Unreliable inter process communication in Ethernet: migrating to RINA with the shim DIF

Sander Vrijders[1], Eleni Trouva[2], John Day[3], Eduard Grasa[2], Dimitri Staessens[1],
Didier Colle[1], Mario Pickavet[1], Lou Chitkushev[3]
[1]Ghent University - iMinds, INTEC, Gaston Crommenlaan 8 bus 201, 9050 Gent, Belgium
E-mail: firstname.lastname@intec.ugent.be
[2]i2CAT Foundation, Jordi Girona, Barcelona, Spain
E-mail: firstname.lastname@i2cat.net
[3]Computer Science, Metropolitan College, Boston University, Massachusetts, USA
E-mail: day@bu.edu, ltc@bu.edu

***Abstract*—There is often a requirement to interface a new model to a legacy implementation by creating a shim between them to make the legacy appear as close to the new model as possible. This is a common exercise, usually fraught with frustrations, but here we find the exercise reveals fundamental aspects about nature of layers that were previously not well understood. Here we will be primarily concerned with creating a shim between RINA and IEEE 802.1q (VLANs). The Recursive InterNet Architecture (RINA) proposes a network architecture derived from the fundamentals of InterProcess Communication (IPC). This yields a recursively layered architecture of Distributed IPC Facilities (DIFs).**

I. Introduction

There is often a requirement to interface a new model to a legacy implementation by creating a shim between them to make the legacy appear as close to the new model as possible. This is a common exercise, usually fraught with frustrations, but here we find the exercise reveals fundamental aspects about nature of layers that were previously not well understood. Here we will be primarily concerned with creating a shim between RINA and IEEE 802.1q (VLANs) [1]. The Recursive InterNet Architecture (RINA) proposes a network architecture derived from the fundamentals of InterProcess Communication (IPC). This yields a recursively layered architecture of Distributed IPC Facilities (DIFs).

RINA provides explicit mechanisms for enrollment of processes in a layer/DIF, flow allocation and data transfer between processes. Unlike the Internet, the mode of operation of the data transfer, whether connection or connectionless is internal to the layer and not visible to the user. The user characterizes the kind of flow desired and the layer determines how to best provide it using connection-like or connectionless mechanisms. Applications request flows by naming the destination and never see addresses and there are no well-known ports.

For migrating from current TCP/IP over Ethernet networks to RINA, the concept of a shim DIF was introduced. The task of a shim DIF is to put as small as possible veneer over a legacy protocol to allow a RINA DIF to use it unchanged. To minimize the impact of the transition, a shim DIF only provides the capability of the legacy layer. It does not try to enhance it to be a fully functionally RINA DIF. Applications in the current internet model can keep on using the existing layers in this way, while applications using the RINA API can use the advantages RINA has to offer. A shim DIF allows a current layer in the current internet model to appear as if it was a regular DIF.

The rest of the paper is organized as follows. After a quick review of the Ethernet frame formats in Section II, we will describe the Recursive InterNet Architecture in more detail in Section III. Inter process communication in Ethernet will be discussed in Section IV. Next, we will describe how flow allocation is performed in the upper layer, in Section V. In Section VI, we present the shim DIF over Ethernet. We will explain the migration strategy of RINA in Section VII, form a conclusion in Section VIII and finally present some future work in Section IX.

II. Ethernet frame format

Following the OSI model, IEEE 802.3 [2] describes the physical and data link layer's medium access control mechanism of wired Ethernet. Ethernet does not provide a reliable service. This is not required. While the purpose of any layer should be to meet the requirements of its users, the purpose of the traditional Data Link Layer, in particular, is to provide sufficient error control that end-to-end error control (at the Transport Layer) is cost-effective. Generally, this means that the rate of loss at the data link layer should be no worse and less than the loss due to congestion at the Network Layer. For Ethernet, this requirement is easily met by the nature of the media, the Manchester encoding, and the checksum on the frames. (Some media may require more robust mechanisms.) Ethernet was first published in 1985. Since then, it has had several revisions, mostly focused on the physical layer. Before the standard was created, there was already a specification released by DEC, Intel, and Xerox, called Ethernet II framing. The structure of an Ethernet II frame can be seen in Table I. It is the most common frame structure used today. Recently, an optional IEEE 802.1q tag [1] has been included to support VLANs, and is in widespread use. The tag consists of the Ethertype 0x8100 (2 bytes), and Tag Control Information (also 2 bytes), which holds the VLAN identifier among other fields.

