Advanced Computer Networks

Professor Ibrahim Matta

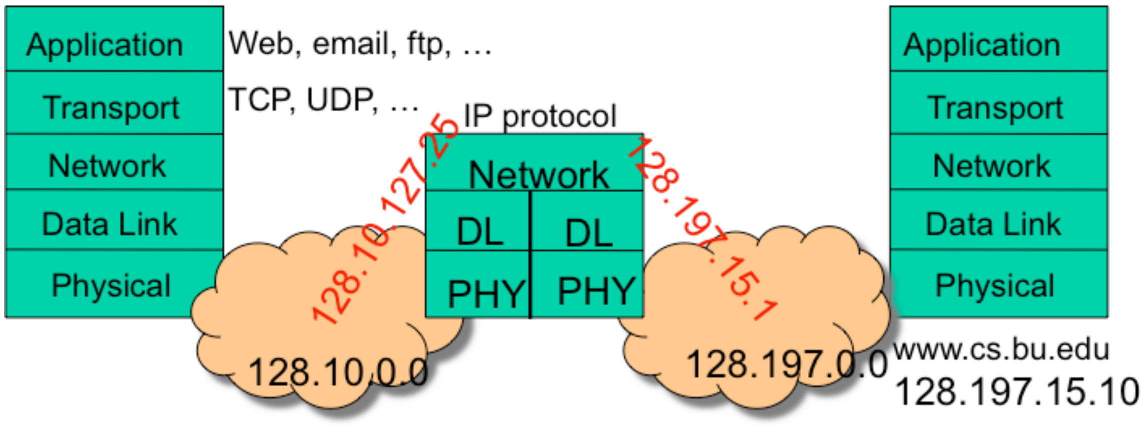
Naming and Addressing

We discuss the important issue of naming and addressing in the context of our new Recursive service-based Inter-Network Architecture called RINA. In these notes, we give a brief overview of RINA. They serve as complement to the slides discussed in class and posted on the course website. The RINA website can be found at: http://csr.bu.edu/rina/

Overview of RINA

Today's Network Architecture

The current Internet architecture is built around layers of *different* functions, where the Network Layer provides a technology-independent abstraction on top of a large set of autonomous, heterogeneous networks. The Internet Protocol (IP) is one mechanism for achieving such an abstraction. By making the choice for a rudimentary ``best-effort'' service, the Internet has not been able to effectively respond to new requirements (security, manageability, wireless, mobility, *etc.*) Today, Internet Service Providers (ISP's) may be willing to provide better than best-effort service to their customers or their peers for a price or to meet a Service Level Agreement (SLA). The lack of a structured view of how this could be accomplished, given the current IP model, has led to numerous ad hoc solutions that are either inefficient or incomplete. There is a lack of a *building block* that can be composed to provide a wide scope service that meets given user requirements.



How Is RINA Different?

The current Internet architecture has its shortcomings:

(1) Naming and addressing interfaces rather than nodes. By *early binding* communication to an IP address, which identifies a particular connection, it becomes hard to take advantage of other connections that a multi-homed node has.

(2) Exposing addresses to applications. By hard-coding IP addresses in applications, having them public, together with well-known ports, the network becomes vulnerable to attacks.

(3) Artificially isolating functions of the same scope[1]. This isolation makes it hard for layers of the same scope to interact and exchange relevant information. For example, the transport layer may not have access to the exact reasons of packet losses inside the network and may react to mobility related errors in the same way it reacts to congestion-induced losses.

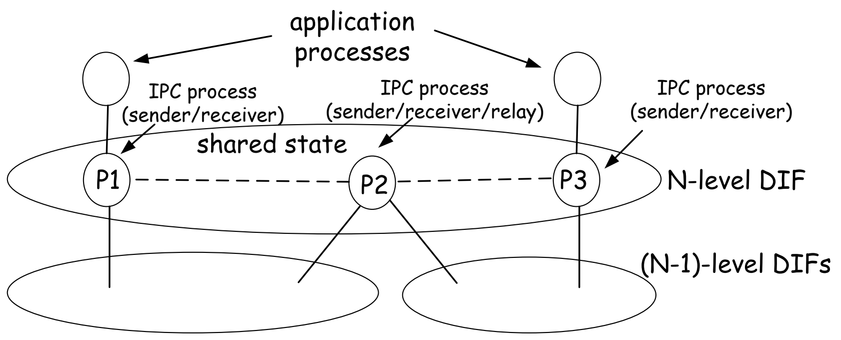
(4) Artificially limiting the number of layers (levels). The Internet is basically designed around a two-tiered routing system when different levels of management may better control different parts of the Internet.

