs and applications). As a consequence RINA supports mobility and multi-homing inherently [23]

Protocol Design

A protocol can effectively serve different applications with a wide range of requirements as long as this is the goal from the beginning of the protocol design. In RINA, policy and mechanism are separated, resulting in a framework than can be fine-tuned through policy specification. The mechanism of a protocol may be tightly bound, such as the headers of a data transfer packet, or loosely bound, as in some control and acknowledgment messages. The use of common mechanisms in conjunction with different policies, rather than the use of separate protocols brings greater flexibility.[18] Each DIF can use different policies to provide different classes of quality of service to further adapt to the characteristics of the physical media (if the DIF is close to the lower end of the stack) or to the characteristics of the applications (if the DIF is close to the upper end of the stack).

By separating mechanism and policy, RINA dramatically simplifies networking. There is no longer a different set of modules and protocols for each capability in each layer. Instead, the elements of the DIF combine to accomplish functions that have previously required a multitude of individual specifications. The implications of this are significant: rather than hundreds of handcrafted protocols each with their own optimizations, there is a consistent repeating structure of two protocols (an application protocol, a data transfer protocol) and a common header and roughly a half dozen modules. In this approach there is thus far less to go wrong. A single replicable layer of a half dozen or so modules can be much more effectively engineered to be free of bugs and security holes than the literally hundreds of protocols in conventional Internetworking. Furthermore, the inherently constrained nature of policies makes it possible to circumscribe their side effects and ensure their proper behaviour. It is possible to define properties of policies that can be proved or tested that ensure proper behaviour.

The basis of the data transfer control protocol in RINA is the Delta-T protocol.[6] Watson proved that the necessary and sufficient conditions for reliable transfer is to bound three timers. Delta-T is an example of how this should work. It does not require a connection setup (such as TCP's SYN handshake) or tear-down (like TCP's FIN handshake) for integrity purposes. In fact, TCP uses the same three timers! So RINA avoids unnecessary overhead. Watson showed that synchronization and port allocation are distinct functions. Port allocation is part of DIF management, while synchronization is part of the data transfer protocol. In fact, maintaining the distinction also improves the security of connection setup, and avoids the need for separate protocols like IPSec, since multiple transport connections can be associated to the same port allocation.

By separating mechanism from policy, RINA dramatically reduces the number of protocols required in the architecture, while still allowing each layer to be customized to provide different levels of quality of service. Applying Watson's theory of separating port allocation from synchronisation enables simpler, more robust and more reliable data transfer protocols.

Security

The recursive model of the RINA architecture provides a clear security model in which the trust relationships between layers (DAFs or DIFs) and between members of a single layer are well identified. Figure 6 illustrates these trust boundaries, which facilitate the placement of the different security functions in the RINA architecture - in contrast to the Internet where security is designed in every protocol instead of at the system-level, making security complex, expensive and brittle. Research on RINA security properties to date has already produced some promising results.

Resiliency to data transport attacks. IPCP (node) addresses are internal to a DIF and not exposed to applications, data connections are dynamically assigned connection-endpoint ids (CEP-ids) that are bound to dynamically assigned ports. Bodappati et al.[24] showed that due to this decoupling of transport port allocation and access control from data synchronization and transfer RINA was much more resilient than TCP/IP to transport-level attacks such as port-scanning, connection opening or data-transfer.

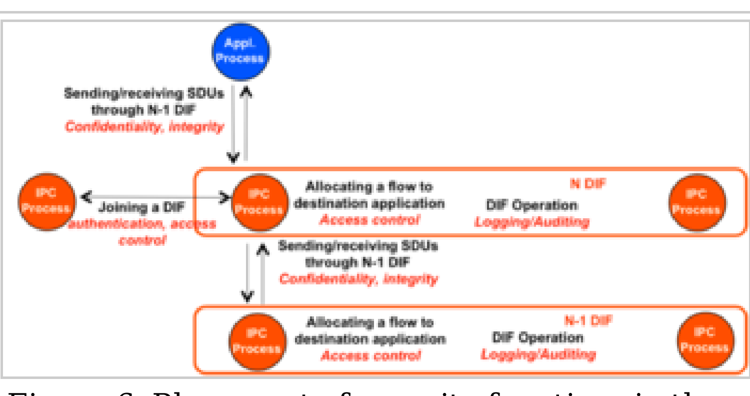
Figure 6. Placement of security functions in the RINA architecture.

