



Intro

RINA

Example

PRISTINE

RINA

Vladimír VESELÝ
6.3.2014





INTRODUCTION

Intro

RINA

Example

PRISTINE

- ◆ Motivation
- ◆ Step-by-step evolution

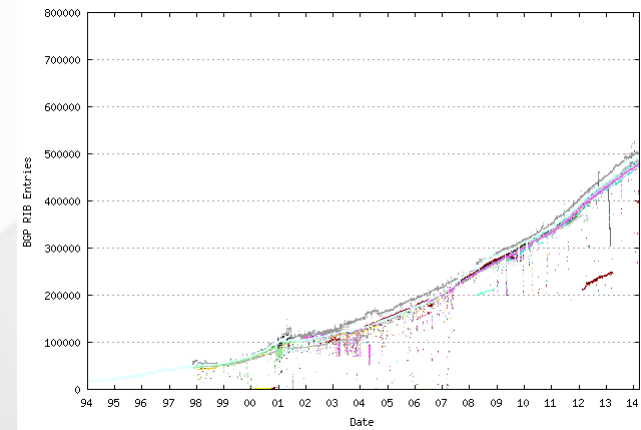




PROBLEMS

◆ DFZ Routing Table growth

- ◆ maximal state vs. minimal state



◆ Kludged multihoming

- ◆ IP address names interface

◆ Cumbersome mobility

- ◆ MobileIP is not really easy to deploy



What is unique address???

◆ Congestion control is not done as close to resource

- ◆ TCP treats any message lost as congestion

◆ Inefficient QoS

- ◆ IntServ too complicated and resource intensive
- ◆ DiffServ too unreliable

Intro

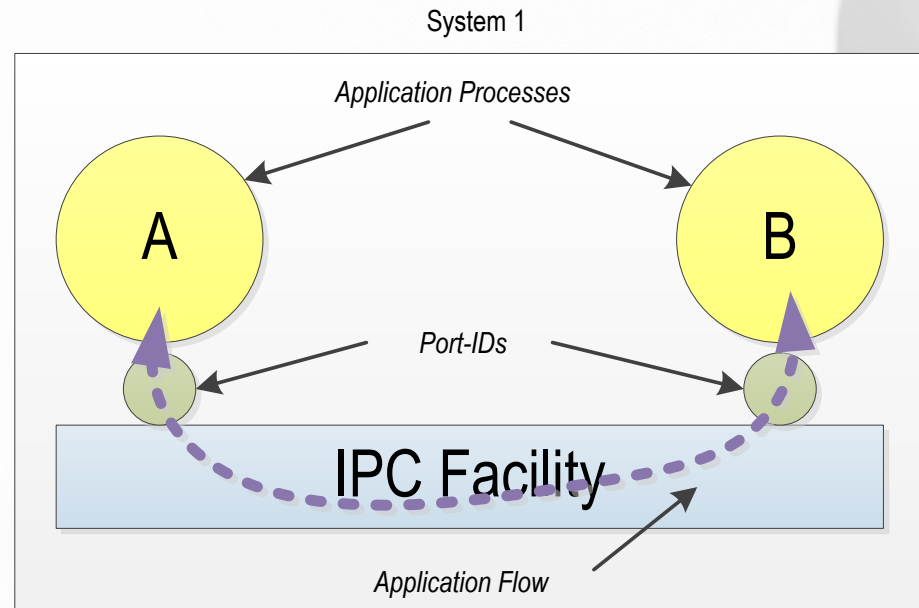
RINA

Example

PRISTINE



TWO APPS COMMUNICATING IN THE SAME SYSTEM



- 1) **A** invokes IPC to create a channel to **B**; **a = Allocate (B, QoS)**
- 2) IPC determines whether it has the resources to honor the request.
- 3) If so, IPC allocates port-id **a** and uses “search rules” to find **B** and determine whether **A** has access to **B**.
- 4) IPC may cause an instance of **B** to be created. **B** is notified of the IPC request from **A** and given a port-id, **b**.
- 5) If **B** responds positively, and IPC notifies **A** (the API could be blocking in which case the assignment of the port-id, **a** would be done now).
- 6) Thru **n**) Then using system calls **A** may send PDUs to **B** by calling **Write(a, buf)**, which **B** receives by invoking **Read(b, read_buffer)**
- 7) When they are done one or both invoke **Deallocate** with the appropriate parameters



TWO APPS COMMUNICATING IN DISTINCT SYSTEMS

Intro

RINA

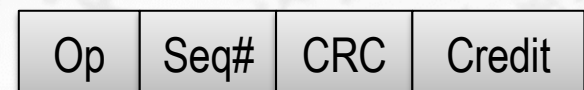
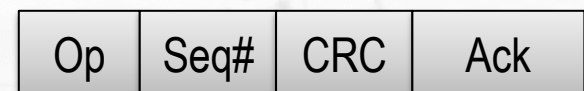
Example

PRISTINE

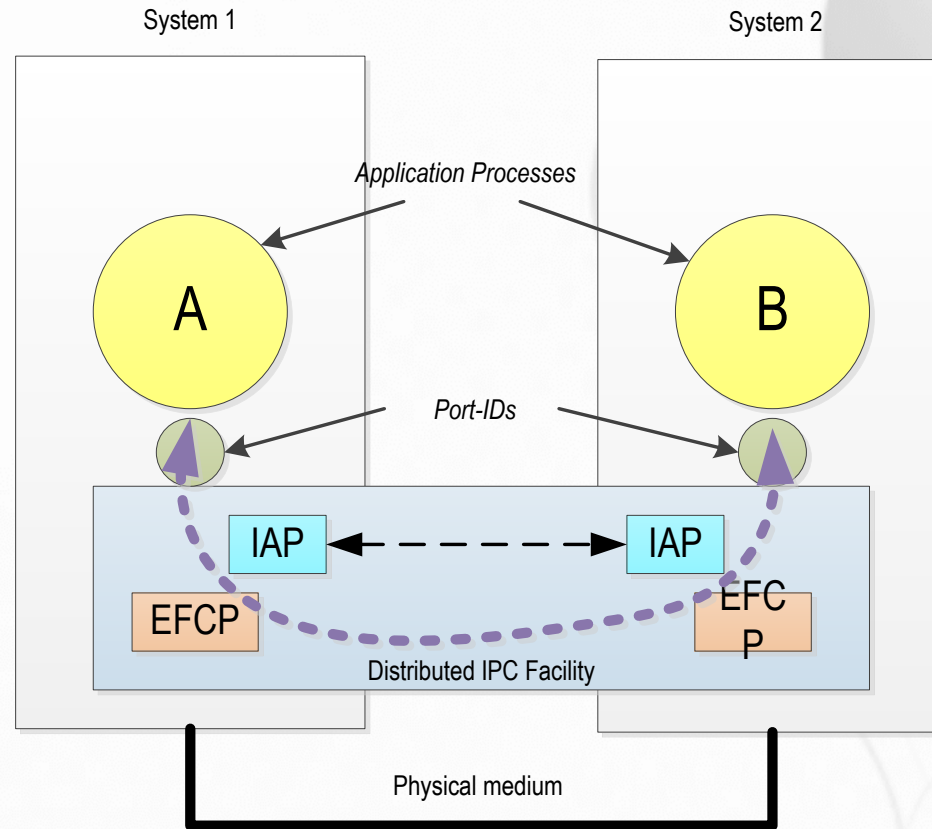
- Management of name space is no longer under the control of a single system
 - Each system no longer knows all available applications
 - Local Access Control can no longer be relied on to provide adequate authorization and authentication
- Need a protocol to carry application names and access control information - **IPC Access Protocol (IAP)**
 - Simple request-response protocol



- We need some kind of **Error and Flow Control Protocol (EFCP)**
 - Bad things can happen to messages in transit
 - Protection against lost or corrupted messages
 - Receiver must be able to tell sender, it is going too fast



TWO APPS COMMUNICATING IN DISTINCT SYSTEMS



- 1) **A** invokes IPC to create a channel to **B**; $a = \text{Allocate}(\mathbf{B}, \text{QoS})$;
- 2) IPC determines whether it has the resources to honor the request.
- 3) Send IAP Request to access **B**, creating an EFCP connection and determines if **A** has access to **B**.
- 4) IPC may cause **B** to be instantiated. **B** is notified of the IPC request from **A** and given a port-id, \mathbf{b} .
- 5) If **B** responds positively, and IPC notifies **A** with a different port-id, \mathbf{a} .
- 6) Thru n) Then using system calls **A** may send PDUs to **B** by calling **Write(a, buf)**, which **B** receives by invoking **Read(b, read_buffer)** over the EFCP connection
- 7) When they are done one or both invoke Deallocate with the appropriate parameters



TWO APPS COMMUNICATING IN DISTINCT SYSTEMS

- ◆ An **application name space** that spans both systems
 - ◆ Should be location-independent so that applications can move
- ◆ A protocol to carry **application names** and access control info
 - ◆ Applications need to know with whom they are talking
 - ◆ IPC must know what application is being requested to be able to find it
 - ◆ For now, if the requested application isn't local, it must be in the other system.
- ◆ A protocol that provides the IPC Mechanism and does Error and Flow Control
 - ◆ To maintain shared state about the communication, i.e. synchronization
 - ◆ To detect errors and ensure order
 - ◆ To provide flow control
- ◆ Resource allocation can be handled for now by either end refusing service

Intro

RINA

Example

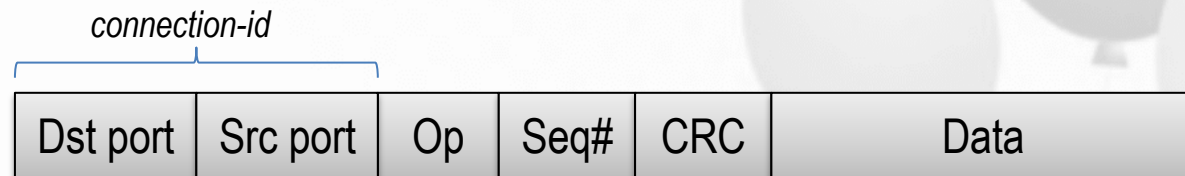
PRISTINE



SIMULTANEOUS COMMUN. BETWEEN 2 SYSTEMS

- Multiple instances of the EFCP

- ◆ Add the ability in EFCP to distinguish one flow from another



- ◆ Include the port-ids in the information sent in IAP to be used in EFCP synchronization (establishment)

- Need for application to manage multiple users of a single resource (**multiplexor** a.k.a. **mux**)

- ◆ The multiplexing application will need to be fast, its functionality should be minimized, i.e. just the scheduling of messages to send
- ◆ To provide QoS, we use the EFCP and scheduling by the Mux

- Application naming gets a bit more complicated than just multiple application-names

- ◆ Must allow multiple instances of the same process

Intro

RINA

Example

PRISTINE



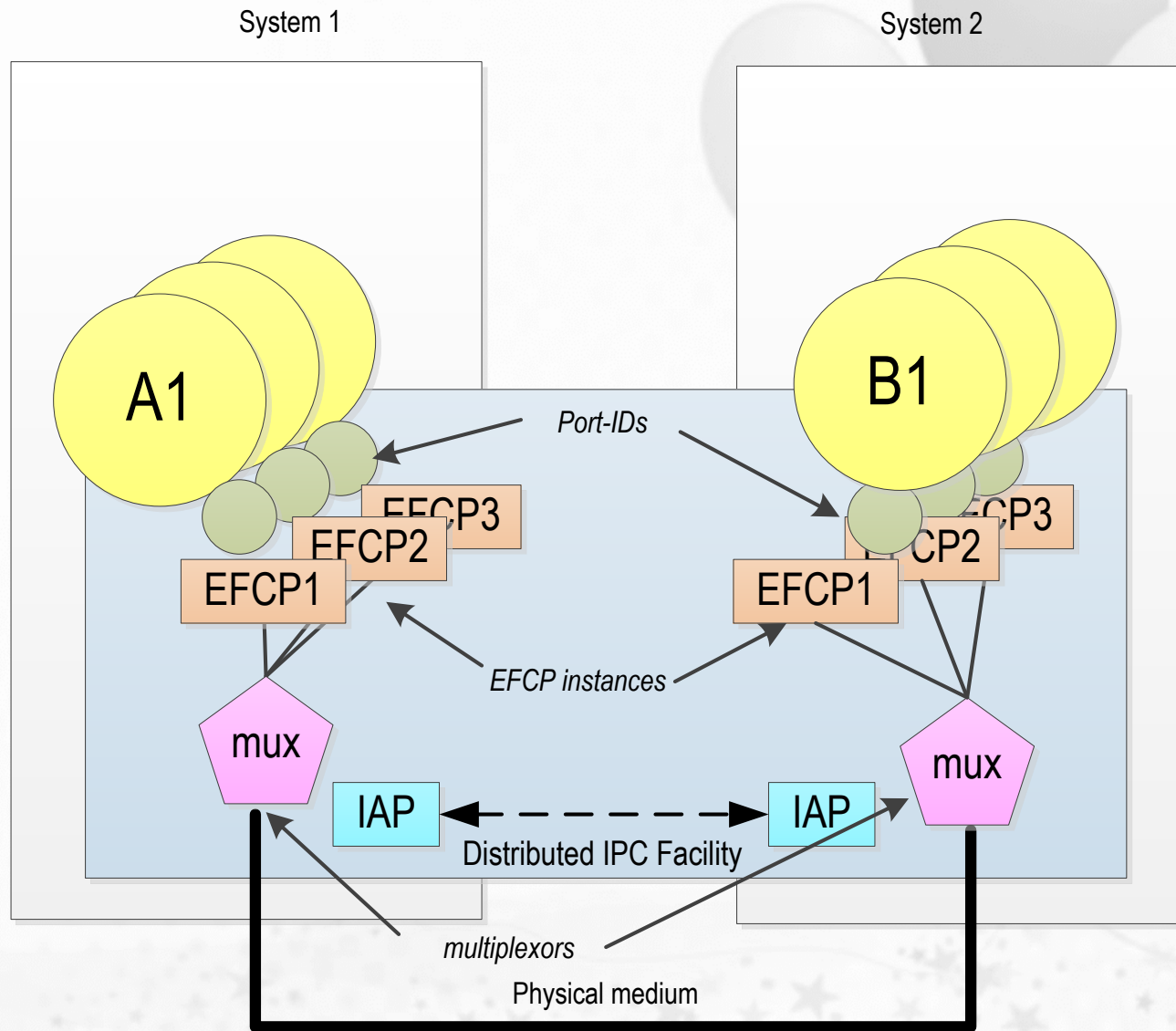
SIMULTANEOUS COMMUN. BETWEEN 2 SYSTEMS

Intro

RINA

Example

PRISTINE





COMMUNICATION WITH N SYSTEMS

- ◆ IPC can find the destination by choosing the appropriate interface
- ◆ IF enough applications THEN create a **Directory** to remember what is where, i.e. what application names are at the other end of which interfaces
- ◆ We can organize interface IPC into modules of similar elements
- ◆ Each one constitutes a Distributed IPC Facility of its own
 - ◆ As required, consists of IAP, EFCP, Mux, Directory, Per-Interface Resource Allocation
- ◆ Need an application to manage IPC processes use and moderate user requests

