SIMPLIFYING MULTI-LAYER NETWORK MANAGEMENT WITH RINA: ANALYSIS OF A MULTI-TENANT DATA CENTER NETWORK

Eduard Grasa, Bernat Gastón
Fundació i2CAT, Barcelona, Spain.
e-mail: eduard.grasa@i2cat.net, bernat.gaston@i2cat.net

Sven van der Meer
Ericsson NM Labs, Ercisson Software Campus, Athlone, Col. Westmeath, Ireland.
e-mail: sven.van.der.meer@ericsson.com

Michael Crotty
Telecommunications Software & Systems Group, Waterford Institute of Technology, Ireland.
e-mail: mcrotty@tssg.org

Miguel Angel Puente
Atos Research and Innovation, Madrid, Spain.
e-mail: miguelangel.puente@atos.net

Paper type
Research paper

Abstract
Computer networks are made of multiple co-operating layers that perform different functions implemented by a diverse set of protocols. The current approach of one function per layer implemented via one or more protocols contributes to increasing the complexity of multi-layer network management systems, causing them to be more expensive, error-prone and less automated that they could be. RINA is a network architecture featuring a single type of layer that recurses as many times as needed by the network designer. This layer, called a DIF, provides Inter Process Communication (IPC) services over a certain scope and ranges of bandwidth, QoS and scale. This paper performs a comparative analysis in the complexity of managing an IP-based and a RINA-based large-scale multi-tenant data centre networks. Configuration management is the main target of the analysis although some hints on performance and security management are also provided. The analysis shows that the commonality built into the RINA architecture and the single type of recursive layer with a uniform API greatly reduces the complexity of the models the Network Management System (NMS) uses to understand the state of the managed network. RINA opens the door not to an unprecedented degree of automation in Network Management, enabling the NMS to perform sophisticated configuration changes in multiple layers of the network at once while minimizing the risk of causing service downtime.

Keywords
Network configuration management, multi-layer, network automation, RINA, multi-tenant data centre.

1. Introduction
Today computer networks are designed as multiple co-operating layers that perform different functions implemented by a diverse set of protocols. Allowing distributed applications to achieve an optimal performance through the network requires the different layers to collaborate, so that application quality requirements can be enforced down to the physical wires. Each layer and protocol needs to be properly configured, and this

configuration must be continuously updated due to the dynamic nature of the network operational environment (node and link failures, degradation of physical layer conditions) or changes in the volume and characteristics of the traffic offered to the network.

Network Management Systems (NMSs) are in charge of keeping the network operating in optimal conditions, monitoring the behavior of the different layers that make up the network, reasoning about their current and desired states, and taking any corrective actions to update the network configuration if needed (due to misalignments in performance or reliability and security issues). Through years the role of NMSs has switched from performing low-level control decisions to taking part in higher levels of decision, since more and more autonomic control functions such as routing, resource allocation, flow recovery or congestion control have been embedded into the network layers. The main driver for this trend is that events in a network occur too fast for a human to be in the loop, thus they can't follow the network's peace of change. Thus, we can characterize the goal of modern NMSs to be that of "monitor and repair", as illustrated in **Figure 1**. It is important to clarify that monitoring is not trying to micro-manage the autonomic control functions, but to guide them so that they stay closer to their desired state.

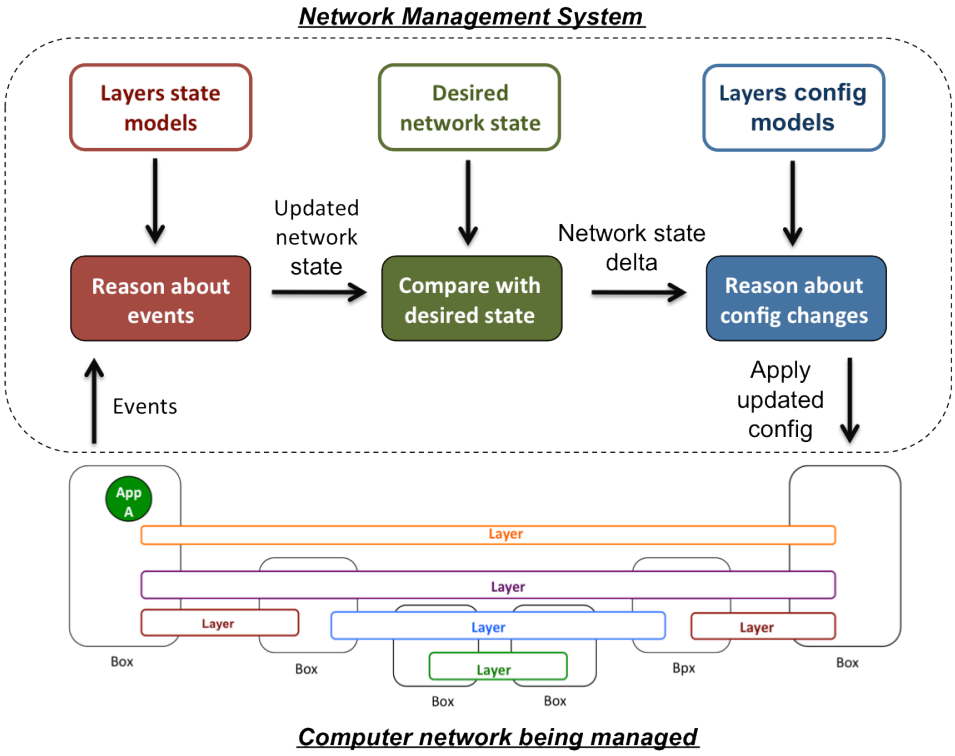

Figure 1 Model of the Network Management System as a closed control loop

Figure 1 conceptualizes the main functions of a NMS, forming a closed control loop with the computer network being managed [1]. The NMS receives events (notifications, reports) from the network, which contain relevant information about changes in the state of one or more layers. The first step of the NMS is to reason about these events, performing data analytics in order to understand what is the updated network state. To do so, the NMS needs models about the state of the different layers that compose the network. Once it has understood the updated stated, the NMS has to compare it with a reference desired state – provided by the network managers – and obtain the "state delta" between the desired network state and the current network state. Finally, this state delta is used as the input to a module that needs to reason about the changes in the network configuration that need to be applied in order to get as close as possible to the desired network state (the "repair" function of the NMS). This stage requires the NMS to have models of the network layers configuration. After applying the configuration changes, the NMS will get more events that will allow it to reason about the effectiveness of the new configuration (closing the loop of the control system).

The current Internet architecture is based on the paradigm of functional layering: each layer performs a different function, usually implemented by multiple protocols. For example: the transport layer performs end-to-end error

and flow control via protocols such as TCP, UDP or SCTP (Stream Control Transport Protocol). The network layer performs relaying over a number of data link layers of different technology. IP is the universal protocol at this layer, but a number of other protocols perform the dynamic control functions of the network layer, such as routing or resource reservation. Data link layers take care of the transmission of data over physical links of different characteristics; therefore different protocols are used depending on the link characteristics. In addition to the layers present in the theoretical architecture (L1 to L4 with applications on top), practical network deployments include also other type of layers, such as "layers 2.5" like MPLS – Multi Protocol Label Switching, separated customer and provider data link layers in Provider Backbone Bridging (PBB) or "virtual layers" on top of transport in virtual network overlays to allow for multi-tenancy in data centres (NVGRE – Network Virtualization using Generic Routing Encapsulation or VXLAN – Virtual eXtensible Local Area Network [2]).

