2010-10-15

On Supporting Mobility and Multihoming in Recursive Internet Architectures

Technical Report BUCS-TR-2010-035, Computer Science Department, Boston University

On Supporting Mobility and Multihoming in Recursive Internet Architectures

Vatche Ishakian · Joseph Akinwumi · Flavio Esposito · Ibrahim Matta

Abstract 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. Mobility can be viewed as a special case of multihoming—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 multihoming (and consequently mobility), which has led to the emergence of several custom point-solutions. The Recursive InterNetwork Architecture (RINA) was recently proposed as a clean-slate solution to the current problems of the Internet. In this paper, we present a specification of the process of ROuting in Recursive Architectures (RORA). We also perform an average-case cost analysis to compare the multihoming / mobility support of RINA, against that of other approaches such as LISP and Mobile-IP. Extensive experimental results confirm the premise that the RINA architecture and its RORA routing approach are inherently better suited for supporting mobility and multihoming.

Keywords Routing · mobility support · Multihoming · Loc/id split · recursive architecture · performance analysis · simulation.

This work has been partially supported by National Science Foundation awards: CISE / CNS #0963974, CISE / CCF #0820138, and CISE / CSR #0720604. An earlier version of this work will appear in the IEEE Globecom 2010 Workshop on Network of the Future (FutureNet-III), Miami, Florida, December 2010.

Vatche Ishakian, Joseph Akinwumi, Flavio Esposito, Ibrahim Matta
Boston University
Computer Science Department
Boston, MA, USA
E-mail: {visahak,akin,flavio,matta}@cs.bu.edu

1 Introduction

Support for multihoming and mobility was not a primary goal in the original design of the Internet. As a result, the Internet's naming and addressing architecture is incomplete. Specifically, the address of a multihomed host specifies a particular interface (connection), rather than the node itself. Because routing is done based on this interface *i.e.*, Internet Protocol (IP) address, if this active interface goes down, it is costly to switch to another operational interface.

There have been several attempts to fix this addressing problem, including the Location ID Separation Protocol (LISP)—currently being tested at Cisco [11, 16]—and Mobile-IP [20]. The basic idea behind LISP is to assign the multihomed node a provider-independent (location-independent) identifier (ID). A border router maps a destination ID to the node's location, which is the address of another border router that is known to have a path to the node. Routing is then done from the source's border router to the destination's border router. If the latter (node's location) changes due to path failure or mobility, it becomes costly to propagate that change over the whole Internet (to all possible source border routers).

Mobile-IP (MIP) allows a mobile host to seamlessly move from its home domain to a foreign location without losing connectivity. This is done by having a foreign agent (router) update the location of the mobile node at its home agent (router). Since mobility is a special (dynamic) form of multihoming, MIP can also be used to handle a change in the active interface (due to failure or re-routing) leading to a multihomed node, where a home agent directs traffic to the currently active (operational or "better") interface. However, this location update can be costly since it needs to propagate from the foreign agent to the home agent.

Note that both LISP and Mobile-IP (and combination thereof) help reduce the size of the routing tables at the core of the Internet, since several IDs can map to one location and hence be represented by one routing entry. Further elaboration on the benefits of LISP can be found in [21].

RINA [6] is a recently proposed Recursive InterNetwork Architecture. It uses the concept of Distributed Inter-process communication Facility (DIF) to divide communication processes into manageable scopes across network subsystems, which results in a reduced routing table size per DIF. RINA routes *hop-by-hop* based on the destination's node address, *not* its interface. At each hop, the next-hop node address is mapped to the (currently operational) interface to that next-hop node. This *late* binding of a node's address to its interface (path) allows RINA to effectively deal with interface changes due to multihoming or mobility. The cost of such *late* binding is relatively small since its scope is local to the routing "hop" that traverses the underlying DIF. By recursing the DIF structure to make the DIF scopes small enough, the cost of such *late* bindings (location updates) can be made arbitrarily small.

1.1 Our Contribution

We present a specification of the process of ROuting in Recursive Architectures (RORA) adopted in RINA, and highlight its inherent support for mobility and multihoming. We present a cost model to quantitatively assess the effectiveness of LISP, MIP, and RINA, in supporting multihoming / mobility. To the best of our knowledge, this paper presents a first cost comparison of these approaches. Our definition of "cost" captures both the average number of packets generated by a source node to a (multihomed or mobile) destination node, as well as the average path length from the source to the destination (as indication of delays or bandwidth usage). In our model, we compute the overall average cost for a *single* interface change experienced by the multihomed or mobile destination node. We validate our analytical model for mobility using simulation and for multihoming using trace-driven simulation.

1.2 Organization of the Paper

The rest of the paper is organized as follows. Section 2 reviews MIP, LISP, and RINA. Section 3 presents the RORA routing process. We present our general cost model in Section 4, and then we instantiate it for the various approaches. Section 5 presents numerical results for grid topologies. Section 6 evaluates the cost of supporting mobility using simulations, and Section 7 evaluates the cost of supporting multihoming using real packet traces from CAIDA [29]. Section 8 reviews related work and Section 9 concludes the paper.

2 Background

This section provides a basic background on the various architectures we study, namely MIP, LISP, and RINA—for more details, we refer the reader to references herein.

2.1 Mobile-IP

Mobile-IP (MIP) [20] has been mainly standardized to deal with the mobility of nodes. As mentioned earlier, since mobility is merely a (dynamic) form of multihoming, the MIP concept can also be used to deal with interface (path) change to a multihomed node.

In MIP, two basic mechanisms are identified: (1) a *discovery* mechanism, which allows a node to detect its new point-of-attachment, and (2) a *registration* mechanism, which allows a node to register itself with an agent that represents it at its home network.

Figure 1 shows a source node (SN) sending packets to a destination node (DN) in another Autonomous System (AS). The destination moves to a new AS and acquires a care-of-address at the Foreign Agent (FA). The FA then updates the corresponding Home Agent (HA) with DN's new location.

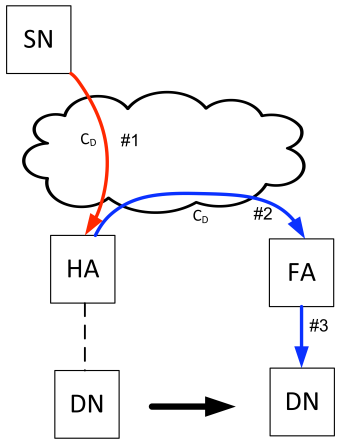

Fig. 1 Mobile-IP Protocol.

The basic delivery process of data packets from a source node to a destination node is as follows (highlighted as sequence 1–3 in Figure 1):

1. The datagram is delivered to HA via standard routing.

2. The HA intercepts the datagram and tunnels it to the destination's current location (care-of-address).
3. The FA at the current location intercepts the datagram and delivers it to the destination node.

2.2 LISP

The Locator/ID Separation Protocol (LISP), proposed by Farinacci *et al.* [9], separates the address space into end-systems' identifiers (EID) for source and destination hosts, and routing locators (RLOCs) where border routers act as RLOCs for the end-systems inside their local domain. The mappings, referred to as *EID-to-RLOC* mappings, are stored in a Mapping Server (MS).

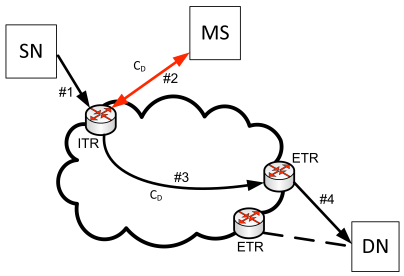

Fig. 2 LISP Architecture.

The basic delivery process of data packets from a source node (SN) to a destination node (DN) is as follows (highlighted as sequence 1–4 in Figure 2):

1. The source forwards the packet to its border router called Ingress Tunnel Router (ITR).
2. The source ITR performs a lookup query for a destination EID-to-RLOC mapping [8].
3. ITR transparently tunnels the data packets to the destination's RLOC referred to as Egress Tunnel Router (ETR).
4. Upon intercepting the packet, the destination's ETR forwards the packet to the destination.