Intro

RINA

Example

PRISTINE

COMMUNICATION WITH N SYSTEMS

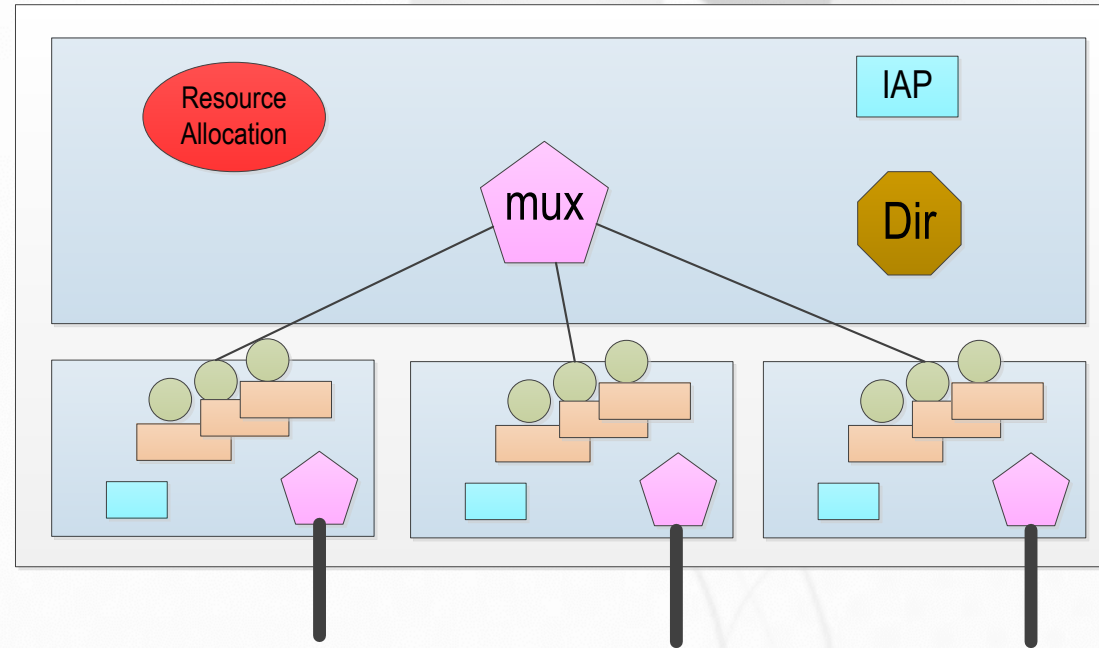
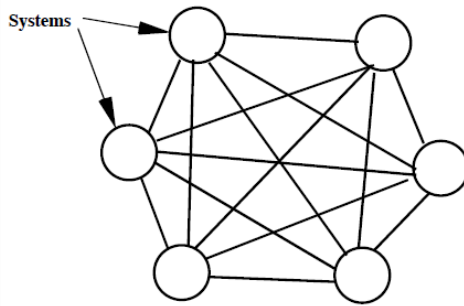
Full-mesh network

Intro

RINA

Example

PRISTINE



- ◆ *We need to dedicate some systems to IPC to reduce the number of lines required...*



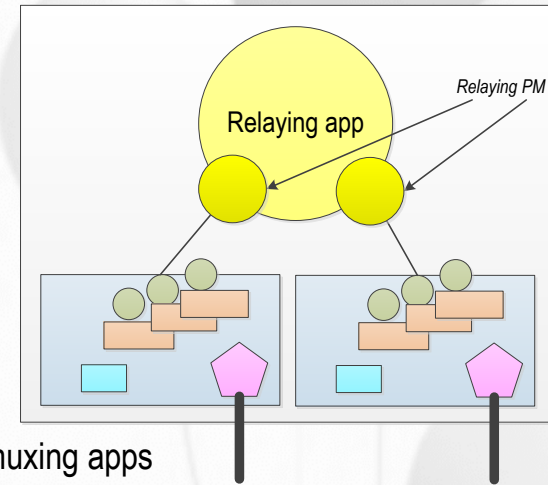
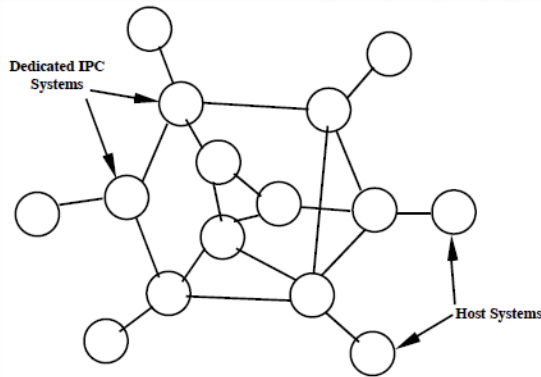
COMMUNICATION ON THE CHEAP

Intro

RINA

Example

PRISTINE



- ◆ Need systems dedicated relaying and multiplexing
- ◆ That requires some new elements
 - ◆ Globally accepted names for source and destination muxing apps
 - ◆ Relays require names for routing, they have to know where you are to determine where to go next
 - ◆ Need routing applications too, which will need to exchange information on connectivity
- ◆ Need a header on all PDUs to carry the names for relaying and multiplexing
 - ◆ Interface IPC Facilities will need one too IF they are multiple access.

Dst app	Src app	Dst port	Src port	Op	Seq#	CRC	Data
---------	---------	----------	----------	----	------	-----	------

- ◆ But relaying systems create problems too
 - ◆ Momentary congestion from time-to-time
 - ◆ Bit errors can occur
- ◆ EFCP needs to operate over the relays to ensure required QoS reliability parameters



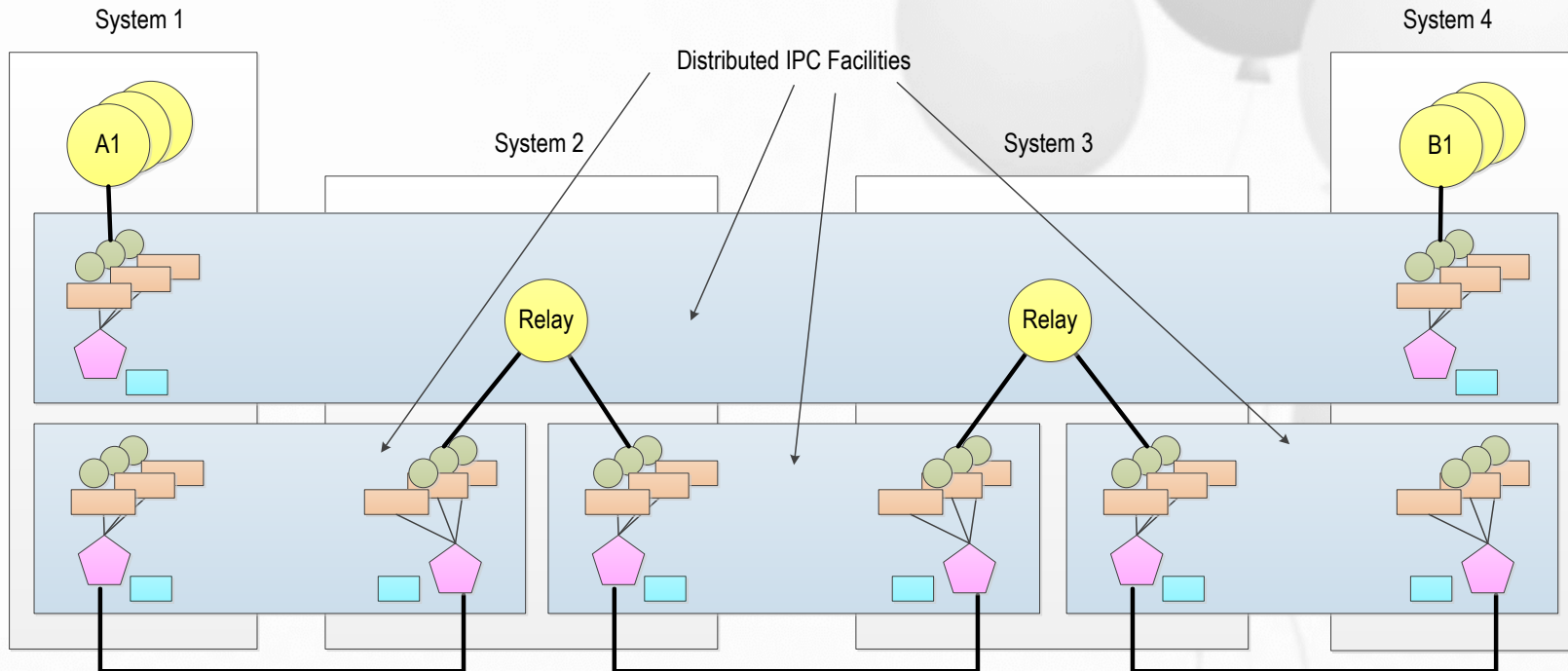
IPC MODEL

Intro

RINA

Example

PRISTINE



- ◆ A **layer** is a distributed application that manages IPC consisting of a collection of SDU protection, muxing, EFCP, and their associated routing and resource management tasks



RECURSIVE INTERNETWORK ARCHITECTURE

Intro

RINA

Example

PRISTINE

- ◆ Principles
- ◆ DIF
- ◆ Naming
- ◆ DAF
- ◆ IDD
- ◆ CDAP
- ◆ IPC components





RINA PRINCIPLES

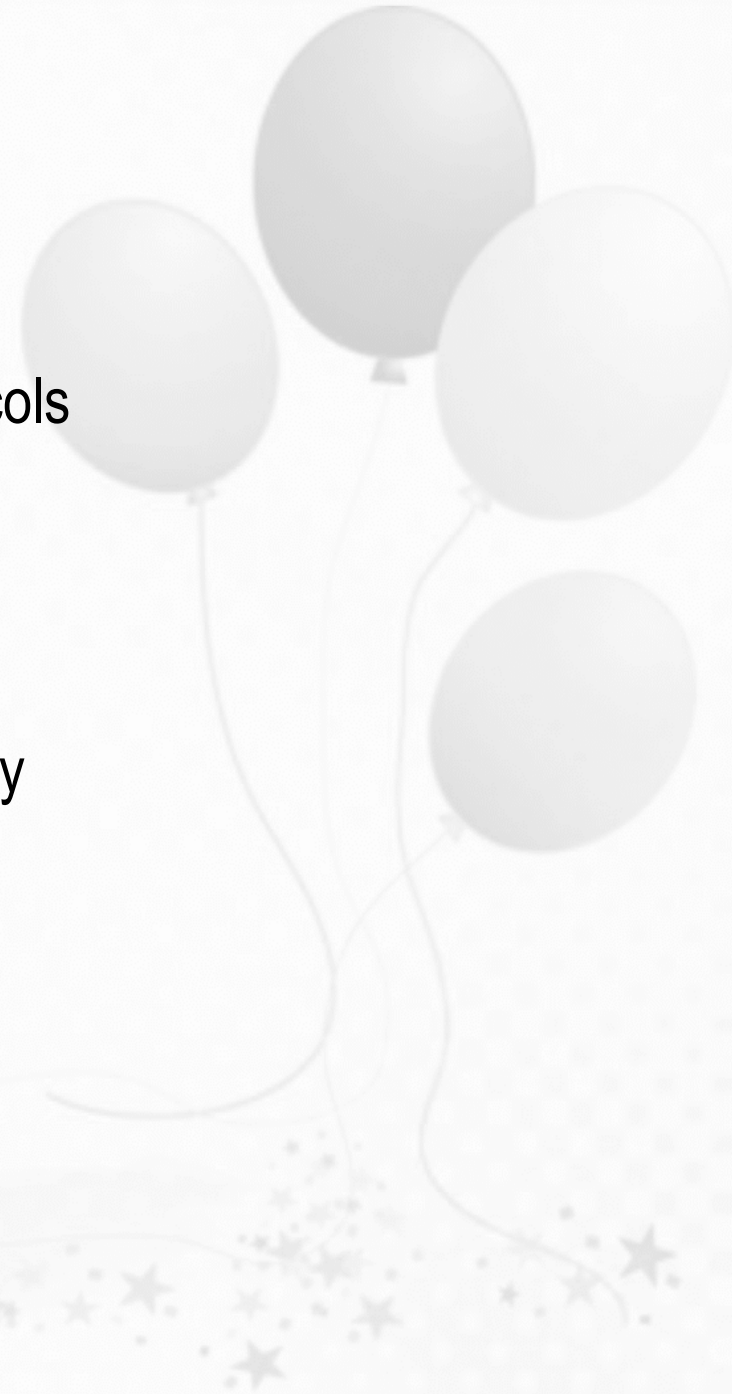
- 1) Resolving CO/CL
- 2) Nature of applications and protocols
- 3) Watson's Synchronization
- 4) Separating Mechanism and Policy

Intro

RINA

Example

PRISTINE





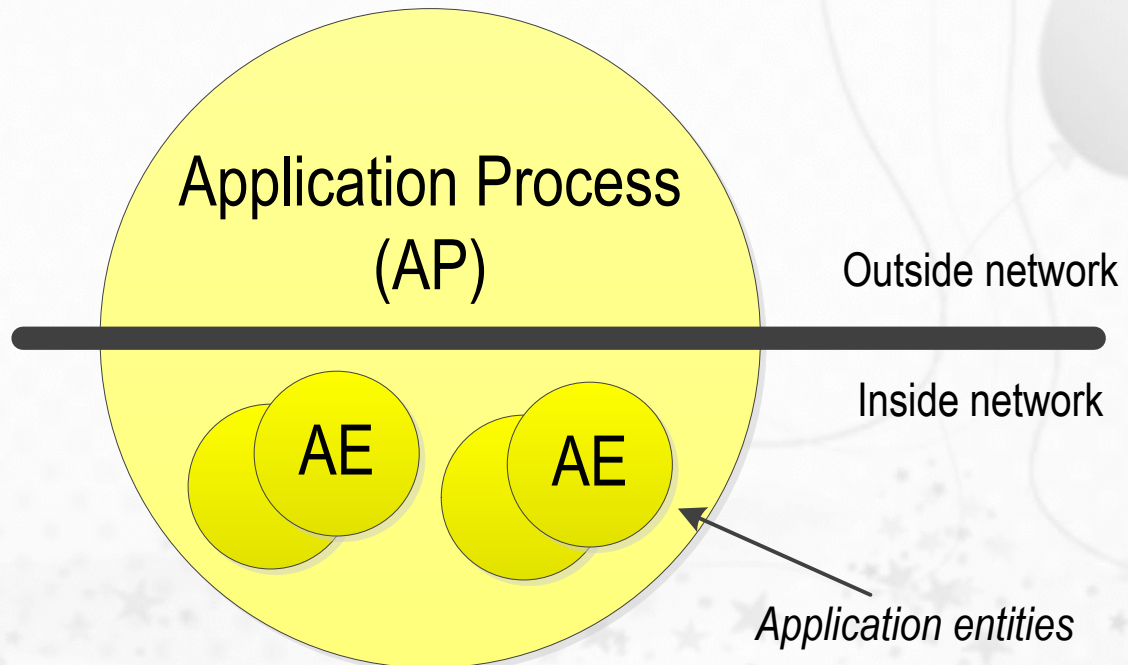
1) CO vs. CL

- ◆ Connection oriented vs Connection-less is a function of the layer and it should not be visible to applications
 - ◆ Connectionless is characterized by the maximal dissemination of state information and dynamic resource allocation
 - ◆ Connection-oriented mechanisms attempt to limit dissemination of state information and tends toward static resource allocation
- ◆ IF applications request the allocation of comm resources THEN layer determines what mechanisms and policies to use
 - ◆ Tends toward CO when traffic density is high and deterministic.
 - ◆ CL when traffic density is low and stochastic.



2) APPLICATIONS

- ◆ The **Application-Entity (AE)** is that part of the application concerned with communication, i.e. shared state with its peer.
 - ◆ An Application Process may have multiple AEs for different **application protocols** (govern by application protocol machine A-PM)
- ◆ The rest of the **Application Process (AP)** is concerned with the reason for the application in the first place.