Managing networks made up of functional layers becomes complex very fast, since each layer has a different state, configuration and service model that the NMS must understand. What is more, interaction between layers varies depending on the type of layer and protocol; therefore predicting what will be the effect of a configuration change across multiple layer becomes very hard. Last but not least, there are multiple network management protocols (the protocols used to get notifications from the network and update its configuration) such as SNMP (Simple Network Management Protocol), CMIP (Common Management Information Protocol), WBEM (Web-Based Enterprise Management) or NETCONF and multiple languages to model the functionalities of the different layers, such as SMI (Structure of Management Information), SID (Information Framework), CIM (Common Information Model) or YANG. The combination of NETCONF as a management protocol and YANG as a modelling language [3] is lately gaining traction in the ISP industry, but it still does not provide a complete coverage of all the different network layers found in a typical provider network today. SDN, Software Defined Networking, has also been proposed to simplify network management, by reducing the number of autonomous functions present in the hardware and logically centralizing them at certain network points. However it does not solve the inherent problem of functional layering: each layer is different.

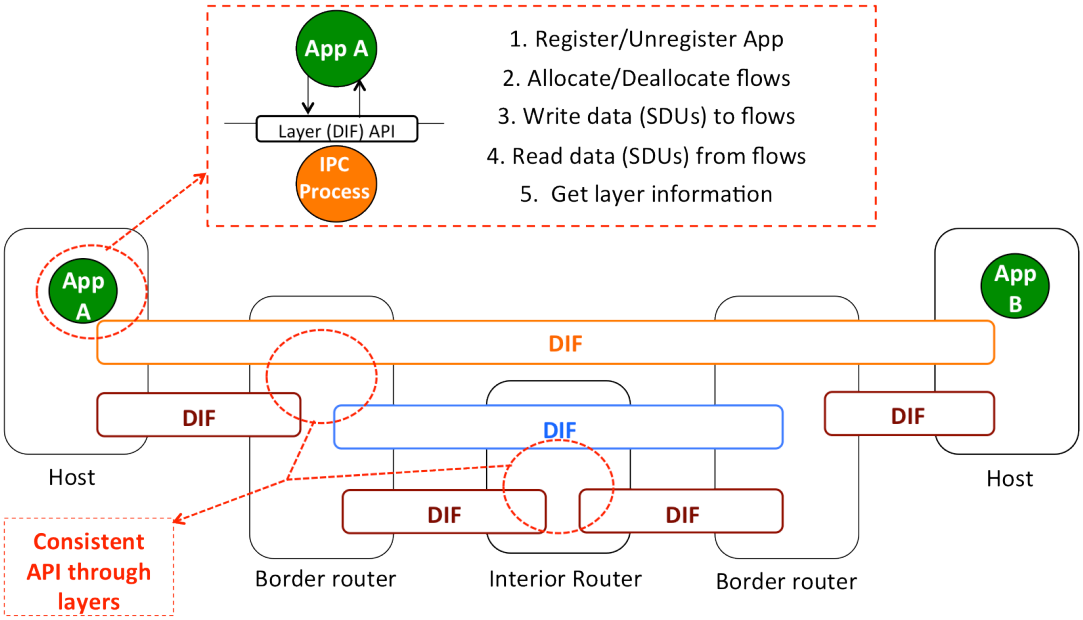

Figure 2 Structure of the Recursive InterNetwork Architecture (RINA)

RINA, the Recursive InterNetwork Architecture [4], [24], is a layered network architecture where layers are used to isolate different scopes, not to perform different functions. As shown in Figure 2, in RINA all layers provide the same service (Inter Process Communication or IPC between two or more application processes), and perform the same functions, but provide service over a different range of bandwidth, QoS and scale. All the functions of a layer can be separated in to the fixed parts (mechanism) and the variable parts (policies), therefore the behavior of a layer can also be tailored to its operational environment. Managing RINA networks is radically more simple than managing the networks of today, because: i) there is a single type of layer, therefore the NMS only needs to understand one state and configuration model; ii) since all layers are the same and provide the same API, if is much easier to reason about the interaction of multiple layers and last but not least iii) there is a single management protocol that is sufficient to manage all layers of the network. The simplicity gains are huge: not

only network management will become cheaper, but automating the troubleshooting and reconfiguration of multi-layer networks will become feasible, allowing the network operator to predict the effect of updating the configuration of multiple layers at once.

Figure 3 shows the internal organisation of the functions of a layer in RINA. A layer is a distributed application that provides and manages the resources for IPC. A (N)-layer is made up of a number of "IPC Processes" that can communicate with each other by using some number of (N-1)-layers. Ultimately a (N-1)-layer is the physical media. An IPC Process is merely a process that provides and manages IPC. The functions of each layer can be divided into three categories characterized by decreasing cycle time and increasing complexity: data transfer, data transfer control and layer management.

- *SDU Delimiting* groups the functions required to adapting SDUs (the data written by the N+1 layer to an N-port) to the characteristics of the layer (like maximum PDU size). These functions include fragmentation/reassembly and concatenation/separation.
- *Error and Flow Control Protocol (EFCP)* – This protocol is based on Richard Watson's work [5] and separates mechanism and policy. There is one instance of the protocol state for each flow originating or terminating at this IPC Process. The protocol naturally cleaves into Data Transfer (sequencing, lost and duplicate detection, fragmentation/reassembly, and delimiting), which updates a state vector; and Data Transfer Control, consisting of retransmission control (ack) and flow control. One could consider the Data Transfer half of EFCP to be UDP+IP and when Data Transfer Control is present, it is similar to (but not the same as) TCP/IP.
- *Relaying and Multiplexing (RMT)* – makes forwarding decision on incoming PDUs and multiplexes multiple flows of outgoing PDUs onto one or more (N-1)-interfaces. There is one RMT per IPC Process.
- *SDU Protection* – does integrity/error detection, e.g. CRC, encryption, compression, etc. Potentially there is a different SDU Protection for each (N-1)-interface.
- The state of the IPC Process is stored in the *Resource Information Base* (RIB) and accessed via the RIB Daemon. This all part of Layer Management, which cooperates with other IPC Processes in the layer to allocate resources, (*enrollment*, *managing the use of the (N-1)-layers*, *flow allocation*, *routing*, etc.) Coordination within the layer uses the *Common Distributed Application Protocol (CDAP)* to maintain timely information in the RIB.

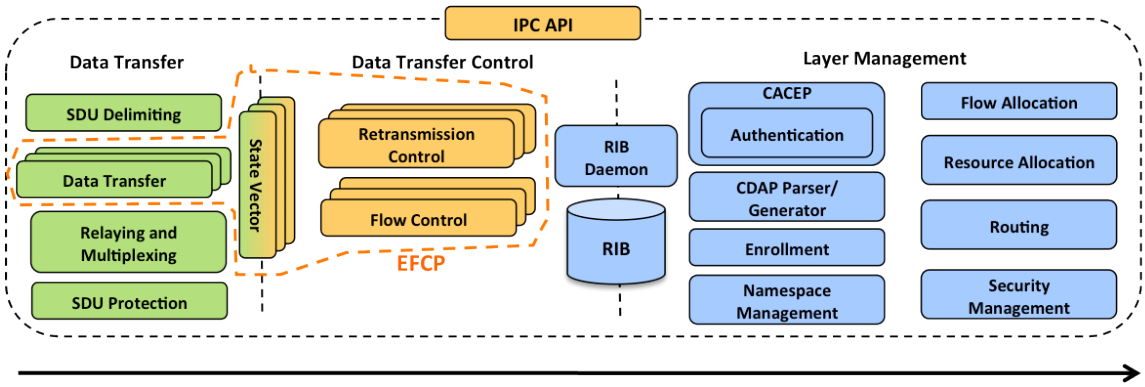

Figure 3 Internal organisation of a layer in RINA

The work on this paper compares the structural complexity of an IP-based and a RINA-based multi-tenant data centre (DC) network, and how this differences in complexity impact network management. The paper focuses its analysis in network configuration management, but also discusses implications for network performance and security management. The rest of this paper is structured as follows: section 2 introduces the multi-tenant DC network and its realisation with IP-based and RINA-based technologies; section 3 analyses the complexity of managing the configuration of the DC fabric; section 4 analyses the complexity of managing the configuration of the different tenant overlays; section 5 discusses implications for performance and security management and finally section 6 concludes and provides hints on future work.