DIFs are securable containers, no firewalls are necessary. Small et al. [25] performed a threat analysis at the RINA architecture level, concluding that DIFs are securable containers. That is, if proper authentication, authorization, confidentiality, integrity protection and auditing policies are put in place (as identified in section 2.1) a DIF is a structure used to transport data that can be made to be not subject to thread. No external tools such as firewalls are required.

Complexity of RINA security is lower than that of the current Internet security. This is a consequence of the different approach to each of the architectures: the system design approach adopted by the RINA architecture, is based on identifying the proper placement of all the functions within the architecture - in contrast to the disorganized evolution of the Internet "protocol-suite" which leads to unnecessary redundancy and complexity. This is particularly evident in comparing the number of protocols and mechanisms required to add security to both RINA and the Internet: Small[26] showed that the current Internet security had for 4-5 times the overhead of RINA security. Less overhead means not only less cost but also more effective security, since complexity is the worst enemy of security [27]

Other aspects

Quality of Service. Another shortcoming of the TCP/IP architecture is that there is no built-in mechanism that allows the network to provide specific QoS levels. Applications have no way of asking for certain QoS parameters, since the sockets API only allows to specify a reliable (TCP) or unreliable (UDP) transport service. Within RINA 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. The DIF API allows applications (being general applications or other DIFs) to provide QoS requirements when they request an IPC service.

Greater robustness and more effective response to change. Response to change is far faster thanks to load balancing, quicker convergence due to much smaller routing table sizes, more responsive flow management, and, on the manpower side, simpler and more effective operational management. RINA provides all of the flexibility and survivability of connectionless networking while supporting all the service capabilities of connection-oriented networking. Data flows under hostile environments are far more reliable due to highly tuneable, situation-specific policies. As new demands arise, systems can be quickly reconfigured to accommodate them simply by configuring new policies.

A more competitive marketplace. The particular "best-effort" architecture adopted by the Internet does relegate providers to a commodity business. The Transport Layer (TCP) effectively seals the providers off in the lower layers with IP providing a best-effort service. This implies that everyone must do the same thing or nearly so, leaving little room for differentiation and competition and relegates them to a commodity business. RINA creates the robust feedback needed for a healthy marketplace.[28] An application API allows applications to request service with specific QoS requirements from the layer below. 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. This distributed IPC can be configured to not only provide the fundamental services of the traditional networking lower layers but also the services of application relaying, e.g. mail distribution and similar services; transaction processing, and peer-to-peer.

Internetworking. In RINA the public Internet can be seen as a public e-mall floating on top of multiple providers. This is what we are trying to do today but lacking the right tools to do it. In RINA each level of the hierarchy has an independent address space, managed by the provider it belongs to. As opposed to what we are used today, other e-malls potentially more upscale are possible with other characteristics such as tighter security for joining, perhaps specialized to certain market segments, creating a real Internetwork. An example of several Internetworks is shown in Figure 7. Lower level DIFs belong to ISPs, which decide for the configuration, the address space and policies in these DIFs. Higher layer DIFs can be accessible by everybody. For example, Facebook could be considered boutique e-malls in contrast to the mega-malls like the Internet. It could benefit from the improved security and ability to create focused communities with tighter controls. There is no public network or address space to which one must belong. Any network you are part of is by choice. Networks have considerable flexibility in who they provide their services. This would encourage alliances among groups of providers with complementary interests to provide QoS services in competition with groups of other providers.

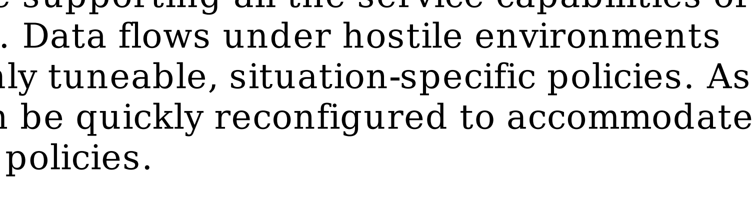
Figure 7. Multiple RINA networks supporting several internetworks.