2) APPLICATION PROTOCOL

- ◆ All application protocols are stateless, the state is in the application
- ◆ Application protocols modify state outside the protocol
- ◆ Everything is just an object outside the protocol
- ◆ There is only one application protocol – **Common Distributed Application Protocol CDAP**
- ◆ CDAP contains six primitive commands governing objects:
 - ◆ Read and Write
 - ◆ Create and Delete
 - ◆ Start and Stop
- ◆ Each set of objects yields different AE
 - ◆ One protocol, potentially shared objects, different state machines

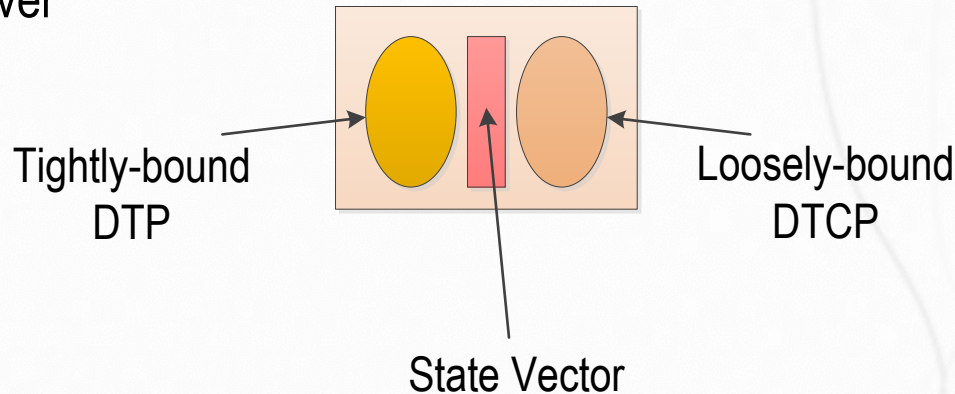


3) SYNCHRONIZATION

- ◆ Delta-T assumes
 - ◆ all connections exist all the time
 - ◆ decoupling of “port allocation” from synchronization
- ◆ Watson proves that the conditions for distributed synchronization are met IFF three timers are bounded:
 - ◆ Maximum Packet Lifetime (= MPL)
 - ◆ Maximum number of Retries
 - ◆ Maximum time before Ack (= a parameter)
- ◆ Thus no explicit state synchronization, i.e. hard state, is necessary (SYNs, FINs are unnecessary)
 - ◆ All properly designed data transfer protocols are soft-state
- ◆ After period of 2-3 MPL of no traffic, all synchronization state should be discarded, which effectively resets the connection

4) STRUCTURE OF PROTOCOLS

- ◆ Separation of mechanism and policy
- ◆ Two kinds of mechanisms
 - ◆ **Tightly-Bound**: Those that must be associated with the Transfer PDU; policy is imposed by the sender
 - ◆ **Loosely-Bound**: Those that don't have to be; policy is imposed by the receiver



- ◆ Following mechanism are loosely coupled through a **state vector**
- ◆ There is one data transfer protocol with a small number of encodings (called **DTP + DTCP** that performs **EFCP**)



Intro

RINA

Example

PRISTINE

RINA IMPLICATIONS

- ◆ DTP modify state internal to the protocol and CDAP modify state external to the protocol
- ◆ A layer provides IPC to either another layer or to a **Distributed Application** using a programming model
 - ◆ The application protocol is the “assembly language” for distributed computing
- ◆ Every protocol is soft-state with Delta-T synchronization
 - ◆ Hard state only occurs for some uses of application protocols
 - ◆ Storing in a database may be hard-state
- ◆ Separating mechanism and policy in a delta-t like protocol will yield the entire range from UDP-like to TCP-like.

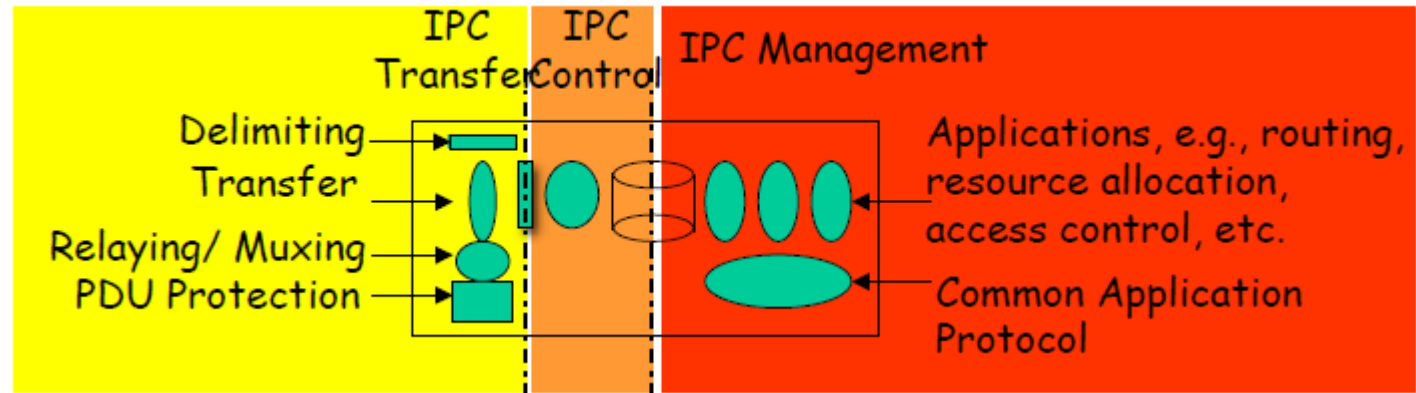
IPC PROCESS

Intro

RINA

Example

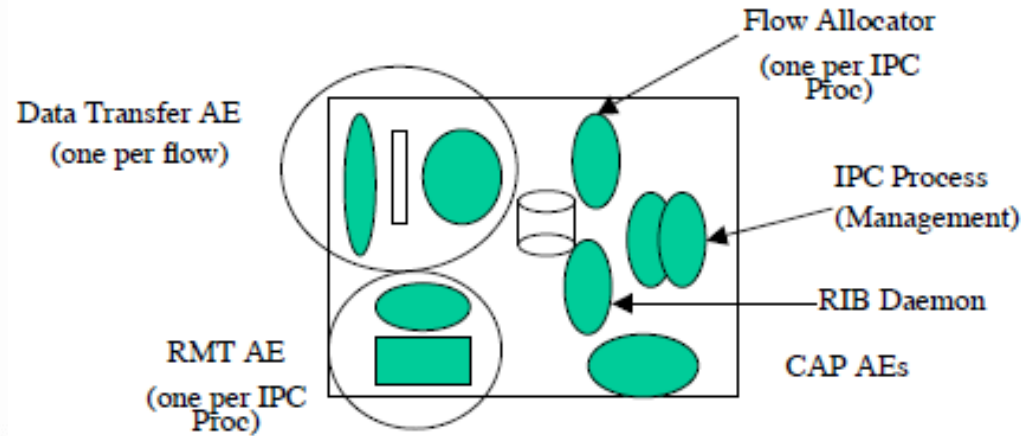
PRISTINE



- ◆ **IPC Transfer** actually moves the data (\approx IP + UDP)
- ◆ **IPC Control** (optional) for retransmission (ack) and flow control
- ◆ **IPC Layer Management** for routing, resource allocation, locating applications, access control, monitoring lower layer
- ◆ Processing at three timescales, decoupled by either a **State Vector** or a **Resource Information Base**



ELEMENTS OF IPC PROCESS



- ◆ Managing IPC means
 - ◆ Resource Allocation
 - ◆ Maintaining a Resource Information Base
 - ◆ Routing
 - ◆ Security Management
- ◆ **Flow Allocator** manages flows, finds the destination and does access control, manages the binding of connection-endpoint-ids to port-ids
- ◆ **Data Transfer AE** is the error and flow control protocol
- ◆ **Relaying/Multiplexing Task (RMT) AE** consists of the relaying and multiplexing task and SDU Protection
- ◆ **RIB Daemon** maintains the local RIB information

Intro

RINA

Example

PRISTINE



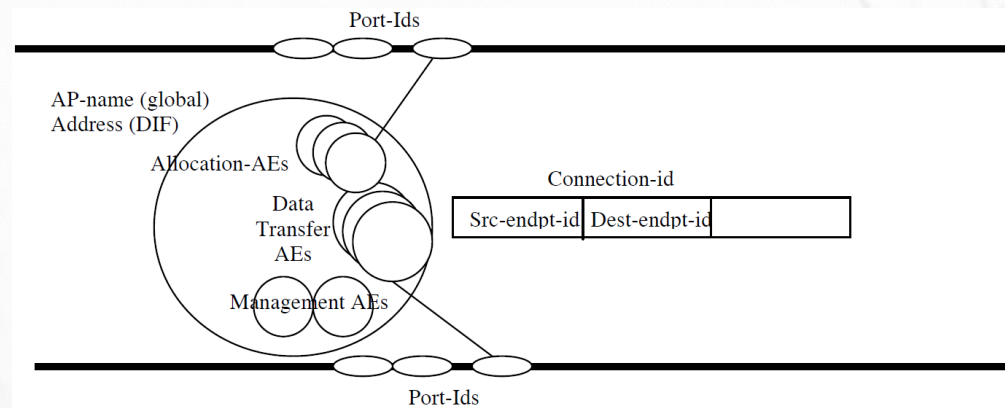
IPC PROCESS NAMING

Intro

RINA

Example
PRISTINE

- ◆ IPC process has an unambiguous **Application Process Name (APN)**, commonly called **address**
- ◆ Flow allocator AE instance id (FAI) a.k.a. **port-id**
 - ◆ port-id is the handle returned to the calling application to refer to this instance of communication, unique within its AE
- ◆ Data Transfer AE instance id a.k.a. **CEP-id**
 - ◆ connection-endpoint-id (CEP-id) identifies the shared state of one end of a flow/connection, unique within its AE
 - ◆ **connection-id** identifies flows between the same two IPC Processes, formed by concatenating CEP-ids, unique within the pair

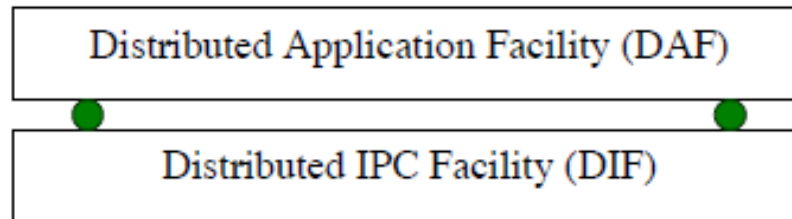


- ◆ **Distributed Application Name (DAN)** is globally unambiguous name for the set of all Application Processes in a Distributed Application, e.g. DIF



DIF AND DAF

- ◆ **DIF** a.k.a. **Distributed IPC Facility** a.k.a. **layer** is a distributed application that does IPC
 - ◆ System may be member of 0-n DIFs
 - ◆ System has IPC process for every DIF
- ◆ **DAF** a.k.a. **Distributed Application Facility** is a collection of two or more AP on one or more processing systems, which exchange information using IPC and maintain shared state
 - ◆ Each member of DAF is **Distributed Application Process (DAP)**
 - ◆ DAF operates usually over single DIF to obtain IPC services



- ◆ Every DAF has two whatevercast names
 - ◆ first that returns all members of DAF
 - ◆ second that resolves to a single member (nearest or designated member)



NAMING SUMMARY

Intro

RINA

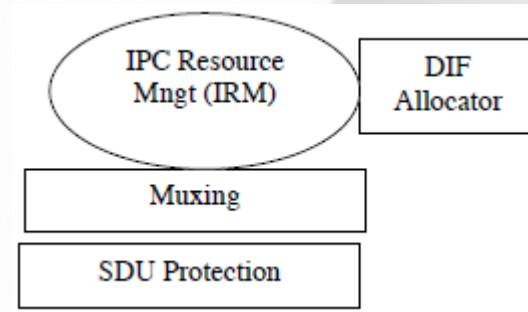
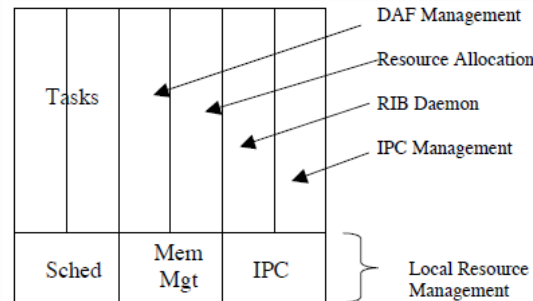
Example

PRISTINE

Common Term	Scope	Application Term
Application Process Name (APN)	Global (unambiguos)	Application Process Name
Address	Layer (unambiguous)	Synonym for IPC Process' Application Process Name
Port-id	Allocation AE (unique)	Allocation AE-Instance-Identifier
Connection-endpoint-identifier (CEP-id)	Data Transfer AE (unique)	Data Transfer AE Instance-Identifier
Connection-id	Src/Dest Data Transfer AE (unique)	Concatentation of data-transfer-AE-instance-identifiers
DIF Management Updates	IPC Process (unambiguous)	AE-identifier
Distributed Application Name (DAN)		
Distributed Application Process Name (DAP-Name)		



ELEMENTS OF DAP



Intro

RINA

Example

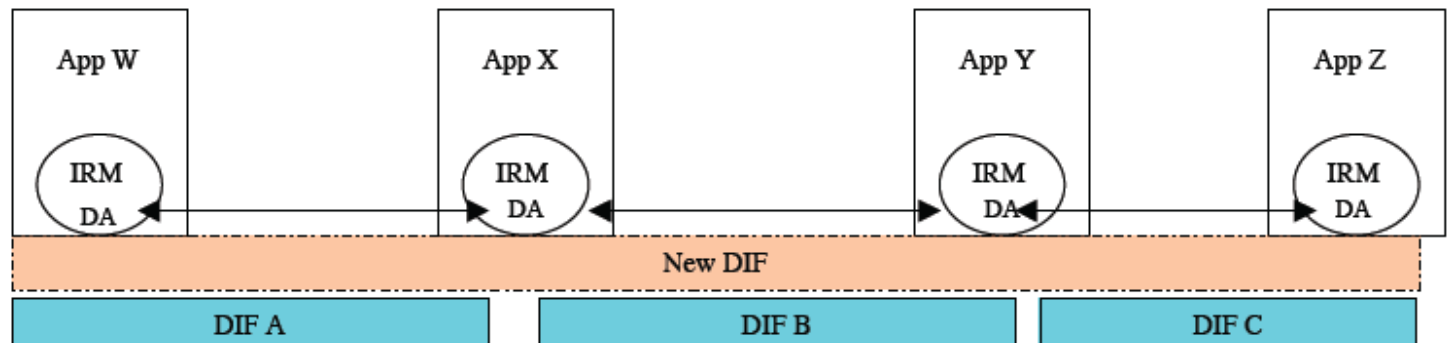
PRISTINE

◆ IPC Resource Management (IRM)

- ◆ The primary role of the IRM is managing the use of supporting DIFs and in some cases, participate in creating new DIFs
- ◆ When an App submits an Allocate Request to the IRM with an APN, the IRM determines if the application is reachable on any of the DIFs
 - ◆ IF true THEN the DIF is selected and the Allocate is processed normally
 - ◆ IF false THEN query its peer IRMs

◆ DIF-Allocator (DA)