TABLE I
ETHERNET II FRAME FORMAT

Preamble	MAC dest	MAC src	802.1Q tag (optional)	Ethertype	Payload	FCS	Interframe gap
7 bytes	6 bytes	6 bytes	4 bytes	2 bytes	42-1500 bytes	4 bytes	12 bytes

TABLE II
802.3 FRAME FORMAT

Preamble	SOF delimiter	MAC dest	MAC src	802.1Q tag (optional)	Length	Payload	FCS	Interframe gap
7 bytes	1 byte	6 bytes	6 bytes	4 bytes	2 bytes	42-1500 bytes	4 bytes	12 bytes

The other Ethernet frame structure that is used today is the IEEE 802.3 frame structure, followed by a Logical Link Control (LLC) frame, standardized as IEEE 802.2 [3]. LLC is the upper sublayer of the data link layer in the OSI model. The structure of the IEEE 802.3 frame resembles the structure of the Ethernet II frame with the difference that the Ethertype field now represents the frame length, and a start of frame delimiter is added (see Table II). The IEEE 802.3 frame doesn't need a type field, because an LLC header provides Source and Destination Service Access Points (SSAP and DSAP, respectively (see Table III)). A Service Access Point is a generic OSI Reference Model term, i.e. applies to all layers, for the entity named by a connection-end-point-identifier at the layer boundary.

TABLE III
LOGICAL LINK CONTROL FORMAT

DSAP	SSAP	Control	Information
1 byte	1 byte	1-2 bytes	M bytes ($M \geq 0$)

III. A SHORT DESCRIPTION OF THE RECURSIVE INTERNET ARCHITECTURE

The Recursive Internet Architecture (RINA) described by John Day in 2008 [4] [5] is a unified theory of networking. RINA starts from the premise that networking is inter process communication (IPC) and only IPC. From this premise, a model of repeating layers of IPC arises. Here a layer is a distributed application that provides IPC, called a DIF (Distributed IPC Facility), to other distributed applications (including other DIFs) over a certain scope. In other words, DIFs recurse and provide IPC to one another. A DIF is configured by a set of policies, so it operates well for a given range of bandwidth and QoS. In the current internet architecture, protocols were designed for a specific situation, usually wired networks, which means a new protocol is needed for each new situation, even though there is considerable commonality in the functions/mechanisms used. This is one of the main advantages of RINA. In RINA, there is separation of mechanism (e.g. function) and policy. As a consequence, this IPC model recognizes the 3 phases all communication must progress through, is more secure than the current networking model, has inherent support for QoS, multihoming and mobility without the need for extra protocols. In Figure 1 an example with two DIFs is depicted.

A DIF consists of one or more IPC processes. As seen in Figure 1, an IPC process consists of different components that offer a different functionality (configurable by a certain policy). These components are divided in three groups that increase in complexity and decrease in frequency of use: Data Transfer, Data Transfer Control, and Layer Management. The functionality of a DIF is offered through the IPC API. This API allows applications (an IPC process is also an application) to allocate new flows with a certain Quality of Service (QoS), Read/write data from/to these flows, and deallocate them again. Applications can also register with a DIF so that they are reachable and unregister when they should no longer be reached.

All inter process communication (not only in RINA) goes through three phases: Enrollment, allocation (or establishment), and actual data transfer.