Upon failure of an active interface, a multihomed destination node sends an update to its ETR, which in turn updates the EID-to-RLOC MS. The sequence of messages is shown in Figure 3.

Different variants of LISP only differ in how the EID-to-RLOC mapping is maintained [8]. The use of caching for lookup has also been recently explored in [12].

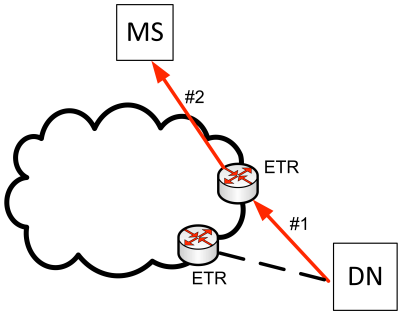

Fig. 3 LISP cost of update.

2.3 RINA

In RINA, application processes or services have globally unique names, and networking is viewed as distributed Inter-Process Communication (IPC) [6].

If an application process in RINA needs to communicate with another application process, it requests service from the underlying Distributed IPC Facility (DIF). This DIF maps the destination application name to a node (process) address. A DIF in RINA can (recursively) provide transport services between source and destination application processes, using services of underlying (lower-level) DIFs.

Routing: The route to the destination node address (to which the destination application process is connected) is computed as a sequence of intermediate node addresses. At each routing hop, the next-hop node address is in turn mapped (recursively) to a lower-level node address by the underlying DIF. This lower-level node address is viewed as the point-of-attachment of the higher-level node. Thus, RINA's addresses are relative: A node address at a DIF level (N) is considered a node name by a lower-level (N-1) DIF. At the (N-1)-DIF, this name needs to be mapped to a node (N-1)-address by the DIF's directory service. Eventually, the node (process) address maps to a specific path (interface). This *late binding* to a specific interface (path) makes it easier for RINA to deal with mobility (and multihoming). If an active interface (path) to a node fails, RINA maps the (next-hop / destination) node address to another operational interface (path). The cost of such interface/location update is small because the update is only *local* to the routing hop—the next-hop / destination node address is mapped to the lower-level node address that resides within the operational lower-level DIF.

On the contrary, in the current Internet model, the interface address (*i.e.*, IP address) names both the node itself and the interface (path) to that node—this static binding makes mobility (and multihoming) difficult to manage.

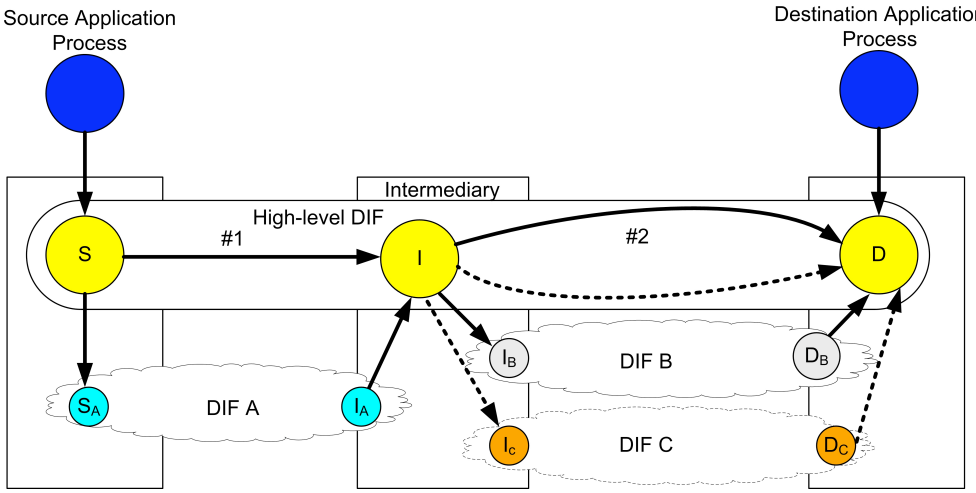

Fig. 4 A RINA Network.

RINA Example: Without loss of generality, Figure 4 shows a source process sending packets to a destination process using the services of the underlying DIFs. Note that in RINA, a single system may have multiple processes which are members of different DIFs at different levels [6]. The source and destination processes form a (high-level) DIF with an intermediate process, which we call "intermediary", such that the intermediary is connected to the destination process using two separate interfaces over two different underlying DIFs. This 3-node DIF can be thought of as an "overlay" (or private network) to which the source, destination, and intermediary had subscribed. When a packet reaches the intermediary, it forwards it based on the current best / operational interface (underlying DIF) leading to the destination process.

Remark: It is important to highlight the difference between how BGP [22] and RINA handle route / interface failures. In BGP, even if there is a specific path failure to a specific prefix (node), BGP may still broadcast a path to the destination since it relies on advertising reachability to *aggregate* destination prefixes. On the other hand, RINA would handle such failures using *hop-by-hop* routing within the DIF of the destination process. In Figure 4, if the (solid) overlay link I–D that uses the underlying DIF *B* goes down, node I would locally adapt and start using the (dotted) overlay link I–D that uses the underlying DIF *C*. Thus, RINA provides finer grained control over routing to multihomed destinations.

Upon mobility, a node (process) may need to join or leave a DIF through a registration or unregistration procedure [6].

In the remaining sections we present the necessary inter and intra DIF operations, such as, registration, unregistration, node-address mapping, and routing, necessary to support mobility (and multihoming) in RINA. We then present an analytical model to compare the cost of supporting mobility/multihoming in RINA with that of other solutions, namely, MIP and LISP variants. Moreover, we validate our analysis using simulations.

3 Protocol Specification

In this section we present the specifications of the process of Routing in Recursive Architectures (RORA) adopted in RINA. Naturally, the RORA functions are *recursive*, whereby each function invocation (instance), in reality, represents processing at a certain DIF level.

In our specifications, we assume the existence of a data structure, which we refer to as RIB (Resource Information Base), in each DIF. Among other information, the RIB contains a set of pairs (n, a), where n is the node name, and a is its corresponding node address.

The RINA architecture consists of registration and unregistration phases to support the subscription and unsubscription of processes as they join and leave DIFs, respectively. RINA also requires translation / mapping functionalities and the actual recursive routing process.

3.1 Registration

In RINA, the registration process is done in a top-down fashion. As a node (process) moves from one DIF to another, it sends a registration request to a *registration* node located in that DIF. After being authenticated (a mechanism outside the scope of our discussion) [1],

[1] Security aspects of RINA are highlighted in [3].

the requester is assigned an address (line 5). The registration process is recursively propagated to the underlying DIFs to which lower-level processes on the same machine subscribe (line 8). During the registration process, and after allocating an address to the node, the DIF updates its RIB (line 6). Once the registration process is complete, a registration response (shown in Figure 6) is propagated upwards to each requesting DIF node (line 9).

Register_Request():
Require: n:String, l:String
1: **if** ($l == 0$) **then**
2: Return {End of recursion; reached bottom DIF}
3: **end if**
4: **if** authenticate(n, l) **then**
5: $a \Leftarrow$ allocate_address(n)
6: RIB_Update(n,a,"ADD")
7: **end if**
8: Register_Request(a,$l-1$) {Recursively register in the lower DIF}
9: Register_Response(n,a)

Fig. 5 RRegister_Request Recursive function.

Register_Response():
Require: n : *String*, a : *String* {respond with the allocated address information to the requesting node}

Fig. 6 Register_Response function.

Registration Example: We illustrate the registration process of RINA using the network shown in Figure 4. Assuming that the source wants to register. It starts by calling *Register_Request(source, l)* where l is the topmost level. Once it is authenticated, it will be assigned an address S, which will be registered recursively at layer $l-1$ and in turn assigned an address S_A.

