Follow the Model: How Recursive Networking Can Solve the Internet's Congestion Control Problems

Michael Welzl, Peyman Teymoori, Stein Gjessing, Safiqul Islam
Department of Informatics, University of Oslo
Email: {michawe, peymant, safiquli, steing}@ifi.uio.no

***Abstract*—The Recursive InterNetworking Architecture RINA describes a new way to look at networking; it offers a point of view that is fundamentally different from today's networks. This paper explains how designing congestion control strictly in line with this model almost automatically leads to a conceptually cleaner, and quite possibly altogether better design than what we have in the Internet today. We give an overview of how far the OCARINA research project has come with the development of the RINA congestion control elements that follow from this design. Then, we conclude with an explanation of how the shift in thinking that RINA suggests can be applied to more gradual Internet developments related to congestion control.**

***Index Terms*—congestion control, RINA, recursive networks**

I. Introduction

Internet congestion control has many issues: unnecessary latency, the inability to quickly saturate a bottleneck's capacity, lack of a meaningful fairness notion, difficulty in efficiently utilising more than one network path, and ignorance of the underlying medium, which renders controls unable to react to swift and drastic capacity changes (as expected in 5G networks). Nevertheless, the Internet works—but it really *only just works* [1].

In this paper, we briefly discuss these problems, and explain why they are an inevitable result of the Internet's architectural design. Beginning with ideas from the X-Bone's Virtual Internet Architecture, the Recursive Network Architecture (RNA) [2], the Recursive Internetworking Architecture (RINA) [3] and the compositional architecture described in [4] (which, in turn, is based upon RINA), have all recognized that:

1) today's reality of layering is quite different from the original layer model envisioned for the Internet;
2) the essential role of layers should *not* be to represent a collection of a specific set of functions (i.e., the link layer carries out typical link layer functions, the network layer carries out typical network layer functions, and so on), but instead to define a communication *scope* (e.g., IP routing does not need to know about MAC addresses);
3) some essential communication functions should exist at every layer, although they might be instantiated differently depending on the environment. This repetition of functions constitutes the recursive aspect of these architectures.

RINA *demands* a form of congestion control that solves the problems above (it satisfies all the requirements that we derive from these problems in the next section), and this design results from decisions that seem obvious in a RINA context. We will make this clear with a discussion of an Internet-like RINA configuration in Section III, where we then proceed with an overview of our developed solutions and ongoing work in the OCARINA[1] research project. Section IV concludes.

II. Problems of Internet Congestion Control

Here we discuss congestion control issues and derive requirements from them. We cover the problems very briefly because they are generally well known and amply discussed in the broader congestion control literature.

A. Latency

With the exception of Explicit Congestion Notification (ECN), which does not see any significant level of deployment as of yet, TCP relies on implicit feedback: once a bottleneck's capacity is exceeded and queues have grown to the point of overflowing, a packet is dropped and this is taken as a sign of congestion. In the process, TCP creates latency by making queues grow. Countless approaches to reduce this latency exist; to name one example, one of the earliest ideas was to back off in response to delay growth (TCP Vegas [5]). All practical solutions rely on feedback from queues, and hence need to create at least a little bit of latency before they can react.
Derived requirement: *R1: Do not create latency.*

B. Capacity utilization

Approximately 15 years ago, TCP's inability to saturate a long-delay, high-capacity path with a bulk data transfer was considered one of the biggest problems in congestion control. This problem was tackled by many research teams [6], and it is now solved by the wide deployment of experimental congestion control mechanisms such as Cubic [7] (one of the proposals from that time, enabled by default in Linux for many years), and, more recently, BBR [8]. Nowadays, the majority of data transfers are short or application-limited, and the focus is on reducing latency in general. Latency, as perceived by end users, is not only a function of the time that a single packet needs from one end to the other—for web surfing, for example, the prevalent factor influencing the user-perceived latency is the flow completion time. This, in turn, depends on the ability to quickly saturate the bottleneck capacity, so as to transfer all the data within the minimum number of Round-Trip Times (RTTs). Cutting round-trips is the motivation for a

