Seamless network renumbering in RINA: automate address changes without breaking flows!

Eduard Grasa*, Leonardo Bergesio*, Miguel Tarzan*, Diego Lopez†, John Day§ and Lou Chitkushev§
*Internet Architecture and Services, Fundacio i2CAT, {eduard.grasa, leonardo.bergesio, miquel.tarzan}@i2cat.net
†Telefonica Investigation y Desarrollo S.A., diego.r.lopez@telefonica.com
§Computer Science Department, Metropolitan College, Boston University, {day, ltc}@bu.edu

***Abstract*—Network renumbering in the IP world is a complicated and expensive procedure that has to be carefully planned and executed to avoid routing, security (firewall, ACLs) and transport connection integrity problems. The source of most of these issues is in the lack of a complete naming and addressing architecture in the TCP/IP protocol suite. This paper analyses the issues related to IP networks renumbering, identifying its root causes. Then it looks into how these issues affect renumbering in networks based on RINA, a network architecture with a complete naming scheme. Theoretical analysis backed up by experimentation results indicate that renumbering in RINA networks not only is seamless (can be done without impacting existing flows) but also does not require any special mechanisms.**

I. Introduction and motivation

Most real networks have to be eventually renumbered [1]: a subset or all the addresses assigned to network entities must be updated. It may be happen that the network has grown to the point that its current addressing plan is no longer effective or does not scale. Or perhaps the network is changing upstream providers and must get new addresses from the new provider (provider-based addresses are the norm in the current Internet). Whatever the reason, renumbering in IP networks is a complex procedure involving a number of steps: IP addresses need to be assigned to interfaces on switches and routers, routing information must be propagated, ingress and egress filters must be updated - as well as firewalls and access control lists -, hosts must get new addresses and DNS entries have to be updated.

An overview of the problems associated to renumbering of IP networks is provided in [2]. Since TCP and UDP connections are tightly bound to a pair of IP addresses, changing any of them will destroy the flow. Since DNS is an external directory - not part of the network layers - the renumbering process usually leads to stale DNS entries pointing to deprecated addresses. Even worst, applications may operate through the direct use of IP addresses, which will require an update in the application code, its configuration or both. Router renumbering usually requires an exhaustive manual and error-prone procedure for updating control plane Access Control Lists or firewall rules. Moreover, static IP addresses are usually embedded in numerous configuration files and network management databases [3].

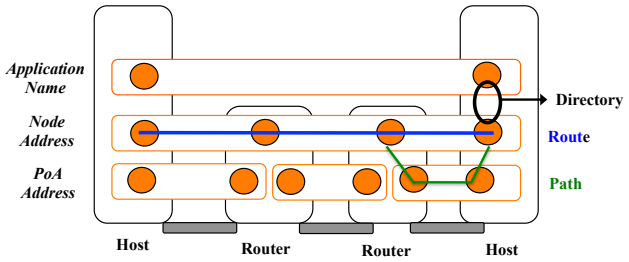

Fig. 1. Elements of a complete naming and addressing scheme

Most if not all of the issues described are rooted in the incomplete naming and addressing architecture of IP networks. In 1982 Saltzer [4] described the elements of a complete naming architecture for computer networks as well as their properties: application names that are location independent so that applications can move, node addresses that are location dependent but route independent to facilitate routing, and names for points of attachment to the network. Application names are mapped to node addresses via a directory, as illustrated in Figure 1. In the IP world there are no application names - domain names are synonyms for IP addresses resolved outside of the network layer - and end to end communication flows are created between transport layer endpoints identified by an IP address and a transport port number. If flow requests to the network were based on application names and the network internally resolved those names to network addresses renumbering would be completely transparent to applications.