3.2 Unregistration

Figure 7 highlights the unregistration process in RINA which is similar to the registration process. The node n issues an *UnRegister_Request* to a node in a DIF. The node that receives the unregistration request removes n from the RIB (line 6), and subsequently issues a recursive unregistration request (line 8) to the lower DIFs. Once this process is complete, an unregistration response (shown in Figure 8) is propagated upwards to each requesting DIF node (line 9).

Unregistration Example: We illustrate the unregistration process of RINA using the network shown in Figure 4. Assuming that the source wants to unregister. It starts by calling *UnRegister_Request(source, l)*

UnRegister_Request():
Require: n:String, l:String
1: **if** ($l == 0$) **then**
2: UnRegister_Response(n)
3: **end if**
4: **if** authenticate(n, l) **then**
5: $a \Leftarrow$ allocate_address(n)
6: RIB_Update(n,a,"REMOVE")
7: **end if**
8: UnRegister_Request(a,$l-1$) {Recursively unregister in the lower DIF}
9: UnRegister_Response(n)

Fig. 7 UnRegister_Request function.

UnRegister_Response():
Require: n:String {respond with the unregistration confirmation to the requesting node}

Fig. 8 UnRegister_Response function.

where l is the topmost level. Once authenticated, its address will be removed and the unregistration will be processed recursively until all processes on the same machine have unsubsribed from their respective DIF.

3.3 Mapping Functions

The mapping functions are a set of primitives that update and query a DIF's RIB data structure. They are called by a node to obtain a mapping between a name and its address.

Figure 9 highlights the *RIB_Update* method, which is called whenever node information needs to be added to or removed from the RIB. $RIB \cup (n, a)$ and $RIB \setminus (n, a)$ should work as any database add and remove function, respectively.

RIB_Update():
Require: n : *String*, a : *String*, *type* : *string*
if *type* == "*ADD*" **then**
RIB $\cup$ (n, a)
else
RIB $\setminus$ (n, a) {remove entry}
end if

Fig. 9 RIB_Update function.

The *Map_Request* function (Figure 10) queries for node n in the RIB, and when found, its address is returned.

The *Map_Response*(n, a) (Figure 11) function establishes a connection based on the node address, or signals an error (*e.g.* timeout, credential not found).

Map_Request()
Require: n : *String*
  if $\exists n \in$ RIB **then**
    $a \Leftarrow$ Find_address(n) {Returns the address of node n.}
    Map_Response(n,a)
  end if

Fig. 10 Map_Request function.

Map_Response()
Require: n : *String*, a : *String* {replies to the requesting node with either successful connection to node address or error}

Fig. 11 Map_Response function.

3.4 Recursive Routing

The recursive routing function (Figure 12) is considered the core of RORA, and requires a source and a destination node, s and d. We denote by i any intermediate node.

The function starts by recursively obtaining the source and next-hop / destination addresses using the *Map_Request* function (cf. Figure 13). Based on the routing policy adopted, the next-hop (or intermediate node) to the destination is obtained by calling $getNextHop(d)$. Whenever the message reaches its destination process at the lowest DIF, the message is decapsulated and delivered to the higher level DIF directly using the function *Deliver_up* (Figure 14). Whenever the message reaches its next-hop, it continues to be sent recursively down to its next-hop / destination. Eventually the message reaches its final destination and gets delivered to the destination application process using the function *Deliver_App*.

RRoute():
Require: s:String, d:String, m:String
1: **if** "me" == d **then**
2: Deliver_up(m) {If my address is the same as the destination, then deliver to the upper layer DIF}
3: **else**
4: i = getNextHop(d)
5: $m' = (s,d) \parallel m$ {add header}
6: send_down("me",i, m')
7: **end if**

Fig. 12 RRoute function.

Send_down():
Require: s:String, d:String, m:String
1: s' = Map_Request(s)
2: d' = Map_Request(d)
3: RRoute(s',d',m)

Fig. 13 Send_down function.

Deliver_up():
Require: m:String
1: **if** header(m) == ϕ **then**
2: Deliver_App()
3: return
4: **end if**
5: (s,d) = header(m)
6: $m = m \backslash (s, d)$ {decapsulate}
7: **if** "me" == d **then**
8: Deliver_up(m)
9: **end if**
10: RRoute(s,d,m)

Fig. 14 Deliver function.

Routing Example: We illustrate the routing process of RINA using the network shown in Figure 4. Assuming that source S wants to send a message to D. It starts by calling *RRoute(S,D,m)* where m is the message to be delivered. The function finds out the next-hop node (I in this case), and sends the message down to the lower-level DIF, which maps the source and next-hop addresses to their lower-level addresses and calls *RRoute* recursively. In particular, the lower layer process S_A will forward the message to I_A, which in turn will deliver it up to node (process) I. Node I will repeat the same process to send the message to D.

4 Cost Model

In this section we study the average (communication) cost of supporting mobility under MIP, LISP and RINA architectures. For the LISP architecture, we also analyze extended variants that employ caching for EID-to-RLOC mappings, or Mobile-IP running over basic LISP.

4.1 Assumptions, Cost Definitions, and Parameters

We assume a single source-destination model where the source sends data packets at a constant rate. We analyze the average cost of managing a single interface (path) change to the destination due to the mobility of the destination node.

The cost of delivery of a single packet is denoted by C_D. The total cost per interface change, denoted by C_T, is a function of the *location lookup* cost (C_L), the *location update* cost (C_U), and *location inconsistency* cost (C_I). *Location lookup* cost is defined only for LISP, to capture the cost of querying a mapping server for information about the destination's RLOC given the destination's EID. In computing the *location inconsistency* cost, we assume that packets delivered to the wrong location due to inconsistency of location / routing information, need to be delivered again.

In our model, we assume that the inter-arrival times of data packets and the lifetime of the destination's interface, each follows an exponential distribution, denoted by $f_p(t)$ and $f_m(t)$, respectively. We define the following two parameters:

- λ: the mean packet arrival rate, *i.e.*, $f_p(t) = \lambda e^{-\lambda t}$.
- μ: the rate at which the interface to the destination changes or mobility rate, *i.e.*, $f_m(t) = \mu e^{-\mu t}$.

Assuming that both packet arrival and interface lifetime processes are independent, the mean number of data packets received by the destination per single interface change is given by: $\rho = \frac{\lambda}{\mu}$.

We define P to be the probability that the source has the correct (*i.e.*, consistent) location / interface information. For example, under MIP, P defines the probability that the home router contains consistent routing / location information. Under LISP, P defines the probability that the Mapping Server contains correct EID-RLOC mapping information. Under RINA, P defines the probability that the DIF contains correct routing information.

In steady state, P can be defined as the probability that the interface to the destination has not changed since the last packet delivery. Let t_p be the exponential random variable representing the packet inter-arrival time, and t_m be the exponential random variable representing the residual time during which the interface to the destination node does not change[2]. Thus, we have:

$$P = Prob(t_p < t_m) \tag{1}$$

$$= \int_{t_p=0}^{\infty} f_p(t_p) \int_{t_m=t_p}^{\infty} f_m(t_m) dt_m dt_p \tag{2}$$

$$= \int_{t_p=0}^{\infty} \lambda e^{-\lambda t_p} \int_{t_m=t_p}^{\infty} \mu e^{-\mu t_m} dt_m dt_p \tag{3}$$

$$= \frac{\lambda}{\lambda + \mu} \tag{4}$$

The total cost per destination's interface change, C_T, is given by:

$$C_T = C_L + C_U + \rho(P \times C_D + (1 - P) \times C_I) \tag{5}$$

where C_I is defined as $(C_D + C_D^{\text{OLD}})$, and C_D^{OLD} is the cost of packet delivery to the old location / interface. Henceforth, we take $C_D^{\text{OLD}} = C_D$, assuming that packets delivered to the wrong location need to be redelivered to the correct location at the same cost.

Table 1 summarizes our parameters.

[2] Recall that the *residual* time of an exponentially distributed time is also exponential due to the memoryless property.

