Application-Driven Network Management with ProtoRINA

Yuefeng Wang Ibrahim Matta Nabeel Akhtar
Computer Science Department, Boston University
{wyf, matta, nabeel}@bu.edu

***Abstract*—Traditional network management is tied to the TCP/IP architecture, thus it inherits its many limitations. Additionally there is no unified framework for application management, and service providers have to rely on their own ad-hoc mechanisms to manage their application services. The Recursive InterNetwork Architecture (RINA) is our solution to achieve better network management. RINA provides a unified framework for application-driven network management along with built-in mechanisms. It allows the dynamic formation of secure communication containers for service providers in support of various requirements. In this paper, we focus on how application-driven network management can be achieved over the GENI testbed using ProtoRINA, a user-space prototype of RINA. We use video multicast as an example, and experimental results show that application-driven network management enabled by ProtoRINA can achieve better network and application performance.**

I. Introduction

Software-Defined Networking (SDN) [1] and Network Functions Virtualization (NFV) [2] both aim to provide better and more flexible network management. SDN simplifies network management by enabling programmability of the network through high-level network abstractions. NFV implements network functions as software instead of dedicated physical devices (middleboxes) to virtualize and consolidate network functions onto industry standard servers. However most work on SDN and NFV is tied to the TCP/IP architecture, and inevitably inherits many of its limitations, notably its static management and one-size-fits-all structure.

The Recursive InterNetwork Architecture (RINA) [3], [4] is a network architecture that inherently solves the problems of the current Internet. RINA's management architecture [5] is our solution to achieve better network management, and it inherently supports SDN and NFV concepts [6], [7]. Most importantly, RINA supports application-driven network management, where a federated and secure communication container can be dynamically formed to support different application requirements.

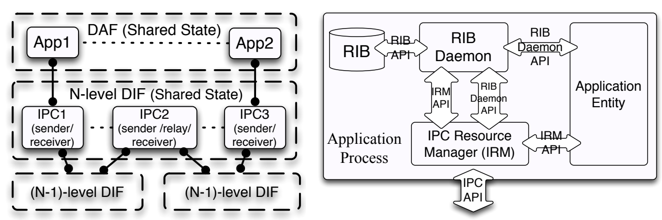
In this paper, we explain how application-driven management can be achieved using our policy-based management architecture with ProtoRINA [8], [9], a user-space prototype of RINA, and as an example we illustrate how video can be efficiently multicast to many clients on demand. Comparing to our previous published work [5], [6], [7], this paper is a detailed description of our application-driven management architecture and its support for programmability via application or IPC process relays.

II. Background

A. RINA Architecture and ProtoRINA

The Recursive InterNetwork Architecture (RINA) [3], [4] is a new network architecture which inherently solves the communication problem in a fundamental and structured way. It is based on the fundamental principle that *networking is Inter-Process Communication (IPC) and only IPC*.

1) **Distributed Application Facility***:* As shown in Figure 1(a), a *Distributed Application Facility (DAF)* is a collection of distributed application processes with shared states. Each DAF performs a certain function such as video streaming and weather forecast. Particularly, a *Distributed IPC Facility (DIF)*, *i.e.*, a collection of IPC processes, is a special DAF whose job is only to provide communication services over a certain scope (*i.e.*, range of operation) for application processes. Recursively, a higher-level DIF providing larger scope communication services is formed based on lower-level DIFs that provide smaller scope communication services. Different DAFs use the same mechanisms but they may use different policies for different purposes and over different scopes.



(a) (b)

Fig. 1: (a) RINA overview. (b) Application process components and APIs

2) **ProtoRINA***:* ProtoRINA [8], [9] is a user-space prototype of the RINA architecture. It provides a framework with common mechanisms which enable the programming of recursive-networking policies (supported by network applications). It can be used by researchers as an experimental tool to develop network applications, and also by educators as a teaching tool in networking and distributed systems classes. A *RINA node* is a host (or machine) where application processes and IPC processes reside. A *DIF Allocator* is a management DAF with application processes running on RINA nodes to manage the use of various existing DIFs and can create new DIFs on demand to provide communication services or meet different application-specific requirements.

B. Application-Driven Network Management

By application-driven network management, we mean given the physical topology, virtual networks can be built on the fly to satisfy application-specific demands and achieve better network performance. In RINA, each virtual network is actually a secure transport container providing inter-process communication. Processes inside such transport containers are authenticated and instantiated with policies that meet the needs of applications running atop, and such policies include

978-1-5090-0223-8/16/$31.00

private addressing, access control, routing, resource allocation, error and flow control, *etc.* A DIF is such a secure transport container. Each DIF has its own scope, and DIFs all use the same RINA mechanisms but can have different policies.

