Guaranteeing QoS Requirements in Long-haul RINA Networks

Sergio Leon[1*], Jordi Perelló[1], Davide Careglio[1] and Miquel Tarzan[2]**
[1] *Universitat Politècnica de Catalunya (UPC)-BarcelonaTech,*
Jordi Girona 1-3, 08034 Barcelona, Spain
[2]*Internet Architecture and Services, Fundacio i2CAT*
*E-mail: *slgaixas@ac.upc.edu, **miquel.tarzan@i2cat.net*

ABSTRACT
In the last years, networking scenarios have been evolving, hand-in-hand with new and varied applications with heterogeneous Quality of Service (QoS) requirements. These requirements must be efficiently and effectively delivered. Given its static layered structure and almost complete lack of built-in QoS support, the current TCP/IP-based Internet hinders such an evolution. In contrast, the clean-slate Recursive InterNetwork Architecture (RINA) proposes a new recursive and programmable networking model capable of evolving with the network requirements, solving in this way most, if not all, TCP/IP protocol stack limitations. Network providers can better deliver communication services across their networks by taking advantage of the RINA architecture and its support for QoS. This support allows providing complete information of the QoS needs of the supported traffic flows, and thus, fulfilment of these needs becomes possible. In this work, we focus on the importance of path selection to better ensure QoS guarantees in long-haul RINA networks. We propose and evaluate a programmable strategy for path selection based on flow QoS parameters, such as the maximum allowed latency and packet losses, comparing its performance against simple shortest-path, fastest-path and connection-oriented solutions.
Keywords: Data transport networks, long-haul, RINA, QoS

1. INTRODUCTION

A plethora of innovative distributed applications has been developed in the last years, each one with specific communication requirements. This has led to the evolution of networking scenarios, especially transport network infrastructures, which need to evolve hand-in-hand with new requirements. The networking panorama has considerably changed from the Internet origins, where only few distinct applications populated the network. Currently, the network has to provide connectivity between billions of end users, running a large variety of distributed applications, also allowing their mobility in most cases. With the wide use of smartphones and the increasing number of connected devices, as well as the future arrival of 5G networks, the network is also encountering new changes with services growing closer to the users. However, the TCP/IP-based Internet model cannot cope with such unprecedented evolution, in part due to its static layering and almost complete lack of a built-in QoS support.

As a replacement to the current TCP/IP Internet model, the Recursive InterNetwork Architecture (RINA) [1] proposes a new recursive and programmable networking model. With a recursive layering divided in networking scopes, complete layer configurability and full QoS support, RINA not only adapts to the current networking requirements, but also provides an environment capable of evolving with them. RINA layers share all the same Application Programming Interface (API), and with this API and the recursive nature of RINA, it becomes easier to ensure QoS guarantees. On the one hand, lower layers, with their smaller and more manageable environments, can easily be informed of the requirements of upper layers and take the necessary steps to ensure them. On the other hand, upper layers can be duly informed about any problems arising in lower layers that could prevent meeting those requirements. Taking advantage of this dynamic knowledge, network managers can improve and easily automatize the configuration of each layer to get its full potential. In this context, this work focuses on the assurance of QoS guarantees in long-haul RINA networks, where traffic flows have to traverse large physical distances end-to-end.

The rest of this paper is presented as follows. In Section II, we shortly introduce RINA and its QoS support. In Section III, we focus on the relation between QoS requirements and the selected path. In Section IV, we propose a QoS aware path selection method based on a heuristic approach. Section V compares our path selection method versus simple shortest-path, fastest-paths and versus a connection-oriented scenario. Finally, Section V concludes the paper.

2. THE RINA ARCHITECTURE

The Recursive InterNetwork Architecture, RINA, is a clean-slate architecture for computer networking based on the idea that networking is distributed inter-process communication (IPC) and only IPC [1][2]. Unlike the TCP/IP protocol stack, RINA presents a single type of layer, called Distributed IPC Facility (DIF), which repeats as many times and levels as deemed necessary by the network designer. These DIFs are capable of performing any of the functions needed to provide IPC services to upper layers' processes while offering a common set of

DOI 10.1109/ICTON.2017.8025071

services (data-transfer, security, etc.) and APIs to these services. While providing all DIFs the same functionality, they can be fully adapted to different requirements via policies, namely, sets of variable behaviours that can customise the different mechanisms available in the IPC processes (e.g., enrolment, routing, namespace management, scheduling, etc.). This programmability allows network administrators to properly configure each DIF with the policies that better adapt to its scope, operating environment and offered levels of service.