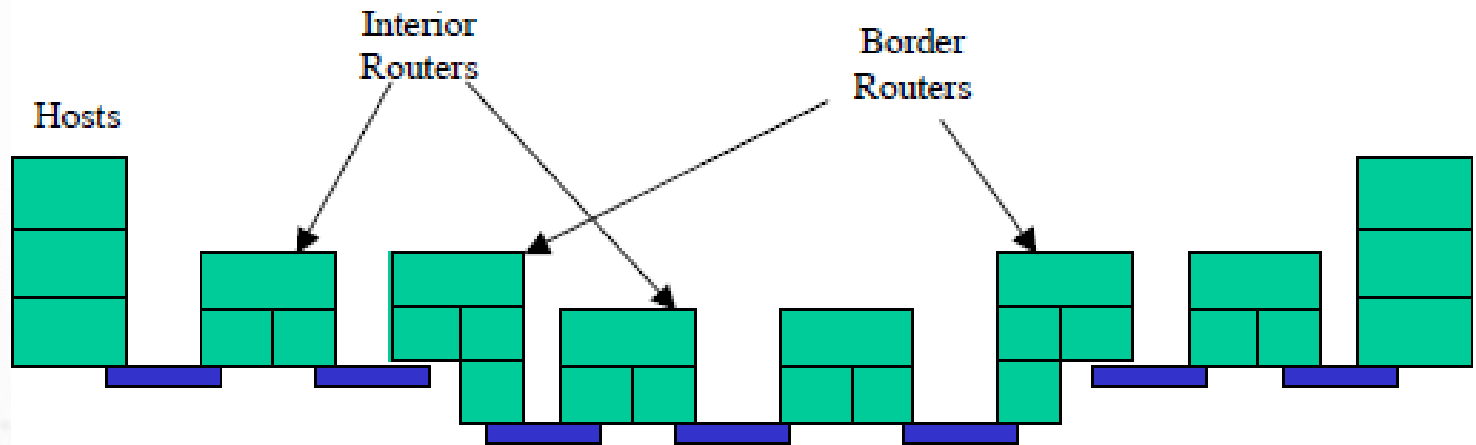
- ◆ ① to find the DIF that supports the requested DIFs or DAFs, that is not available to this IRM, and ② to make available a DIF by either joining existing DIFs or creating new DIFs





RINA NODES

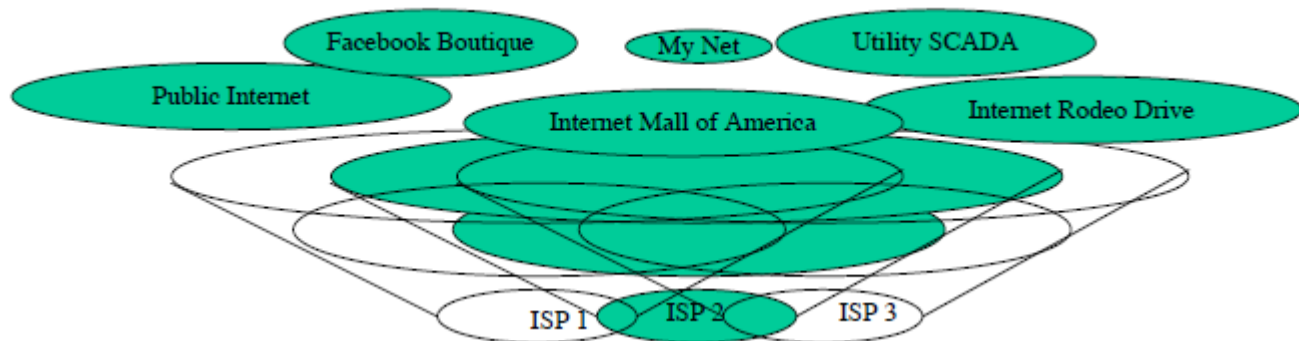
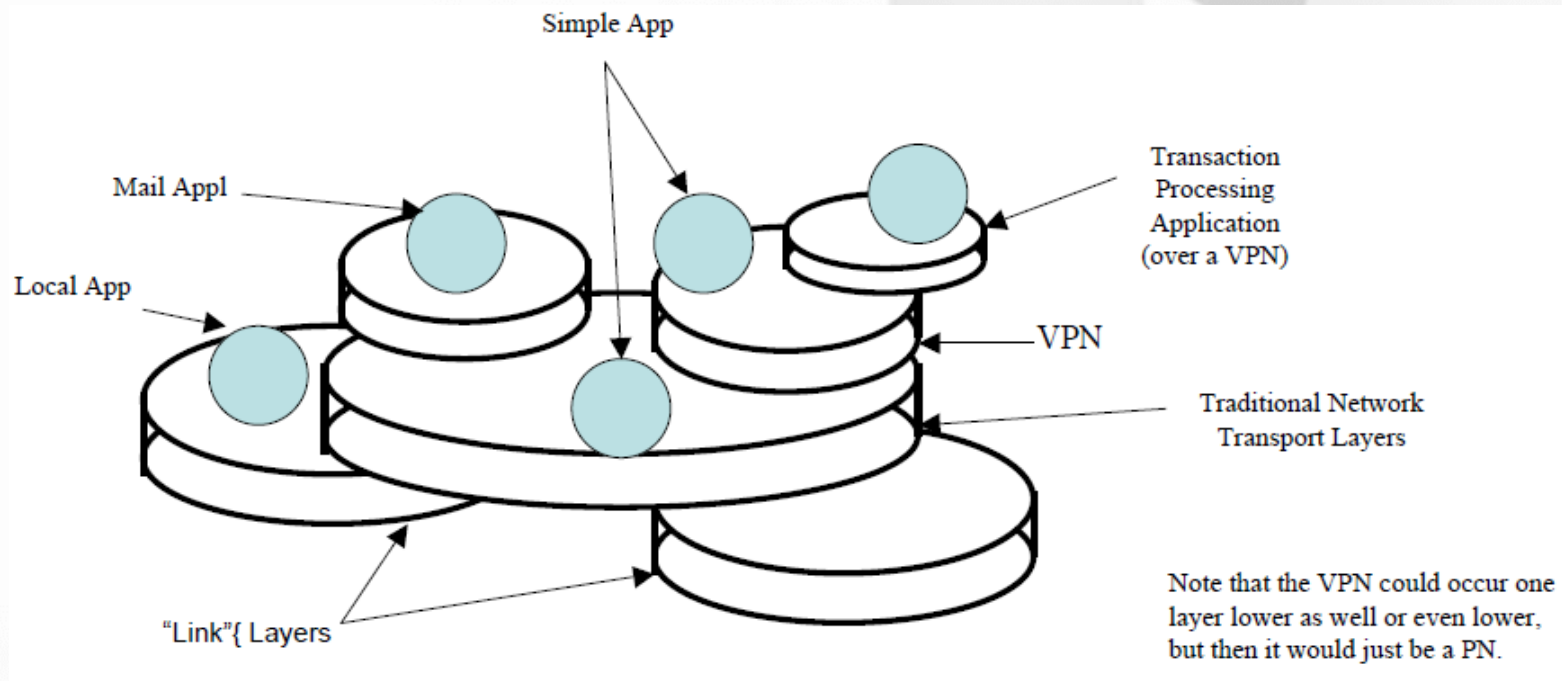
- ◆ **Hosts** running applications
 - ◆ members of DAF
- ◆ **Interior routers**
 - ◆ dedicated IPC processes providing relaying between DIF members
- ◆ **Border routers**
 - ◆ providing relaying between different DIFs





DIF'S INTERNET

- ◆ User Applications use whatever layer has sufficient scope to communicate with their apposite



Intro

RINA

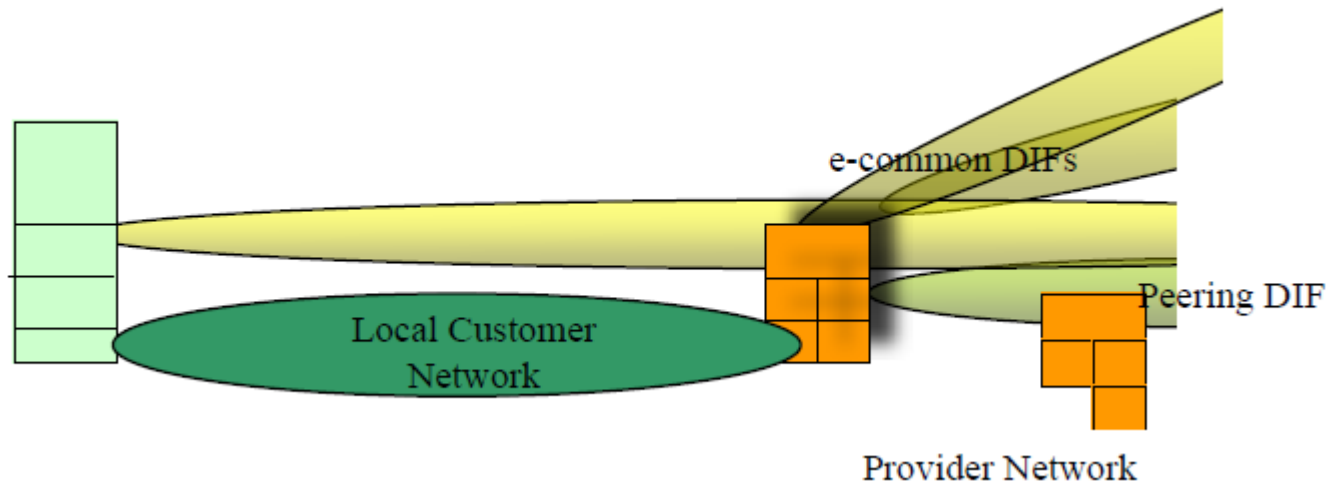
Example

PRISTINE



HOST PERSPECTIVE ON DIF'S INTERNET

- ◆ A customer network has a border router that makes several e-mails available
- ◆ A choice can be made who joins DIF whether
 - ◆ the entire local network
 - ◆ a single host
 - ◆ a single application



Intro

RINA

Example

PRISTINE

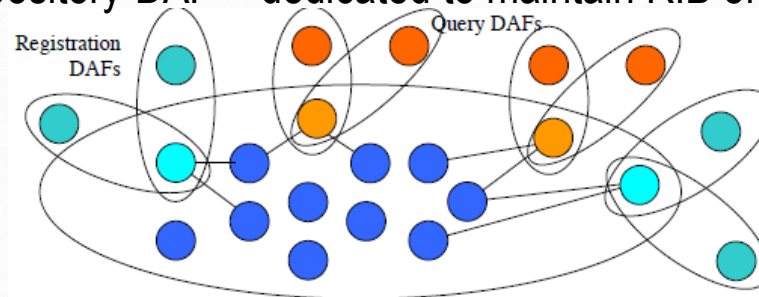


NSM AND DA

◆ Name Space Management (NSM)

◆ Heterogenous DAF with four kinds of DAPs involved

- 1) Query-DAP – user of NSM
- 2) Registration-DAP – (de)registerer of name in NSM
- 3) Server-DAP – service requests from 1) and 2)
- 4) Repository-DAP – dedicated to maintain RIB of NSM

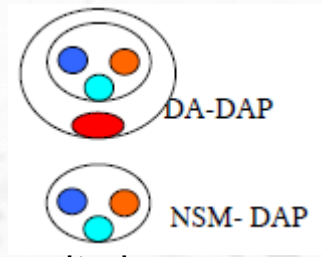


◆ DIF Allocator (DA)

◆ DA may be stand-alone DAF which uses NSM-DAP

◆ Two sets of forwarding tables

- ◆ FT-s for a name resolution requests to forward to next repositories
- ◆ FT-n is required to forward PDUs through one or more DA-DAPs when the destination DA-DAP is not one hop away



Intro

RINA

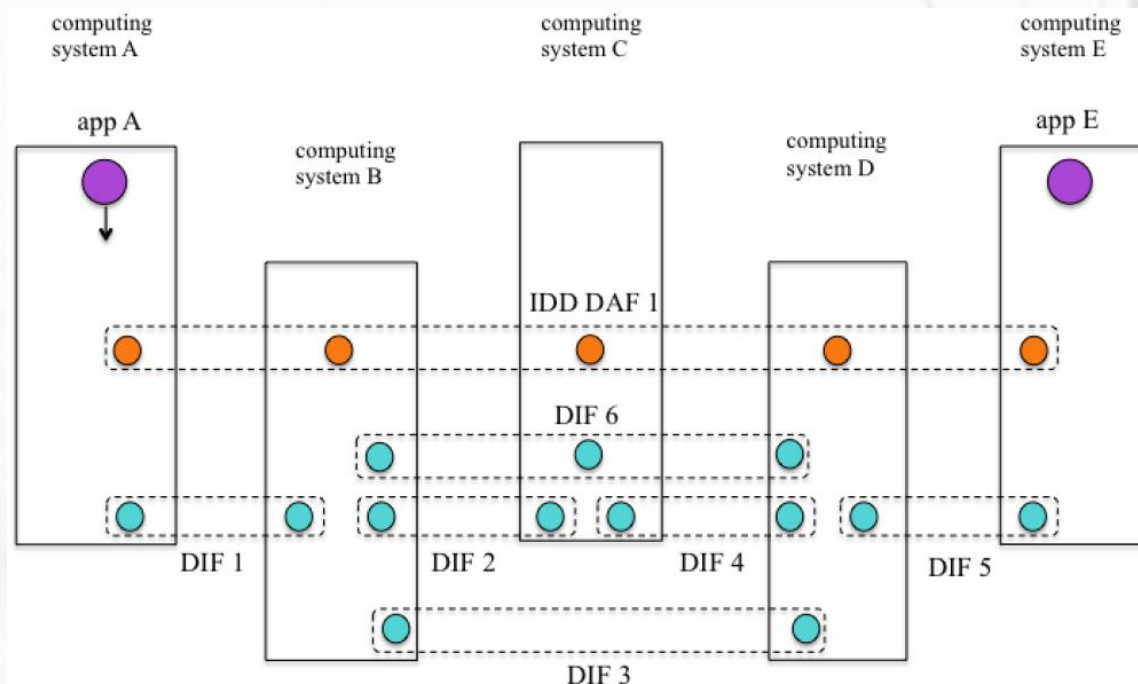
Example

PRISTINE



CHOOSING A LAYER

- Hosts do not have to see all of the wires or all the “Nets”
- Need for directory service
 - database that maintains mappings between two name spaces
 - mappings of application names to list of supporting layers (DIFs)
- InterDIF Directory (IDD) = DIF-Allocator of IRM**
 - Responsible for two main distinct functions
 - a) Discovery of the application
 - b) Creation of the supporting DIF



Intro

RINA

Example

PRISTINE



IDD STRUCTURE

Intro

RINA

Example

PRISTINE

FT-s of DA

FT-n of DA

Naming Information

IDD Application Process Name
synonyms (optional)

Search Table

Application Process Name	List of Peer IDD's Application Process Names
--------------------------	--

Neighbor Table

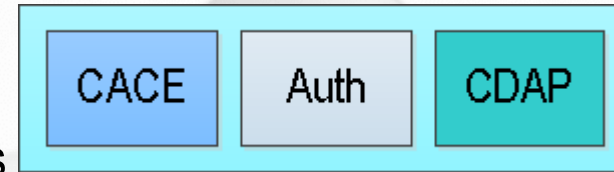
Peer IDD Application Process Name	List of Peers IDD's Application Process Names
-----------------------------------	---

Directory

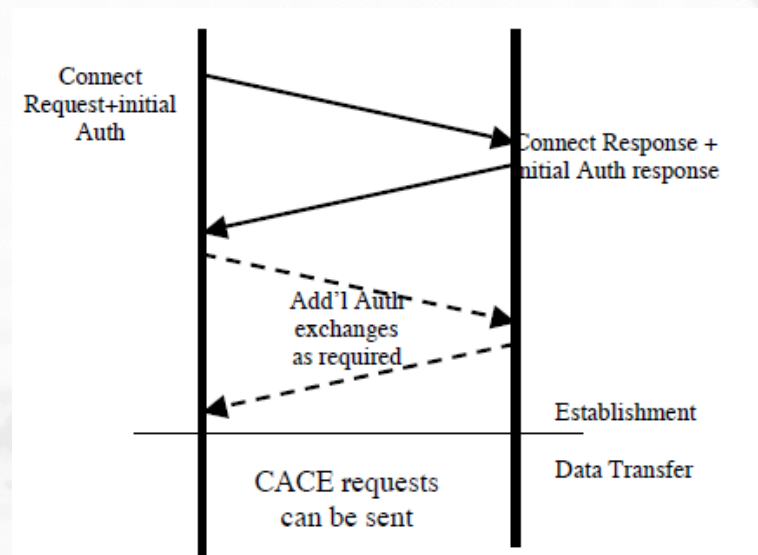
Application Process { Name, Access Control Information }
List of supporting DIFs { Name, Access Control Information, supported QoS }