2. Multi-tenant data centre (DC) network

Modern web-scale DataCentre Networks (DCNs) usually are designed with a multi-stage Clos topology to provide high cross-bisectional bandwidth and high levels of redundancy using a large number of moderate cost

networking devices. **Figure 4** provides an illustration of the physical layout of the modular Facebook DCN [6], which features a modular design to allow DCs to scale as their computing, storage and networking requirements evolve. The basic unit of modularity is the POD, which is a bunch of computing racks interconnected by a single-stage Clos network. Each rack has Z servers, connected to a ToR (Top of the Rack switches). Each POD has M racks (and therefore M ToRs), which are interconnected via 4 Fabric switches. PODs are connected to each other via a number of Spine switches organized in 4 different planes: each Spine plane has Y Spine switches. Traffic that doesn't leave the DC (East-West traffic) is not processed by any additional network devices; but for North-South traffic (in/out the DC), the DC has a number of Edge switches (organized into groups, 4 switches per group) that are also connected to the switch planes as shown in **Figure 4**. The whole design can be characterized by the parameters shown in **Table 1**.

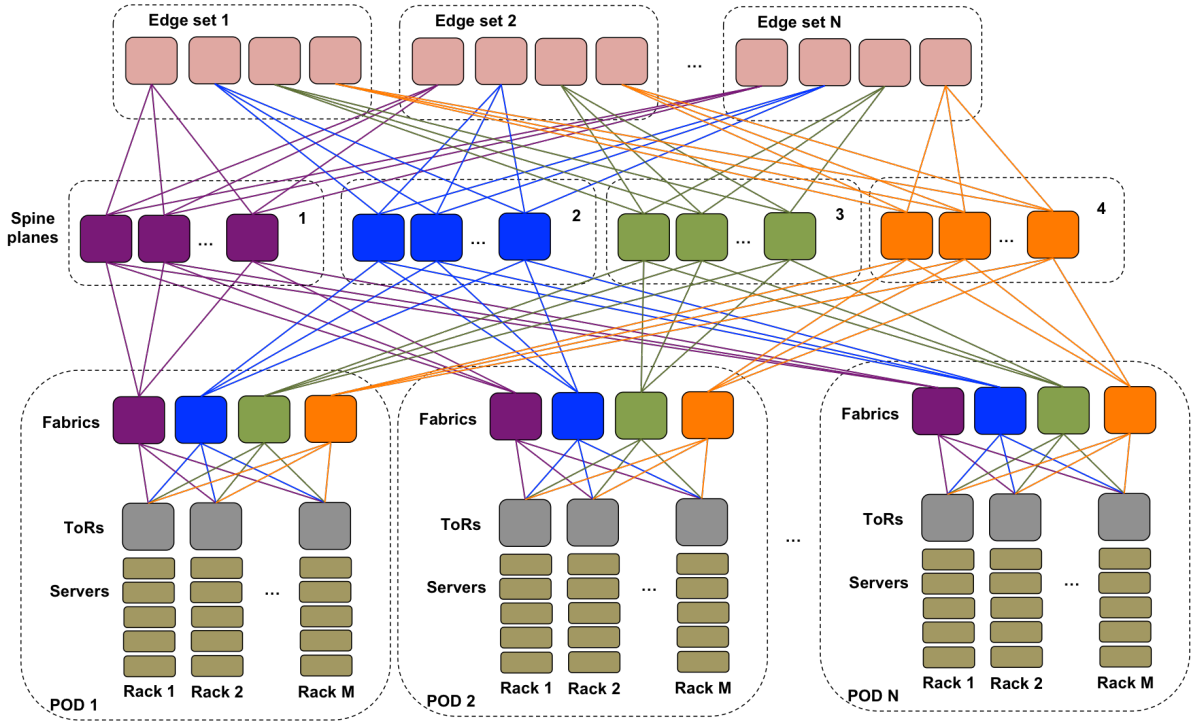

Figure 4 Physical layout of the modular Facebook DataCentre Network (DCN)

Parameter name	Max. value (for a full DC)
# Servers per rack (Z)	48
# Racks per POD (M)	48
# Spine planes (X)	4
# Spines per plane (Y)	48
# Edge sets (U)	4
# PODs	95

Table 1 Parameters defining the Facebook DCN and its max value

The following subsections analyze how a DC such as the one described above can provide its customers a service consisting in a set of Virtual Machines interconnected by isolated networks, independent of those from other tenants. DCs that are capable of providing this service are usually referred to as supporting "multi-tenancy" (where each tenant is allocated a subset of the DC computing, storage and networking resources).

2.1 IP-based DCN

There are multiple ways to design the network of a DC based on a multi-stage Clos Fabric, such as the one described by Facebook. Layer 2 (L2), Layer 3 (L3) or a combination of both approaches could be used in different parts of the DCN fabric, depending on the types of applications to be deployed in the DC and other requirements. For example, L2 technologies such as Q-in-Q or MAC-in-MAC could be used between ToR and

Fabric switches, with L3 technologies being used between Fabrics and Spines. In this paper we focus on an "all L3 DCN fabric" approach for two reasons: i) to minimize the number of protocols and technologies in the DCN – since we will perform a "best-case" comparison with RINA; and ii) since this is also the approach followed by Facebook. In order to implement the multi-tenant overlays on top of the DCN fabric we will use Ethernet VPN (EVPN) [7] technologies - a VXLAN data plane with a BGP control plane for MAC and host IP address learning-, since EVPN enables scalable and flexible L2 and L3 overlay networks over an L3 DCN fabric in a protocol-efficient way. Finally, we assume that the DC Network Management system can manage all the devices and protocols in the DCN via NETCONF and YANG [3].

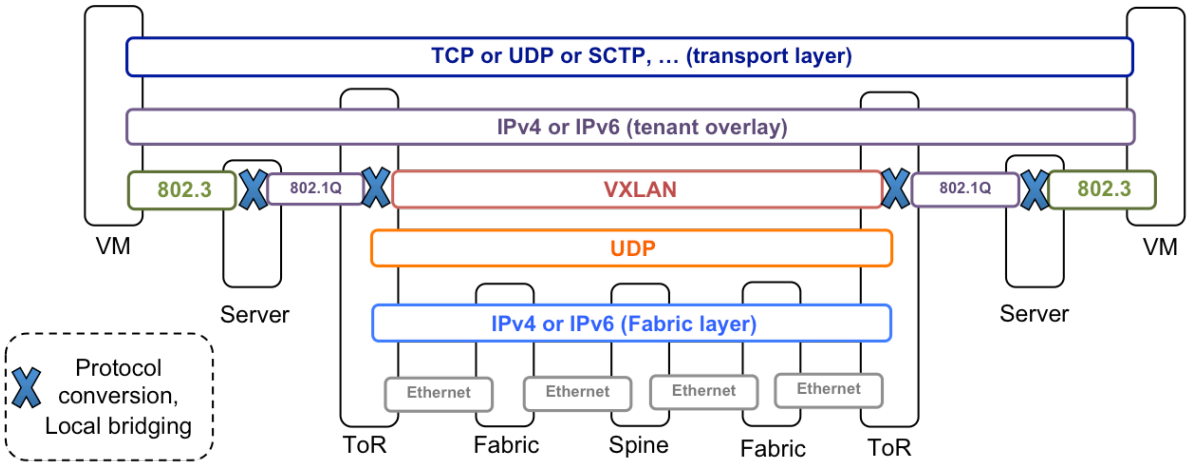

Figure 5 Data plane for VM to VM communication

Figure 5 shows the data plane of the IP-based solution, for VM to VM communication (VMs can communicate to external machines in the public Internet or a customer's internal network via a Gateway, but this paper only focuses on the analysis of inter-VM communication within the DC). VMs have one or more virtual Ethernet interfaces, which are connected by internal procedures to one of the Server's software Ethernet bridges. There the server typically tags the traffic generated by different VMs with a different VLAN id, and sent to an upstream Top of Rack switch (ToR). The ToR encapsulates the tagged Ethernet frame into a VXLAN frame (VLAN to VXLAN mappings have been populated by the eVPN control plane), and sends the resulting packet to the destination IP address in the VXLAN frame through the DC fabric. Equal Cost Multi-Pathing (ECMP) is heavily used in the data plane to leverage the large number of paths between each pair of nodes (either for load balancing or resiliency purposes). The fabric control plane – based on eBGP – keeps the ECMP groups up to date, adding or removing members as Ethernet links go up and down.