[1]https://www.mn.uio.no/ifi/english/research/projects/ocarina/

larger initial window of TCP [9], and it is also a core aspect of the design of QUIC [10].
Derived requirement: ***R2: Quickly utilize available capacity.***

C. Fairness

Fairness is an aspect of congestion control which faces a gap between theory and practice. On the practical side, TCP equally shares the bottleneck capacity between multiple long-lasting permanently-sending connections that have the same RTT. Often, this type of fairness is not very meaningful [11] (e.g., would it make sense to equally divide the capacity between a VoIP and P2P file sharing traffic?). On the theoretical side, Network Utility Maximization (NUM), introduced by Kelly et al in [12], is a well-established and widely accepted notion of fairness in networks. Here, the idea is to maximize the utility per sender under the constraint of the available network resources; with a logarithmic utility function, NUM yields the well-known "proportional fairness". NUM optimizes the send rate at end hosts according to the cost it receives from the network. Cost should be an additive measure, but this is not available as an input to congestion control in the Internet.
Derived requirement: ***R3: Implement NUM.***

D. Multipath communication

TCP was originally designed to operate across a single end-to-end network path with not much flexibility in routing. If in-network load balancing mechanisms such as Equal-Cost Multi-Path (ECMP) would operate on a per-packet basis, they would introduce reordering in TCP connections—but TCP is sensitive to reordering and easily interprets it as a sign of congestion (the latest TCP versions address this problem to some degree). Therefore, ECMP usually balances connections rather than packets, which requires routers to look at the transport header.

With the growing pervasiveness of multi-homing (e.g., WiFi and cellular connectivity of cell phones), using multiple paths at the same time has become attractive. Multipath TCP (MPTCP) [13] achieves this with a *coupled* congestion control mechanism, where sub-flows of an MPTCP connection attempt to better use the total available capacity while avoiding to be more aggressive than a single flow in case they share a bottleneck. Sub-flows of one connection can also be placed on different paths by ECMP, enabling more efficient network usage even with single-homed end systems [14].

Despite its obvious benefits, the MPTCP approach to multipath communication has a limitation: ECMP only uses shortest paths. Combining congestion control with an in-network view could enable the occasional use of longer paths. This idea has been proposed in [15], but the proposed scheme is limited in that each load balancing point can only use congestion information of its immediate outgoing links.
Derived requirement:
R4: Support efficient multi-path usage inside the network.

E. Reacting to link layers

We have already discussed that (not MP-)TCP assumes a single network path and is sensitive to reordering. There are more implicit assumptions: TCP expects that all packet losses indicate congestion, and it does not expect quick changes of the physical bottleneck capacity. A large body of research on cross-layer optimization [16] is testimony to TCP's poor performance when these assumptions are wrong—but none of the proposed solutions are deployed in the Internet (we will briefly discuss the reasons for this failure in Section IV).
Derived requirement: ***R5: Use knowledge of lower layers.***

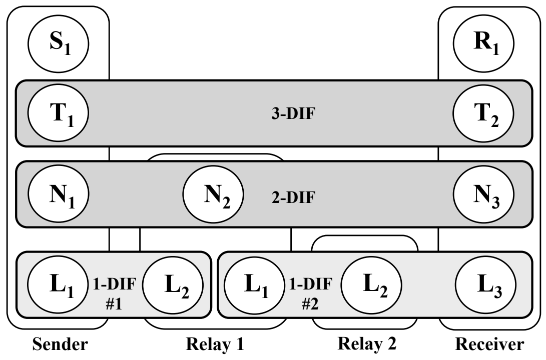


Fig. 1: A RINA configuration which resembles an Internet scenario, with the 1-DIFs as link layers, the 2-DIF as the network layer and the 3-DIF as the transport layer.