CDAP



- ◆ The Basic Form consists of three components
 - ◆ the common application connection establishment (**CACE**), based on ACSE
 - ◆ the authentication module (**Auth**)
 - ◆ **CDAP**, based on CMIP
- ◆ Three primary objectives
 - 1) Joining a DIF (Establishing a management connection between an new IPC Process and the members of the DIF, authenticating it, initializing it, and assigning it a forwarding-id)
 - 2) Maintaining Resource Information Base
 - 3) Flow Allocation
- ◆ DAF asks DIF to allocate flow, then sets up application connection



Intro

RINA

Example

PRISTINE



PHASES OF COMMUNICATION

◆ *Application protocols*

◆ **Initialization**

- ◆ Establishment of communication between APs

◆ **Operation**

- ◆ Operations are performed on objects external to the protocol

◆ *DTP protocols*

◆ **Enrollment**

- ◆ Operations to create sufficient state within the network to allow an instance of communication to be created

◆ **Allocation a.k.a. Establishment**

- ◆ Operations required to allocate an instance of communication creating sufficient shared state among instances to support the functions of the data transfer phase

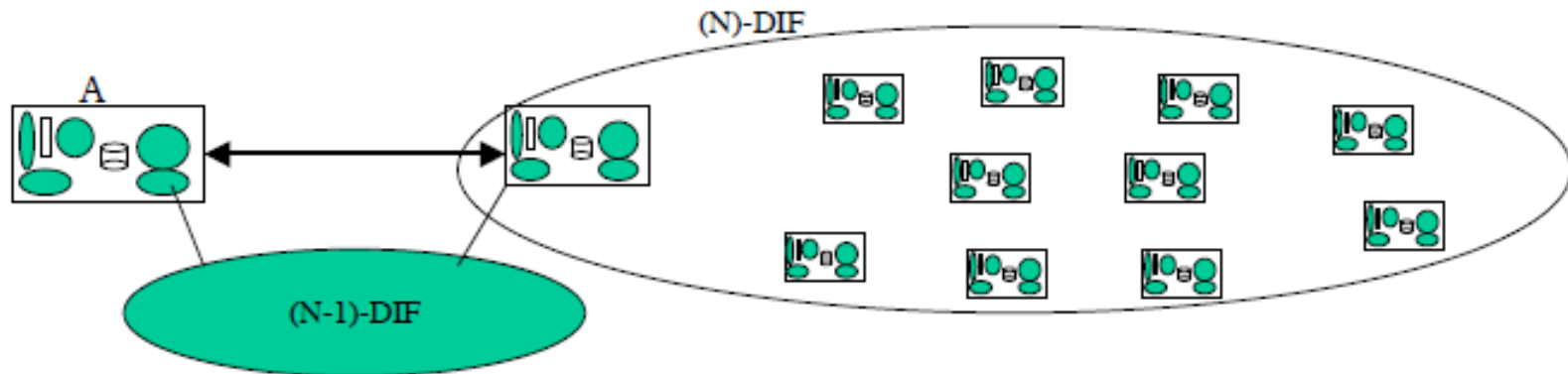
◆ **Data Transfer**

- ◆ Operations to provide the actual transfer of data and functions which support it



DIF ENROLLMENT

- Joining a DIF is application establishing communication (for management)



- Steps involved

- 1) Authenticating that **A** is a valid member of the (N)-DIF
- 2) Initializing it with the current information on the DIF
- 3) Assigning it a synonym to facilitate finding IPC Processes in the DIF, i.e. an address

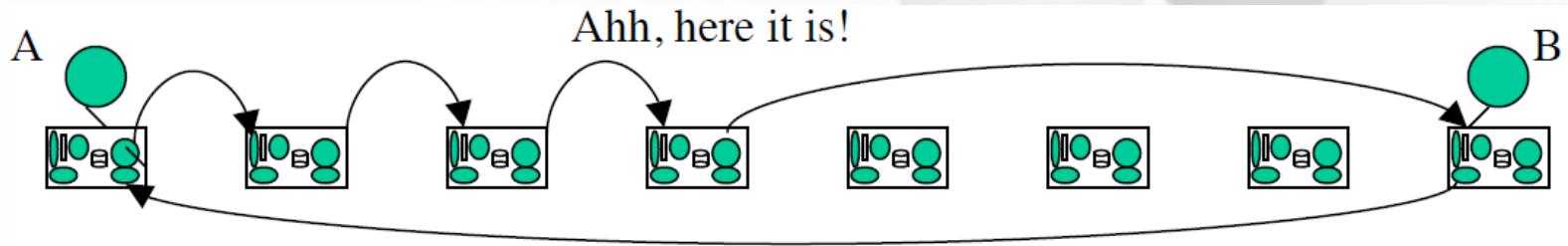
DIF ESTABLISHMENT

Intro

RINA

Example

PRISTINE



Steps involved

- 1) **A** asks IPC to allocate communication resources to **B**
- 2) Determine that **B** is not local to **A**, then use search rules to find **B**
- 3) Keep looking until we find an entry for it
- 4) Then go see if it is really there and whether we have access.
- 5) Then tell **A** the result

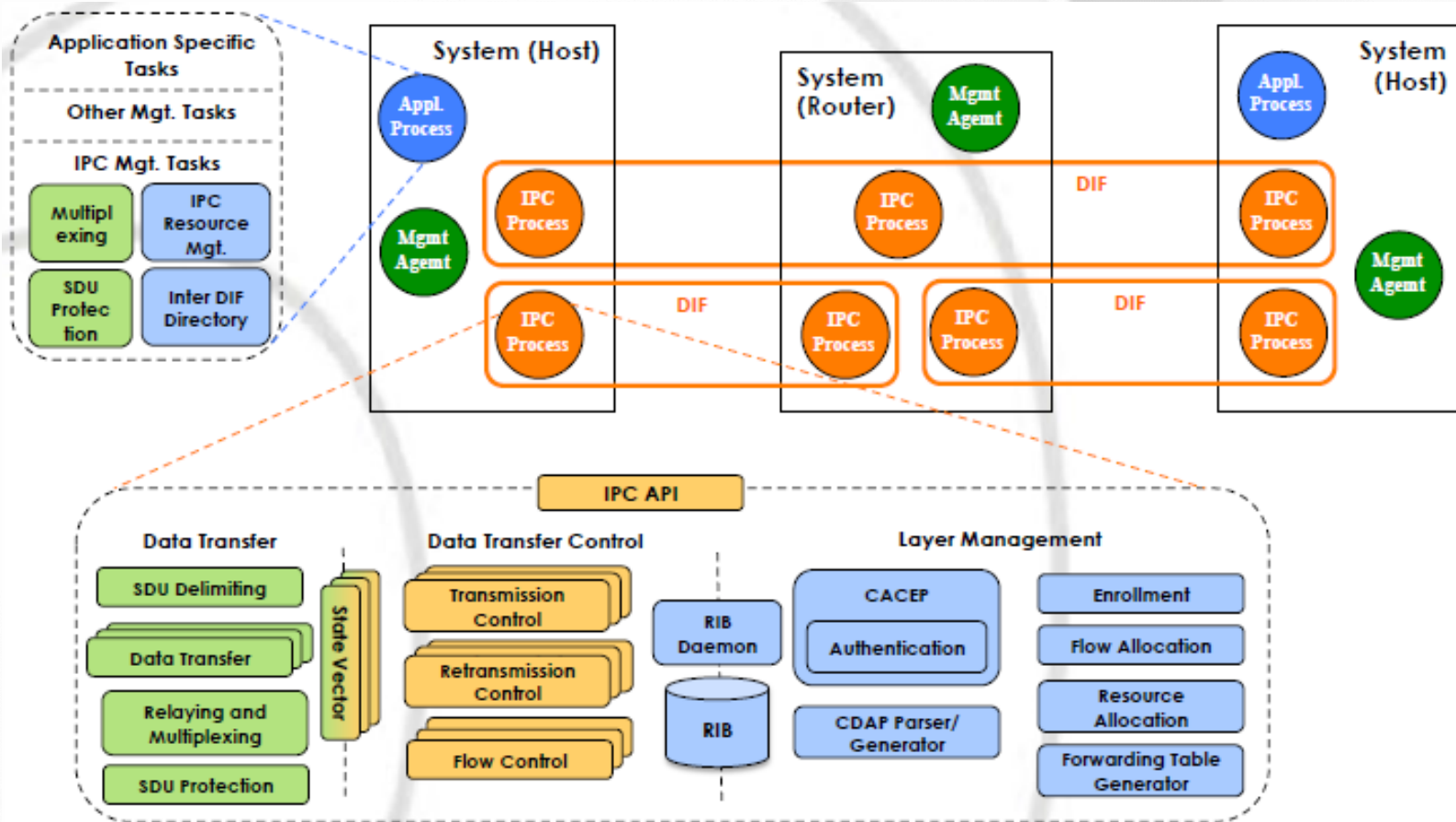


ELEMENTS OF IPC PROCESS REVISITED

Intro

RINA

Example
PRISTINE





DELIMITING

- ◆ IPC may fragment or combine SDUs but will deliver same SDUs to the destination
 - ◆ External and Internal delimiting are possible
 - ◆ For a flow that appears to be streaming, the entire flow is a single SDU and early delivery is allowed
- ◆ This module is entirely policy

Intro

RINA

Example

PRISTINE



SDU PROTECTION

- ◆ SDU Protection is also entirely policy and may include
 - ◆ Data Corruption Protection (CRCs or FECs)
 - ◆ Time to Live
 - ◆ Integrity and Confidentiality (encryption)
 - ◆ Compression

Intro

RINA

Example

PRISTINE





EFCP = DTP + DTCP / DATA TRANSFER + REXMSN CONTROL

- ◆ Based on Delta-t with mechanism and policy separated
- ◆ Naturally cleaves into DTP and DTCP
 - ◆ Data Transfer consists of tightly bound mechanisms
 - ◆ Roughly similar to IP + UDP
 - ◆ Data Transfer Control, if present, consists of loosely bound mechanisms
 - ◆ Flow control and retransmission (ack) control
- ◆ One instance per flow
- ◆ Policies driven by the QoS parameters

DTP Protocol Data Unit (PDU)

PCI

User data

Information required to perform data transfer mechanisms tightly bound to the transported SDUs: addressing, sequencing, ...

Partial, one or more SDUs from the layer above

DTCP Protocol Data Unit (PDU)

PCI

Information required to perform data transfer mechanisms loosely bound to the transported SDUs: transmission control, retransmission control, flow control

Intro

RINA

Example

PRISTINE



Intro

RINA

Example

PRISTINE

DTP PDU

- ◆ Version: 8 Bit
- ◆ Destination-Address: Addr-Length
- ◆ Source-Address: Addr-Length
- ◆ Flow-id: Struct
 - ◆ QoS-id: 8 Bit
 - ◆ Destination-CEP-id: Port-id-Length
 - ◆ Source-CEP-id: Port-id-length
- ◆ PDUType: 8 bits
- ◆ Flags: 8 bits
- ◆ PDU-Length: Length-Length
- ◆ SequenceNumber: SequenceNumber-Length
- ◆ Sequence User-Data{DelimitedSDU* | SDUFrag}
- ◆ User data





DTCP PDU

Intro

RINA

Example

PRISTINE

Common Control PDU

Version: 8 Bits

Destination-Address: Addr-Length

Source-Address: Addr-Length

Flow-id: Struct

QoS-id: 8 bits

Destination-CEP-id: CEP-id-Length

Source-CEP-id: CEP-id-length

PDUType: 8 bits

Flags: 8 bits

PDU-Length: LengthLength

SequenceNumber: SequenceNumberLength

Ack/Flow ControlPDU

Common Control PDU

PDU TYPE = X'800C' Ack only

PDU TYPE = X'800D' Ack and Flow Control

PDU TYPE = X'8009' Flow Control only

Ack: Integer(SeqNbrLength)

RightWindowEdge: SequenceNbrLength

NewRate: RateLen

TimeUnit: TimeLen

Selective Ack PDU

Common Control PDU

PDU TYPE = X'8004'

Ack/Nack: Integer(SeqNbrLength)

Ack/Nack List Length: Integer(8)

Ack/Nack List: Sequence(StartingNbr Integer (SeqNbrLength), Ending Integer(SeqNbrLength))

Forced NackPDU

Common Control PDU

PDU TYPE = X'8006'

Ack/Nack: Integer(SeqNbrLength)

Ack/Nack List Length: Integer(8)

Ack/Nack List: Sequence(StartingNbr Integer(SeqNbrLength),

Ending Integer(SeqNbrLength))