The issues on ACLs and firewalls, which use rules based on IP addresses, have a similar origin. Since the IP address is both the identity of protocol machines (nodes) in the IP layer and also the name used for forwarding IP packets, there is an issue if the network is renumbered: ACL and firewall rules have to be updated to reflect the new address assignment. Obviating the fact that in well-formed architectures firewalls are not necessary [5], we can generalise the problem to that of setting access control rules in the IP layer. There is clearly a need for a stable, location-independent name that identifies the node (protocol machine) in the layer, and another location-dependent name that is used for forwarding packets between protocol machines (the addresses). In this scenario access control rules can be written in terms of the location-

978–1–5386–3873–6/17/$31.00

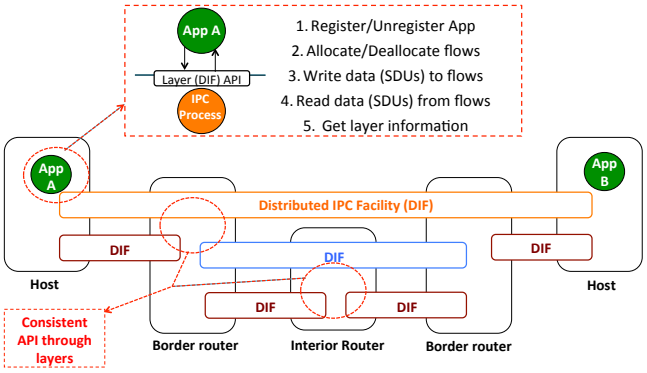

Fig. 2. Layers in RINA (down), and grouping of functions of protocol machine in the layer(IPC Process, up)

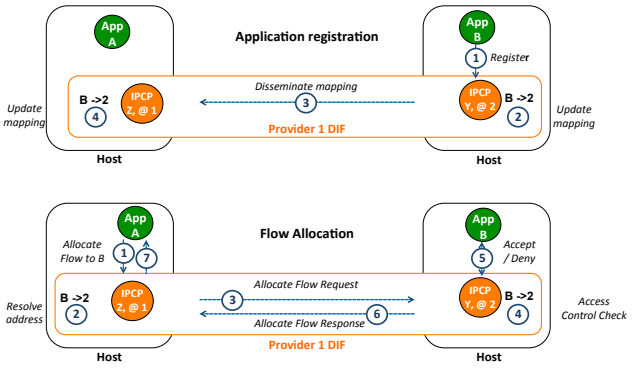

Fig. 3. Application registration in DIF (up); allocation of flow in DIF (down)

independent names; if addresses change access control rules do not need to be updated. Similar considerations apply to management-related problems: if the system being managed is identified by a location independent (management) application name, all problems related to stale addresses in configuration files and network management databases are just avoided.

The rest of the paper is structured as follows. In section II we discuss the RINA design principles that are relevant to naming and addressing and facilitate achieving seamless renumbering. In Section III we make a functional analysis of the renumbering procedure in RINA networks. In Section IV we describe the changes performed to the IRATI RINA implementation to support dynamic network renumbering; in section V we describe experimental scenarios and present results and finally Section VI provides concluding remarks and discusses future work.

II. Can renumbering cause issues in RINA networks?

RINA [6], [7] the Recursive InterNetwork Architecture, is a fundamental network architecture that relies on the premise that networking is nothing more and nothing less than distributed Inter Process Communication (IPC). RINA decomposes networks into layers of generic protocols that can be configured via policies to optimally serve their operational requirements. As seen in Figure 2, in RINA there is a single type of layer - called a DIF, Distributed IPC Facility - that repeats as many times as needed by the network designer.

A DIF itself is just a distributed application that performs and manages IPC. The application processes that are members of a DIF are called IPC Processes (IPCPs). As described by Saltzer, within the RINA framework application processes (APs) are assigned names that are location-independent. In order for an AP to be reachable via a certain DIF, it has to register to the DIF. Registration creates a local binding between the AP and the IPC Processes it is registering to. This binding is disseminated through the DIF via a DIF directory update that maps the registered AP name to the address of the IPCP through which it is available.

Figure 3 provides an example of this procedure. AP named *B* registers to the DIF called *Provider 1*. IPCP *Y* is the DIF representative at the system where AP *B* is executing. Upon receiving the registration request the IPCP updates its internal directory map and disseminates the new registration information through the DIF. Eventually IPCP *Z* learns this information and updates its internal directory map. Note that the procedures for maintaining the distributed directory in the DIF are a policy: they change from DIF to DIF, depending on its operational environment. Small DIFs may use fully replicated directories as the one described in this example or exhaustive search such as ARP (the Address Resolution Protocol), larger DIFs may use partially replicated directories with structure (hierarchical maps similar to DNS, the Domain Name system) or without it (Distributed Hash Tables [8]).