Most recent work on network management, such as SDN management platforms (*e.g.*, [10], [11]) or NFV management platforms (*e.g.*, [12], [13]), focuses on managing the network in a flat way where there is only one scope that includes all elements (physical components, *i.e.*, devices, and logical components, *i.e.*, processes) of the network. And they do not allow dynamic instantiation of such transport containers with different subscopes (subset of network elements) based on application requirements. Some work has been done to support network virtualization based on application requirements (*e.g.*, [14] and [15]), but their virtual network is limited to routing and not for transport purpose, and they do not support the dynamic formation of virtual networks. With the development of new service models (*e.g.*, Software as a Service), as well as the demand for different SLAs (Service-Level Agreements), we believe application-driven network management is necessary and will become the norm.

III. RINA Mechanisms for Application-Driven Network Management

A. DAF-Based Management Architecture

A DAF is a collection of distributed application processes cooperating to perform a certain function (Section II-A1). RINA's management architecture is DAF-based [5], *i.e.*, application processes providing management functionalities form different management DAFs, and the same DAF-based management structure repeats over different management scopes.

We would like to highlight two forms of management based on scope. The first is *DIF management*, *i.e.*, managing the DIF itself to provide communication service within a small scope. Examples of such management include different policies for routing traffic or establishing transport flows among IPC processes. The second is *network management*, *i.e.*, managing various DIFs that form the whole network. Examples of such management include the dynamic formation of new DIFs to provide communication services between remote application processes. In the former case, the *Management Application Entity* [9] of each IPC process inside the DIF forms the management DAF, and in the latter case, the *DIF Allocator* forms the management DAF for the whole network (Section II-A2).

Our previous work [6] focused on the DIF management where policies of a single DIF can be configured to satisfy different requirements, while in this paper we focus on network management where new higher level DIFs can be formed in support of application-specific demands. Our management architecture is built on top on the RINA architecture, but it is general, and can be used as a stand-alone management system overlayed on top of any infrastructure, such as the Internet.

B. Application Process Components and RINA APIs

Figure 1(b) shows the common components of an application process in ProtoRINA. The *Resource Information Base (RIB)* is the database that stores all information related to the operations of an application process. The *RIB Daemon* helps other components of the application process access information stored in the local RIB or in a remote application's RIB. Each application process also has an *IPC Resource Manager (IRM)*, which manages the use of underlying IPC processes belonging to low-level DIFs that provide communication services for this application process. The *Application Entity* is the container in which users can implement different management (or application-specific) functionalities. ProtoRINA (Section II-A2) provides two sets of APIs, *RIB Daemon API* and *IRM API*, for users to write management (or regular) applications and to support new network management policies. The RIB Daemon API is based on a publish/subscribe model and allows application processes to retrieve or publish network information from/to other application processes. The RIB Daemon also supports the traditional pulling mechanism to retrieve information. The IRM API allows allocating/deallocating a connection (flow) to other application processes, and sending/receiving messages over existing connections. More details about RINA programming APIs can be found in [9].

IV. Video Multicast With ProtoRINA

Next we explain how video can be efficiently multicast to different clients on demand as an example of application-driven network management. In order to support RTP (Real-time Transport Protocol) video streaming over the RINA network, RTP proxies (server proxy and client proxy) are used as shown in Figure 2. The RTP server proxy is connected to the video server over the Internet, and each RTP client proxy is connected to a video client also over the Internet. The RTP server proxy and RTP client proxies are connected over the RINA network which consists of DIFs. Namely, the RTP server proxy redirects all RTP traffic between the RTP server and RTP client to the communication channel provided by the RINA network. In our experiments, we use the VLC player [16] as the video client, and the Live555 MPEG Transport Stream Server [17] as the RTP video server. The video file used in the experiments is an MPEG Transport Stream file, which can be found at [18].

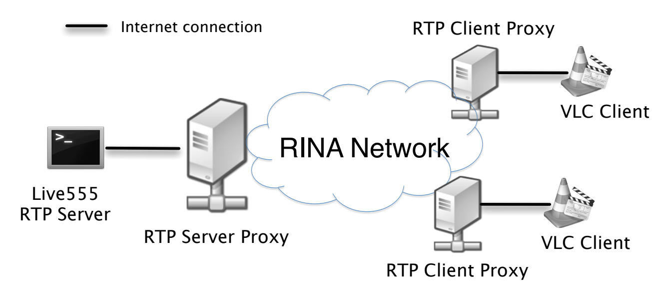


Fig. 2: Video clients (VLC players) are connected to the RTP video server through RTP proxies over a RINA network

Figure 3 shows a scenario, where the whole network is made up of four enterprise (or university) networks. The RTP server and RTP server proxy are running in `Network A`, and they provide a live video streaming service. There are two video clients along with RTP client proxies (one in `Network C` and the other one in `Network D`) that would like to receive video provided by the RTP video server. `Network A` and `Network B` are connected through `DIF 1`, `Network B` and `Network C` are connected through DIF 2, and `Network B` and `Network D` are connected through `DIF 3`. `DIF 1`, `DIF 2` and `DIF 3` are three level-zero DIFs that can provide communication services for two connected networks.