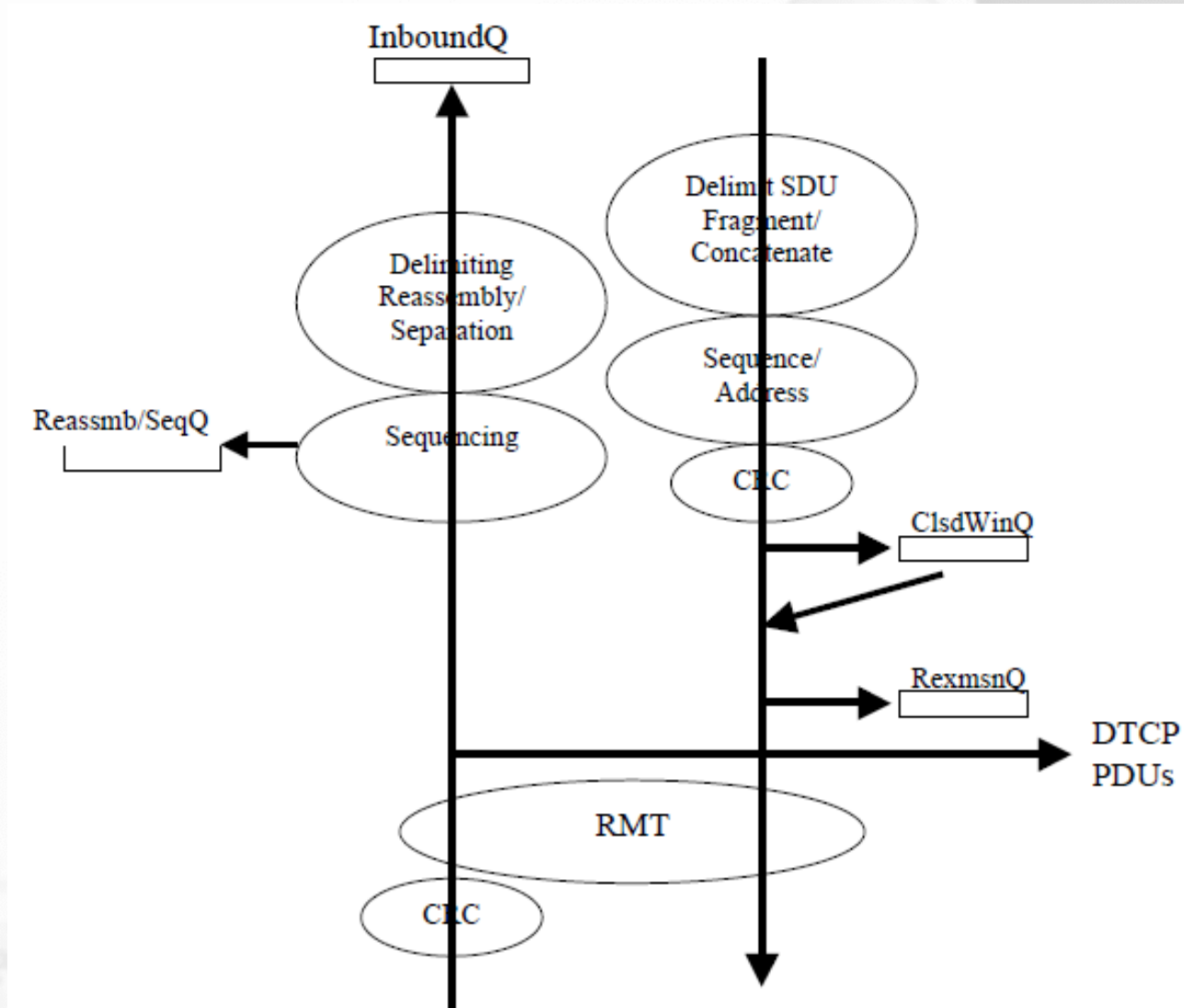
DATA TRANSFER PATH

Intro

RINA

Example

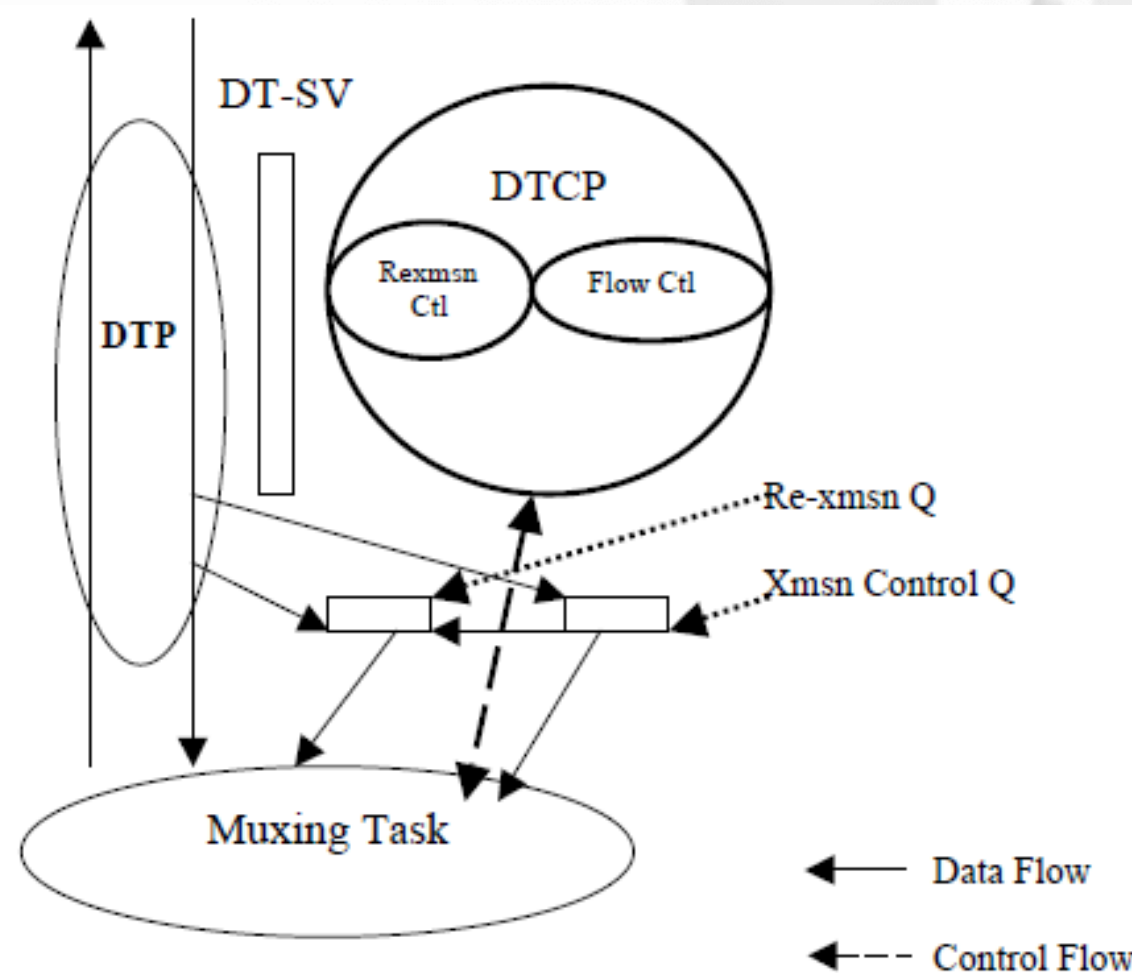
PRISTINE





DATA TRANSFER CONTROL PATH

- Control stays out of the main data flow
- This module will not exist for flows that don't need it



Intro

RINA

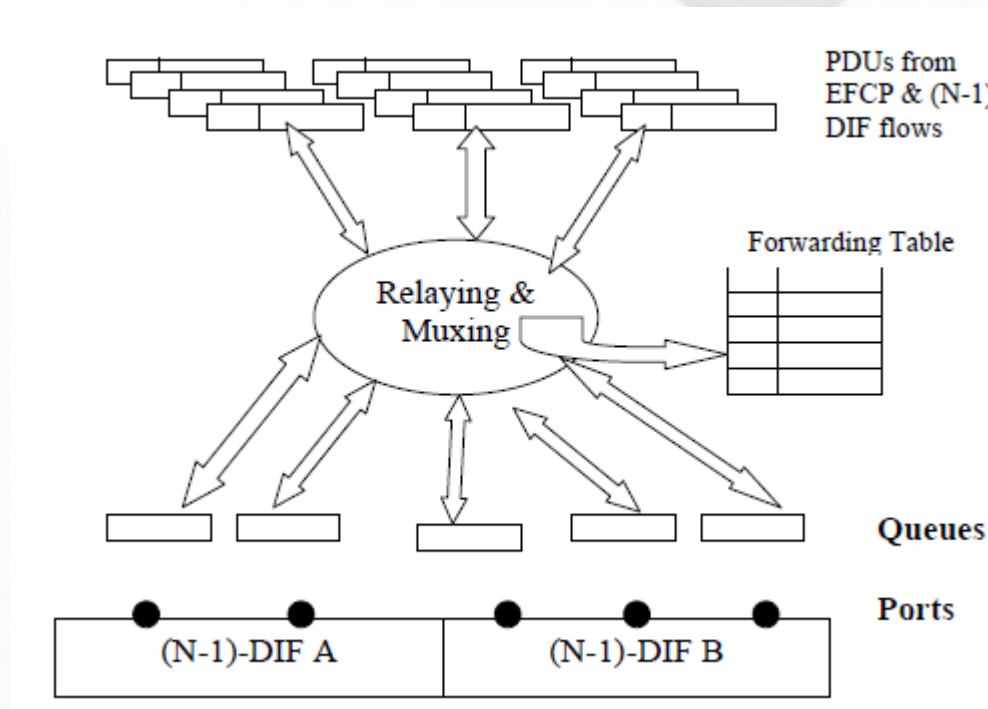
Example

PRISTINE



RELAYING AND MULTIPLEXING TASK (RMT)

- ◆ RMT multiplexes the PDUs coming from the N-flows originated at the IPC Process into one or more N-1 flows (in its multiplexing role)
- ◆ RMT forwards the incoming PDUs to another IPC Process through the adequate N-1 flow based on the PDU's destination address (in its relaying role)



- ◆ Queues at the top are at least one per QoS Class
- ◆ Queues at the bottom are created by the Resource Allocator and may be related to the number of QoS Classes provided by the lower DIF



RESOURCE ALLOCATOR (RA)

- ◆ Component that decides how the resources in the IPC Proc. are allocated
 - ◆ Dimensioning of the queues, creation/suspension/deletion of RMT queues, creation/deletion of N-1 flows, and others
- ◆ The RA has a set of **meters** and **dials** that it can manipulate
 - ◆ The meter fall in 3 categories:
 - ◆ Traffic characteristics from the user of the DIF
 - ◆ Traffic characteristics of incoming and outgoing flows
 - ◆ Information from other members of the DIF
 - ◆ The Dials
 - ◆ Creation/Deletion of QoS Classes
 - ◆ Data Transfer QoS Sets
 - ◆ Modifying Data Transfer Policy Parameters
 - ◆ Creation/Deletion of RMT Queues
 - ◆ Modify RMT Queue Servicing
 - ◆ Creation/Deletion of (N-1)-flows
 - ◆ Assignment of RMT Queues to (N-1)-flows
 - ◆ Forwarding Table Generator Output

Intro

RINA

Example

PRISTINE



FLOW ALLOCATOR (FA)

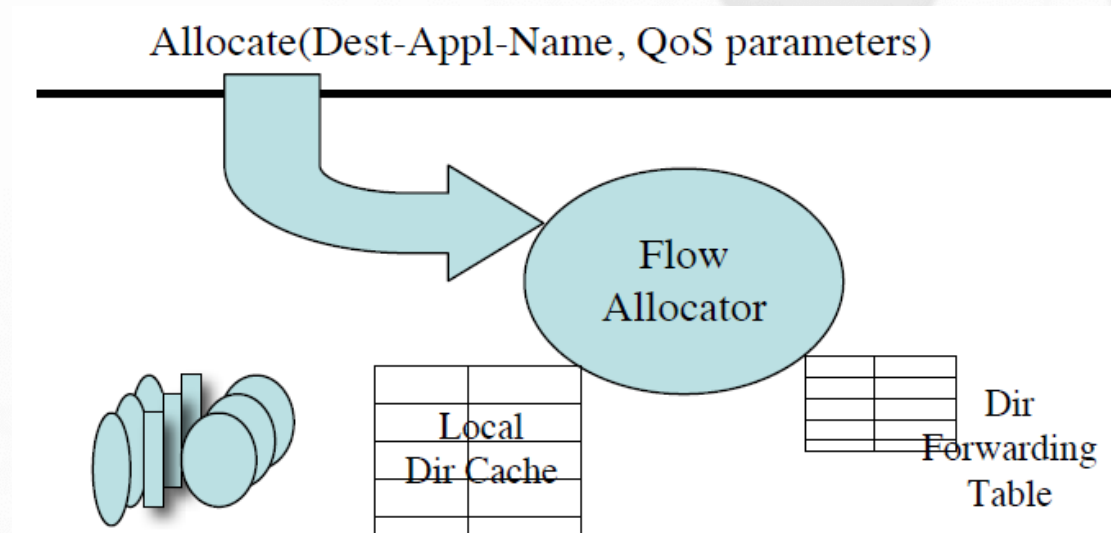
- ◆ When AP generates an Allocate request, the FA creates a flow allocator instance (FAI) to manage each new flow
- ◆ The FAI is responsible for managing the flow and deallocating the ports
 - ◆ DTP/DTCP instances are deleted automatically after 2MPL with no traffic

Intro

RINA

Example

PRISTINE



- ◆ When it is given an Allocate Request it does the following:
 - 1) It inspects the **Allocate request** and maps the parameters to the appropriate QoS Class and the associated policy set.
 - 2) It instantiates a DTP (and DTCP if necessary) for this flow.
 - 3) Checks its local directory cache for the destination application name.
 - ◆ IF found THEN it sends a **Create Flow request** to destination address;
 - ◆ ELSE consults the **Dir Forwarding Table** and sends the **Create Flow request** to the address noted there.
 - 4) When it receives an **Create Flow Response** it executes an **Allocate**
 - ◆ Response API call and modifies the state of the DTP as necessary.



RIB DEAMON

- ◆ The RIB is a logical representation of the information known by a an AP
 - ◆ Doesn't need to be a database, the information may be stored in the different AP components
- ◆ The RIB Daemon provides an API to perform operations on the RIB (both objects in the local RIB and objects in remote application processes' RIBs)
- ◆ Three facilities
 - ◆ routing update
 - ◆ event management
 - ◆ management agent
- ◆ Performs
 - ◆ periodically request or notify all or a subset of members of the current value of selected information (routing update);
 - ◆ upon certain events, notify all or a subset of members of the current value of selected information;
 - ◆ the latter implies that all event notifications occurring within the DIF should be delivered to the RIB Daemon because there may be a subscription that is triggered by the event, (event management);
 - ◆ given that elements of the DIF members should be able to request immediate notification of an event's arrival, have it recorded in the RIB, or both (event management);
 - ◆ if a log of events received is to be kept (the black box function), then the RIB Daemon is the natural place or it (event management);
 - ◆ given that the RIB Daemon responds to requests for information from other members of the DIF, then it is the natural place to respond to external requests for information from the DIF Management System (DMS), (management agent).