Parameters/Costs	Definitions
λ	sending rate of the source
μ	mobility rate of destination or rate of interface failure for multihomed destination
ρ	$\frac{\lambda}{\mu}$
C_L	Cost of lookup
C_U	Cost of location update
C_D	Cost of delivery
C_I	Cost of inconsistency

Table 1 Definitions of Parameters and Costs.

4.2 MIP Cost Analysis

For MIP, we define the cost terms in Equation (5) as follows:

- $C_D = C_{\text{SN-HR}} + C_{\text{HR-DN}}$,
  where the cost of delivery of a single packet, C_D, is the sum of $C_{\text{SN-HR}}$, representing the cost of delivering a packet from the source node (SN) to the home router (HR), and $C_{\text{HR-DN}}$, representing the cost of delivering the packet from HR to the destination node (DN).
- $C_U = C_{\text{DN-FR}} + C_{\text{FR-HR}}$,
  where the cost of updating the destination's interface / location is the sum of $C_{\text{DN-FR}}$, which represents the cost of updating the foreign router, and $C_{\text{FR-HR}}$, which represents the cost of updating the home router.

Note that in MIP, $C_L = 0$, since the home router readily maintains the location of the destination node, and does not look up any mapping service.

4.3 LISP Cost Analysis

Under LISP, we define the cost terms in Equation (5) as follows:

- $C_D = C_L + C_{\text{SN-DN}}$,
  where the lookup cost, C_L, represents the cost of querying the EID-RLOC Mapping Server (MS) to identify the location of the destination Tunnel Router (TR). This lookup cost is incorporated in the delivery cost of every single data packet.
- $C_U = C_{\text{DN-TR}} + C_{\text{TR-MS}}$,
  where C_U, the cost of updating the MS, is the sum of $C_{\text{DN-TR}}$, which represents the cost of location update from the destination node to its TR, and $C_{\text{TR-MS}}$, which represents the cost of updating the MS.

4.4 RINA Cost Analysis

Support for mobility is inherent in the RINA architecture [6]. As described earlier, a data packet is delivered *hop-by-hop* to the destination across limited-scope Distributed Inter-process communication Facilities (DIFs). If the destination's interface changes, then the mapping from the destination node's address to the new interface is locally propagated. This local update involves unsubscription / withdrawal from/of the old interface (underlying DIF), and subscription / registration to/of the new interface (underlying DIF), which in turn results in updating the routing information to map to the new interface.

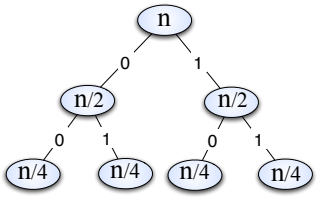

Fig. 15 RINA DIF Structure.

As described in Section 3, registration in RINA is done in a top-down fashion where a node registers at a higher level DIF and gets assigned an address, which in turn serves as the node name for the lower level DIF. Thus, a communication request for that destination name can be readily resolved at the lower level DIF to a node address at that level. This process is repeated recursively over all RINA DIFs.

For ease of analysis we define the DIF structure of RINA as a binary tree, where a tree node represents a network node that is subscribed to a DIF of size indicated in Figure 15, as well as to all lower level DIFs in its subtree of half the size each. Thus, to route over the scope of the whole network, say of size n nodes, routing can start from the root of the tree and proceed recursively down toward the lowest level DIF to which the destination is connected.

To assign addresses to nodes, we assign to each tree edge a binary value of zero or one. Each node gets assigned an address whose prefix is derived from the tree edge values. For example, a node that resides in the leftmost lowest level DIF gets allocated an address whose prefix is 00.

When a destination node moves from one lowest level DIF to another, routing along the tree gets updated to point to its current location. The cost of update is determined by the total number of nodes that will need to be updated as a result of a user's mobility. We define l as the level (height) of routing propagations up the tree, which is given by taking the exclusive-or (XOR) of the destination's current address prefix and its previous address prefix, and computing l as the position of the most significant (leftmost) bit being set to one (assuming the position of the least significant bit is 1).

The total cost for routing updates is equal to:

$$\sum_{j=0}^{l-1} 2 \times \frac{D}{2^{h-j}}$$

where D is the diameter of the network, and h is the height of the tree.

Example: Referring to Figure 15, assume that a node with address prefix 00 moves to the nearby lowest level DIF to the right, then the node address prefix changes to 01. In this case, 00 XOR 01 = 01, so $l = 1$, and the total update cost is equal to $2\frac{D}{2^2} = 2\frac{D}{4}$ (given the height of the tree $h = 2$). This is the case since the parent node (with address prefix 0) needs to update its routing to point toward the new lowest level DIF instead of the old DIF. This requires the propagation of routing update across two lowest level DIFs, each of which spans a delay equal to fourth the diameter delay across the whole network. Note that we further multiply the update cost by two to account for acknowledgements.

Since our analysis deals with average costs, our goal is to compute the average value of l over possible mobility between different lowest level DIFs. To this end, we define an event β_i such that given m bits, bit i is flipped and bit $i+1$ to m remain unchanged—in other words, β_i represents the probability of movement of a node that requires route updates to propagate up i levels, given a certain node mobility model. We also define the probability of bit i flipping as α_i. Thus, the probability of event $\beta_i = \alpha_i \prod_{j=i+1}^{m}(1-\alpha_j)$. The expected value of the level of route update propagations l is given by $E[l] = \sum_{i=1}^{m} i\beta_i$.

Thus under RINA, we define the cost terms in Equation (5) as follows:

- $C_D = C_{\mathrm{SN-DN}}$,
  since RINA strives to maintain a "direct" route to the destination.
- $C_U = \sum_{j=0}^{E[l]-1} 2 \times \frac{D}{2^{h-j}}$,
  which is the cost of routing updates upon mobility of the destination node.

As in MIP, $C_L = 0$ since each node (process) readily maintains the next-hop (routing) information to the destination node, and does not look up any mapping service.

4.5 LISP-MIP Cost Analysis

Farinacci *et al.* [9] propose the use of MIP as a means to managing fast mobility in LISP. This LISP-MIP variant can be generally used to deal with a change of destination's interface whether because of mobility or rerouting to a multihomed destination.

Figure 16 highlights the cost of message delivery under the LISP-MIP architecture. The source is sending a packet to the destination node that has already moved to another domain and got a new care-of-address and updated its home agent, following the MIP protocol. Once the home agent intercepts the message, it tunnels it to the new location. An additional lookup is needed to obtain the address of the current destination Tunnel Router (TR).

Thus under LISP-MIP, assuming no caching of location information, we define the cost terms in Equation (5) as follows:

- $C_D = C_{\text{SN}-\text{L}} + C_{\text{SN}-\text{HR}} + C_{\text{HR}-\text{L}} + C_{\text{HR}-\text{DN}}$, where $C_{\text{SN}-\text{L}}$ and $C_{\text{HR}-\text{L}}$ represents the cost of querying the EID-RLOC mapping server at the source's TR, and at the destination's home TR, respectively.

The cost of update C_U in LISP-MIP is the same as that of MIP.

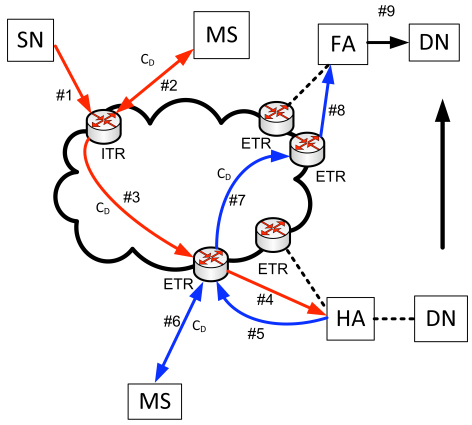

Fig. 16 LISP-MIP cost of packet delivery.