When an AP wants to communicate to another one, it requests the allocation of a communication flow to the destination AP, just providing its name. The local IPCP that processes the request queries the DIF directory to resolve the address of the IPCP through which the destination AP is reachable, and forwards the flow allocation request to the resolved IPCP. The destination IPCP does an access control check and notifies the local AP, who has the last say in accepting or rejecting the flow. An example of this procedure is shown in the bottom part of 3. An AP called *A* requests a flow allocation to AP *B*. IPCP *Z* processes the allocation request, queries the directory and finds out that AP *B* is reachable via the IPCP whose address is *2*, therefore it sends a flow allocation request to this address. IPCP *Y* receives the flow allocation request, notifies AP *B*, who accepts the flow, and sends a reply to IPCP *Z*, who notifies AP *A* that the flow is now allocated.

Notice that in all this procedure the addresses of the IPCP are never exposed outside of the DIF, therefore if IPCP addresses change the ability to create new flows is not compromised and the identity of old flows is not lost. Moreover, since DIFs are also distributed applications - which means IPCPs are just application processes - the naming problems of IPCPs at each layer are also solved. IPCPs have a stable location-independent AP name that use as their identity, while they also have one ore more temporary location-dependent synonyms (addresses) that are used for forwarding the PDUs of the DIFs data transfer protocol. Addresses are temporary by design, therefore renumbering is just part of the normal lifecycle of the DIF, as it is explained in the next section.

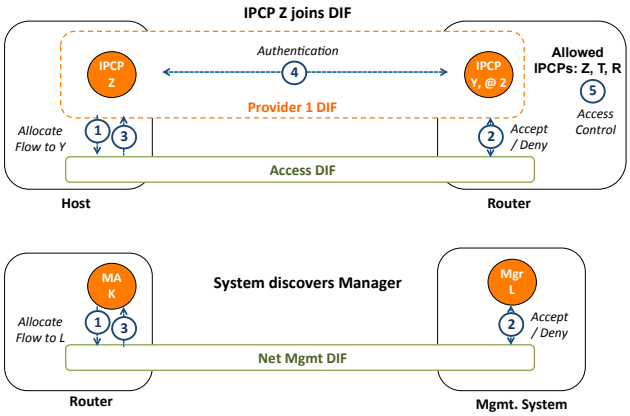

Fig. 4. IPCP joining a DIF (up); Network Manager contacting a managed system (down)

To better understand how this solves the problems related to access control rules and management when a DIF is renumbered we will describe an example using the scenario in Figure 4. The upper part of this Figure shows an scenario in which an IPCP called *Z* is joining the DIF named *Provider 1 DIF*. To do so, it requests the allocation of a flow to *Y* to a lower level DIF (in fact it would allocate a flow to the DIF name, but we will assume the flow is to *Y* to simplify the explanation). When the flow is allocated, IPCP *Z* authenticates to IPCP *Y*, using its application name. After authentication, IPCP *Y* checks if *Z* is allowed to join the DIF - here it could have access control rules based on the IPCP name, or something more elaborate. If the access control check is positive, IPCP *Z* joins and is initialised with enough information to start operating as a DIF member (including getting one or more addresses). The address of IPCP *Z* is never used for authentication or access control, in fact *Z* does not even have an address until it has joined the DIF.

The bottom part of Figure 4 illustrates an scenario in which a managed system (in this case a router) discovers its Manager. The Management Agent AP in the Router System requests a flow allocation to the Manager AP - called *L* in the Figure. Once the flow is allocated the Management Agent can authenticate to the Manager and start exchanging information with it. Again, no address of any IPCP is used in the scenario, just location-independent application names; therefore renumbering does not cause issues.

III. A WALK-THROUGH A RENUMBERING EVENT

When an IPCP is renumbered it obtains a new synonym that better represents the location of the IPCP within the graph of the DIF (facilitating aggregation and routing). Once the IPCP has acquired the new address it starts using it and deprecates the old one. In order to achieve these goals no new mechanisms are required, only normal DIF operations need to be used.