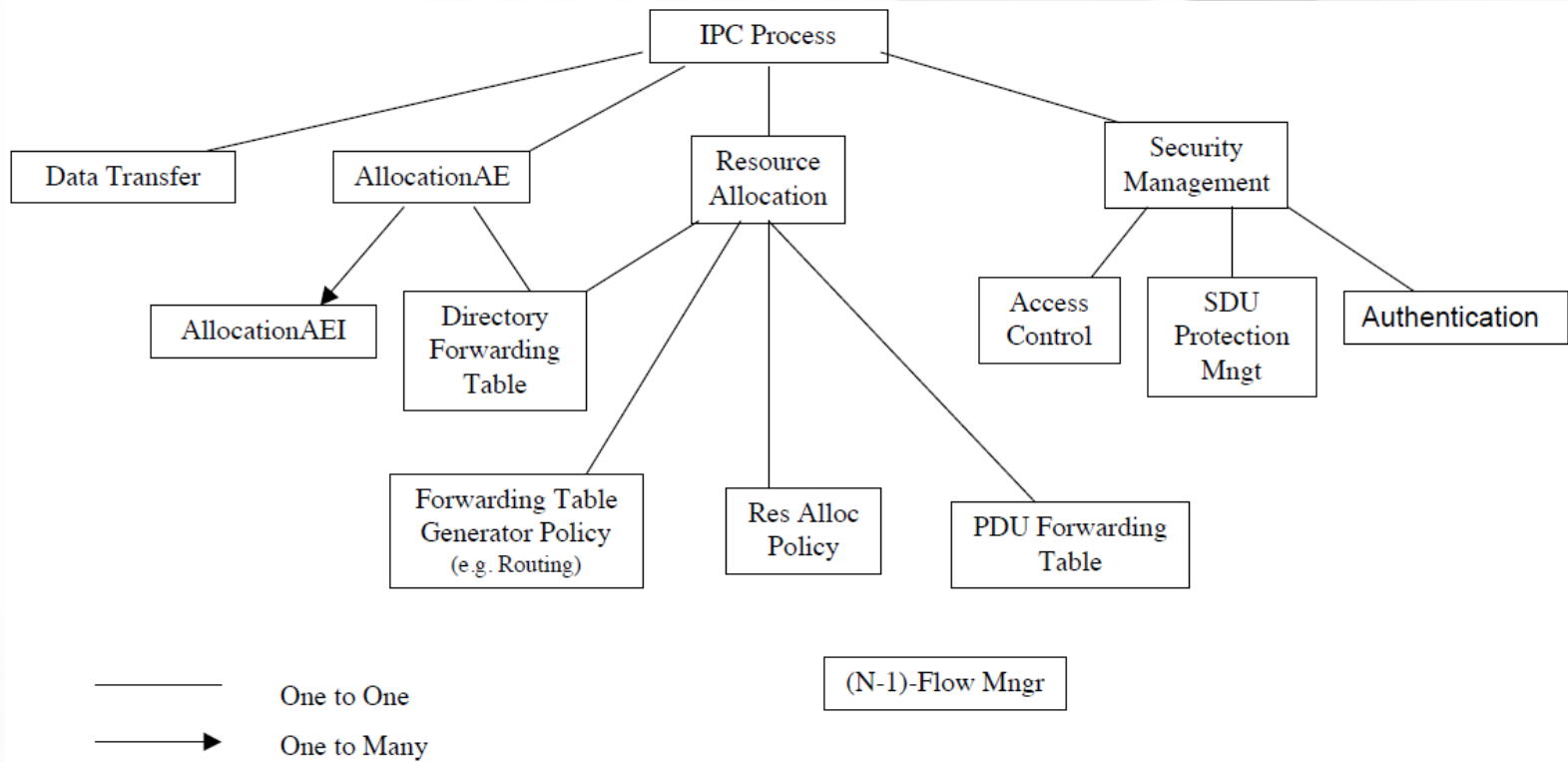


MIB ①

Intro

RINA

Example
PRISTINE



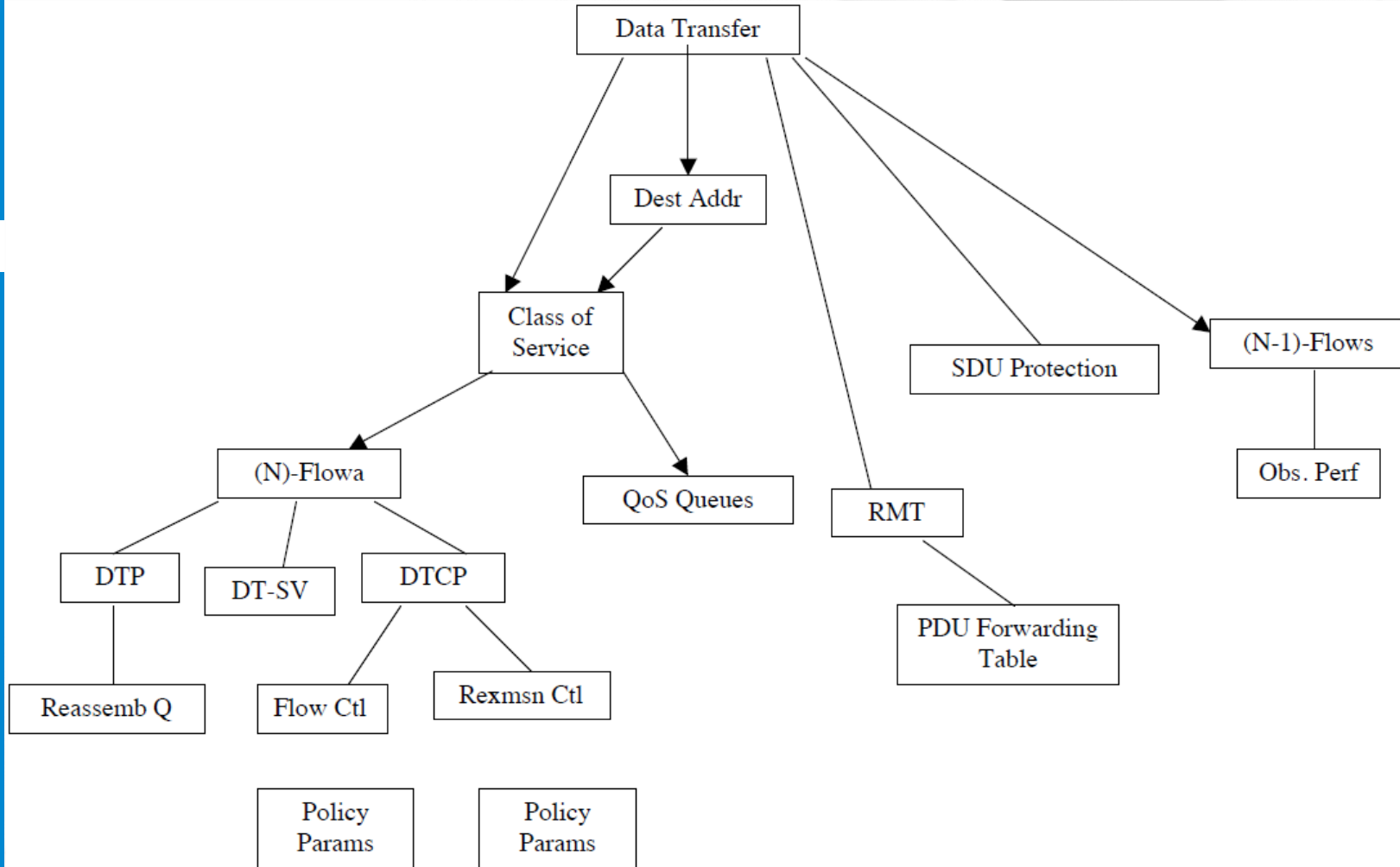


MIB ②

Intro

RINA

Example
PRISTINE





EXAMPLES

Intro

RINA

Examples

PRISTINE

- ◆ Flow Allocation
- ◆ Application Connection
- ◆ Application Search





ESTABLISHING EXAMPLE: BEGINNING

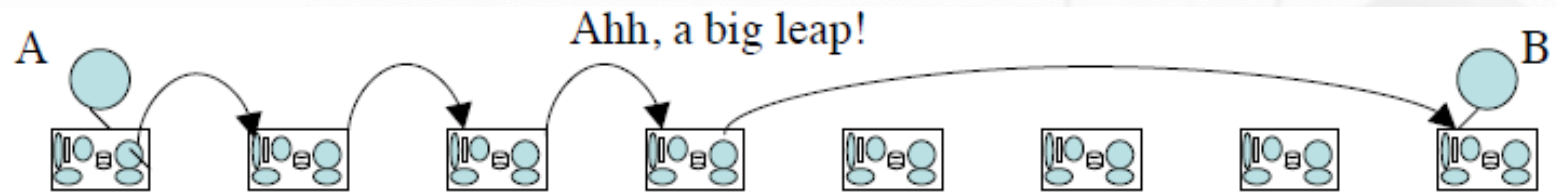
Intro

RINA

Examples

PRISTINE

- Means flow allocation in (N-1)-DIF

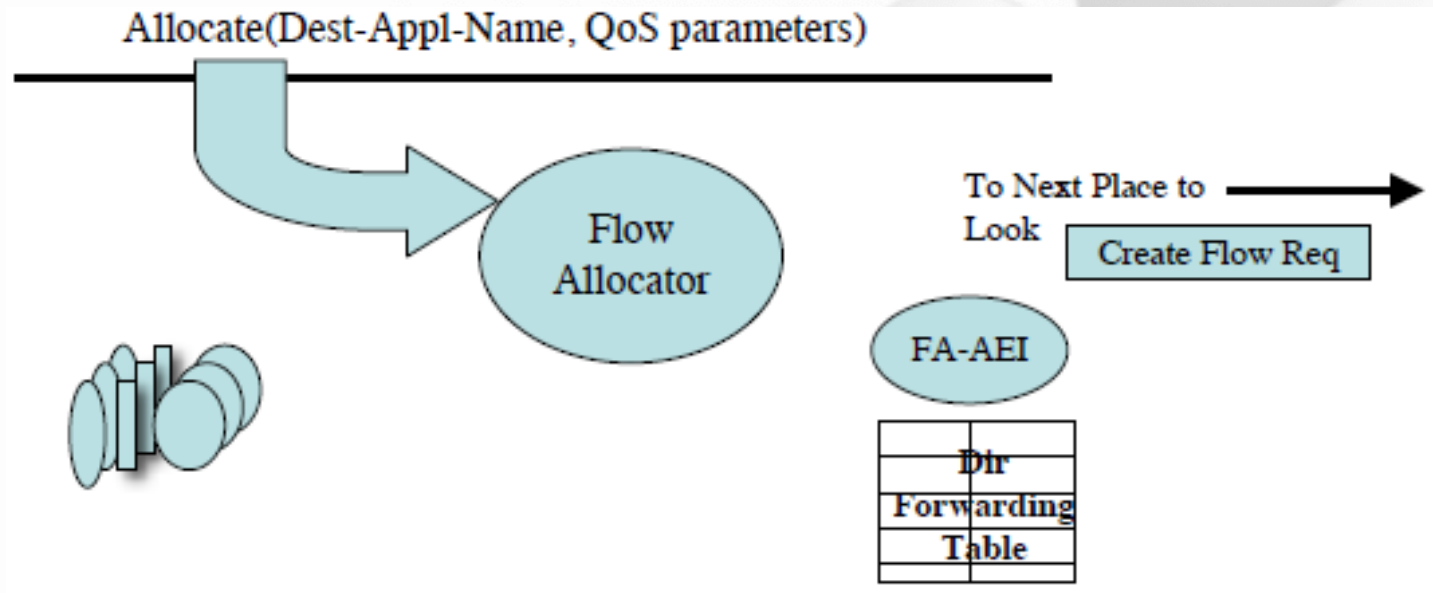




FA SOURCE

Intro
RINA

Examples
PRISTINE



- 1) An application issues an **Allocate Request** to the Flow Allocator.
- 2) IF well-formed THEN spawns an instance to manage the request and flow.
- 3) The FAI looks up the destination-application-name in its local cache and finds an address to look for the requested application.
- 4) It then instantiates an EFCP instance (whose id is the CEP-id).
- 5) Forms a Create flow request and sends it.

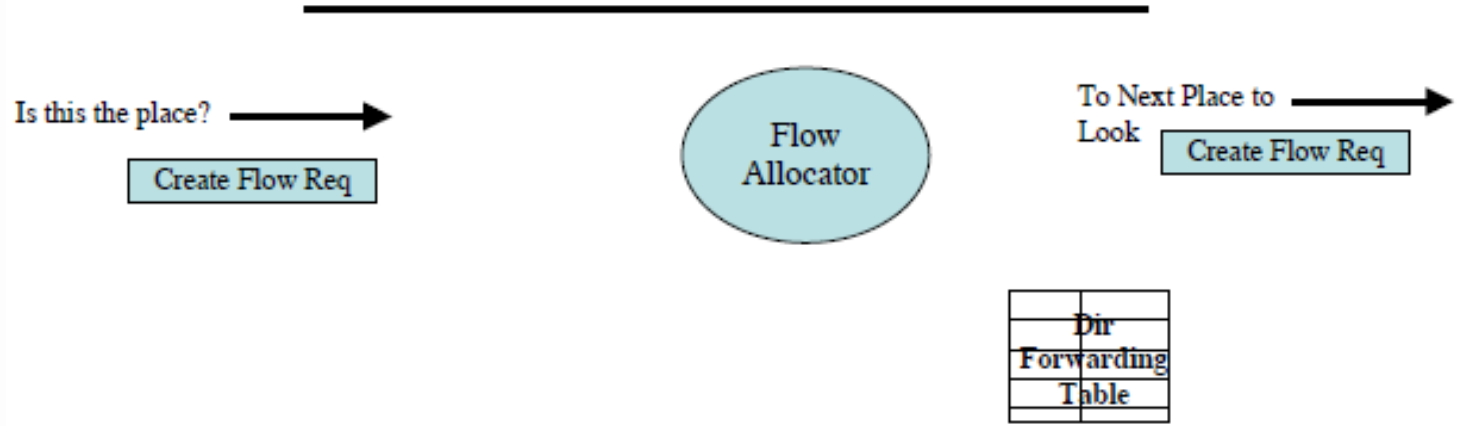


FA INTERIM

Intro
RINA

Examples

PRISTINE



- 1) Create Flow Request arrives at the next place to look.
- 2) FA looks up the destination application name in its “local cache”
- 3) Address returned isn’t ours, so not here.
- 4) Send it there



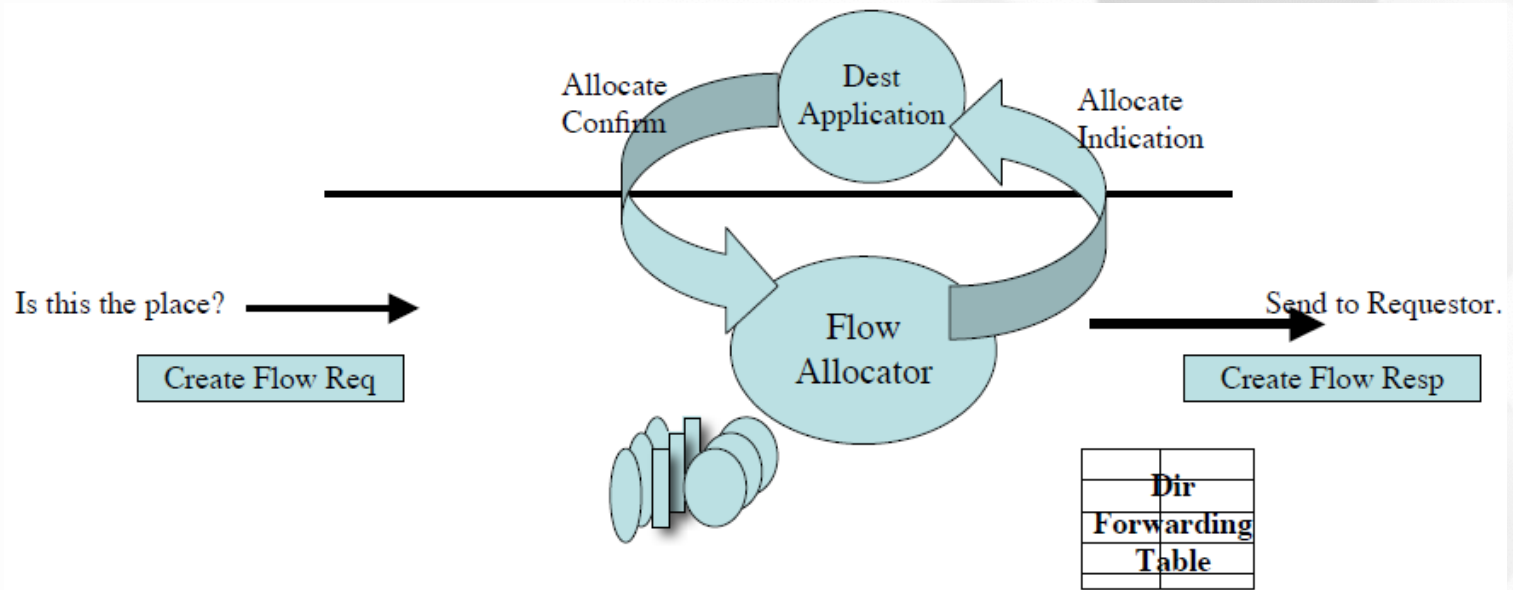
FA DESTINATION

Intro

RINA

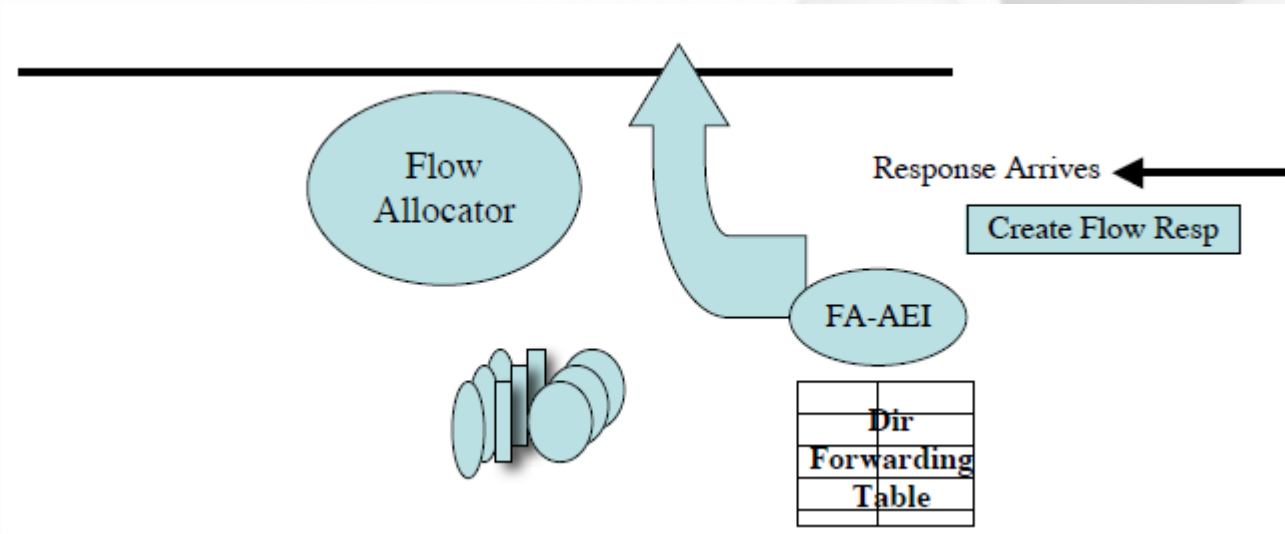
Examples

PRISTINE



- 1) Create Flow Request arrives at yet another place to look.
- 2) Flow Allocator looks up the destination application name in its “local cache”
- 3) Address returned is ours. Its here! Check requestors credentials, if okay.
- 4) Deliver Allocate indication to the destination application
- 5) Which accepts or rejects, if accepts sends a Create Flow Response
- 6) Instantiates a EFCP-instance
- 7) The connection is established.