The creation and management of DIFs is performed by the Network Management System (NMS), a distributed application composed of one or more Manager(s), together with the Management Agents (MA) located at each RINA device [3]. The NMS is responsible of the creation of IPC processes in the devices and their configuration, according to the requirements of each DIF. Thanks to the recursive layering of DIFs and the control of the NMS, changes on the connectivity of lower layers can be completely hidden or at least their negative effects mitigated. In addition, the use of a common protocol for management (called CDAP) greatly simplifies the management of RINA networks compared to legacy architectures.

The approach of RINA towards QoS assurance is achieved by defining QoS Cubes, namely, QoS classes for flows providing statistical bounds on multiple metrics like latency or losses. While providing the same kind of services, the characteristics of those vary from DIF to DIF. In RINA, each DIF defines its own set of supported QoS Cubes and aims to ensure them under normal network conditions. This allows for applications and upper processes to request specific bounds on the experienced end-to-end QoS of flows, afterwards mapped to the closest valid QoS Cube provided by the DIF. To ensure QoS, the behaviour of a DIF must be programmed via appropriate policies, so as to adapt to the specific operational environment. Specifically, two key policies toward effective QoS assurance are the scheduling and forwarding ones.

The scheduling policy in particular becomes crucial when a network suffers high congestion, since it decides on how to differentially treat the incoming packets at an IPC process. By means of QoS Cubes, the specific QoS requirements of each traffic flow are well known, which allows smarter resource sharing of network resources. To this goal, the use of policies based on the idea of degradation of quality (ΔQ) have been proposed [4][5]. ΔQ policies consider all parameters related to the quality of flows, providing differentiated QoS assurance in terms of delay and losses, separately. With ΔQ-based scheduling policies [6], a fine-grained QoS differentiation per flow can be enforced, delivering appropriate service level to most sensitive (i.e., top priority) flows, but without excessively worsening the service levels experienced by such less sensitive (i.e., lower priority) ones.

While an appropriate scheduling policy has a great effect in congested (or overbooked) network scenarios, queuing latencies incurred by packets at network nodes due to congestion may be negligible against the propagation delays over long end-to-end physical routes, particularly in long-haul transport network infrastructures. This effect highlights the importance of the forwarding policy toward providing QoS assurances, as the selected path may affect the viability of fulfilling the service requirements of flows. In this regard, the selection of the path with the smallest hop count may not be enough, even selecting the minimum latency between two points. On the contrary, in this context of providing QoS assurances, it becomes important to consider how other flows are placed in the network and treated by the scheduling policy at network nodes, as this also has an effect on the final degradation suffered by the flow.

3. PROPOSED PATH SELECTION METHOD

As commented above, the propagation delay in long-haul transport networks can arise as the main factor in latency degradation. This delay not only can be very substantial, but also can largely vary depending on the chosen path, up to the point that the path traversing less nodes (the shortest path), may not be the one ensuring the smallest latency. This propagation delay cannot be completely avoided, as even a direct link would impose some latency, but we can minimize it considering other metrics more related to it (e.g. link distance). Doing it for all flows, however, would not only increase the length (in hops) of some paths, but also the amount of flows traversing the network links comprised in fastest routes across the network (typically those in the network central region), which would lead to congestion and excessive packet losses. Indeed, there exists a trade-off between delay, losses and bandwidth utilization, which is the rationale behind ΔQ. Given the specific requirements of a QoS Cube, it is easy to know if a given path would assure or potentially fail to support it, considering not only the path latency but also the maximum degradation that any hop in the path would add upon congestion.

An optimal approach to QoS forwarding should consider all candidate flow setups in a connection-oriented way, where all flows with similar requirements between the same pairs of nodes are aggregated over the same path, so that their QoS requirements are ensured even in the worst congestion scenario. This approach is not always viable, however, not only due to the high complexity necessary to contemplate all possible candidate paths, but also due to the number of forwarding entries required in connection-oriented scenarios. Considering that, we propose an intermediate approach where, for each QoS Cube, each destination is reached by a forwarding tree ensuring that the required QoS levels in its paths from all other nodes are met. To avoid adding too much degradation to the shortest flows, we also impose limits in the additional quality degradation that flows can suffer with respect to their best path. Limiting forwarding to only these valid trees ensures that QoS requirements are met for all flows, while requiring only simple forwarding entries per destination and QoS Cube.

Furthermore, as multiple valid trees are available for each destination at the embedding stage, rather than only the best ones, that gives us a larger solution space that allows for a more optimized usage of resources.