A very simple way to meet clients' requirements is as follows. Two video clients can receive live streaming service from the video server through two unicast connections supported by two separate DIFs as shown in Figure 3. The unicast connection between `RTP Client Proxy 1` and the video server proxy is supported by `DIF 4`, which is a level-one DIF formed based on `DIF 1` and `DIF 2`. The unicast connection between `RTP Client Proxy 2` and the video server proxy is supported by `DIF 5`, which is a level-one DIF formed based on `DIF 1` and `DIF 3`. However, it is easy to see that the same video traffic is delivered twice over `DIF 1`, which consumes unnecessary network bandwidth. In order to make better use of network resources, it is necessary to use multicast to stream the live video traffic. Next we show two different solutions of managing the existing DIFs to support multicast, *i.e.*, two ways of application-driven network management.

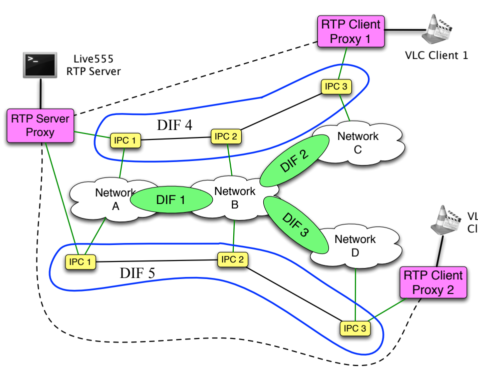


Fig. 3: Video streaming through unicast connections, where same video traffic is delivered twice over `DIF 1` consuming unnecessary network bandwidth

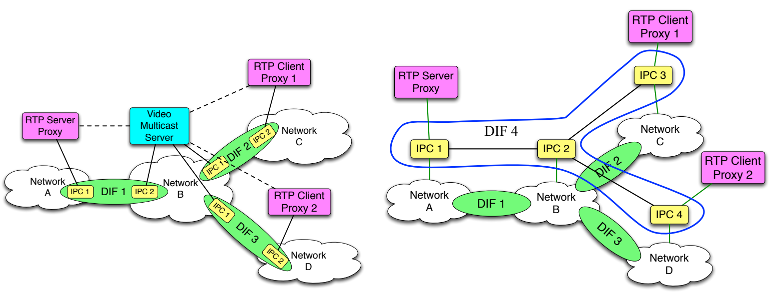
A. Solution One: Application-Level Multicast

The first solution is enabled through a video multicast video server as shown in Figure 4a. The connection between the video server and the video multicast server is supported by `DIF 1`. The connection between the video multicast server and `RTP Client Proxy1` is supported by `DIF 2`, and the connection between the video multicast server and `RTP Client Proxy 2` is supported by `DIF 3`. The video server streams video traffic to the video multicast server, which multicasts video traffic to each client through two unicast connections supported by `DIF 2` and `DIF 3`, respectively. We can see that the video traffic is delivered only once over `DIF 1` compared to Figure 3. And we only rely on existing level-zero DIFs, and no new higher-level DIF is created.

Actually the video multicast server provides a VNF (Virtual Network Function [2]) as in NFV (Network Function Virtualization), *i.e.*, RINA can implicitly support NFV. In a complicated network topology with more local networks, if there are more clients from different local networks needing the live streaming service, we can instantiate more video multicast servers, and place them at locations that are close to the clients, thus provide better video quality and network performance (such as less jitter and bandwidth consumption).

B. Solution Two: DIF-level Multicast

The second solution is supported using the multicast service provided by the DIF mechanism. As shown in Figure 4b, we form a level-one DIF `DIF 4` on top of existing level-zero DIFs. The video server proxy creates a multicast channel through `DIF 4`, and streams live video traffic over this multicast channel. Each client joins the multicast channel to receive the live video traffic. Note that the allocation of a multicast connection is the same as the allocation of a unicast connection, and both are done through the same RINA API.