The enrollment phase creates, maintains, distributes and deletes the information within a layer that is necessary to create instances of communication. This includes setting addressing information into the appropriate directories and routing tables, access-control rules are established. The enrollment phase in the current internet architecture is there but has often been ignored because this has to be performed as configuration, or setup, often manually. Certain applications are currently used for some aspects of this, such as DHCP. More well-formed enrollment phases, similar to that found in RINA, can be found in the IEEE 802.11 (WiFi) [6] and 802.1q (VLAN) [1] environments. Enrollment in RINA is used by an IPC Process when it wishes to join a DIF, or if it is the first member, to create a new DIF. During enrollment in a DIF, information specific for this DIF, such as the maximum packet size, is exchanged between a member of the DIF and the member that wants to join.

The allocation phase creates, maintains and deletes the shared state necessary to support the functions of the data transfer phase for a particular instance of communication. This creates initial shared state in the communicating protocol machines in order to support the functions of the protocol. In RINA, the allocation phase is performed by the Flow Allocator. Flow allocation is responsible for creating and managing an instance of IPC, also known as a *flow*. In RINA, the term flow is used to designate the construct seen by the user of the DIF; within the DIF a connection exists between EFCP-entities that may use connectionless or traditional connection mechanisms. A flow allocation request is sent to an IPC

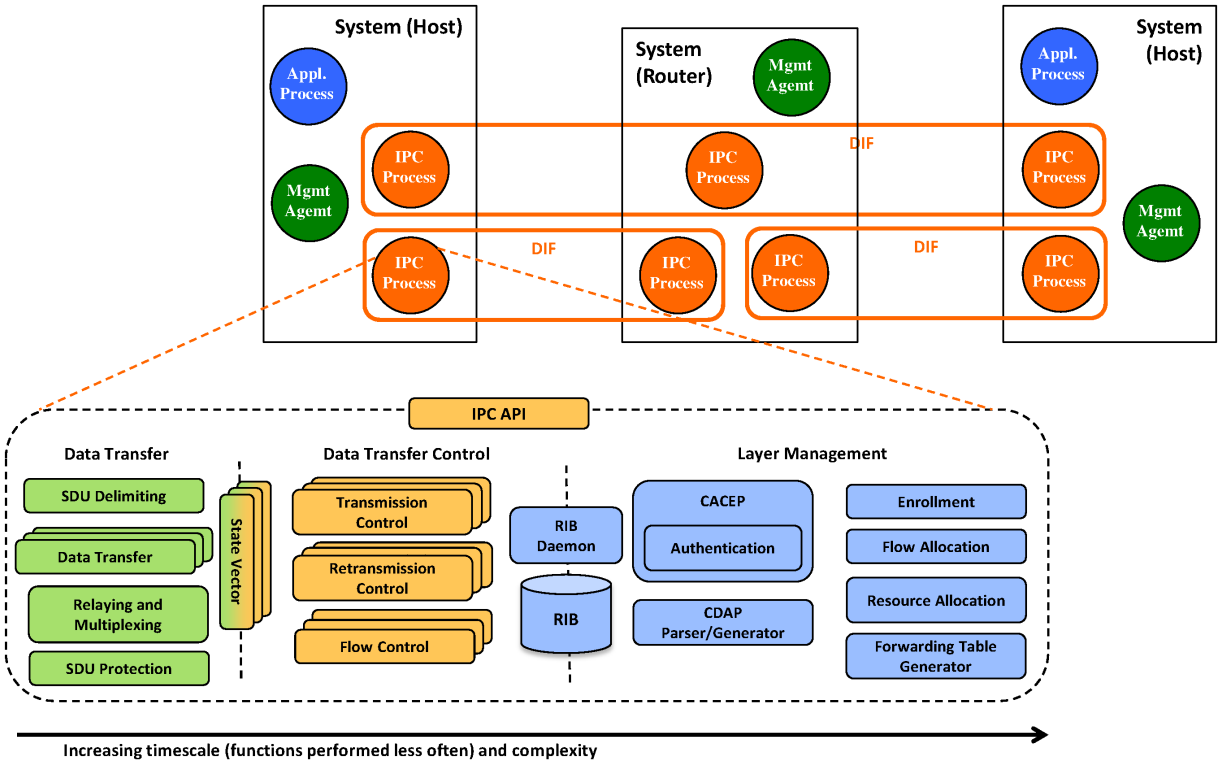

Fig. 1. The RINA Architecture.