III. Developing RINA Congestion Control

We explain our design with a RINA configuration that mimics the Internet with its "end-to-end"[2] congestion control. One of RINA's fundamental principles is that computer networking is just Inter-Process Communication (IPC). Communicating entities are called "IPC Processes (IPCPs)", and the collection of such entities in a layer is accordingly called "Distributed IPC Facility (DIF)". DIFs provide a common set of APIs to other DIFs. Figure 1 shows a simple RINA network where a sender S_1 talks to the IPCP T_1 of the layer immediately below it to transmit data to the other end, eventually reaching the receiver R_1. IPCPs of this DIF in turn make use of lower DIFs—if this were the Internet, the 1-DIFs would be link layers, the 2-DIF would be the network layer, and the 3-DIF would be the transport layer; relay 1 would be an IP router and relay 2 could, e.g., be an Ethernet switch.[3]

Considering congestion control in this setup, it might be tempting to think that it would only be implemented at the transport layer, i.e. in the 3-DIF between T_1 and T_2.

[2]The "end-to-end" argument is an oft-quoted design guideline; it recommends to carefully consider the trade-offs between implementing functions inside the network versus end systems, and it advises against implementing application-specific functionality in the network [17]. It is common to misquote the argument to say that the network must be kept "as dumb as possible". While such design would certainly be in line with the argument, it is a too strict interpretation—e.g., it would also forbid using complex routing algorithms. Congestion is network-specific, not application-specific; it should therefore arguably be resolved as close as possible to where it happens.

[3]The primary of layers in RINA is to establish a scope of communication. Because the 2-DIF and the 3-DIF have equal scope in the diagram, a typical RINA configuration would normally not contain the 3-DIF, but have its functionality included in the 2-DIF. We show them as separate DIFs here to facilitate the discussion of RINA vs. the Internet.

Accordingly, to emulate the Internet, RINA congestion control would have to be instantiated as empty policies[4] in the 1-DIFs and the 2-DIF. This however, draws a very incomplete picture of congestion control in the Internet; next, we will see how a closer look can show us the way to a better design.

A. From the Internet...

Starting at the 1-DIFs (link layers), there is in fact usually some form of overload (congestion) control implemented there, called Medium Access Control (MAC). We need this in today's networks for various reasons, in addition to congestion control at higher layers—this need is a testament to the fact that, as recursive architectures tell us, the same basic functionality is required at all layers. Without recognizing this necessity as part of the architecture however, MAC mechanisms are developed separately from upper-layer congestion control, and there is no vertical information flow between these controls; link layer developers make static design-time assumptions about TCP running on top of their technology, and vice versa. This narrows the design space on both ends, as future mechanisms should be able to operate over (or under) previously developed systems. MAC algorithms are usually only concerned with the transmission of a single frame at a time—but senders at that layer would be in a good position to derive a longer-term average rate. Congestion control at the 2-DIF could benefit if each 1-DIF could tell it a sending rate, but such interfaces are not available—and: is there even any congestion control at the 2-DIF (i.e., the network layer) at all?

Internet congestion control operates at the transport layer with the exception of Active Queue Management (AQM), which is at the network layer. This layer split becomes particularly clear with ECN, which uses two bits of the IP header such that AQM algorithms can easily access them, yet the "ECN Echo" bit, which routers do not need to access, is in the TCP header. Every transport protocol that needs ECN marks at the sender side must include this feedback as part of its design; IP header space has become a too valuable resource to be spent on information that routers do not need to see.

Because the control loops operate at the transport layer, multiple connections between the same pair of hosts work independently, competing for resources in the network. Solutions to this problem (such as the Congestion Manager [18] or other congestion control coupling approaches [19, 20] and [21, 22]) have a hard time getting deployed because ECMP may put transport connections on separate paths. Since congestion control was primarily implemented into the transport layer, there is no explicit interaction between congestion control and ECMP—instead, again, static design-time assumptions are made.