To compute a solution for this problem, we first need the set of valid trees for each pair of destination and QoS Cube. As these sets can be really large depending on the network, we instead can consider only a small fraction of those, selected at random. For that, we use the algorithm depicted in Fig. 2 to compute forwarding trees for each destination that ensure QoS requirements. With the expected traffic matrix, we also know the required capacity of each tree for each link in the network. Now, in order to solve this problem, we propose the use of the Simulated Annealing (SA) metaheuristic depicted in Fig. 3. In this SA, we consider our solution space to be that of all combinations of tree-sets that contain one valid tree for each destination and QoS Cube. Given a solution, its N-neighbourhood is compressed by those solutions that result in replacing up to N trees in it. Then, in order to compute the weight of a solution, we take the capacity required in the most loaded link plus **α** times the total capacity requested for a small **α**. **α, $\mathbf{k_{max}}$**, **$\mathbf{max_{neis}}$** and the temperature function (temp(k)) being dependent on the size of the network and supported traffic.

- Start with empty tree with destination as *root*
- While all nodes are still not in the tree:
  - Compute the distance* from *root* to the remaining nodes, extending the current tree, and select the farthest one (*n*).
  - Select a random valid path** from *n* to *root* and add its links to the tree.

* Metric dependent on the QoS requirements, commonly a joint metric contemplating hop count and latency.
** Any path that extends the tree, meeting the QoS Cube requirements and under allowed degradation from optimal.

Figure 2. Pseudo-code for computing random valid trees per destination and QoS Cube

```
$s_{best}$ = $s$ = random(SolutionSpace)
For $k$ = 1 … $\mathbf{k_{max}}$
   t = random(SolutionSpace);
   for q = 1 … $\mathbf{max_{neis}}$ do
      t = best( t, random(N-neighbour(s) )
   if $t$.cost < $s_{best}$.cost then : $s_{best} = s = t$;
   else if accept ($t$, $s$, $k / \mathbf{k_{max}}$) then : $s = t$;
return $s_{best}$;
accept($t$, $s$, $k'$)
   return (t.cost < s.cost)
      or exp((s.cost – t.cost)/temp(k’)) < uniform(0,1);
```

Figure 3. SA metaheuristic pseudo-code for computing sub-optimal solutions

4. NUMERICAL RESULTS

In order to test the proposed solution, we used a 17-Node backbone network (depicted in Fig. 4) based on a German backbone network (DTAG). We considered a uniform traffic matrix where the offered traffic between each pair of nodes is divided according to their QoS requirements. To this end, only their latency requirements are considered for simplicity. The resulting traffic profile is as follows: 10% of QoS Cube A (minimum latency); 30% of QoS Cube B (low latency); and 60% of QoS Cube C (no strict latency requirement). Instead of fixing the amount of traffic and minimizing overbooking, we have considered the opposite case, scaling rates to the capacity of the most loaded link.

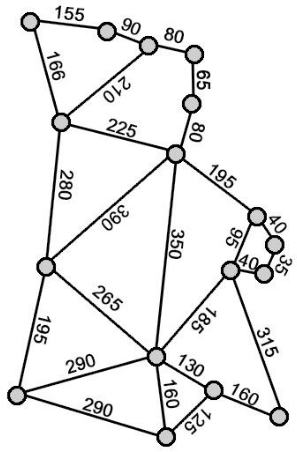

Figure 4. 17-Node backbone network based on German backbone network (DTAG)

First, we aim to analyse how simple QoS differentiation behave against giving the same treatment to all QoS classes (either using the shortest or fastest paths). We considered three metrics, one per QoS Cube. For QoS Cube A we consider distance (in Km) as metric. For QoS Cube B we consider hops + distance/20 as metric. Finally, for QoS Cube C consider the number of traversed hops as a metric. With these metrics, distance-vector routing is used to select the best paths for each QoS Cube over the network. As comparison, we considered two solutions using only the shortest (hops) and fastest (distance) paths, respectively. Fig. 5 shows the required capacity on each link, as well as the worst case and average for each solution, relative to the usage in the most loaded link in the shortest path solution. The first notable result is that, while fastest paths improve the delivered QoS, this is achieved at expenses of requiring a higher capacity on some links, especially those covering smaller distances (note that the peak is not even on one of the most loaded links in other solutions). On the other hand, the separation per QoS, not only provides better services to the more requiring flows, but also reduces slightly the load on the most used link. This has an interesting meaning as, while specific to this network and the traffic matrix, results imply that considering QoS does not have to worsen, but could even improve the usage of the network.