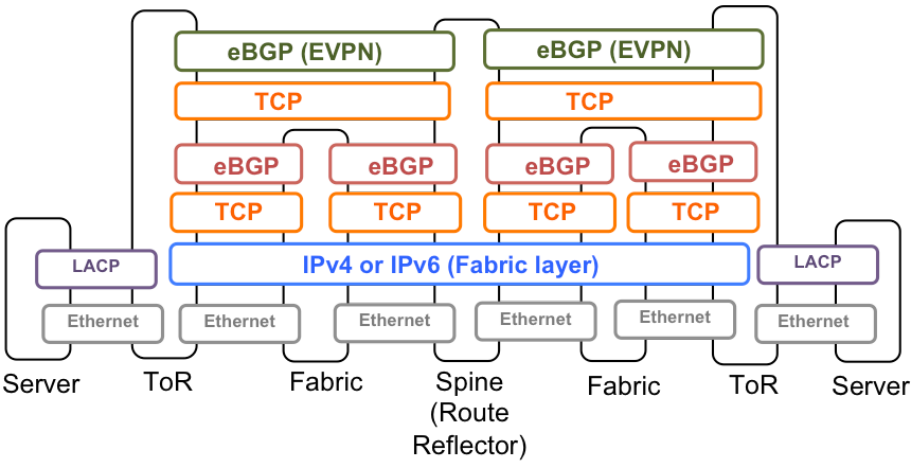

Figure 6 Control plane for VM to VM communication

The control plane of the IP-based DC-solution is shown in **Figure 6**. As opposed to the data-plane structure, in which the overlay layers are multiplexed over the fabric layers, the control-plane features an essentially flat

structure and is overlaid on top of the fabric IP layer (this is a key difference with the RINA structure). eBGP is deployed as the only routing protocol of the data centre fabric, following the design in [9], which divides the different DCN fabric switches into private Autonomous Systems (AS). All spine switches are grouped together as a single AS, all the fabric switches of the same group are another AS and finally each ToR with all the servers connected to it form another AS. Expressing the number of ASs as a function of the number of PODs in the DC (N) we obtain the following expression: N*(racks per POD + 1) + 1. Therefore the maximum number of ASs in a full DC – taking the numbers of **Table 1** – is of 4656. As explained in [8], this number is larger than the number of private AS numbers (considering 2-byte AS numbers), therefore special BGP configurations to allow for re-using private AS numbers have to be employed. BGP is also the key protocol in the eVPN overlay control plane, in which ToRs exchange eVPN routes amongst them (using BGP multi-protocol extensions and a dedicated address family). In order to avoid setting up a full mesh of BGP sessions between ToRs, some spine switches (or alternatively dedicated servers) can be configured as BGP route reflectors. Finally, in order to increase the availability of the servers, ToRs can be designed as two separate chassis that are connected to each server with redundant connections. In this configuration, the Link Aggregation Control Protocol (LACP) is required to provide the servers with transparent multi-homing over the separate physical Ethernet links to each ToR chassis.

2.2 RINA-based DCN

Figure 7 shows a conceptual diagram of the layers in the RINA-based Data Centre Network solution. As opposed to the IP-based design, in RINA both the data transfer (data plane) and layer management (control plane) functions of a layer are part of the same layer (DIF in the RINA terminology). The RINA-design that is equivalent to the IP-based designed features three different types of layers: i) a DC-Fabric DIF, which brings together all the ToR, Fabric and Spine switches as a large distributed switch; ii) multiple tenant DIFs, allowing tenants to connect dedicated computing resources (VMs) via performance-isolated and secure networking and iii) multiple Point-to-Point (PtP) DIFs, providing connectivity over individual physical links.

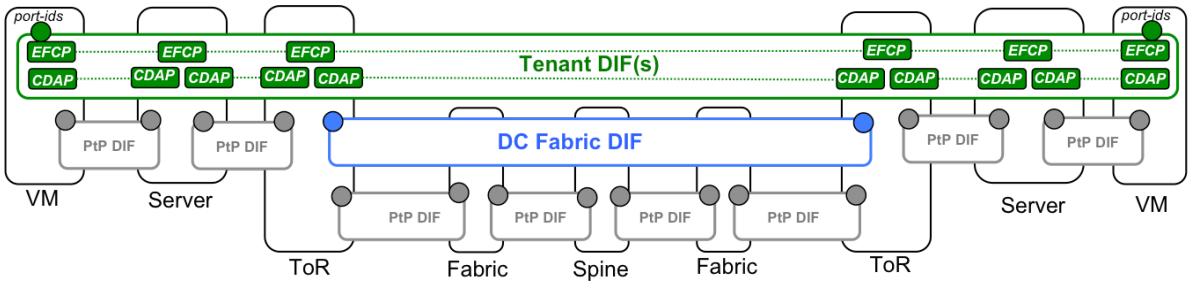

Figure 7 Layers in the RINA-based DCN

The Tenant DIFs is expanded to show a simplified view of the protocol processing performed by each IPC Process in that layer (the protocol structure of the other layers is identical, only specific policies change). EFCP is the single data transfer/data transfer control protocol, used to provide end-to-end flows to applications using the DIF (where the ends are defined by the scope of the layer). IPCPs that are at the endpoints of the flow encapsulate and decapsulate SDUs, optionally provide flow and/or retransmission control and forward the resulting PDUs to the next hop towards the destination IPCP. Intermediate IPCPs relay the PDUs belonging to different flows according to a forwarding policy maintained by the layer management functions (routing, flow allocation, resource allocation, security management, namespace management).

In parallel, all layer management functions exchange information (encoded as objects) with its neighbours via the CDAP protocol. CDAP allows layer management functions to perform 6 operations (create, delete, read, write, start, stop) targeting objects of its neighbour IPCPs. Therefore what changes from layer management function to layer management function are the objects and operations carried by CDAP, not the protocol. This is a big simplification in terms of network management and operation compared to IP-based designs, even more taking into account that this model is consistently followed by all layers.

In terms of specific policies for each layer, we will briefly describe the ones used for routing and forwarding in the DC Fabric DIF; policies for each tenant DIF could be different and customized to the tenant DIF goals. Taking into account that the DC has a highly regular connectivity graph (as seen in **Figure 4**), the use of topological addressing would minimize the need for exchanging routing information and the amount of entries in the forwarding tables. Each IPCP would know how to forward PDUs to all destination address by just inspecting the destination address and comparing it to the addresses of all direct neighbour IPCPs. Only failed links would

need to be disseminated via routing updates; upon learning about a failed link the routing function would compute an exception to the default forwarding rules and add an entry to the IPCP forwarding table. Since it is the case that usually multiple paths to the destination IPCP will exist, the forwarding policy would include a ECMP-style logic in order to load-balance PDUs of different flows over the different paths.