process when an application requests a new flow. The receiving application (or IPC process, which is also an application) can decide to accept or reject the request for a new flow. The application requesting the flow is notified of this result. If the flow allocation was successful, a port-id is assigned to the application requesting the flow, which it can use for future API calls. Flow Allocation also allocates an EFCP-instance (Error and Flow Control Protocol instance) in the N-1 DIF, which provides the actual data transfer. Each EFCP-instance is identified by a connection-endpoint-id (CEP-id), which is unique in the IPC process. Figure 2 shows the state after flow allocation.

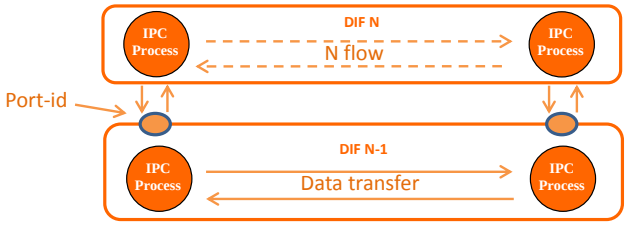

Fig. 2. After flow allocation.

Finally actual data transfer can happen. To achieve this, RINA uses an Error and Flow Control Protocol (EFCP), based on delta-t [7]. The Error and Flow Control Protocol (EFCP) provides the necessary synchronization between two IPC processes with each flow N. Delta-t is based on the result that, for synchronization it is necessary and sufficient to bound three timers: Maximum Packet Lifetime, Maximum Delay on ACK, and Time to Complete Maximum Retries. It encompasses Data Transfer, and Data Transfer Control, which are connected to each other through a State Vector (SV).

Our investigation of the separating mechanism and policy in this class of protocols revealed a natural cleavage between control and data. Consequently, EFCP is really two simple state machines coordinated through a state vector: Data Transfer, which only does sequencing and fragmentation/reassembly and Data Transfer Control for the feedback mechanisms, flow and retransmission control. In RINA, a service like UDP is merely Data Transfer without Data Transfer Control.

SDU Delimiting performs delimiting of SDUs (Service Data Units). Delimiting is necessary so the identity of the SDU is preserved, because the IPC process may find it necessary to concatenate it with other SDUs, or fragment it.

SDU Protection protects the SDUs before transmitting them. A variety of techniques can be used here, ranging from a simple checksum to encryption, again, depending on the policy enforced in the DIF, and exchanged during enrollment.

Protocol Data Units (PDUs) are sent between communicating IPC processes and can be viewed as SDUs wrapped by Protocol Control Information (PCI). PDUs that are ready for transmission are sent to the Relaying and Multiplexing component (RMT). The primary job of the RMT is to pass PDUs to the correct DIF via an (N-1)-port-id based on the information in the Forwarding Table, which is generated by the Forwarding Table Generator. The Forwarding Table holds entries mapping an address in this DIF (N) to a certain QoS

and one or more (N-1) port-ids. (Note how the port-id provides isolation between layers.) If a PDU arrives on a (N-1) port with an address that does not belong to this IPC process, it is forwarded based on information in the forwarding table.

The remaining components are all layer management functions. The Common Distributed Application Protocol (CDAP) is used for this, which is a platform for building distributed applications. CDAP consists of three parts: Common Application Connection Establishment (CACEP), for initializing the application connection. Auth for authenticating the correspondent(s), and a protocol for performing fundamental operations like create, delete, start, stop, write, read.

The Resource Information Base (RIB) is the local representation of the view of the DIF in the IPC Process. It can logically be viewed as a partially replicated distributed database. For instance, information such as all state information maintained by the IPC Tasks, the Flow Allocator, Resource Allocator is maintained and stored here. Access to the RIB is controlled by the RIB Daemon.

Resource Allocation uses information in the RIB to determine the resource allocations for this IPC Process. Each system also has a DIF Management System (DMS) that is responsible for network management functions.