The first thing that needs to be done by the IPCP that obtained the new address is to start advertising it through the routing system, so that when the IPCP starts using the new address the other IPCPs in the DIF know how to forward PDUs to it. If the DIF is using link-state routing, this can be achieved by advertising new routing updates to the direct neighbours of the IPCP. Neighbors will update their link-state database and propagate the new advertisement to other IPCPs in the DIF. After a while, all the DIF members know how to forward PDUs to both the old and new address.

After a wait period the renumbered IPCP can start using the new address. This wait period may be very low or even zero depending on why and how the network is being renumbered. In the case of an IPC Process in a mobile node having moved too far away and getting a new address, it is possible and beneficial to start using the new one almost right away. Since the address is location-dependent, routing to the new address and to the old one will be the same until packets are close to the area where the IPCP is located. When that happens routing in the neighbourhood of the renamed IPC Process will have already converged (more experimentation is required to properly understand any limitations in this approach). In the mobility scenario it is beneficial to start using the new address as soon as possible because the path to the new address will have a higher probability of introducing less latency than the path to the old one. If a lot of IPC Processes change its addresses at approximately the same time but very infrequently (e.g. because the network has grown too much and needs a new addressing plan), then it may be necessary to define a guard time before start using the new address; depending on the addressing plan, routing policy or distribution of flows.

Going back to the details on how to start using the new address, if the renumbered IPCP is providing any flows to applications or higher-level DIFs it will issue flow update messages to the IPCPs at the other end of the flows; to notify them about the address change. Then the renumbered IPCP just needs to start using its new address as the source address of outgoing data transfer protocol PDUs. When the IPCPs at the other end of the flows receive the Flow update message, they will see that the address has changed, update the data transfer protocol state and start using this address as the destination address for outgoing data transfer protocol PDUs. If the renumbered IPCP has any local applications registered in the DIF, it will have to update its directory and disseminate it through the DIF according to the directory dissemination policy. This way new flow allocation requests to the applications registered in the renumbered IPCP will also use the new address.

After these procedures the new address is fully operational and in use. Upon the expiration of a timeout - set with a value that is high enough to guarantee that the old address is no longer used by other IPCPs in the DIF - the IPCP can free the old address. To do so it just needs to issue a routing update deprecating the old address. For example, if link state is the routing policy of the DIF, the IPCP will advertise all the link state objects containing the old address as deprecated to its neighbours, who will remove them from the flow state database and propagate the message to other IPCPs in the DIF. After that the old address will have disappeared from the

forwarding tables of the IPCPs in the DIF.

IV. Implementation in IRATI

In order to experimentally validate the behaviour of renumbering in a RINA network, we have used the IRATI [9] RINA implementation. Although RINA allows an IPC Process to have multiple addresses, the IRATI implementation was still not prepared to exploit this feature, therefore we extended the codebase to support it. IRATI is a Linux-based RINA implementation that splits the functionality of IPC Processes between the kernel and user spaces [10]. The fast path functions - IPCP data transfer and data transfer control - belong to the kernel, while the slow path functions - IPCP layer management (routing, resource allocation, enrolment, etc) - belong to user space. IRATI also facilitates programmability of IPC Process functions via a Software Development Kit that allows plugging in well-defined units of functionality called policies [11].

The authors have extended the IRATI codebase to support multiple addresses and the ability to change them, specifically: i) data structures that were assuming a single address per IPC Process have been updated; ii) the link-state routing policy has been adapted and iii) the synchronisation mechanisms between user-space and kernel IPC Process components have been extended. A new namespace management policy was implemented to trigger the IPCP address changes, as a quick way to control the renumbering periods in experiments. The Namespace Manager is the layer management task of the IPC Process that manages its naming and addressing. The policy is configured with an address range and an address change period. It then sleeps a random amount of time within the configured period and when it wakes up changes the address of the IPC Process. When the address change occurs, the Namespace Manager sends an event to the other tasks of the IPC Process, which react to it accordingly. In particular:

- The routing policy (link-state is the only one implemented to date) starts advertising the new address to its neighbors via routing updates. It sets a timer to deprecate the old address (part of the IPCP configuration).
- The core IPCP component updates its internal data structures and sends a message to the kernel to notify about the address change.
- The kernel updates the internal RMT (Relaying and Multiplexing Task) data structures (it will now accept PDUs whose destination address is the old one or the new one) and sets two timers: one to start using the new address and another one to deprecate the old address (the timers are part of the IPCP configuration).
- When the first timer fires (start using new address), the kernel modifies the data structures containing the data of active connections in order to start using the new address as the source address of all outgoing PDUs.
- When the second timer fires (deprecate old address), the kernel removes the old address from the RMT data structures, which causes the IPCP to no longer accept PDUs with the old address.

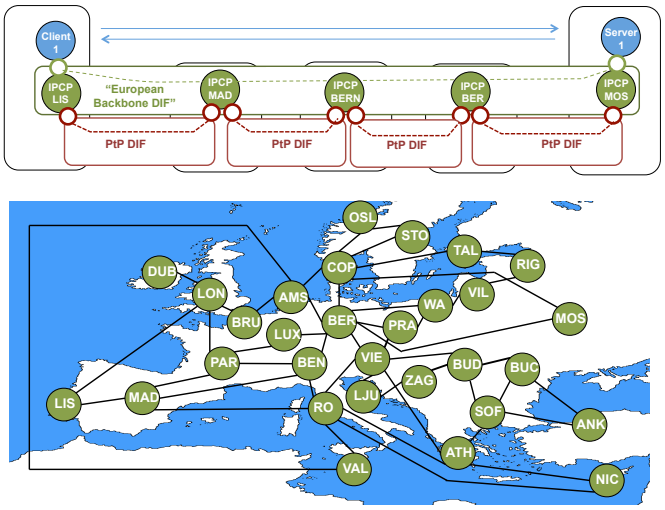

Fig. 5. Layer diagram of the single DIF Experiment (up); Connectivity graph of the European backbone DIF (down)

- The second timer will also fire in user-space, causing the link-state routing policy to advertise the old address as deprecated to its neighbours. The next time the PDU Forwarding table is computed all entries associated to the old address will be removed.

V. Experimentation and validation

We have conducted a set of experiments to validate the behaviour of renumbering in RINA networks using the IRATI implementation. In order to stress the network we have designed a scenario in which all IPC Processes of all DIFs involved in the experiment change their addresses periodically. This scenario does not match real use cases, which are quite less demanding: either only a subset of the IPC Processes change their addresses (in the case of mobile networks), or a significant number of the IPCPs in the network change its address but very infrequently (e.g. merging two networks after an acquisition). With this very demanding scenario we will be able to understand if the ability of the network to keep operating effectively is compromised due to the renumbering process. In addition to the IRATI implementation we will also be using the *Demonstrator* tool [12] to configure and setup the experimental scenario. The *Demonstrator* is a Python-based tool that automates the deployment and configuration of RINA networks in virtualised environments. Given an initial configuration file, it sets up a number of RINA-enabled Linux Virtual Machines (VMs), connects them through Ethernet software bridges in the physical machine, configures RINA in each VM and initiates the enrolment and neighbor discovery procedures in all DIFs.

The first experimental scenario is depicted in Figure 5. This scenario consists of a single DIF - called *European backbone DIF* - on top of several point-to-point Ethernet links. The goal of this experiment is to evaluate the degradation of performance experimented by different flows while renumbering is taking place in a single DIF in the network. The bottom part of Figure 5 shows the connectivity graph of

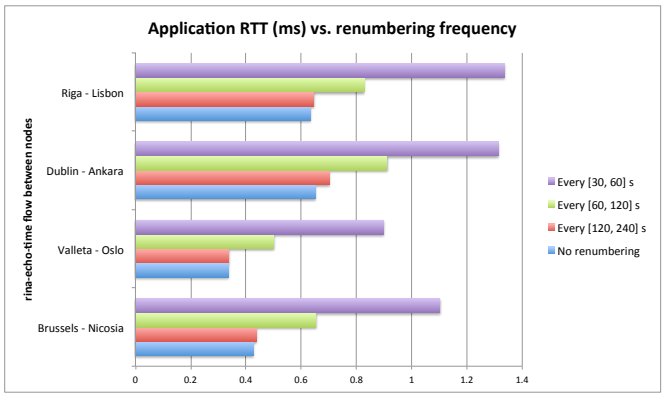

Fig. 6. Application RTT (ms) vs. renumbering frequency for flows in the European backbone DIF