B. ... to RINA — "following the model"

As we have seen, an Internet-like configuration is quite awkward in a RINA setting: in the example in Figure 1, there is no communication about congestion control between the 1-DIFs and the 2-DIF, leading to static design assumptions (e.g., about the 2-DIF, by 1-DIF designers). The 2-DIF contains policies (AQM) which make no sense on their own, but assume a certain operation of the 3-DIF (TCP or similar).

RINA prescribes the same vertical interface between DIFs everywhere, and DIFs do not make static assumptions about upper or lower DIFs. Thus, each DIF should have congestion control in a form that is suitable for its own specific environment. This means that we have multiple control loops along a path from a sender to a receiver, and the number of these loops is a matter of configuration—thus there is a need to consider stability as these loops interact. Information can be handed over vertically between DIFs to form a push-back signal.

In earlier work, we have carried out two preliminary analyses of RINA congestion control. In [23], we investigated what would happen if we would simply install TCP-like congestion control in RINA—and found that some improvements happen just as a result of network configuration: consecutive control loops yield a behavior similar to a Performance Enhancing Proxy (PEP),[5] and if a DIF operates on aggregates coming from DIFs above it, it can yield benefits similar to mechanisms that couple congestion controls. In [25], we investigated recursive feedback for a broad class of loss-based congestion controls; this work showed that using overflowing queues as a feedback method between DIFS can produce very large delays. These delays occur in addition to the delays caused by the congestion controls themselves (see the discussion of **R1**).

From these early works, we have concluded that installing a control loop at every DIF has the potential to yield many benefits, but a congestion control which uses implicit (loss- or delay-based) feedback is not a recommendable approach at all. Due to the multitude of queues that can individually be pushed to the limit by each loop in a RINA network, the delay from a TCP-like congestion control could in fact become far worse than in the Internet. Therefore, any reasonable form of RINA congestion control needs to use explicit feedback that can be produced early, before queues grow.

Earlier approaches on explicit feedback congestion control such as XCP [26] and RCP [27] require precisely the same behavior in all routers and end systems, with no room for flexibility, and they need several bits in packet headers. The Internet's notion of ECN as an indicator of congestion seems more convenient—and the much more recent Data Center TCP (DCTCP) has introduced the use of consecutive single-bit ECN marks as a compound multi-bit congestion signal [28]. However, ECN cannot easily be used as a signal before the capacity is exceeded. In the DCTCP way of using ECN, marks are assumed to come from only one congested queue at a time, and the marking probability reflects the degree of congestion

[4]In RINA terminology, the general functions that exist at every layer are "mechanisms"; mechanisms can be customized/programmed differently per DIF; this is done via "policies".

[5]PEPs are in-network mechanisms or devices which carry out operations specific to transport layer protocols, typically TCP, to improve their performance [24]. A TCP link splitter is a typical example: by spoofing IP addresses and acting like a receiver towards the sender and like a sender towards the receiver, two shorter, more reactive connections are created. This is often beneficial, e.g. in the face of heavy packet loss on only one part of the path.

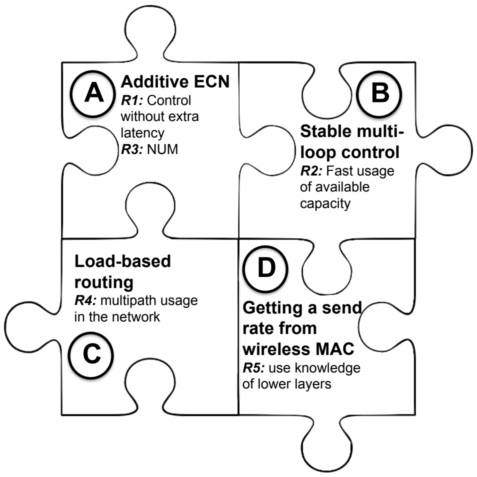

A
Additive ECN
R1: Control without extra latency
R3: NUM
B
Stable multi-loop control
R2: Fast usage of available capacity
Load-based routing
R4: multipath usage in the network
C
D
Getting a send rate from wireless MAC
R5: use knowledge of lower layers

Fig. 2: Pieces of the RINA congestion control puzzle.

at that queue. If we mark earlier, marking at multiple relays is the norm and not the exception, and the result is distorted.