IV. Inter process communication in Ethernet

In this section we will discuss inter process communication in Ethernet. In the Recursive InterNet Architecture, the two lowest (packet) layers of the current internet architecture (Ethernet), the physical and data link layer, can be viewed as one DIF. The same applies to the two layers above (IP and TCP/UDP/RTP/...).

In Ethernet, there is no explicit enrollment phase. The address is assigned to the network interface card at manufacture. In RINA, an address is obtained when enrolling to a DIF. The length of the address in the DIF is a matter of policy, and will depend on the scope of the DIF. Also, the address only has to be unique in the namespace of the DIF. By being a globally unique ID, MAC addresses have often been used for other purposes. Some which could compromise privacy.

In Ethernet II framing, an Ethertype is used [8]. The Ethertype identifies the syntax of the encapsulated protocol. Because the layer below needs to know the protocol of the layer above, fields like this both here and in protocols like IP have always been controversial as whether they are a layer violation. Having an Ethertype per protocol assumes there is a single flow between an address pair. There is no other field to identify the connection-endpoint-id (CEP-id) of the flow. The MAC address doubles as the one CEP-id.

When there is no explicit flow allocation phase, there is an implicit agreement between nodes that all Protocol Data Units (PDUs) will be accepted, and that the CEP-ids are fixed and implicit. In Ethernet there are no dynamically assigned port-ids that are assigned upon accepting a new flow.

With LLC (IEEE 802.2), the Ethertype field is unnecessary and is used as a length field. Here, the Source and Destination SAP are the CEP-ids. They are still fixed CEP-ids, and all traffic that arrives will be accepted, but they allow more than one flow between two communicating nodes to be differentiated. Hence there is reason to suspect that a layer without port-ids or its equivalent is a false layer boundary.

V. Flow allocation in the upper layers

In this section we will very briefly discuss how flow allocation is performed in the transport layer.

UDP has the same problem as Ethernet. There is no flow allocation phase, with no feedback mechanisms. There is minimal requirement for synchronization of state. UDP should have a flow allocation phase to allocate ports, but with UDP the user of the layer has to know the UDP port number of the application it wishes to communicate with. UDP requires either manual configuration or some other protocol to allocate a flow. The fact that UDP doesn't have flow allocation allows for a variety of possible attacks, since an attacker most likely also knows the port number and no access control is performed. Even if the attacker doesn't know the port number, there are only 2^{16} ports, and the ports can be scanned.

The problem with TCP is that it overloads the uses of the TCP port, both as a port-id and as a CEP-id. This results in the server having to rely on client generated identifiers to distinguish flows rather than identifiers the server generated.

This means the port numbers are known up front, which as said before, allows a variety of attacks. TCP does have a synchronization phase, but there is no decoupling of the flow allocation and the data transfer phase, which could mean a TCP connection is ended during long times of no traffic, although more traffic may follow, which would require setting up a new TCP connection. This also introduces another security risk, since there is no way to verify if the new connection for the same "flow" was established by the same user.

VI. The Shim DIF over Ethernet

In the case of the shim DIF over Ethernet, the shim IPC process wraps the Ethernet layer with the IPC process API. The goal is not to make legacy protocols provide full support for RINA and so the shim DIF should provide no more service or capability than the Ethernet layer provides. An Ethernet shim DIF spans a single Ethernet segment. This means relaying is done only on the MAC addresses. This also means we assume the number of users of the same shim DIF is small. It is not the case that all stations on an Ethernet segment are by default members of the same shim DIF. Each shim DIF is identified by a VLAN (IEEE 802.1Q) id, which is in fact the shim DIF name. Each VLAN is a separate Ethernet Shim DIF. All the traffic in the VLAN is assumed to be shim DIF traffic. All members of a VLAN are assumed to be members of the same shim DIF. Thus joining the VLAN is considered enrolling in the shim DIF.