4.6 LISP-Cache

Iannone *et al.* [12] studied the use of caching at the source Tunnel Router (TR) under LISP. Naturally, caching would decrease the per-packet cost of looking up the EID-RLOC mapping information, as long as the cached location information is accurate. The packet delivery process is still the same as that of Figure 2 with the only difference being that the lookup is only done once per cache entry lifetime (which, we assume, corresponds to the expected inter-failure time of the destination's interface). Thus we define the cost terms in Equation (5) as follows:

- $C_{\text{L}} > 0$, which represents the cost of querying an EID-RLOC mapping server to identify the location of the destination TR. This lookup is done *once* whenever the destination's interface changes and then cached for subsequent data packets.
- $C_D = C_{\text{SN}-\text{DN}}$, where we assume that looking up the cache for the location information is negligible, and thus does not contribute to the cost of delivery of every single packet.
- $C_U = C_{\text{DN}-\text{TR}} + C_{\text{TR}-\text{SN}_{\text{cache}}}$, where $C_{\text{DN}-\text{TR}}$ represents the cost of location update from the DN to its TR, and $C_{\text{TR}-\text{SN}_{\text{cache}}}$ represents the cost of invalidating the source TR's cache due to the change in the destination's interface.

4.7 LISP-MIP-Cache

As a last LISP variant, we augment the LISP-MIP model described above with caching to reduce the cost of looking up location information. The delivery process still follows the same pattern as shown in Figure 16, the only difference is that the lookup is only done once per cache entry lifetime (which, we assume, corresponds to the expected inter-failure time of the destination's interface). We define the cost terms in Equation (5) as follows:

- $C_L = (C_{\text{SN}-\text{L}} + C_{\text{HR}-\text{L}}) > 0$, which represents the costs of querying a mapping server at the source's TR and the destination's home TR, respectively. We note that these lookup costs are only incurred once whenever the destination's interface changes. The location information is then cached for future use. Thus these lookup costs do not contribute to the delivery cost of every single data packet.
- $C_D = C_{\text{SN}-\text{HR}} + C_{\text{HR}-\text{DN}}$, which defines the cost of delivery of a single packet. The cost of looking up the cached location information is assumed to be negligible.
- $C_U = C_{\text{DN}-\text{FR}} + C_{\text{FR}-\text{HR}}$, which defines the cost of updating the destination's location at its home router.

A summary of the costs under all schemes is shown in Table 2.

Costs	Mobile IP	RINA	LISP	LISP-Cache	LISP-MIP	LISP-MIP-Cache
C_D	$C_{\mathrm{SN-HR}} + C_{\mathrm{HR-DN}}$	$C_{\mathrm{SN-DN}}$	$C_L + C_{\mathrm{SN-DN}}$	$C_{\mathrm{SN-DN}}$	$C_{\mathrm{SN-L}} + C_{\mathrm{SN-HR}} + C_{\mathrm{HR-L}} + C_{\mathrm{HR-DN}}$	$C_{\mathrm{SN-HR}} + C_{\mathrm{HR-DN}}$
C_U	$C_{\mathrm{DN-FR}} + C_{\mathrm{FR-HR}}$	$C_{\mathrm{DIF-UNREG}} + C_{\mathrm{DIF-REG}}$	$C_{\mathrm{DN-TR}} + C_{\mathrm{TR-MS}}$	$C_{\mathrm{DN-TR}} + C_{\mathrm{TR-SN_{CACHE}}}$	$C_{\mathrm{DN-FR}} + C_{\mathrm{FR-HR}}$	$C_{\mathrm{DN-FR}} + C_{\mathrm{FR-HR}}$
C_I	$C_D + C_D^{\mathrm{OLD}}$	$C_D + C_D^{\mathrm{OLD}}$	$C_D + C_D^{\mathrm{OLD}}$	$C_D + C_D^{\mathrm{OLD}}$	$C_D + C_D^{\mathrm{OLD}}$	$C_D + C_D^{\mathrm{OLD}}$
C_L	0	0	$C_{\mathrm{TR-MS}} + C_{\mathrm{MS-TR}}$	$C_{\mathrm{TR-MS}} + C_{\mathrm{MS-TR}}$	$2(C_{\mathrm{TR-MS}} + C_{\mathrm{MS-TR}})$	$2(C_{\mathrm{TR-MS}} + C_{\mathrm{MS-TR}})$

Table 2 Components of total cost in response to a single interface change.

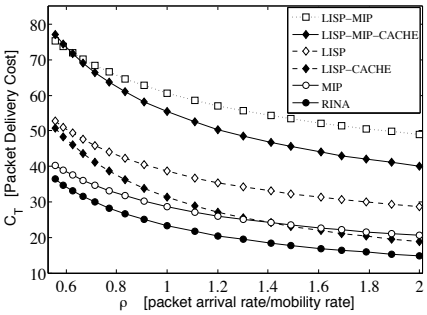

Fig. 17 Numerical results for an 8×8 grid.

5 Numerical Results

We present numerical results using the cost equations defined above for grid topologies. As mentioned earlier, we define costs in terms of average path lengths between communicating entities, *e.g.*, between a source's TR and a mapping server in LISP.

For an $N \times N$ grid topology, the average distance between any two nodes is given by $1.333(N/2)$ hops. We use this average distance as the cost of communication between two nodes that are *not* on the same network. On the other hand, if the communicating nodes are on the same network, the cost is relatively smaller (and independent of the size of the topology) — we take the cost to be two hops between a node and its TR, and one hop otherwise. For RINA we model a binary DIF tree on top of the grid topology such that each leaf (lowest level) DIF contains two network nodes.

Figure 17 presents results for an 8×8 grid for the various schemes as ρ takes on different values. The height of our RINA binary tree is 5.

We assume a skewed probability distribution for the movement of nodes between (lowest level) DIFs such that the probability of moving from the leftmost DIF to the rightmost DIF is minimum — the probability of address bit i being flipped is $1/2^i$ (cf. Section 4.4).This is a reasonable assumption since non-local movements would not be practical in reality. Given the above mobility distribution, $E[l] \approx 3$.

As ρ increases, the total cost for all schemes decreases (as expected). RINA has the lowest total cost, while LISP has the worst cost. It is worthwhile to mention that the total cost of location update in RINA is higher than that of MIP, but due to the "direct" path to the destination, RINA's total cost of packet delivery is lower.

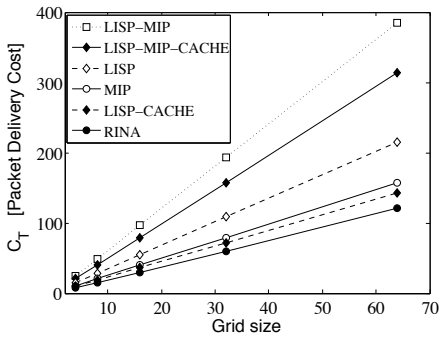

Fig. 18 Numerical results for varying grid sizes.

Figure 18 shows the total costs of the various schemes for varying grid sizes N for $\rho = 2$. As N increases, the total cost for all schemes increases, with RINA incurring the lowest cost at a sublinear increase rate.

6 Mobility Simulation Results

We validate our cost model using simulation. In our simulations, "cost" is represented by average packet delay or inverse of packet delivery ratio. To obtain an internet-like topology, we use the BRITE topology generator [15] to generate a network of autonomous systems (ASes) and their router topologies. We use the top-down generation model of BRITE which is based on two phases. In the first phase, an AS topology is initially generated using the Barabasi-Albert model with

incremental growth type and preferential connectivity. In the second phase, a router-level topology is generated for each AS, where router nodes are placed randomly on the 2D-plane and connected using the Waxman model. The average path length between nodes in the generated topologies is 14 hops, consistent with Internet measurement studies [19].