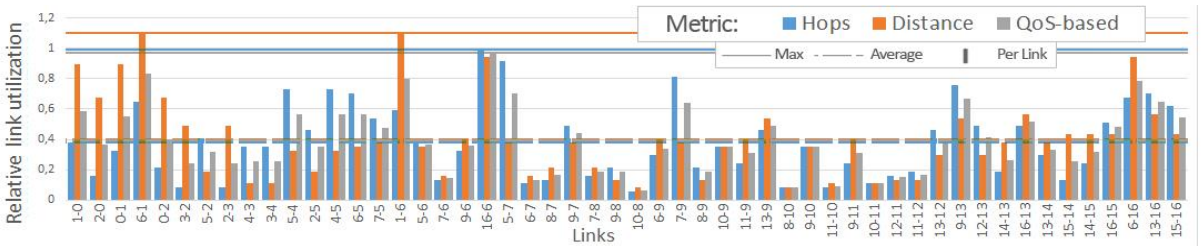

Figure 5. Link usage comparison between metrics: hops, distance and the described per-QoS metric.

After establishing the benefits of using different metrics per QoS, we aim to quantify how the freedom to decide whether to allow path degradation (within QoS requirements) can improve the network usage, identifying at the same time the drawbacks of the proposed approach with respect to a traditional connection-oriented one. We compare the results from the previous QoS-based metric with those of connection-oriented and the proposed approach, both with different accepted path degradation for QoS Cube C. In this regard, Fig. 6 shows significant improvements, greatly reducing the required capacity on the most loaded links when accepting either 1 or 2 extra hops. Given these results, if for example we could provide 100Mbps between peers using the first solution, this would mean that one extra hop for QoS Cube C could improve the load to 120Mbps per flow, 150Mbps if we allowed 2 extra hops instead. At the same time, this is done without dramatically affecting the average load, as seen in Fig. 7, meaning that only few flows need to be degraded to reach such network performance improvements. However, allowing higher degradations, while still provides some improvement, results in a huge increment of the average utilization, as a high number of flows needs to be degraded to reach that improvement. In addition, comparing connection-oriented and tree-based solutions, we see that both have the same worst cases, only differing in a slightly higher average load with the use of trees.

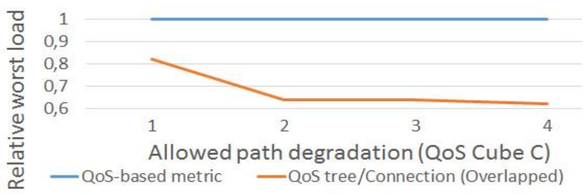

Figure 6. Comparison of worst link utilization between QoS-based metric, QoS trees and connection-oriented approaches

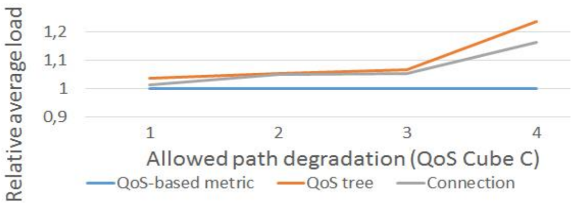

Figure 7. Comparison of average link utilization between QoS-based metric, QoS trees and connection-oriented approaches

5. CONCLUSIONS

When considering QoS differentiation in an Internet-wide network, it is important to provide quality guarantees in each networking level. When covering large distances, this has a great importance given the large path latencies. In this work, we have discussed the use of metrics per-QoS in long-haul RINA networks, as well as proposed a heuristic for path selection. The proposed heuristic results in paths that minimize the worst load in the network, while providing paths that ensure QoS guarantees under normal utilization. In future work, the effects of network and traffic changes will be considered, as well as mechanisms to dynamically adapt to changes in the traffic and recover after failures.

ACKNOWLEDGEMENTS

This work has been partly funded by the Spanish National project SUNSET (FEDER-TEC2014-59583), and the European Commission through the H2020 ARCFIRE project (Grant 687871).

REFERENCES

[1] J. Day, I. Matta, and K. Mattar. “Networking is IPC: A Guiding Principle to a Better Internet”. In CoNEXT’08: Proceedings of the 2008 ACM CoNEXT, pp. 1-6, New York, NY, USA, 2008. ACM.
[2] J. Day, “Patterns in network architecture: a return to fundamentals”, Prentice Hall, January 2008.
[3] Eduard Grasa, et al., “Simplifying multi-layer network management with RINA”. In TNC16, Prague, Czech Republic, June 2016.
[4] N. Davies, “Delivering predictable quality in saturated networks,” Technical Report, September 2003, available online at http://www.pnsol.com/public/TP-PNS-2003-09.pdf
[5] N. Davies, J. Holyer and P. Thompson, “An operational model to control loss and delay of traffic in a network switch,” in third IFIP Workshop on Traffic Management and Design of ATM Networks, 1999.
[6] S. Leon Gaixas, et al., “Assuring QoS Guarantees for Heterogeneous Services in RINA Networks with ΔQ”. In NetCloud 2016, Luxembourg, December 2016.