Because this Ethernet shim DIF does not use LLC, there can only be a single user of the Ethernet shim DIF. Other shim DIFs that use LLC may be defined later on. Therefore, the only applications that can register in an Ethernet shim DIF

are IPC Processes. Moreover, since Ethernet doesn't provide explicit flow allocation, there can only be one instance of an IPC Process registered at each Ethernet shim IPC Process. There is also only one QoS cube, namely the unreliable QoS cube.

The fields in the Ethernet header (see Table I) are translated as follows in the shim DIF:

- Destination MAC address: The MAC address assigned to the Ethernet interface the destination shim IPC Process is bound to.
- Source MAC address: The MAC address assigned to the Ethernet interface the source shim IPC Process is bound to.
- 802.1q tag: The DIF name.
- Ethertype: Although it is not strictly required to have a special Ethertype for the correct operation of the shim DIF (since all the traffic in the VLAN is assumed to be shim DIF traffic), it is handy to define an Ethertype for RINA (if, for no other reasons, to facilitate debugging). Therefore the Ethernet frames used within the shim Ethernet DIF will use the 0xD1F0 value for the Ethertype field.
- Payload: Carries the upper DIF SDUs. The maximum length of the SDU must be enforced by the upper DIF, since the Ethernet shim DIF doesn't perform fragmentation and reassembly functions. The maximum length can be higher if Jumbo frames are used.

Each shim IPC Process is assigned to an Ethernet interface. The shim IPC Process will receive all the Ethernet frames belonging to the VLAN of the DIF addressed to the MAC address of that interface. The shim IPC Process needs the shim IPC process AP name, the OS specific name of the Ethernet interface to be bound to, and the shim DIF name, which is in fact the VLAN id, in order to operate effectively.

Instead of implementing its own directory mechanism, the Ethernet shim DIF reuses ARP in request/response mode to perform this function. ARP resolves a network layer address into a link layer address; that is, in the context of the shim DIF, mapping the application process (AP) name to a shim IPC Process address, exactly the function that the directory in RINA provides. Figure 3 shows where the shim IPC process is located with respect to the ARP protocol machine and Ethernet layer.

When an IPC Process registers ("register") with the shim IPC process, it will pass its naming information to the shim IPC process. Depending on the configuration the operation is accepted or denied. When the application is registered the shim IPC processs ARP cache gets populated with a static entry, mapping the AP name to the MAC address of the interface the shim IPC Process is bound to. When an application unregisters ("unregister"), the shim IPC process removes the corresponding static entry from the ARP cache, so future queries of the application name are ignored.

For IPC communication, the state diagram in Figure 4 is used. When an IPC process (source, A in Figure 4) wants to communicate with another IPC process (destination, B in

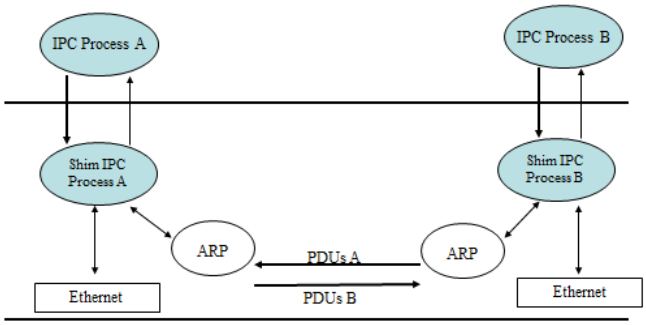

Fig. 3. The shim IPC process location

Figure 4), it uses the allocateRequest primitive. If there is already a flow established to the destination application (there is already a port-id, in the ALLOCATED state), or we are currently trying to setup a flow (there is also already a port-id and the port-id is in the INITIATOR ALLOCATE PENDING state), a negative reply is returned to the top IPC process. If there is an entry in the ARP table for the destination application, the shim IPC process creates a new port-id, not currently in use in the namespace of the shim IPC process and a positive reply is returned to the top IPC process. The port-id transitions to the ALLOCATED state. If there isn't one, again the shim IPC process creates a new port-id, not currently in use in the namespace of the shim IPC process and an ARP request is generated. The port-id transitions to the INITIATOR ALLOCATE PENDING state.