We simulate a single source-destination pair where the source sends packets at a rate λ while μ defines the rate at which the destination interface changes as a result of node mobility. We adopt a random walk mobility model where the destination node moves within a specified hop radius from its current location. For MIP, we assume that the cost of update is the round-trip propagation delay between the mobile destination node's current location and its designated home router. For LISP, we assume that updating the EID-RLOC mapping server takes an exponentially distributed time with a mean value that corresponds to the average path length, upper bounded by the network diameter. For simplicity, we assume the delay of one hop is 1 ms.

For RINA, we assume a two-level hierarchy where at the AS level, border routers form the higher level DIF, whereas internal routers of each AS constitute the lower layer DIFs. We simulate *hop-by-hop* routing in RINA, and at the higher level DIF, whenever the destination's interface changes due to mobility, we calculate the cost of updating the "intermediary" leading to the destination to be the round-trip propagation delay between them. If there is no path to the destination from the "intermediary", we assume the source needs to be updated to route to a new "intermediary" leading to the destination. The cost of updating the source is calculated as the round-trip propagation delay between the source and the destination.

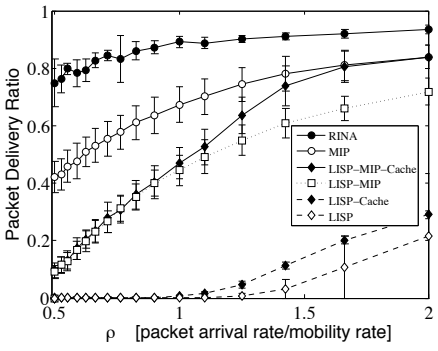

Fig. 19 Packet Delivery Ratio.

Figure 19 and 20 show the packet delivery ratio and the average packet delivery time under the various approaches. All results are presented with 90 percent confidence intervals. The results are consistent with our analytical results. RINA yields the lowest cost in terms of packet drop ratio, delivering packets at the lowest possible delay due to its local routing adaptation within the scope of the lower level DIFs connecting the intermediary and destination. LISP-MIP has higher packet delivery ratio compared to LISP, but higher average packet delivery delay.

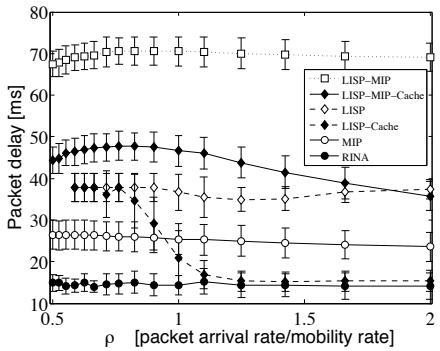

Fig. 20 Average Packet Delivery Time.

As the mobility rate decreases, approaches that utilize caching like caching over LISP and caching over LISP-MIP gain a significant advantage over non cached approaches. In Figure 19, and contrary to our analytical results, MIP and LISP-MIP perform better than LISP in terms of packet delivery ratio. This is due to the fact that the communication between the source node and the home router does not suffer any losses, which leads to better packet delivery ratio. However, MIP and LISP-MIP do incur a higher packet delay, which is consistent with our cost model.

7 Multihoming Trace-driven Simulation

In this section we validate our analytical results using trace-driven simulation based on CAIDA's anonymized packet traces [29]. This simulation considers only multihoming, so we do not include experimental results for Mobile-IP. We select two datasets from two Equinix locations: Chicago and San Jose (dated 20090219-045912 and 20090219-060100, respectively). The traces consist of anonymized tcpdump packets from different source-destination pairs. Each trace file contains more than a million records. Since the traces provide only source-destination pairs and packet arrival times, we use the BRITE topology generator [15] to generate an underlying AS and router network topology. Due to unavailability of real Internet topologies, and the difficulty of mapping the packets to any real topology since they are anonymized, we generate a topology using BRITE in the same way described in Section 6.

To keep the simulation and generated topologies manageable, we only consider the first 74123 packets from each packet trace, and make the simplifying assumption that all IP addresses which have a common 16-bit prefix belong to the same AS. Table 3 highlights properties of our two simulated topologies.

DataSet	Chicago	San Jose
Unique ASes	66	97
Packets	74123	74123
Nodes	2178	2425
Edges	4488	5041

Table 3 Topology Properties.

We utilize the packet timestamp as the packet arrival time. Furthermore, the time between link failures follows an exponential distribution. To simplify our simulation model, we assume that a single link (interface) fails at a time. We also make sure that interface failures occur only on destinations that are multihomed.

7.1 Results

In this section, we present the results of our simulations. We measure packet drop ratio, and packet delivery delay. We experimented using topologies generated using BRITE's top-down approach, where in the initial phase, the AS topology is generated using either the Barabasi-Albert (BA) model or Waxman's model [30]. Figures 21 and 22 show the results of packet drop ratio using simulations based on the two datasets. The results confirm our analytical model. RINA drops around 2% and 2.5% of the packets, respectively, while BGP, LISP, and LISP with caching, drop around 4% and 8% of the packets, respectively.

Figures 23 and 24 show the average packet delivery time. The delivery time of RINA and BGP is smaller due to the fact that there is no need to contact a mapping server. The benefit of caching for LISP is highlighted by smaller average packet delivery time.

Note that BGP's delay is slightly lower than that of RINA, since BGP's lack of fine-grained routing control makes it incapable of adapting to a failure of the shortest path to a specific destination node, however, for those packets that get delivered when the shortest path is up, their delivery delay is smallest. On the other hand, RINA enables the construction of an "overlay" network between the source node, destination node, and an intermediate node (intermediary) that is capable of re-routing around failed paths (interfaces). Thus, under RINA, more packets are successfully delivered, but

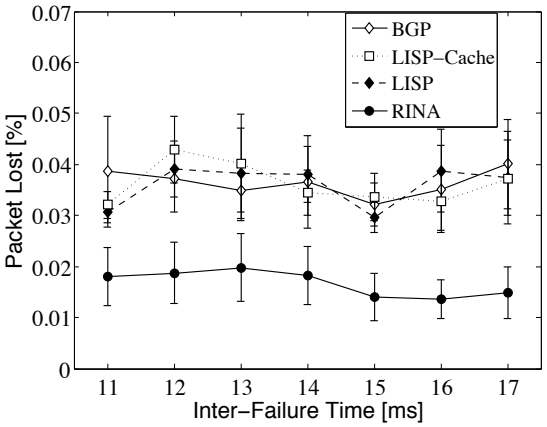

Fig. 21 Packet Drop Ratio (Chicago dataset, Waxman AS-topology).

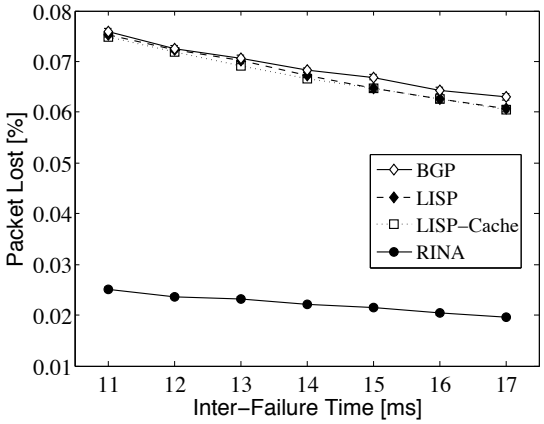

Fig. 22 Packet Drop Ratio (San Jose dataset, Waxman AS-topology).

those packets taking alternate paths when the primary paths are down, experience slightly higher delay.

We also observe that under LISP, the delay is almost double that of RINA and BGP, since LISP requires a mapping lookup which adds extra delay that is in the order of the average path length of around 14 hops (msec) in our topologies.

Results for the BA-generated AS-topology shown in Figures 25 and 26 for the San Jose dataset, and in Figures 27 and 28 for the Chicago dataset, are consistent with our Waxman AS-topology results. RINA yields the lowest cost in terms of packet drop ratio, delivering packets at the lowest possible delay due to its local routing adaptation within the scope of the overlay involving the source, destination, and "intermediary" node.