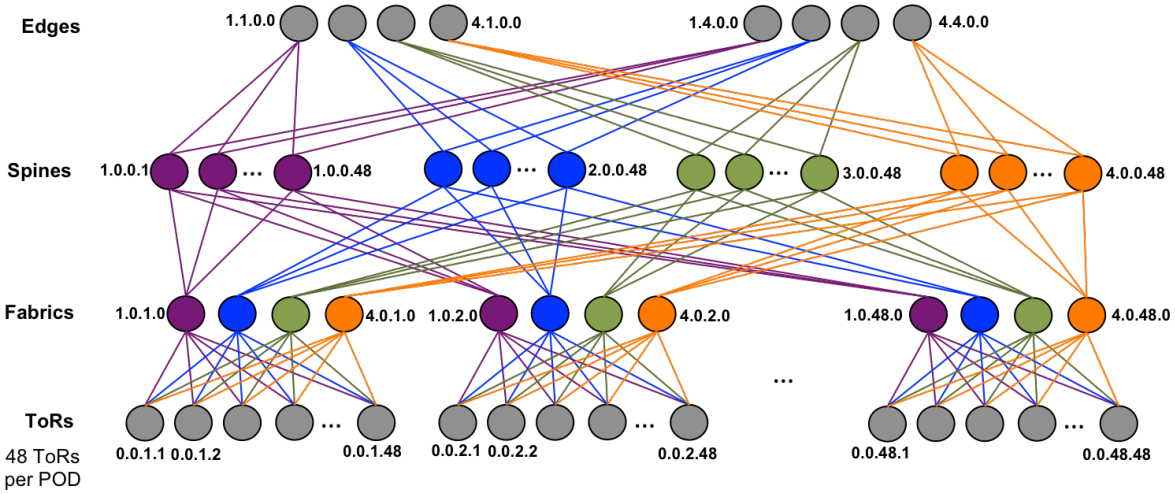

Figure 8 A possible topological addressing plan for the DC-fabric DIF

Figure 8 presents an example of a topological addressing plan for the DC-fabric DIF. Circles represent IPC Processes, while links represent N-1 flows. Each address has 3 bytes, structured as *a.b.c.d* (where a and b are 4 bits long, and c and d 8 bits long). Each type of IPCP is addressed as follows:

- Edge IPCPs: *a.b.0.0*, where *a* is the id of the spine plane and *b* the id of the edge set.
- Spine IPCPs: *a.0.0.c*, where *a* is the id of the spine plane and *c* the id of the spine within the plane.
- Fabric IPCPs: *a.0.d.0*, where *a* is the id of the spine plane and *d* is the id of the POD.
- ToR IPCPs: *0.0.d.e*, where *d* is the id of the POD and e the id of the rack within the POD.

The scheme scales up to 16 spine planes, 16 edge sets, 256 PODs and 256 racks per POD.

3. Configuration of DC Fabric

This section compares the complexity in configuring the fabrics of the RINA and IP-based solutions. Configuring the fabric involves configuring the ToR, Fabric, Spine and Edge devices. For simplicity we will assume that the DC Fabric provides no support for traffic differentiation and therefore a simple FIFO scheduling policy would suffice. Section 5 briefly discusses RINA's QoS model comparing it to the IP protocol suite in terms of network management.

3.1 IP-based solution

We will assume that when a device bootstraps all Ethernet interfaces are enabled by default. All devices in the IP-based DC fabric require a similar configuration, consisting in getting IP addresses for all the Ethernet interfaces (MAC addresses are already assigned to the physical interface by the interface vendor), configuring BGP sessions and ECMP groups. In order to configure BGP, at least the following information is needed: AS number, router-id (IP address), configuration of routing policies, maximum number of entries per ECMP group and for each session to each directly connected neighbour an IP address and its AS number. The following table summarizes the main entities managed in each layer.

Interfaces	Ethernet interfaces, need unique MAC address (one per interface)
Data transfer protocol syntax	IEEE 802.3 (Ethernet)
Management protocol	NETCONF
Management models	yang-common-types [9], yang-interfaces [10]

Table 2 Main managed entities for Point-to-Point Ethernet Links

Interfaces	IPv4 interfaces, need IP address (one per interface), unique in the layer.
Data transfer protocol syntax	IPv4 syntax, TCP syntax (TCP is used by the control plane)
Forwarding entity	router, one per device in the layer, has FIB entries (forwarding table)

Forwarding strategy	Longest prefix matching, ECMP
Scheduling strategy	FIFO (needs max-queue size) – assuming no traffic differentiation in the fabric
Routing protocol	BGP with different routing policies. Needs AS numbers, router-id (IP address), neighbours' IP addresses and AS numbers. Maintains RIB.
Management protocol	NETCONF
Management models	yang-common-types [9], yang-interfaces [10], yang-ip [11], yang-routing [12], yang-bgp [13]

Table 3 Main managed entities for the DC-fabric IPv4 layer

3.2 RINA-based solution

Similar to the IP-based case, we assume that the IPC Processes (IPCPs) belonging to the point-to-point DIFs are setup when the device (ToR, Fabric, Spine or Edge) bootstraps. Then the Management System would create a single IPCP per device belonging to the DC-Fabric DIF, configure them with an address and the policies described in **Table 5**. After that the Management System would instruct each IPC Process to enrol with all directly connected neighbours (so that neighbour IPCPs are able to exchange layer management information). Note that the Manager could just configure a few IPCPs and let all the other IPCPs in the DC layer automatically obtain their configuration by enrolling to those already-configured IPCPs.

Interfaces	Physical wire driver
Data transfer protocol syntax	EFCP (length of fields in PCI optimized for physical media). No addresses.
Management protocol	CDAP
Management models	daf-common-mom [14], dif-common-mom[14]

Table 4 Main managed entities for Point-to-Point DIFs

Interfaces	Port-ids to N-1 flows, just need port-id (locally –device- unique identifier)
Data transfer protocol syntax	EFCP (length of fields in the PCI optimized for the layer). Need address (one per device in the layer), unique in the layer
Forwarding entity	Relaying and Multiplexing Task (RMT), one per device in the layer, has forwarding table entries.
Forwarding strategy	Longest prefix matching, ECMP
Scheduling strategy	FIFO (needs max-queue size) – assuming no traffic differentiation in the fabric
Routing protocol	CDAP with link-state routing policy and topological addressing. Maintains RIB.
Directory protocol	CDAP with centralized directory policy. Maintains Directory Forwarding Table.
Management protocol	CDAP
Management models	daf-common-mom [14], dif-common-mom[14], dif-default-policies

Table 5 Main managed entities for the DC-fabric DIF

As explained in section 2.2, EFCP (data transfer) and CDAP (layer management, network management) are the only protocols required in every DIF. The fields in the EFCP PCI (source/destination addresses, qos-id, source/destination cep-ids, sequence number, length) are the same across layers, with only its length changing from layer to layer. All layer management functions (such as routing or the directory) are CDAP policies that manipulate different objects via the CDAP protocol actions (create, delete, read, write, start, stop).

The complete naming and addressing architecture of RINA [15] also contributes to reducing the management overhead by minimizing the number of addresses that needs to be configured in each layer. **Figure 9** shows the number of addresses needed in the DCN Fabric as a function of the number of PODs. In the IP-based solution each interface needs two addresses (MAC and IPv4), therefore the number of addresses is a function of the number of links plus the number of devices (since BGP needs a router-id per BGP-speaking device). In the RINA-based solution Point-to-Point DIFs don't need addresses (data can only have a possible source and a possible destination) and the DC-Fabric DIF just needs one address per device in the DIF (i.e. per IPCP). The simple formulas below express the number of addresses required for both solutions:

- ***IP, # of addresses***: 4*Links in DCN fabric + Nodes in the DCN Fabric (IP) = 436N + 976
- ***RINA, # of addresses***: Nodes in the DCN Fabric = 52N + 208

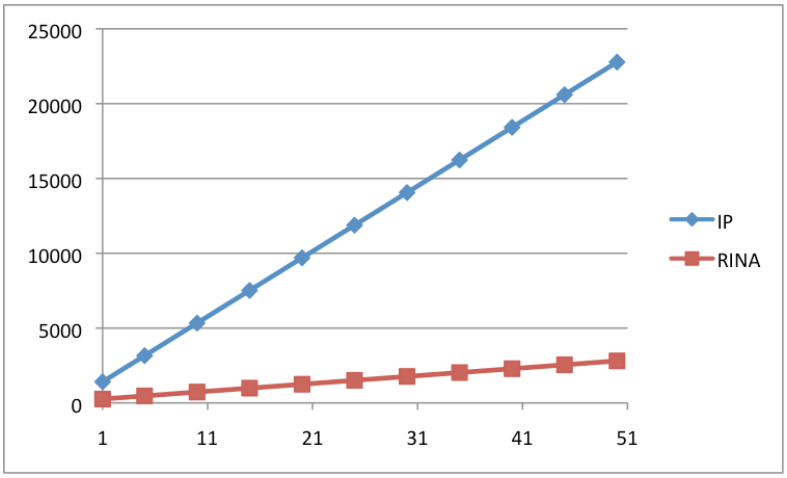

Figure 9 # of addresses in the DCN fabric as a function of the # of PODs in the DC

4. Configuration of Tenant Overlays

Tenant overlays have to bring together a number of VMs under a performance and security-isolated connectivity domain, crating the illusion that each tenant has its own dedicated "slice" of data centre resources. Configuring a tenant overlay requires updating the state of ToR and server devices, as well as instantiating VMs hosted in the servers. The size of a tenant overlay may go from small to moderate, but in any case is expected to be at least one or two orders of magnitude smaller than the size of the DC Fabric layer (considering a large-scale data centre with a capacity of 100k servers as shown in **Table 1**).