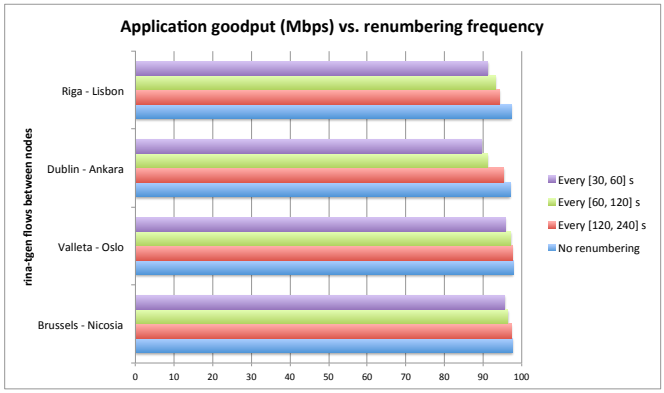

Fig. 7. Application goodput (Mbps) vs. renumbering frequency for flows in the European backbone DIF

the *European backbone DIF*, composed of 32 IPC Processes connected via 52 N-1 flows (which essentially wrap point to point Ethernet links). After instantiating the experiment with the *Demonstrator* tool, we setup four flows over this DIF (Riga-Lisbon, Dublin-Ankara, Valletta-Oslo and Brussels-Nicosia) and run the *rina-echo-time* application over them in order to measure the application round trip time and the data loss rates. *rina-echo-time* is configured to send 100.000 *Echo Requests*, waiting 2 ms between each request. Upon completing the execution of the four *rina-echo-time* instances, four new flows are setup, this time by the *rina-tgen* traffic generation application. *rina-tgen* reports the application goodput after sending data as fast as possible (limited by flow control carried out by the *European backbone DIF*) during 60 seconds. This experiment is repeated four times, for different rates of IPCP address change: i) no address change; ii) addresses change randomly every 30 to 60 seconds (high renumbering rate); iii) addresses change randomly every 60 to 120 seconds (mid renumbering rate) and iv) addresses change randomly every 120 to 240 seconds (low renumbering rate).

Figures 6 and 7 show how the RTT and goodput degrade for different rates of IPCP address change. Packet loss is not reported because it was 0 in all cases, therefore we can conclude that renumbering does not affect packet loss. The ability to create new flows is not impaired by the renumbering

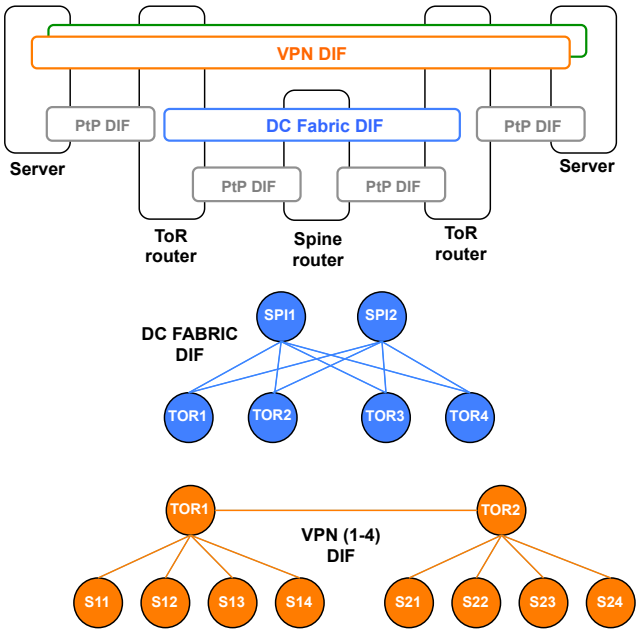

Fig. 8. Layer diagram of the Data Centre Experiment (up); Connectivity graph of the DC Fabric and VPNs (1-4) DIF (down)