FA BACK AT SOURCE



- 1) The Create Flow Response arrives from the Destination
- 2) The Application is notified that the Allocate was successful and is given a port-id (in Unix this might be a file descriptor), i.e. the FA-AEI-identifier.
- 3) The application can now start exchanging information.

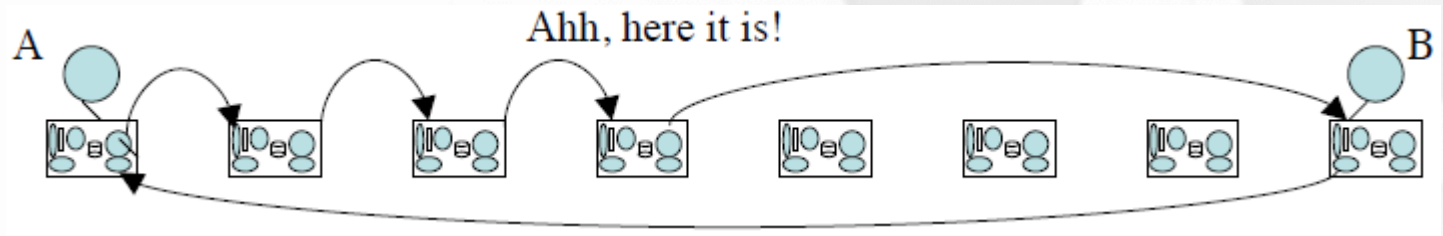
ESTABLISHING EXAMPLE: END

Intro

RINA

Examples

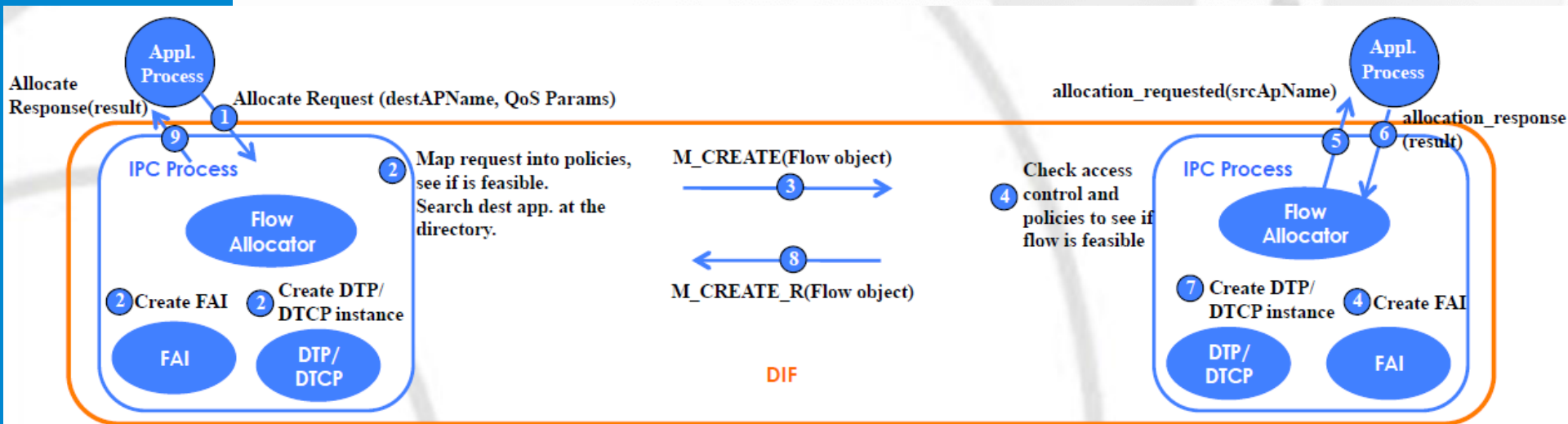
PRISTINE



- ◆ The Create Flow Response is sent back to the originator.
- ◆ Data can flow now
 - ◆ EFCP is instantiated at the source before start
 - ◆ Just in case data gets back before the Create Response
- ◆ Either end may start the Application Protocol exchange.



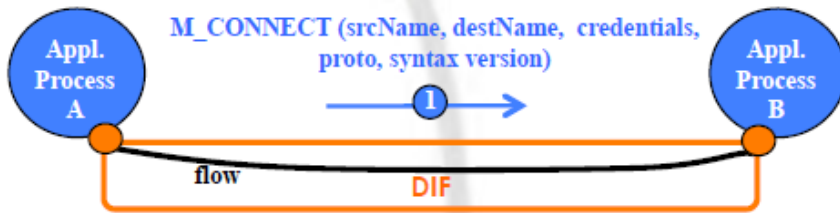
APPLICATION CONNECTION - FA





APPLICATION CONNECTION - CACE

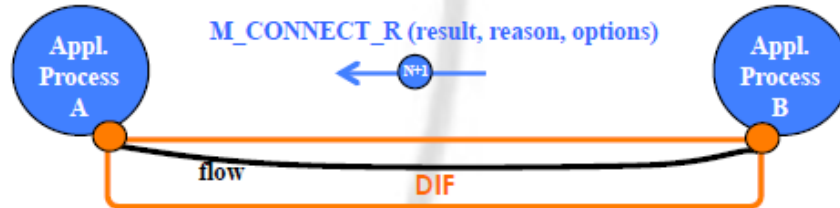
1)



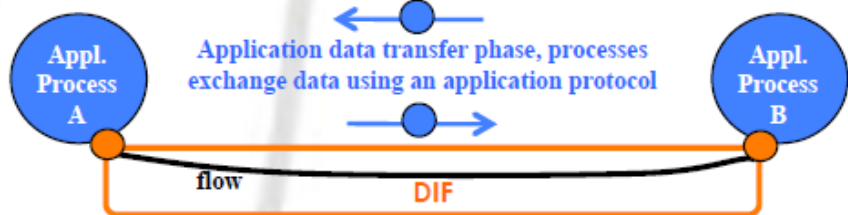
2)



3)



4)



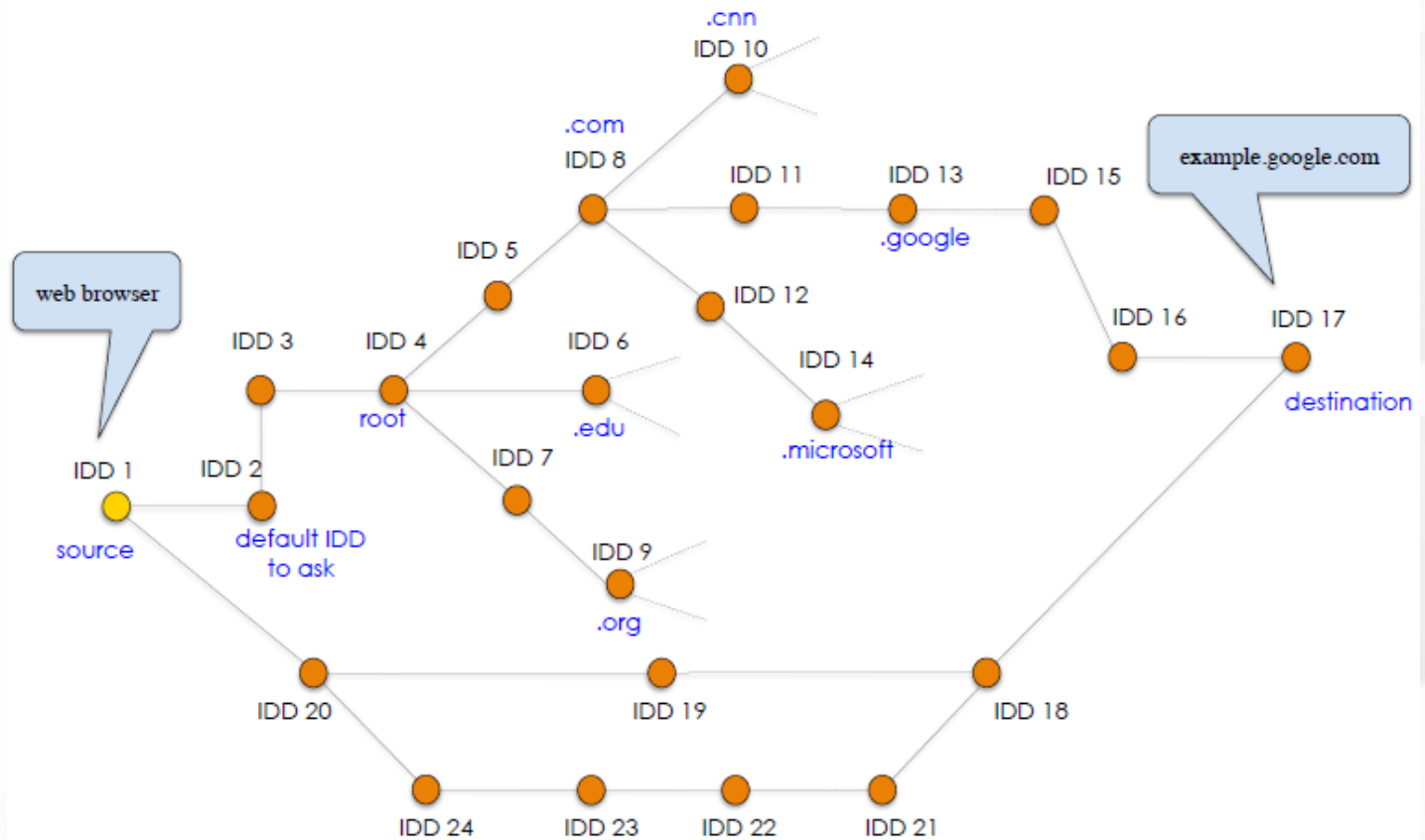


IDD SEARCH EXAMPLE

Intro
RINA

Examples

PRISTINE



	IDD 1	IDD 2	IDD 3	IDD 4	IDD 5
Neighbor Table	IDD2 - IDD2 ...	IDD 4 - IDD3 ...	IDD 4 - IDD4 ...	IDD8 - IDD5 IDD6 - IDD6 IDD9 - IDD7 ...	IDD8 - IDD8 ...
Search Table	* - IDD2	* - IDD 4	-	.com - IDD8 .edu- IDD6 .org- IDD9	-



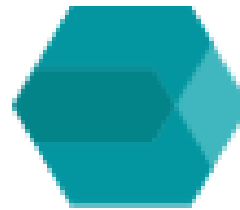
PRISTINE

Intro

RINA

Example

PRISTINE



Pristine

- ◆ Programmability In RINA for European supremacy of virTualised Networks
- ◆ <http://ict-pristine.eu/>



BASIC FACTS

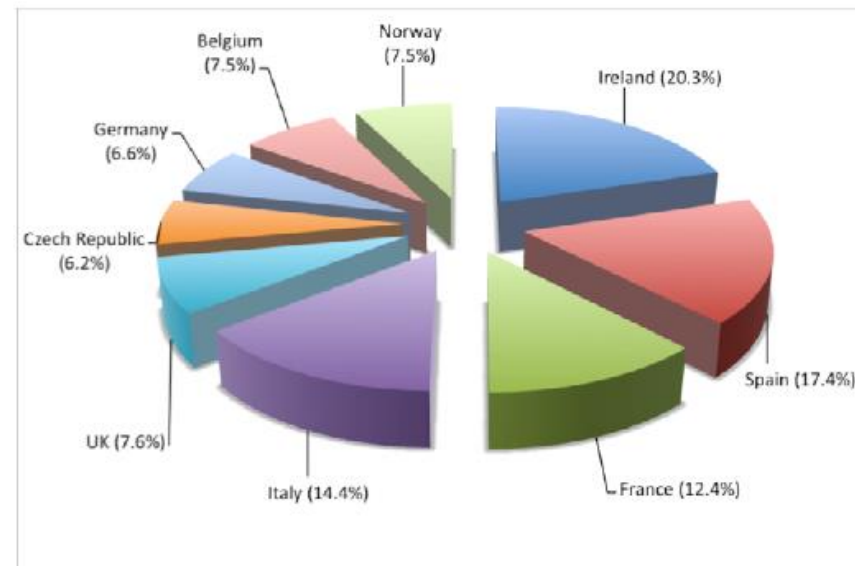
- ◆ Call: FP7-ICT-2013-11
- ◆ Date: January 2014 – June 2016
- ◆ Budget: 5 mil. €
- ◆ Participants: 15 consortium members
- ◆ FIT BUT: 40 pm

Intro

RINA

Example

PRISTINE





OBJECTIVES

Objetive (Task) / Partner	WIT-TSSG	i2CAT	TID	LMI	NXW	TRT	Nexedi	BISDN	Atos	JUN	UiO	FIT-BUT	IMT-TSP	CREATE-NET	iMinds
1 SDK (T2.2, T2.3)	C	L	L		L			C	C						C
2 Congestion Control (T3.1)				C						C	L	C			
3 Res. allocation (T3.2)		C				C			C	L	C		C	C	
4 Topological address. (T3.3)		C										C	L		
5 Secure DIF enablers (T4.1)			C			L		C				C	C		
6 Security coord. (T4.2)			C			C	L			C		C			
7 Resilient networks (T4.3)	C				C			C				C			L
8 Multi-layer management (T5.1, T5.2, T5.3, T5.4)	L	C	C	L	C	L		L	C					C	
9 Use case trials (T2.1, T6.1, T6.2, T6.3)	L	C	R	R	C	C	R		L,R	R	C			L	C
10 Simulator (T2.4)							C					L	C	C	



RESEARCH INTEREST

Intro

RINA

Example

PRISTINE

- ◆ RINA Simulator models in OMNeT++ are the main responsibility
- ◆ However, many other topics to explore
 - ◆ Routing, currently no routing protocol
 - ◆ IDDs (which uses DHT instead of FQDN)
 - ◆ Automatic enrollment (i.e. DHCP like behavior)
 - ◆ Security, of RINA
 - ◆ Deployment (OpenFlow, Linux SDK)