C. Elements of a solution

Making ECN additive is the first piece of the congestion control puzzle (piece **A** in Figure 2) that we have developed in the OCARINA project. Assume that each router i marks packets of a flow with probability p_i on their path towards the destination. In this case, the end-to-end marking probability, p_r, is equal to $1 - \prod_i(1 - p_i)$. If p_i is much smaller than 1, then $\sum_i p_i \approx 1-\prod_i(1-p_i)$ meaning that p_r is approximately additive, and this has been the justification behind using ECN in the NUM framework in previous work (e.g. See [29]). However, in modern congestion controllers such as DCTCP that converge at higher marking probabilities, ECN does not provide the additivity property. In [30], we have shown that, by lower- and upper-bounding the marking probability at each router and applying a logarithm when interpreting the ECN signal in the end system, we can turn ECN into an *additive* signal that conveys a useful measure of congestion from multiple relays. Instead of p_r, senders just need to use $-\log_\phi(1-p_r)$ as the congestion signal where ϕ is the system parameter affecting the equilibrium marking probability.

As with DCTCP, this is even achievable with commodity router hardware that supports the RED AQM algorithm. An additive ECN signal can be related to an additive path cost—this allows us to i) use ECN as a feedback signal before a queue even grows, i.e. without any extra latency (**R1**), and ii) implement NUM, which is based upon an additive path cost signal (**R3**). We note that, while [30] proves the ability to implement NUM, we have not yet validated our claim that our additive ECN signal can indeed be used before a queue grows; this is planned as one of our immediate next steps.

In [30], we configured RED in a special manner: we set the minimum threshold to 1, maximum threshold to 220 KB, and max$_p$ to 1. This means that, similar to DCTCP, we disable queue averaging, but different from DCTCP, we use a marking probability that grows with the queue length instead of a step function. This allows a wide range of marking probabilities, from 0 to 1, and marking can start when there is at least one backlogged packet in the queue to react faster to congestion (DCTCP requires a larger value due to its step function). Feeding the end-to-end marking probability into our logarithmic function yields

$$-\log_\phi(1-p_r) = -\log_\phi\Big(\prod_i(1-p_i)\Big) = -\sum_i \log_\phi(1-p_i)$$

which is equal to the sum of the "cost" of routers in the path.

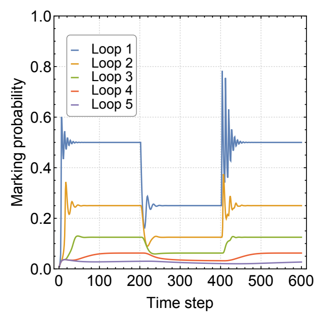
Next, connected to puzzle piece **A**, we need a congestion control mechanism based on the additive ECN feedback that can address **R2** (quickly converging to full utilization of the available capacity) and can be shown to be stable when used with multiple consecutive or overlapping control loops—piece **B** of our puzzle. We found Logistic Growth to provide a suitable answer, and accordingly developed a congestion control mechanism, "Logistic Growth Control" (LGC), which we described and analyzed in [31]. Assuming rates are normalized, LGC updates the send rate using

$$x = x + xr(1 - x - p_r) \qquad (1)$$

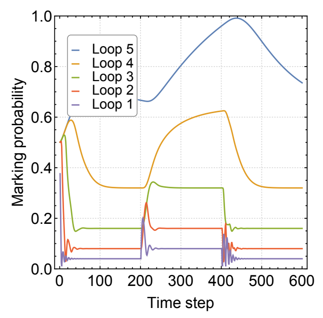
where x and r denote the send rate and the growth rate, respectively. Accordingly, in [30], we also discuss the application of the additive ECN signal to LGC. Logistic growth has long been considered in models involving multiple loops—specifically, the Lotka-Volterra model which is commonly used to describe the dynamics of competing animal species (each growing according to LG). It is called "predator-prey" model if some species (preys) have positive effects on the growth of some other species (predators). Stability properties and boundary conditions of this model have been established in biology literature. On this basis, we are currently developing a stability proof for our multi-loop LGC.