procedures, since we created all flows in the experiment while renumbering was taking place. Figure 6 reports the round-trip delay perceived by the *rina-echo-time* application. It can be seen that the delay is approximately the same in the no-renumbering situation than in the *low renumbering rate situation*. However, delay increases specially in the *high renumbering situation*, in which the average round-trip delay is twice the no renumbering round-trip delay. Taking into consideration that there are 32 IPC Processes changing addresses every 45 seconds on average, it means that on average every 1.5 seconds at least one IPC Process in the DIF is changing addresses. Since all the DIF is a single link-state domain, it implies there is a forwarding table recomputation at least once per second. Our conjecture is that most of the extra delay is due to the non-optimised implementation of the fast-path functions in IRATI, which for example cannot keep forwarding data while the forwarding table is being updated. However, even if this was not the case, the performance degradation is quite acceptable given the extreme scenario in which it is occurring.

Figure 7 shows that application goodput is also reduced in the case of *mid* and *high* renumbering scenarios. This fact is a consequence of the degradation in delay; since the flows are flow-controlled the fact that flow control PDUs take more time to reach the EFCP senders causes a reduction in goodput. Again, given the non-optimal efficiency of the implementation and the high rate of address change events in both scenarios we think that the performance degradation is not too high.

The scenario for the second experiment is depicted in Figure 8. This experiment is designed to understand the effects of renumbering in performance when two DIFs on top of each other are being constantly renumbered at the same time. The scenario chosen is a small datacenter (DC) configuration, in

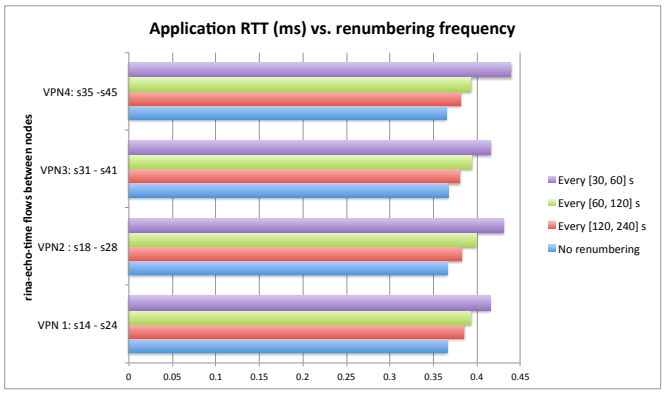

Fig. 9. Application RTT (ms) vs. renumbering frequency for flows in the different VPN DIFs

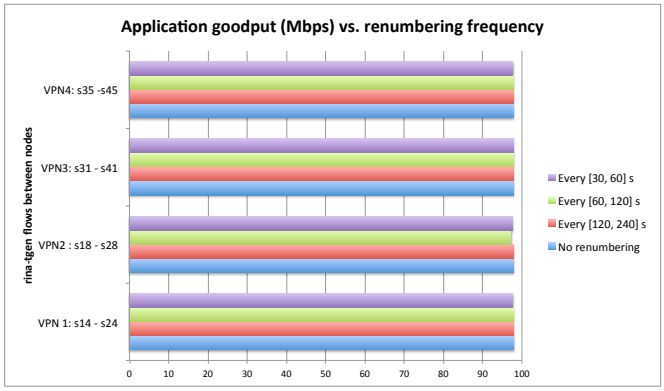

Fig. 10. Application goodput (Mbps) vs. renumbering frequency for flows in the different VPN DIFs

which 4 clusters of 8 servers each one are connected to each other via a leaf-spine fabric composed of 4 Top of Rack (ToR) routers and 2 spine routers. The upper part of Figure 8 depicts the layer diagram of the network. The DC fabric DIF connects together the ToRs and spines, supporting the connectivity and performance requirements of multiple VPN or tenant DIFs. Each tenant DIF provides an isolated connectivity domain to a different customer of the DC. In the experiment there are 4 VPN DIFs, each one connecting 8 servers together. We perform the same actions as in the previous experiment, the differences being that this time we start a separate flow in each of the four VPN DIFs and that renumbering is going on in both the VPN DIFs and the DC Fabric DIFs.