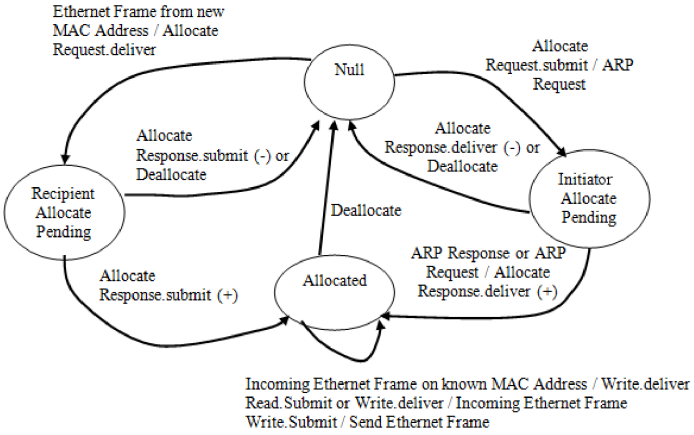

Fig. 4. The shim IPC process state diagram

If the port-id is in the INITIATOR ALLOCATE PENDING state, and an ARP response arrives, a positive reply is generated, which returns the port-id to the top IPC process. The directory is populated with a new entry, mapping destination application to MAC address. In this case the port-id transitions to the ALLOCATED state. If the port-id is in the ALLOCATED state, and an ARP response arrives, the directory gets updated with the new mapping of application name to MAC

address.

If the port-id is in the INITIATOR ALLOCATE PENDING state, and an ARP request arrives from the application that we are trying to reach, the port-id transitions to the ALLOCATED state. This solves the race condition where both applications are trying to reach each other at the same time.

When the shim IPC Process receives an Ethernet frame, it checks if it has seen a frame from the sender before. If this is the case, and the corresponding port-id is in the ALLOCATED state, the SDU is delivered on the port-id corresponding to this flow. If this is not the case, the packet is queued and a port-id is created and it transitions to the RECIPIENT ALLOCATE PENDING state. A message requesting the creation of a new flow is sent to the top IPC process, with the newly created available port-id.

If the top IPC process accepts the new flow, it calls allocateResponse; any queued frames are delivered to the destination application. The port-id transitions to the ALLOCATED state. If the allocateResponse is negative the flow creation has failed and reason indicates the failure reason, all future Ethernet frames from this source MAC address are dropped, until the source application deallocates the flow for this port-id. The port-id transitions to the NULL state in this case.

When a port-id transitions to NULL (because of a deallocation call or the shim IPC process wishes to deallocate all resources concerning the port-id), all corresponding data is removed. This includes the port-id and corresponding state, a queue if there is one.

When the top IPC process calls write on a port-id and the port-id is in the ALLOCATED state, the shim IPC process will create an Ethernet frame and send the SDU. Likewise, when the top IPC process wants to read an SDU from a port-id, and the port-id in the ALLOCATED state, the shim IPC process will wait for the next Ethernet frame to arrive and deliver it.

VII. RINA MIGRATION STRATEGY

RINA can only be of practical use when there is a migration strategy available. Migrating from IPv4 to RINA is easier than migrating from IPv4 to IPv6 using the IETF recommendations. IPv6 was not designed for backward compatibility with IPv4. IPv4 and IPv6 occupy the same place in a fixed protocol stack. Since DIFs are not in a fixed place in the protocol stack, more options are available for a phased adoption. As shown before, Ethernet can be seen as a DIF in RINA with limited capabilities. RINA can be used above Ethernet to allow RINA applications to communicate seamlessly over Ethernet, using the shim DIF, while allowing existing applications that make use of Ethernet to keep on communicating. This also allows reuse of existing hardware.

There is also a shim DIF available for TCP/IP. As with Ethernet, this allows users to run RINA on top of TCP/IP, which allows using existing IP links as a transport medium. It also allows using RINA below TCP/IP, where it can provide network services to TCP/IP hosts, or to connect IP networks. Internet applications on a RINA DIF can be accessed transparently from the Internet and vice-versa with this shim DIF. A backbone network could be built using RINA, supporting both types of upper layers. Since RINA does not depend on globally-unique IP addresses, a single IP address could function as a gateway to a RINA network in which applications communicate using names.