On the other hand, RINA leverages the inter-process communication (IPC) concept. In an operating system, to allow two processes to communicate, IPC requires certain functions such as locating processes, determining permission, passing information, scheduling, and managing memory. Similarly, two applications on different end-hosts should communicate by utilizing the services of a Distributed IPC Facility (DIF). A DIF is an organizing structure – what we generally refer to as a "layer." However, what functions constitute this layer is fundamentally different in this IPC model. A DIF is a collection of IPC processes (nodes). Each IPC process executes transport and management (routing, directory services, access control, resource allocation, security, *etc.*) functions, i.e. all what is needed to manage a "private communication network/community". IPC processes communicate and share state information to provide specific services to their users (application processes) through a well-defined service interface (API). How a DIF is managed, including addressing, is hidden from the applications. RINA's API allows applications to only ask for service by name – a *location-independent* application/service name. In contrast, TCP/IP exposes IP addresses, which name interfaces, and thus are location-dependent! In RINA, the underlying DIF layer itself maps service name to a *location-dependent node address*. This level of indirection allows a service / application to move to a new location while the service name remains the same.

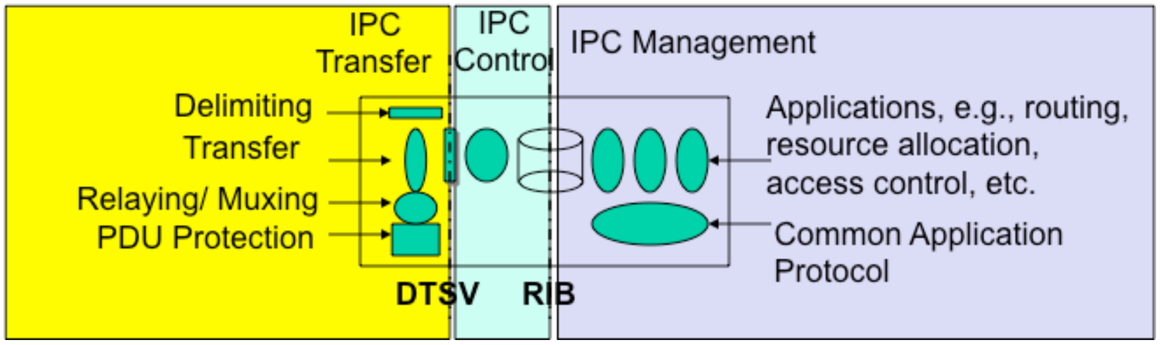
RINA is a clean-slate internet architecture that builds on this very basic premise, yet fresh perspective that networking is not a layered set of different functions but rather a *single layer* of distributed inter-process communication (DIF) that *repeats* over different scopes – i.e., same mechanisms (a la recursive function) but policies (a la parameters) are tuned to operate over different ranges of the performance space (e.g., capacity, delay, loss) so as to accommodate the quality of services provided by lower-level DIFs (service providers) and to meet those requested by its users (application processes, or IPC processes of higher-level DIFs). Specifically, a scope defines a DIF comprised of the set

[1] Transport and routing/relaying functions are split into two separate layers: over the same domain/link they are split into the Data Link and Physical layers. Internet-wide they are split into the Transport and Network layers.

of IPC processes, running on different machines, that collaboratively provide a set of well-defined (transport) flow services to upper application processes. Application (user) processes can themselves be IPC processes of an upper DIF that is providing services over a wider scope. Thus, RINA has one building block (the DIF layer) that can be composed to form a higher-level DIF from the services provided by lower-level DIFs – the figure below illustrates composition by recursion of the DIF layer.


application processes
IPC process (sender/receiver)
IPC process (sender/receiver/relay)
IPC process (sender/receiver)
shared state
P1
P2
P3
N-level DIF
(N-1)-level DIFs

The figure below illustrates all functions implemented by an IPC process in a RINA layer (DIF). RINA separates mechanism (the *how*) from policy (the *what*). This is done by first separating *tightly bound mechanisms*, which involves fast data transfer and PCI (Protocol Control Information, such as source and destination addresses, sequence number, TTL, etc.), from *loosely bound mechanisms*, which involves sending rate adaptation, routing updates, etc. Policies determine what mechanism is activated, for example sending rate adaptation can be activated using a certain flow/congestion control algorithm.