4.1 IP-based

In order to create a tenant overlay three different types of systems have to be configured: Virtual Machines, servers and ToRs. The configuration of Virtual Machines is simple: based on assigning IP addresses to the VM's virtual Ethernet interfaces. Servers must segment traffic coming from different VMs per tenant and multiplex it and send it to one of the two ToRs is connected two via an Ethernet link. The initial configuration of the server requires the creation of a Link Aggregation Group (LAG) with the two Ethernet interfaces that are connected to the uplink ToRs. Then, every time VMs belonging to a different tenant overlay are instantiated a virtual Ethernet bridge is created. The VM's virtual Ethernet interfaces are attached to this bridge. The logical Ethernet interface representing the LAG group is also "partitioned" into multiple VLANs, one per tenant DIF. Each VLAN interface is attached to the virtual bridge belonging to the tenant network.

When creating a tenant overlay a number of VMs will be instantiated on a number of servers, which will extend the layer 2 connectivity of the VMs to a set of ToRs as explained in the previous paragraph. In order to complete the tenant overlay, these set of ToRs must be connected together under the same private, L2 domain. L2 Ethernet VPNs over the DC fabric is the proposed way to implement this functionality for the IP-based solution discussed in this paper. In order to do so, a full mesh of VXLAN tunnels is created between ToRs belonging to the same tenant overlay. VXLAN tunnels transport Ethernet traffic over UDP and the IP layer of the DCN fabric.

For each VXLAN tunnel the Management System has to instantiate two Virtual Tunnel Endpoints (VTEPs), one at each end of the tunnel. Each VTEP needs to be bound to a local IP address and UDP port number, and associated to the IP address and UDP port number of the remote endpoint. Finally an Ethernet VRF (E-VRF) instance is created in each ToR, in order to connect together VTEPs and 802.1q interfaces belonging to the same tenant overlay. The E-VRF instance is like an Ethernet bridge, with the exception that it doesn't use data-plane learning techniques to populate its MAC forwarding table: the table is populated via routes learned by BGP. Therefore, a full mesh of BGP instances has to be configured between all ToRs participating in the provisioning of E-VPN services - or alternatively a number of spine switches can be setup as route reflectors to increase the scalability of the control plane, as **shown in Figure 6.**

Interfaces	Ethernet interfaces: need MAC address (one per interface) 802.1q interfaces: need VLAN-id VTEP interfaces: need VXLAN-id, local IP address and UDP port, remote IP address and UDP port

	IPv4 interfaces: need IP address (one per interface), unique in tenant overlay
Data transfer protocol syntax	IEEE 802.3 (Ethernet), IEEE 802.1q, IPv4, UDP, VXLAN, TCP
Forwarding entity	router: one per VM Ethernet bridge: one per server per tenant overlay E-VRF: one per ToR per tenant overlay
Forwarding strategy	Exact address matching
Scheduling strategy	FIFO (needs max queue size) – assuming no traffic differentiation
Routing protocol	BGP with multi-protocol extensions. Needs route distinguisher and VPN targets.
Directory protocol	DNS (resolve domain names of apps executing in the tenant DIF to IP @s)
Redundancy protocol	Link Aggregation Control Protocol – needs local Ethernet interface addresses
Management protocol	NETCONF
Management models	yang-common-types [9], yang-interfaces [10], yang-ip [11], yang-bridging [18], yang-routing [12], yang-bgp [13], yang-vxlan [16], yang-evpn [17], yang-lacp [19]

Table 6 Main managed entities at the IP-based tenant overlay layer

4.2 RINA-based

Unlike the IP-based solution for tenant overlays described in the previous section, the configuration of VMs, servers and ToRs when a new tenant overlay DIF is instantiated is quite similar [20]. In all the systems belonging to the tenant DIF the Management system has to instantiate a single IPCP belonging to this DIF, and configure the data transfer and layer management policies as in the DC-fabric DIF case. Supporting N-1 flows between IPCPs will be established over Point to Point DIFs (between servers and ToRs) or over the DC-Fabric DIF (between ToRs), but this is transparent to the IPCPs in the tenant overlay DIF: all the DIFs provide the same service API to its users, regardless of its internal policies or implementation.

Interfaces	Port-ids to N-1 flows, just need port-id (locally –device- unique identifier)
Data transfer protocol syntax	EFCP (length of fields in the PCI optimized for the layer). Need address (one per device in the layer), unique in the layer
Forwarding entity	Relaying and Multiplexing Task (RMT), one per device in the layer, has forwarding table entries.
Forwarding strategy	Longest prefix matching, ECMP (load-balancing/redundancy at server level)
Scheduling strategy	FIFO (needs max-queue size) – assuming no traffic differentiation in the overlay
Routing protocol	CDAP with link-state routing policy. Maintains RIB.
Directory protocol	CDAP with distributed directory policy. Maintains Directory Forwarding Table.
Management protocol	CDAP
Management models	daf-common-mom [14], dif-common-mom[14], dif-default-policies

Table 7 Main managed entities for the tenant overlay DIF

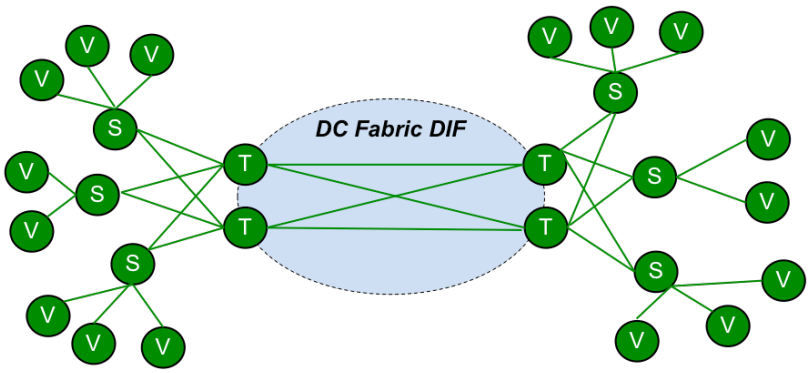

Figure 10 Example connectivity graph of a tenant overlay DIF, interconnecting 16 VMs in two separate racks

Table 7 summarizes the main information that the Manager needs to configure in each IPCP at the different devices (VMs, servers, ToRs). As with the DC-Fabric DIF, EFCP and CDAP are the only protocols used, while RMT is the only type of forwarding entity. The only differences with the DC-fabric DIF are in the policies used for routing and the distributed directory. Since tenant overlay DIFs are significantly smaller than the DC-fabric one, a normal link-state routing policy with flat identifiers would be enough. Since the structure of the graph of

tenant DIFs will be also quite regular, as shown in the example of **Figure 10**, topological addresses could also be used to allow routing to scale better. The small/medium size of these types of DIFs suggests that a directory policy following a distributed approach (similar to the dissemination of routing advertisements in link-state strategies) would be adequate.