Figures 9 and 10 show how the RTT and goodput degrade for different rates of IPCP address change. As in the first experiment there is no packet loss and new flows can be established while IPC Processes change addresses. This time performance degradation is much lower than in the first experiment, with minor increases in delay and almost no decrease in goodput. Even though there are more VMs in this scenario (38 instead of 32), there are less IPC Processes in each DIF, and application flows have to go through less number of hops compared to the first experiment - which hides a bit more the non-optimal implementation of the fast path. In any case we confirm that renumbering multiple layers at once does not disrupt the network, which can continue with normal operation even at high renumbering rates.

VI. Conclusions and Future Work

Shortcomings in the naming and addressing structure of the current Internet protocol suite make network renumbering a tedious, error-prone and expensive procedure. The lack of application names causes the network to bind an application flow to an IP address and a transport layer port number. If the IP address of the source or the destination of the flow changes, the flow identity is lost and the flow is no longer usable. In contrast the comprehensive naming scheme of RINA makes renumbering problems in IP networks non-issues and enables dynamic network renumbering. Flows are associations between application names, only locally bound to IPC Processes via a port-id. Addresses are just location-dependent synonyms of IPC Process names. The identity of IPC Processes is represented by their location-independent application name: authentication and access control operations are performed in terms of AP names, not IPCP addresses. Hence renumbering does not interfere with such procedures.

In this paper we have described how the complete naming and addressing architecture embodied by RINA allows RINA networks to be renumbered live, without significantly impacting the performance perceived by existing flows or impairing the ability to crate new ones. We plan to use this RINA feature to simplify routing and forwarding in DIFs with mobile nodes, as part of one of the experiments carried out within the context of the H2020 ARCFIRE project.

Acknowledgment

This work is partly funded by the European Commission through the H2020 ARCFIRE project (Grant 687871).

References

[1] F. Baker, E. Lear, and R. Droms, “Procedures for renumbering an ipv6 network without a flag day,” IETF Network Working Group. RFC 4192, September 2005.

[2] B. Carpenter, R. Atkinson, and H. Flinck, “Renumbering still needs work,” IETF RFC 5887, May 2010.

[3] D. Leroy and O. Bonaventure, “Preparing network configurations for ipv6 renumbering,” *International Journal of Network Management*, vol. 19, no. 5, pp. 415–426, September/October 2009.

[4] J. Saltzer, “On the naming and binding of network destinations,” *Local Computer Networks*, 1982.

[5] J. Small, “Threat analysis of recursive internetwork architecture distributed ipc facilities,” BU Technical report. Available online at http://pouzinsociety.org/research/publications, 2011.

[6] J. Day, I. Matta, and K. Mattar, “”Networking is IPC”: A Guiding Principle to a Better Internet,” in *Proceedings of the 2008 ACM CoNEXT Conference*, 2008.

[7] J. Day, *Patterns in network architecture: A return to fundamentals*. Pearson Education, 2007.

[8] T. Li, X. Zhou, K. Brandstatter, D. Zhao, K. Wang, A. Rajendran, Z. Zhang, and I. Raicu, “Zht: A light-weight reliable persistent dynamic scalable zero-hop distributed hash table,” *Parallel and Distributed Processing (IPDPS), IEEE 27th International Symposium on*, 2013.

[9] “Irati rina implementation source code,” https://github.com/IRATI/stack.

[10] S. Vrijders, D. Staessens, D. Colle, F. Salvestrini, E. Grasa, M. Tarzan, and L. Bergesio, “Prototyping the recursive internetwork architecture: The irati project approach,” *IEEE Network*, vol. 28, no. 2, 2014.

[11] V. Maffione, E. Grasa, F. Salvestrini, M. Tarzan, and L. Bergesio, “A software development kit to exploit rina programmability,” *IEEE International Conference on Communications*, May 2016.

[12] “Demonstrator source code,” https://github.com/IRATI/demonstrator.