More importantly, RINA offers the faux sockets API. This API is a wrapper around the existing socket API, which means that applications can be recompiled using the faux sockets API, which makes then RINA enabled. Of course, since the application still uses the old socket API, it cannot make use of the advantages RINA has to offer. That's why RINA native applications should be developed and used, or existing applications should be modified, which offer better security, QoS, ... This way, applications can move away from TCP/IP, and an easier migration is possible than transitioning from IPv4 to IPv6.

VIII. CONCLUSION

We discussed RINA, the Recursive InterNetwork Architecture, which is a clean slate architecture with a layered design. In this design, the two lowest layers of the current architecture, commonly referred to as Ethernet, can be viewed as a DIF in RINA. Although Ethernet can be viewed as a layer in the current internet architecture, as a DIF in RINA it would be incomplete because it lacks port-ids. Without port-ids, a layer requires a protocol-id field to identify the syntax of the protocol in the layer above. This choice also limits configurations to a single instance of a given protocol, i.e. only one IP above Ethernet. This complicates so-called virtualization schemes and compromises security. Interestingly, Ethernet with LLC does form a complete layer, minimizing the information shared and not requiring central assignment of port-id values. This would seem to indicate that a complete layer that minimizes the knowledge that one layer must know about the layer above requires the port-id concept for isolation. This would imply that IP and Ethernet without LLC are ill-formed layers in an architecture, while Ethernet with LLC and TCP are at least better-formed layers. (Note that TCP's overloading of the port-ids does compromise security.)

In this paper, we presented the shim DIF over Ethernet. This shim DIF emulates flow allocation, presenting the RINA API to other IPC processes, which makes for transparent use. With the shim DIF in place, migration from the current internet architecture can happen gradually to the recursive internet architecture, which offers reliable data transfer with explicit flow allocation, and access control. Applications can make use of RINA to extend their capabilities, and make use of the inherent QoS, security, multihoming, ...

IX. FUTURE WORK

We are currently developing a prototype in the IRATI project, part of the Future Internet Research and Experimentation (FIRE) objective of the Seventh Framework Programme (FP7). This prototype will use the shim DIF over Ethernet. This prototype will allow us to test the functionality of the shim DIF, and perform experimentation with RINA.

If the current shim DIF over Ethernet proves to be too limiting, other shim DIFs over Ethernet might be defined that make use of LLC, and other IEEE legacy protocols. We will also develop a shim DIF over 802.11, to allow as many applications as possible to make use of the RINA protocol stack.

ACKNOWLEDGMENT

This work is partly funded by the European Commission through the IRATI project (Grant 317814), part of the Future Internet Research and Experimentation (FIRE) objective of the Seventh Framework Programme (FP7).

REFERENCES

[1] "IEEE Standard for Local and metropolitan area networks: Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks," *IEEE Std. 802.1Q-2011*, 2011.

[2] "Part 2: Logical Link Control," *IEEE Std. 802.3-2008*, 1998.

[3] "Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications," *IEEE Std. 802.2-1998*, 2008.

[4] J. Day, *Patterns in network architecture: a return to fundamentals*. Prentice Hall, 2008.

[5] E. Trouva, E. Grasa, J. Day, I. Matta, L. T. Chitkushev, P. Phelan, M. P. de Leon, and S. Bunch, "Is the internet an unfinished demo? meet rina!" 2010.

[6] "IEEE Standard for Information technology: Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std. 802.11-2012*, 2012.

[7] R. W. Watson, "Timer-based mechanisms in reliable transport protocol connection management," *Computer Networks (1976)*, vol. 5, no. 1, pp. 47–56, 1981.

[8] IEEE, "Registration authority programs," 2013. [Online]. Available: https://standards.ieee.org/develop/regauth/ethertype/