IPC Transfer
IPC Control
IPC Management
Delimiting
Transfer
Relaying/ Muxing
PDU Protection
DTSV
RIB
Applications, e.g., routing, resource allocation, access control, etc.
Common Application Protocol

This separation of mechanism from policy yields three loci of processing operating over different timescales from short to long:

- **IPC Transfer** actually moves the data and include *tightly coupled* mechanisms such as fragmenting or packing SDUs (Service Data Units) coming from application processes into packets / PDUs (Protocol Data Units). A PDU encapsulates SDU(s) with PCI information such as source and destination addresses, a TTL field, etc.[2]
- **IPC Control** (optional) for error, flow control, etc. A Data Transfer State Vector (DTSV) maintains the state of a flow (connection), e.g. current transmission window size, RTT estimate,

[2] A PDU is basically a packet that the DIF layer transports and routes to the destination process.

legal sequence numbers, etc.
- **IPC Management** for routing, resource allocation, locating applications, access control, monitoring lower layer, etc. Here, these functions are viewed as application processes which make use of one "stateless" Common Distributed Application Protocol (CDAP) to access objects of different kinds (e.g., routing tables, access control lists) stored in a RIB (Resource Information Base). CDAP supports a limited number of operations on objects like CREATE, DELETE, UPDATE, etc.

Benefits of RINA

The RINA architecture possesses features that intrinsically solve long-standing networking problems. Most notably, the repeating structure of its distributed IPC model allows it to scale indefinitely, thus it avoids current problems of growing routing tables (by limiting the number of processes within each DIF layer), and supports features such as multihoming and mobility, with little cost (by localizing DIF management). Furthermore, RINA views each DIF as a ``privately managed'' network thus it offers intrinsic security features.

The distributed IPC model also creates the robust feedback needed for a healthy marketplace. 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.

How Does RINA Work?

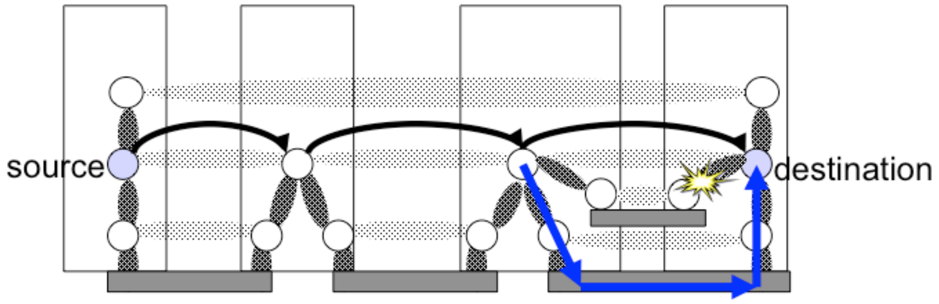
We next highlight two distinguishing features of the RINA architecture: (1) RINA's inherent support for mobility and multihoming, and (2) RINA's separation of security and data-synchronization concerns and support for explicit security and robust soft-state data synchronization.

More details (presentations, papers, *etc.*) on the RINA architecture can be found at the RINA website: http://csr.bu.edu/rina/

Mobility and Multihoming

As the Internet has evolved and grown, an increasing number of nodes (hosts or autonomous systems) have become multihomed, i.e., a node is connected to more than one network. Multihoming can be viewed as a special case of mobility – as a node moves, it unsubscribes from one network and subscribes to another, which is akin to one interface becoming inactive and another active. The current Internet architecture has been facing significant challenges in effectively dealing with mobility (and consequently multihoming), which has led to the emergence of several custom point-solutions. RINA inherently supports mobility and multihoming as it treats them as *local routing updates* as opposed to wider scoped tunnel changes in existing architectures (e.g., LISP) as we explain next.