The instantiation of tenant DIFs in the RINA-based use case is simpler than in the IP-based use case. In RINA only one type of forwarding entity exists (the RMT), compared to the three different forwarding entities used in the IP case (IP routers, Ethernet bridges and Ethernet VRFs). While in RINA there is a single data transfer protocol – EFCP – the IP-based design uses Ethernet with VLANs (IEEE 802.1Q), IPv4, VXLAN, UDP and TCP. The IP-based solution is also more complex in terms of interface types (physical Ethernet, logical Ethernet, Virtual Tunnel Endpoint, IP) compared to the RINA case (just N-1 port-ids). Even if its design has been chosen to minimize the number of control plane protocols, the IP-based solution still features more control plane protocols (BGP, DNS, LACP) than the RINA case, in which only CDAP with routing and directory policies is used. This situation is reflected in the management models of the IP-based solution, which grow in complexity as new requirements are introduced in the design. This fact is due to the lack of commonality and invariants in the IP protocol suite, in which the approach of dealing with different operational environments or new requirements is usually to design new protocols from scratch. In contrast, RINA networks leverage a common structure that captures the mechanisms that are invariant with respect to the requirements of each networking use case, and has built-in hooks for deploying optimized policies into its single data transfer (EFCP) and layer management (CDAP) protocols.

5. Brief overview of performance and security management

Although not the premier target of this paper, this section provides some hints on the complexity of performance and security management of the RINA-based and the IP-based DCN solutions.

5.1 Performance management

The high number of data-plane and control plane protocols in the IP protocol suite and the lack of a clear, consistent model to communicate performance requirements across layers make it hard to manage the performance of a multi-layer network so that it delivers consistent service experiences to its customers. Analyzing the multi-tenant data centre example, the DCN must efficiently multiplex application traffic coming from different tenant overlays while preserving their quality requirements. Some tenant overlays may deal with interactive traffic with stringent delay requirements, while others may transport a high volume of almost delay-insensitive traffic.

The DCN fabric must be made aware of the different quality requirements of the flows it delivers to tenant overlays, identify these flows and configure its internal mechanisms (e.g. scheduling policies) so that individual (or groups of) flows receive the adequate treatment. Since there is no abstract mechanism for the communication of performance requirements between layers, this communication has to be designed on a use-case per use-case basis, usually involving the Network Management System. The layer also must be able to identify the flows belonging to different QoS groups, which depends on the technology being used: DSCP code marking can be used in the case of IP, MPLS traffic engineering based on the values of MPLS labels or VLAN tags for Ethernet. Last but not least, as shown in sections 3 and 4 of this paper, different technologies have different descriptions and definitions for the "packet forwarding entity" and its associated scheduling algorithms, adding more complexity to the network performance management problem.

The RINA model provides solutions to the different cause complicating the management of performance in networks based on the IP protocol suite. As shown in **Figure 2**, in RINA all layers provide the same service API to its users. This API allows users of a layer to request a flow to a destination application with certain characteristics such as bounds on loss and delay, minimum capacity or in-order delivery of data. Therefore layers can pass performance requirements to each other in a technology-agnostic way, without the intervention of the Management System.

DIFs are designed to cover certain ranges of the performance space. A QoS cube is an abstraction of a set of policies that allow the DIF to deliver an IPC service within a certain range of the performance space (e.g. data loss, delay, jitter). Each DIF supports one or more QoS cubes, whose policies (data transfer, resource allocation, scheduling) are designed to ensure the promised performance in the operational environment of the DIF. When an application requests a flow to a DIF, the IPC Process that receives the requests checks the performance requirements for that flow and tries to map it to one of the QoS cubes supported by the DIF. If there is a match,

the IPCP creates a new EFCP instance for the flow, configuring it with the policies specified by the QoS cube. Each QoS-cube has a unique id within the DIF. All EFCP packets of a flow belonging to a QoS cube are marked with the qos-id of that QoS-cube, so that all intermediate IPCPs between the source and destination can identify the flows belonging to the different QoS classes and schedule them accordingly.

Therefore i) the availability of an abstract way of communicating performance requirements between layers; ii) the commonality of the RINA architecture – single data transfer protocol and single type of packet forwarding entity – and iii) a consistent approach towards mapping performance requirements into specific policies via QoS cubes greatly simplifies managing the performance of multi-layer RINA networks compared to their IP counterparts.

5.2 Security management

Management the security of IP networks is no easy task, since usually each protocol provides its own security mechanisms: IPsec, BGPsec, DNSSEC, Transport Layer Security (TLS), DTLS, OSPF security, etc. Add to this the large number of protocols to be secured in the IP protocol suite, and the result is again complexity in the protocol stack and the management model. RINA simplifies multi-layer security management due to two main reasons: i) again, the commonality in its structure and ii) RINA secures layers instead of securing protocols [21].

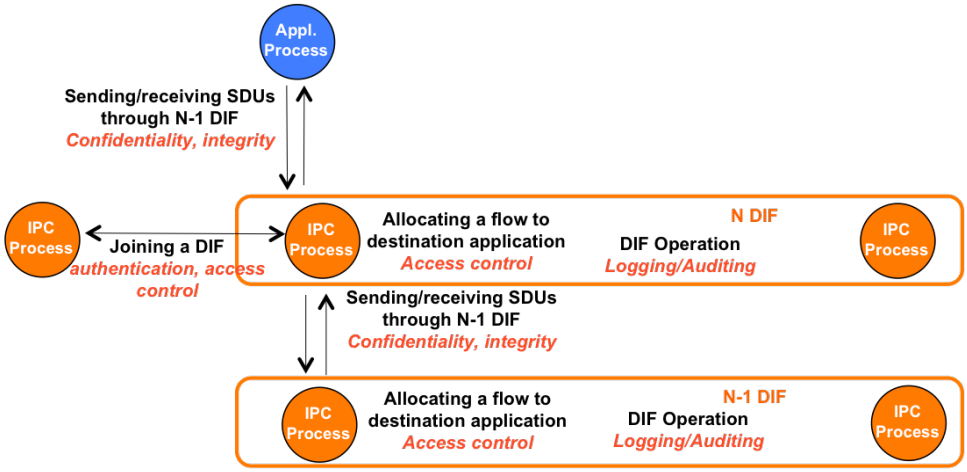

Figure 11 Distribution of security functions within the DIF and across DIFs

Figure 11 illustrates the distribution of security functions within the DIF and across DIFs. In RINA the granularity of protection is a layer, not its individual protocols, which allows for a more simple and comprehensive security model. Users of a DIF need to have little trust of the DIF they are using: only that the DIF will attempt to deliver Service Data Units (SDUs) to some process. Applications using a DIF are ultimately responsible for ensuring the confidentiality and integrity of the SDUs they pass to the DIF. Therefore, proper SDU protection mechanisms (such as encryption) have to be put in place. When a new IPCP wants to join a DIF it first needs to allocate a flow to another IPCP that is a already a DIF member via an N-1 DIF both processes must share in common. Here access control is used to determine if the requesting application is allowed to talk to the requested application. If the flow to the existing member is accepted, the next step is to go through an authentication phase, the strength of which can range from no authentication to cryptographic schemes. In case of a successful authentication the DIF member will decide whether the new IPCP is admitted to the DIF, executing a specific access control policy.

6. Conclusions and future work

This paper has described some of the advantages of managing a RINA network consisting in multiple layers compared to managing an equivalent IP-based network, describing a case study of a large-scale multi-tenant data centre network. The commonality offered by the RINA layers, together with a consistent QoS and security models followed by all layers from the application to the wire allows RINA to minimize the number of management models required to represent all layers and protocols, therefore simplifying a lot the design of Network Managers. Since Managers can reason about simpler models, their behaviour will be able to become more sophisticated and reliable, increasing the degree of network automation.