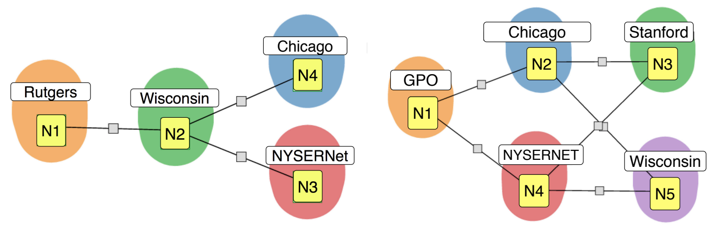
(a) (b)

Fig. 4: (a) Video multicast through an RTP multicast video server. (b) Video multicast through the multicast service provided by the DIF

Here we can see that RINA implicitly supports SDN [1] by allowing the dynamic formation of new DIFs (virtual networks). What's more, it allows instantiating different policies for different DIFs. In a complicated network topology with more local networks, if there are more clients from different local networks accessing the live streaming service, we can either dynamically form new higher-level DIFs or expand the existing DIFs providing the multicast service.

V. Experiments over GENI

GENI [19] is a nationwide suite of infrastructure that supports large-scale experiments, and it enables research and education in networking and distributed systems. Through GENI, users can obtain computing resources (*e.g.*, virtual machines (VMs) and raw PCs) from different physical locations (GENI aggregates), and connect these computing resources with layer-2 (stitched VLAN) or layer-3 (GRE Tunnel) links. In this section, we show our experimental results over GENI.



(a) (b)

Fig. 5: (a) GENI resources from four GENI aggregates. (b) GENI resources from five GENI aggregates

A. Bandwidth Usage

We reserve four VMs from four GENI aggregates (Rutgers, Wisconsin, Chicago and NYSERNet) shown in Figure 5a, and we connect the VMs using stitched VLANs. Each aggregate corresponds to one network in Figure 3, where the RTP server and RTP server proxy are running on VM `N1` in the Rutgers aggregate, the `RTP Client Proxy 1` is running on VM `N4` in the Chicago aggregate, and the `RTP Client Proxy 2` is running on VM `N3` in the NYSERNet aggregate.

Figure 6 shows the bandwidth usage for the unicast solution and the two multicast solutions (cf. Figure 3 and 4). We can see that, as expected, the bandwidth usage for the two multicast solutions are close to half of that of the unicast solution.

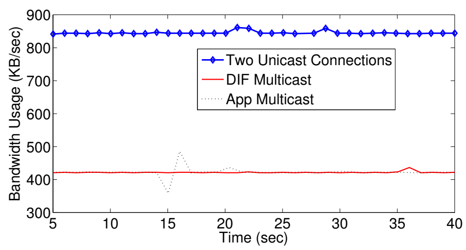
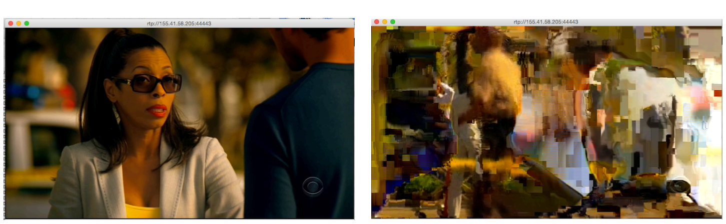


Fig. 6: Comparison of bandwidth usage over `DIF1`: unicast vs. multicast

B. Video Quality

We reserve five VMs from five GENI aggregates (GPO, Chicago, NYSERNet, Stanford, and Wisconsin) shown in Figure 5b, and we connect the VMs using stitched VLANs. The RTP server and RTP server proxy are running on VM `N1` in the GPO aggregate, the `RTP Client Proxy 1` is running on VM `N3` in the Stanford aggregate, and the `RTP Client Proxy 2` is running on VM `N5` in the Wisconsin aggregate.



(a) (b)

Fig. 7: (a) Video observed when the path selected is with less jitter. (b) Video observed when the path selected is with more jitter

The goal is to observe the effect on the video quality at the video client side when placing the video multicast server (cf. Figure 4a) in different locations, *i.e.*, placing the video multicast server either on VM `N2` in the Chicago aggregate or VM `N4` in the NYSERNet aggregate. Since GENI does not yet allow specifying parameters when reserving stitched VLANs, such as capacity and latency, we use a network emulation tool, NetEm [20] to add delay (1000ms $\pm$500ms) on the link between VM `N1` in GPO and VM `N2` in Chicago. To observe video quality, we have VLC players running locally on our BU campus network and connect them to the RTP client proxies running on GENI aggregates (*i.e.*, VM `N3` and `N5`) via Internet connections. Note that the jitter on the Internet connections is negligible, and the jitter in our experiments is mainly from jitter emulated on GENI links.