The figure below illustrates how RINA routes from a source IPC process (node) to a destination IPC process (node) to deliver messages to a given destination application process. After mapping the destination *application name* to a destination (IPC process) *node address*, the DIF layer computes a route as a sequence of (IPC process) node addresses at that DIF layer. At each hop, the underlying lower DIF layer maps the node address (viewed as a name by the underlying lower layer) to a corresponding Point-of-Attachment (PoA) (viewed as a node address in the underlying lower layer). Thus, addressing in RINA is *relative* and not static as in current architectures – a node address is viewed as node name by the lower layer and as a PoA by the higher layer. This yields an architecture where a node address is *late bound* to a PoA. In the figure below, by the time the message reaches the last hop, the mapping of the destination node address has been *locally* updated to the operational/active PoA (interface) over which the message gets delivered.



RINA's approach to routing is in contrast to existing approaches, such as LISP (Location/ID Separation Protocol), which relies on wide scoped tunnels. For example, in LISP, a host is assigned a location-independent identifier (EID). An ingress border router transparently maps the destination EID to the IP address of an egress border router (called Routing LOCator, RLOC) that currently has a path to the destination host. The packet is then encapsulated to that RLOC. A fundamental problem with such loc/id split approaches is that the "loc" is not the location of the ultimate destination node but it specifies some point on the path to the destination node (i.e. PoA). This form of *early binding* of the node name to a PoA is still as problematic as the early binding of Internet name to IP address in the current TCP/IP architecture – if the PoA fails, one has to wait for the new mapping to propagate over the whole Internet!

Transport and Security

RINA's separation of security and data-synchronization concerns enables an explicit organized placement of security mechanisms (authentication, access control, and encryption), and a more general and robust *soft-state* approach to data synchronization.

The figure below illustrates the recursive placement of security functions within the RINA architecture. Application processes authenticate themselves before exchanging data. This is the case before application processes start exchanging data over a DIF, or when application processes communicate to first join a DIF layer and before they start acting as IPC processes exchanging messages among themselves within that DIF. IPC processes encrypt data they send over an underlying DIF layer that they do not trust.

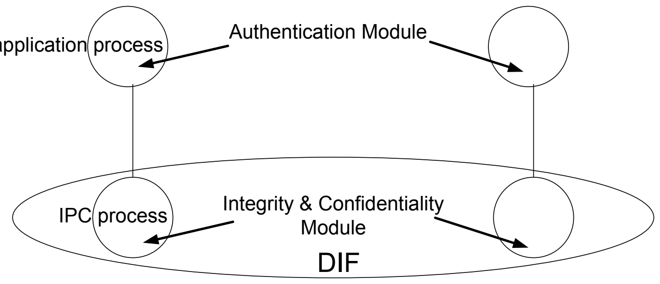


Fig. 6. Security policies applied recursively

Before application processes start authenticating themselves and exchanging data, they have to first request from the underlying DIF that a communication flow be established. If successful, the DIF would return to the application process a handle (port number) through which to send its messages. When establishing a communication flow through a DIF, RINA decouples port allocation and access control from data transfer. The management part of the DIF layer is responsible for allocating source and destination ports (and connection endpoint IDs) and for performing access control. A Flow Allocation management application uses the CDAP protocol for this purpose to carry out a request/reply exchange between the source and destination IPC processes.[3]

Once a flow is allocated, data transfer can start. In a RINA DIF layer, the Data Transfer Protocol (DTP), which includes tightly bound mechanisms, and the Data Transfer Control Protocol (DTCP), which includes loosely bound mechanisms, are modeled after the *soft-state* Delta-t transmission protocol. Different policies support different requirements, e.g. flows without data transfer control (no ACKs or transmission window) are UDP-like. In a soft-state approach, the sender and receiver IPC processes do no need to synchronize their sequence numbers. The sender can start from any initial sequence number. If there is a long idle period, the connection state (DTSV) is discarded, but ports (and connection endpoint IDs) remain. The idle period must be long enough to ensure that all packets and their duplicates have cleared the network. When data transfer resumes, a DTSV is automatically re-created and the sender can again start transmission from any initial sequence number.

Note that connection endpoint IDs can change over the lifetime of the transfer and bound to the same ports. Dynamically allocating port numbers and connection endpoint IDs makes it hard to mount transport-level attacks – no well-known ports, and connection IDs are internal to the DIF layer. Furthermore, to mount a transport-level attack, a process has to join the DIF layer first, a hurdle that does not exist in TCP/IP networks.

[3] Note that for CDAP-based management applications to communicate, they have to *recursively* request from the underlying DIF layer that a flow be established. Once this is successful, they can start exchanging messages through the returned port number.