These benefits are due to RINA's effort in separating mechanism from policy; that is, extracting invariants through all layers of the communication stack. Gains in simplicity compared to all-IP networks will be higher for larger networks with more diversity, like service provider networks. These networks typically feature different segments (access, aggregation, core, interconnection), with different underlying data plane and control plane protocols. In contrast, RINA maintains its generic layer with two protocols, only policies change from layer to layer. The H2020 ARCFIRE project [22] is currently investigating such scenario.

PRISTINE [23] is working on the development of a programmable RINA implementation [24] with a RINA Network Management System. Both tools will be used during the TNC conference to show a demo that provides a simple PoC experiment illustrating the concepts discussed in the paper.

Acknowledgements

The work of the PRISTINE project is partially funded by the European Commission, under Grant Agreement No. 619305. We would like to thank all the PRISTINE members in general and the WP5 partners in particular for the fruitful discussions leading to the work discussed in this paper.

References

[1] S. van der Meer, J. Keeney, L. Fallon. 2015. Dynamically adaptive policies for dynamically adaptive telecommunication networks. Proceedings of the 11th Conference on Network and Service Management (CNSM 2015).
[2] S. Azodolmolky, P. Wieder, R. Yahyapour, 2013. SDN-based cloud-computig networking. ICTON 2013.
[3] J. Schonwalder, M. Bjorklund, P. Shafer, 2010. Network configuration management using NETCONF and YANG. IEEE Communications Magazine, September 2010.
[4] J. Day, I. Matta, K. Mattar, 2008. Networking is IPC: A guiding principle to a better Internet. ACM CONEXT 2008.
[5] R. Watson, 1981. Timer-Based Mechanisms in Reliable Transport Protocol Connection Management. Computer Networks, 5:47–56.
[6] A. Andreyev. 2015. Introducing data centre fabric, the next-generation Facebook data centre Network. Available through: https://code.facebook.com/posts/360346274145943/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/ [Accessed 8 April 2016]
[7] A. Sajassi, J. Drake, N. Bitar, A. Isaac, J. Uttaro, N. Henderickx, 2015. A Network Virtualisation Overlay Solution using EVPN. IETF, L2VPN Working group; draft RFC draft-ietf-bess-evpn-overlay-02.
[8] P. Lapukhov, A. Premji, J. Mitchell, 2014. Use of BGP for routing in large-scale data centres. IETF Network Working Group, draft-lapukhov-bgp-routing-large-dc-07
[9] J. Schoenwalder, 2013. Common YANG data types. IETF, RFC 6991.
[10] M. Bjorklund, 2014. A YANG data model for Interface management. IETF RFC 7223.
[11] M. Bjorklund, 2014. A YANG data model for IP management. IETF RFC 7277.
[12] L. Lhotka, A. Lindem, 2016. A YANG data model for routing management. IETF NETMOD Working Group, draft-ietf-netmod-routing-cfg-21.
[13] A. Shaik, R. Shakir, K. Patel, S. Hares, K. D'Souza, D. Bansal, A. Clemm, A. Zhdankin, M. Jethanandani, X. Liu, 2015. BGP model for service provider networks. IETF interdomain routing group, draft-shaikh-idr-bgp-model-02.
[14] PRISTINE consortium, 2014. D5.1: Draft specification of the common elements of the Management Framework. Available through: http://ict-pristine.eu/?page_id=37 [Accessed 14 April 2016] To be updated in PRISTINE D5.4 (June 2016).
[15] J. Day, 2016. How naming, addressing and routing should work. PSOC Tutorial. Available through: http://pouzinsociety.org/education/rina/mobility_multi_homing_multicast [Accessed 14 April 2016]
[16] H. Fangwei, R. Chen, M. Milligan, Q. Zu, 2015. YANG data model for VXLAN protocol. IETF NVO3 working group, draft-chen-nvo3-vxlan-yang-02.txt.
[17] P. Brissete, H. Shah, Z. Li, A. Liu, K. Tiruveedhula, T. Singh, I. Hussain, J. Rabadan, 2015. YANG data model for EVPN. BESS working group, draft-brissete-bess-evn-yang-01
[18] M. Holness, 2015. IEEE 802.1Q YANG module specifications. IEEE 802.1 working group.
[19] OpenDaylight project, 2015. YANG module for LACP. Avaiable through: https://git.opendaylight.org/gerrit/gitweb?p=lacp.git [Accessed 15 April 2016]
[20] S. Vrijders, V. Maffione, D. Staessens, F. Salvestrini, M. Biancani, E. Grasa, D. Colle, M. Pickavet, J. Day, L. Chitkushev, 2016. Reducing complexity of Virtual Machine Networking. IEEE Communications Magazine, April 2016 issue.

[21] E. Grasa, O. Rysavy, O. Lichtner, H. Asgari, J. Day, L. Chitkushev, 2016. From protecting protocols to protecting layers: designing, implementing and experimenting with security policies in RINA. IEEE ICC 2016.
[22] ARCFIRE Consortium, 2016. H2020 ICT ARCFIRE website. Available through: http://ict-arcfire.eu [Accessed 14 April 2016]
[23] PRISTINE Consortium, 2014. FP7 ICT PRISTINE website. Available through: http://ict-pristine.eu [Accessed 15 April 2016]
[24] V. Maffione, F. Salvestrini, E. Grasa, L. Bergesio, M. Tarzan, 2016. A software development kit to exploit RINA programmability. IEEE ICC 2016.
[24] J. Day, 2007. Patterns in Network Architecture: A retuern to fundamentals. Prentice Hall. ISBN 978-0137063383

Biographies

Dr. Eduard Grasa joined the Optical Communications Group (GCO) of the UPC in 2003, where he did his thesis on software architectures for the management of virtual networks in collaboration with i2CAT, which he joined in 2008. His current interests are focused on the Recursive Internetwork Architecture (RINA), a clean-slate internetwork architecture proposed by John Day. Dr. Grasa is nowadays leading the RINA team at the Distributed Applications and Networks Area (DANA) of i2CAT. He is currently the technical lead of the FP7 PRISTINE project, which is researching RINA in several areas such as security, congestion control, programmability, routing or resource allocation.

Dr. Bernat Gaston holds a graduate in Computer Science Engineering by the Autonomous University of Barcelona (UAB, 2008), a MSc degree in Data Science (Compression, Codification and Security) and a Ph.D. in Distributed & Cloud Big Data Storage (UAB, 2013). In 2007 he joined the Combinatorics, Coding and Security Group, where he worked in several research projects and he completed his thesis on storage optimization for distributed storage systems that received a "Summa Cum Laude" mark. His current interests are focused on Big Data analytics and on the Recursive Internetwork Architecture (RINA), a clean-slate internetwork architecture proposed by John Day.

M.Sc. Micheal Crotty graduated with an honours B.Sc. in Applied Computing in 1995, and completed a M.Sc. through research in 2001. His main interest was in the service management aspects of pervasive computing. Over the last ten years he has worked on several EU and national funded projects. He has worked on service discovery, composition in the Daidalos project, and policy management and configuration in the Daidalos II project. He has worked on the FP7PERSIST project on Personal Smart Spaces as a technical coordinator. He is currently the WP5 leader in the PRISTINE project.

Dr. Sven van der Meer received his PhD in 2002 from Technical University Berlin. He joined Ericsson in 2011 where he is currently a Master Engineer leading a team that will enhance the capabilities of Ericsson's OSS products. Most of his current time is dedicated to designing and building advanced policy systems that can be used to direct the behaviour of complex event management systems. In the past, Sven has worked with Fraunhofer FOKUS, Technical University Berlin and the Telecommunication Software and Systems Group (TSSG) leading teams and projects, consulting partners and customers, and teaching on university level.

Miguel Angel Puente received his M. Sc. in Telecommunications engineering from the Universidad Politécnica de Madrid (UPM) in 2012. In this period he also completed an information technology master degree at the University of Stuttgart (2010-12). Since 2012 he is with Atos Research & Innovation (Spain), where he is involved in European research projects addressing RINA, 5G, LTE, Cloud Computing, Mobile Cloud/Edge Computing and QoE/QoS optimization among other topics. From 2014 he is a PhD candidate at UPM.