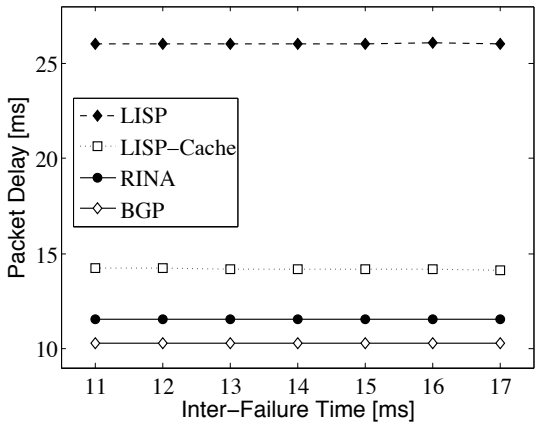

Fig. 23 Average Packet Delivery Time (Chicago dataset, Waxman AS-topology).

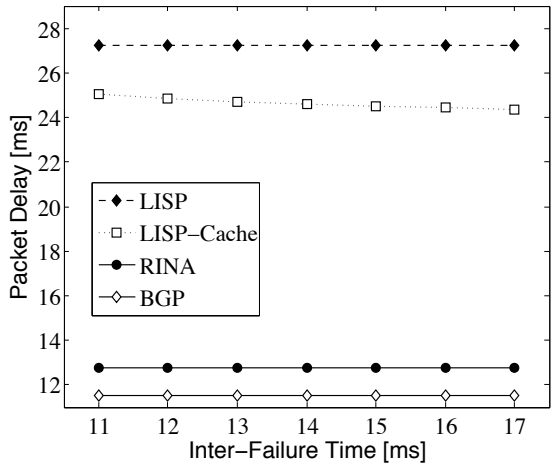

Fig. 24 Average Packet Delivery Time (San Jose dataset, Waxman AS-topology).

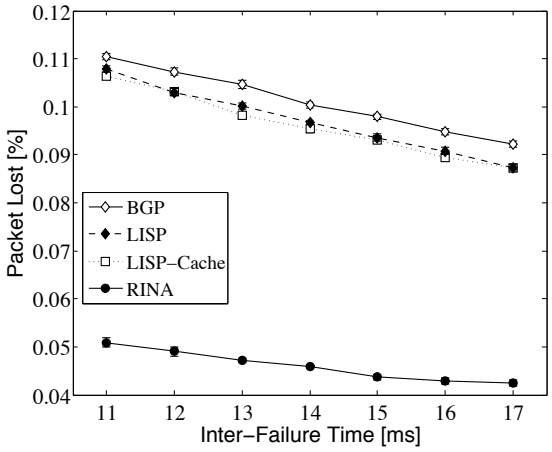

Fig. 25 Packet Drop Ratio (San Jose dataset, BA AS-topology).

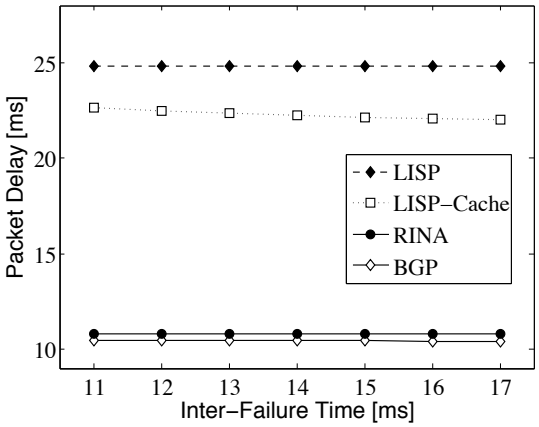

Fig. 26 Average Packet Delivery Time (San Jose dataset, BA AS-topology).

8 Related Work

8.1 Architectural Changes

File transfer and email were the main applications when, in the 70s the Internet protocols were designed. The number of connected hosts have grown from less than 200 in 1980, to 570 million in 2008 [24]. Experience and technological progress that would make a redesign of the Internet nowadays substantially different, together with the deficiencies of the current architecture are motivating research efforts in how the Internet architecture should be. Such research efforts could be classified along two main dimensions:

Approach: Purist versus pluralist. The former supports flexibility to meet current and future application requirements, and the latter envisions parallel protocol stacks able to cope with multiple service requirements. The approach envisioned in RINA is classifiable as both pluralist and purist. This is because we can flexibly compose a diverse set of end-to-end services across several underlying DIFs, where each DIF may run a different set of policies to meet certain local service requirements.

Design: Evolutionary versus clean-slate [23]. For many years, extensive research have been conducted to overcome the Internet impasse, with improvements and patches that would coexist within the design constraints of the current architectures, see *e.g.*, all the efforts on overlays mostly inspired by [1] and [28]. On the other hand, clean-slate approaches ignore such constraints to exploit potential benefits [4,31,7].

Unlike evolutionary approaches, our RINA architecture is a clean-slate design based on the inter-process communication (IPC) principle. Quoting Robert Metcalfe: *" Networking is inter-process communication and*

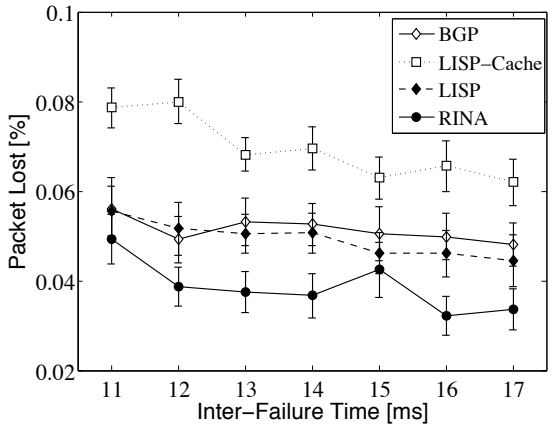

Fig. 27 Packet Drop Ratio (Chicago dataset, BA AS-topology).

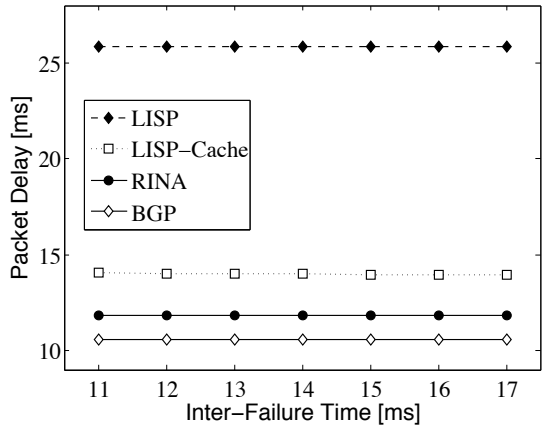

Fig. 28 Average Packet Delivery Time (Chicago dataset, BA AS-topology).

only inter-process communication." (1972) In this view, all network services, *e.g.* transport and internetworking tasks, together constitute a Distributed IPC Facility (DIF) to application processes. RINA applies this concept recursively, whereby the processes that make up a DIF can themselves act as application processes to request services from lower level DIFs.

Recursion has been recently promoted in network architectures, but to the best of our knowledge, this has been limited to tentative proposals of repeated functions of existing layers, and how one may either reduce duplication or create a meta-function (e.g., error and flow control) that could be re-used in many layers, *e.g.*, Touch *et al* [27]. Independently, we have pursued a general theory to identify patterns in network architecture [5,6], which the RINA architecture embodies.

8.2 Multihoming and Mobility

We are certainly not the first to advocate the need for new support for mobility [13]. In the current Internet, a system is identified by its IP / Internet address. As a result, when a system changes its point-of-attachment, its IP address changes. This makes reaching mobile systems difficult.

Multiple efforts have attempted to address this naming / addressing problem, proposing and deploying new mobility and multihoming protocols and architectures, including Mobile IP [20], Mobcast [14], a system based on a *Proxy IP Anycast Service* (PIAS), Internet Indirection Infrastructure [26], Host Identity Protocol [17], and others [2,31].