Research projects

From the publishing of the PNA book in 2008 until 2014 a lot of RINA research and development work has been done. There is a clear need for an international authority that coordinates the different ongoing activities, make sure their results are integrated in the basic reference model but at the same time are able to incorporate new knowledge or fix inconsistencies. An informal group known as the Pouzin Society (PSOC) [29]- named after Louis Pouzin,[30] the inventor of datagrams and connectionless networking - has been taking this role.

BU Research Team

The RINA research team at Boston University [31] is lead by Prof. Abraham Matta and Prof. John Day. BU has been awarded a number of grants from

the National Science Foundation in order to continue investigating the fundamentals of RINA, develop an open source prototype implementation over UDP/IP for Java.[32] and experiment with it on top of the GENI infrastructure.[33] BU is also a member of the Pouzin Society and an active contributor to the FP7 IRATI and PRISTINE projects. In addition to this, BU has incorporated the RINA concepts and theory in their computer networking courses.

FP7 IRATI

IRATI [34] is an FP7-funded project with 5 partners: i2CAT, Nextworks, iMinds, Interoute and Boston University, whose main goal is to produce an open source RINA implementation for the Linux OS on top of Ethernet,.[35][36] FP7 IRATI has already open-sourced the first release of the RINA implementation, called as the project "IRATI".[37] The implementation will be further enhanced by the PRISTINE and IRINA projects.

FP7 PRISTINE

PRISTINE [38] is an FP7-funded project with 15 partners: WIT-TSSG, i2CAT, Nextworks, Telefónica I+D, Thales, Nexedi, B-ISDN, Atos, University of Oslo, Juniper Networks, Brno University, IMT-TSP, CREATE-NET, iMinds and UPC; whose main goal is to explore the programmability aspects of RINA to implement innovative policies for congestion control, resource allocation, routing, security and network management.

GEANT3+ Open Call winner IRINA

IRINA [39] was funded by the GEANT3+ open call, and is a project with four partners: iMinds, WIT-TSSG, i2CAT and Nextworks. The main goal of IRINA is to study the use of the Recursive InterNetwork Architecture (RINA) as the foundation of the next generation NREN and GÉANT network architectures. IRINA builds on the open source RINA prototype developed by the FP7 IRATI project. IRINA will compare RINA against current networking state of the art and relevant clean-slate architecture under research; perform a use-case study of how RINA could be better used in the NREN scenarios; and showcase a laboratory trial of the study.