We run a VLC player locally and connect it with the `RTP Client Proxy 1` running on VM `N3` in the Stanford aggregate. Figure 7(b) shows the video observed when placing the multicast server on VM `N2` in the Chicago aggregate. Figure 7(a) shows the video observed when placing the multicast server on VM `N4` in the NYSERNet aggregate. We can see that by placing the video multicast server at a location experiencing less jitter we can achieve better video quality.

Regarding the time taken to establish/modify the multicast tree, our recent measurements as shown in [21] indicate that it is proportional to the size of the DIF and typically in the order of a few seconds.

VI. Future Work and Conclusion

In this paper, we described how to achieve application-driven network management using ProtoRINA. As an example, we show how video can be efficiently multicast to many clients on demand by dynamically creating a delivery tree. Under RINA, multicast can be enabled through a secure communication container that is dynamically formed to support video transport either through application proxies or via relay IPC processes. We also highlighted RINA's inherent support for envisioned SDN and NFV scenarios. The experimental results over the GENI testbed show that application-driven network management enabled by ProtoRINA achieves better network and application performance. As future work, we plan to investigate how to build a RINA network and compose policies given the physical topology to achieve better network and application performance for different applications. Also we plan to have our ProtoRINA run on a long-lived slice (virtual network) over the GENI testbed to make a RINA network available to researchers and educators so that they can opt-in to offer or access new services.

References

[1] Open Networking Foundation White Paper, "Software-Defined Networking: The New Norm for Networks," April, 2012.

[2] ETSI, "Network Functions Virtualisations (NFV) - White Paper," https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf.

[3] J. Day, I. Matta, and K. Mattar, "Networking is IPC: A Guiding Principle to a Better Internet," in *Proceedings of ReArch'08 - Re-Architecting the Internet (co-located with CoNEXT)*, New York, NY, USA, 2008.

[4] Boston University RINA Lab, http://csr.bu.edu/rina/.

[5] Y. Wang, F. Esposito, I. Matta, and J. Day, "RINA: An Architecture for Policy-Based Dynamic Service Management," in *Technical Report BUCS-TR-2013-014, Boston University*, 2013.

[6] Y. Wang, N. Akhtar, and I. Matta, "Programming Routing Policies for Video Traffic," in *CNERT 2014*, Raleigh, NC, USA, October 2014.

[7] Y. Wang, I. Matta, and N. Akhtar, "Network Functions Virtualization using ProtoRINA," Poster at the 21st GENI Engineering Conference (GEC21), Bloominton, IN, Oct 2014.

[8] Y. Wang, I. Matta, F. Esposito, and J. Day, "Introducing ProtoRINA: A Prototype for Programming Recursive-Networking Policies," *ACM SIGCOMM Computer Communication Review (CCR)*, July 2014.

[9] Y. Wang, F. Esposito, I. Matta, and J. Day, "Recursive InterNetworking Architecture (RINA) Boston University Prototype Programming Manual," in *Technical Report BUCS-TR-2013-013, Boston University*, 2013.

[10] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: A Distributed Control Platform for Large-scale Production Networks," in *OSDI 2010*, Berkeley, CA, USA.

[11] A. D. Ferguson, A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi, "Participatory Networking: An API for Application Control of SDNs," *SIGCOMM Comput. Commun. Rev.*, Aug. 2013.

[12] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici, "ClickOS and the Art of Network Function Virtualization," in *NSDI 2015*, Seattle, WA.

[13] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "OpenNF: Enabling Innovation in Network Function Control," in *SIGCOMM 2014*, New York, NY, USA.

[14] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "FlowVisor: A Network Virtualization Layer," in *Technical Report, OpenFlow-TR-2009-1, OpenFlow Consortium*, 2009.

[15] E. Salvadori, R. Doriguzzi Corin, A. Broglio, and M. Gerola, "Generalizing Virtual Network Topologies in OpenFlow-Based Networks," in *GLOBECOM 2011*.

[16] VLC Media Player, http://www.videolan.org/.

[17] LIVE555 MPEG Transport Stream Server, http://www.live555.com/.

[18] http://csr.bu.edu/rina/RTPVideoSample/.

[19] GENI, http://www.geni.net/.

[20] NetEm. Linux Foundation, "http://www.linuxfoundation.org/collaborate/workgroups/networking/netem."

[21] Y. Wang and I. Matta, "A Recursive Approach to Network Management," in *Technical Report BUCS-TR-2015-014, Boston University*, 2015.