Those attempts may be classified as network or routing-oriented (*e.g.* [9,2,31]), host-centric ([17,20,18]), and hybrid edge-based solutions ([10]).

An example of network-oriented solution is the so-called LISP: location / identifier split [9]. LISP uses the locator not to locate the destination node (*i.e.*, where it is), rather a path that leads to it (*i.e.*, how to get there).

Host-centric solutions instead do routing at the end points of the network (hosts). IP addresses are given both ID and locator semantics. Route selection is done at the host where the ID is mapped to a particular locator, rather than letting routers select the best path. Shim6 [18] is an example of a host-centric solution that addresses multihoming.

The HAIR [10] architecture proposes to separate locators from identifiers in a hierarchical fashion. In fact, it can be seen as a hierarchical version of LISP. If a host moves across adjacent domains at the same hierarchical level, then routing updates do not necessarily have to propagate to the core of the Internet. Although HAIR addresses scalability by restricting the visibility of routing updates as in RINA, it suffers from the same drawbacks of network-based solutions, that is, routing based on interface (IP) addresses rather than node addresses. All these solutions in fact, bind host names to IP/anycast addresses, making it hard to utilize alternate paths in case the corresponding interface goes down.

By adopting and extending Saltzer's naming and addressing schema [25] in a recursive fashion, RINA names applications/nodes rather than interfaces (point-of-attachment). This late binding of a node's address to its interface allows RINA to effectively deal with interface changes due to multihoming or mobility.

9 Conclusion

We highlighted the benefits of ROuting in Recursive Architectures (RORA) as a model for the future internet. We developed a cost model to evaluate the mobility / multihoming support of RINA, LISP, and MIP. RINA incurs the lowest cost, while LISP incurs the highest cost. We also validated our model for both mobility and multihoming using simulation on an Internet-like topology and based on real packet traces from CAIDA. We are currently investigating dynamic DIF formation that optimizes routing in the RINA architecture in the presence of arbitrary node/link failures and mobility. We will also prototype RINA's recursive routing.

Acknowledgements The authors would like to thank members of the RINA team, in particular John Day and Karim Mattar, for their support and valuable feedback.

References

1. David G. Andersen, Hari Balakrishnan, Frans M. Kaashoek, and Robert Morris. Resilient Overlay Networks. In *Symposium on Operating Systems Principles*, pages 131–145, 2001.
2. Hari Balakrishnan, Karthik Lakshminarayanan, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Michael Walfish. A Layered Naming Architecture for the Internet. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 343–352, New York, NY, USA, 2004. ACM.
3. Gowtham Boddapati, John Day, Ibrahim Matta, and Lou Chitkushev. Assessing the Security of a Clean-Slate Internet Architecture. Technical Report BUCS-TR-2008-021, CS Dept, Boston U., June 22 2009.
4. David D. Clark, John Wroclawski, Karen R. Sollins, and Robert Braden. Tussle in Cyberspace: Defining Tomorrow's Internet. In *Procedeing of ACM SIGCOMM*, pages 347–356, 2002.
5. J. Day. *Patterns in Network Architecture: A Return to Fundamentals*. Prentice Hall, 2008.
6. John Day, Ibrahim Matta, and Karim Mattar. "Networking is IPC": A Guiding Principle to a Better Internet. In *Proceedings of ReArch*, Madrid, SPAIN, December 2008.
7. Christian Esteve, Fábio L. Verdi, and Maurício F. Magalháes. Towards a new Generation of Information-oriented Internetworking Architectures. In *Proceedings ACM CoNEXT Conference*, pages 1–6, New York, NY, USA, 2008. ACM.
8. D. Farinacci and V. Fuller. LISP Map Server. online, March 2009. http://tools.ietf.org/html/draft-fuller-lisp-ms-00.
9. D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. Locator/ID Separation Protocol (LISP), March 2009. http://tools.ietf.org/html/draft-farinacci-lisp-12.
10. Anja Feldmann, Luca Cittadini, Wolfgang Mühlbauer, Randy Bush, and Olaf Maennel. Hair: Hierarchical Architecture for Internet Routing. In *ReArch '09: Proceedings of the 2009 workshop on Re-architecting the internet*, pages 43–48, New York, NY, USA, 2009. ACM.
11. L Iannone, D Saucez, and Olivier Bonaventure. OpenLISP: An Open Source Implementation of the Locator/ID Separation Protocol. In *IEEE INFOCOM*, Demo paper, April 2009.
12. Luigi Iannone and Olivier Bonaventure. On the Cost of Caching Locator/ID Mappings. In *Proceedings of the ACM CoNEXT '07 conference*, pages 1–12, New York, NY, USA, 2007. ACM.
13. Deguang Le, Xiaoming Fu, and Dieter Hogrefe. A Review of Mobility Support Paradigms for the Internet. *IEEE Communications Surveys and Tutorials*, 8:2–15, 2006.
14. C.P. Lee, K. Attrey, C. Caballero, N. Feamster, M. Mihail, and J.A. Copeland. MobCast: Overlay Architecture for Seamless IP Mobility using Scalable Anycast Proxies. In *IEEE Wireless Communications and Networking Conference (WCNC).*, pages 3872 –3876, mar. 2007.
15. A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An Approach to Universal Topology Generation. In *Proceedings of MASCOTS*, 2001.
16. David Meyer. *The Locator/Identifier Separation Protocol (LISP)*, 2008.
17. R. Moskowitz and P. Nikander. Host Identity Protocol Architecture. In *Internet Draft, draft-ietf-hip-arch-03*, page 24, 2005.
18. E. Nordmark and M. Bagnulo. Shim6: Level 3 Multihoming Shim Protocol for IPv6. RFC 5533, Internet Engineering Task Force, June 2009.
19. Vern Paxson. End-to-end Routing Behavior in the Internet. *SIGCOMM Comput. Commun. Rev.*, 36(5):41–56, 2006.
20. Charles Perkins. IP Mobility Support for IPv4. RFC 3344, Internet Engineering Task Force, August 2002.
21. Bruno Quoitin, Luigi Iannone, Cédric de Launois, and Olivier Bonaventure. Evaluating the Benefits of the Locator/Identifier Separation. In *MobiArch '07: Proceedings of 2^{nd} ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture*, pages 1–6, New York, NY, USA, 2007. ACM.
22. Y. Rekhter, T. Li, S. Hares, et al. A Border Gateway Protocol 4 (BGP-4), 1995.
23. Jennifer Rexford and Constantine Dovrolis. Future Internet Architecture: Clean-slate versus Evolutionary Research. *Commun. ACM*, 53(9):36–40, 2010.
24. J. Roberts. The Clean-slate Approach to Future Internet Design: a Survey of Research Initiatives. *Annals of Telecommunications*, 64:271–276, 2009.
25. J. Saltzer. Naming and Binding of Objects. In R. Bayer, editor, *Operating Systems, Lecture notes in Computer Science*, volume 60. Springer-Verlag, New York, 1978.
26. Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. Internet Indirection Infrastructure. In *In Proceedings of ACM SIGCOMM*, pages 73–86, 2002.
27. J. Touch, Y-S. Wang, and V. Pingali. A Recursive Network Architecture. Technical report, USC/ISI, October 2006.
28. Joe Touch and Steve Hotz. The X-Bone. *Proceedings of Global Internet Mini-Conference Globecom*, 1998.
29. Colby Walsworth, Emile Aben, kc Claffy, and Dan Andersen. The CAIDA Anonymized 2009 Internet Traces, 2009. www.caida.org/data/passive/passive_2009_dataset.xml.
30. B.M. Waxman. Routing of multipoint connections. *Selected Areas in Communications, IEEE Journal on*, 6(9):1617 – 1622, dec. 1988.
31. Xiaohu Xu, Dayong Guo, Raj Jain, Jianli Pan, and Subharthi Paul. Internet 3.0: Next Generation Internet, 2008.