References

1. Patterns in Network Architecture: A Return to Fundamentals, John Day (2008), Prentice Hall, ISBN 978-0132252423
2. A. McKenzie, "INWG and the Conception of the Internet: An Eyewitness Account"; IEEE Annals of the History of Computing, vol. 33, no. 1, pp. 66-71, 2011
3. R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291 (Draft Standard), February 2006. Updated by RFCs 5952, 6052
4. C.A. Kent and J.C. Mogul. Fragmentation considered harmful. Proceedings of Frontiers in Computer Communications Technologies, ACM SIGCOMM, 1987
5. R. Watson. Timer-based mechanism in reliable transport protocol connection management. Computer Networks, 5:47-56, 1981
6. R. Watson. Delta-t protocol specification. Technical Report UCID-19293, Lawrence Livermore National Laboratory, December 1981
7. J. Day. How in the Heck Do You Lose a Layer!? 2nd IFIP International Conference of the Network of the Future, Paris, France, 2011
8. E.C. Rosen. Exterior Gateway Protocol (EGP). RFC 827, October 1982. Updated by RFC 904.
9. D. Davies. Methods, tools and observations on flow control in packet-switched data networks. IEEE Transactions on Communications, 20(3): 546–550, 1972
10. S. S. Lam and Y.C. Luke Lien. Congestion control of packet communication networks by input buffer limits - a simulation study. IEEE Transactions on Computers, 30(10), 1981.
11. Internet Architecture Board. IP Version 7 ** DRAFT 8 **. Draft IAB IPversion7, July 1992
12. Internet Architecture Board. IAB Recommendations for the Development of Internet Network Management Standards. RFC 1052, april 1988
13. D. Meyer and D. Lewis. Architectural implications of Locator/ID separation. Draft Meyer Loc Id implications, january 2009
14. J. Day. Why loc/id split isn't the answer, 2008. Available online at http://rina.tssg.org/docs/LocIDSplit090309.pdf
15. L. Pouzin. Methods, tools and observations on flow control in packet-switched data networks. IEEE Transactions on Communications, 29(4): 413–426, 1981
16. D. Clark, L. Chapin, V. Cerf, R. Braden and R. Hobby. Towards the Future Internet Architecture. RFC 1287 (Informational), Decembe 1991
17. John Day, Ibrahim Matta and Karim Mattar. Networking is IPC: A guiding principle to a better Internet. In Proceedings of the 2008 ACM CoNEXT Conference. ACM, 2008
18. P. Brinch Hansen. The nucleous of a multiprogramming system. Communications of the ACM, 13(4): 238-241, 1970
19. I. Matta, J. Day, V. Ishakian, K. Mattar and G. Gursun. Declarative transport: No more transport protocols to design, only policies to specify. Technical Report BUCS-TR-2008-014, CS Dept, Boston. U., July 2008
20. K. Egevang and P. Francis. The IP Network Address Translator (NAT). RFC 1631 (Informational), May 1994. Obsoleted by RFC 3022
21. J. Saltzer. On the Naming and Binding of Network Destinations. RFC 1498 (Informational), August 1993
22. D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. Locator/ID Separation Protocol (LISP). Draft IETF LISP 18, december 2011
23. V. Ishakian, J. Akinwumi, F. Esposito, I. Matta, "On supporting mobility and multihoming in recursive internet architectures". Computer Communications, Volume 35, Issue 13, July 2012, pages 1561-1573
24. G. Bodduapati, J. Day, I. Matta, L. Chitkushev, "Assessing the security of a clean-slate Internet architecture," Network Protocols (ICNP), 2012 20th IEEE International Conference on , vol., no., pp.1,6, Oct. 30 2012-Nov. 2 2012
25. J. Small, J. Day, L. Chitkushev, "Threat analysis of Recursive Inter-Network Architecture Distributed Inter-Process Communication Facilities". Boston University Technical Note.
26. J. Small. "Patterns in Network Security: An analysis of architectural complexity in securing Recursive Inter-Network Architecture Networks". Master thesis, 2012. Boston University Library
27. B. Schneier, "Complexity the worst enemy of Security". Computer World, December 2012.
28. J. Day. Creating a viable economic model for a viable internet, 2008. Available online at http://rina.tssg.org/docs/PNAEcon080518.pdf
29. Pouzin Society website: http://www.pouzinsociety.org
30. A. L. Russell, V. Schaffer. "In the shadow of ARPAnet and Internet: Louis Pouzin and the Cyclades network in the 1970s". Available online at http://muse.jhu.edu/journals/technology_and_culture/v055/55.4.russell.html
31. Boston University's RINA research team website: http://csr.bu.edu/rina
32. ProtoRINA github site: https://github.com/ProtoRINA/users/wiki
33. Yuefeng Wang, Ibrahim Matta and Nabeel Akhtar. "Experimenting with Routing Policies Using ProtoRINA over GENI". The Third GENI Research and Educational Experiment Workshop (GREE2014), March 19–20, 2014, Atlanta, Georgia
34. The FP7 IRATI project website: http://irati.eu
35. S. Vrijders, D. Staessens, D. Colle, F. Salvestrini, L. Grasa, M. Tarzan and L. Bergesio "Prototyping the Recursive Internetwork Architecture: The IRATI Project Approach", IEEE Network, Vol. 28, no. 2, March 2014
36. S. Vrijders, D. Staessens, D. Colle, F. Salvestrini, V. Maffione, L. Bergesio, M. Tarzan, B. Gaston, E. Grass; "Experimental evaluation of a Recursive InterNetwork Architecture prototype", IEEE Globecom 2014, Austin, Texas
37. Open IRATI github site, available at: http://irati.github.io/stack
38. The FP7 PRISTINE project website: http://ict-pristine.eu
39. The IRINA project website: http://www.geant.net/opencall/Optical/Pages/IRINA.aspx

External links

- RINA Education page at the IRATI website, available online at http://irati.eu/education/
- RINA document repository run by the TSSG, available online at http://rina.tssg.org
- RINA tutorial at the IEEE Globecom 2014 conference, available online at http://www.slideshare.net/irati-project/rina-tutorial-ieee-globecom-2014