In (1), 1 is the normalized *carrying capacity*. Chaining another LGC loop to (1), which also models a predator and prey relationship, yields another rate update: $y = y+yr(x-y-p_r)$ where y is the send rate of the flows using (eating) flows in the next loop, and p_r is the end-to-end marking probability in the current loop, not the whole path. In this case, x is the carrying capacity of flows (species) y.

One of the benefits of chaining LGC controllers is that each loop can use the send rate of the next loop immediately in its rate update dynamics. We performed a numerical evaluation: in Fig. 3, we used a chain of Primal-Dual controllers with a logarithmic utility function (see [30] for the definition of this controller), and in Fig. 4, LGC was used instead. Both the controllers were tuned for fastest convergence. In this evaluation, we had 5 loops: in each loop, there are two competing flows, which are aggregated in one flow in the next loop. This implies that the send rate of each flow is (should be) divided fairly between the two flows in the previous loop. Therefore, in the last loop, the two flows compete for the bottleneck capacity, and each one should get half of the capacity; in the previous loop, the two flows that are aggregated in the next loop using one of the flows should also get half of the send rate of that

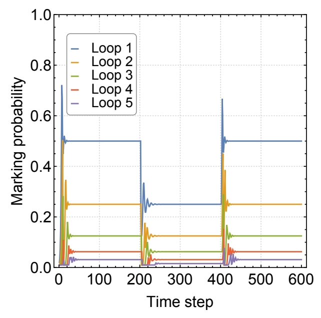


(a) Normalized rate of flows

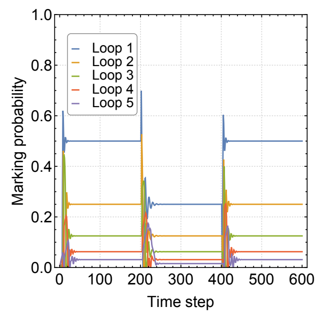


(b) Marking probability of flows

Fig. 3: Chaining 5 loops of a congestion controller with logarithmic utility function



(a) Normalized rate of flows



(b) Marking probability of flows

Fig. 4: Chaining 5 loops of LGC

flow. If we assume a normalized capacity of 1 in the last loop, then last-loop flows get 0.5, each flow in the previous loop gets 0.25, and the values are 0.125, 0.0625, and 0.03125, respectively for the flows in the previous loops. In the figures, we only report the rate of one flow in each loop labeled with "Loop i". At step 200, we cut the capacity of the last loop to 0.5 to observe how fast the chain of congestion controllers can converge. Here, the marking probability is the percentage of marked packets of each flow at the start of the next loop: since we aggregate the two flows of each loop into one flow in the next loop, we assume that the incoming packets from the two flows are queued and then aggregated. The queue, configured with RED as introduced in [30], forms our basis of marking. From the figures we observe that the chain of LGC, because of the direct use of the send rate of the flow in the next loop, can converge faster, and the same happens when the capacity again changes to 1 at step 400.

At the time of writing, puzzle piece **C** (which will satisfy **R4**) is still in early development, but it seems obvious that this mechanism will be able to benefit greatly from piece **A**, i.e. the additive ECN signal that we have developed: if we prescribe symmetric routing for ACKs, then every relay in a DIF obtains a complete picture of upstream (the path segment from the sender to the relay) and downstream (the path segment from the relay to the destination) congestion merely from looking at

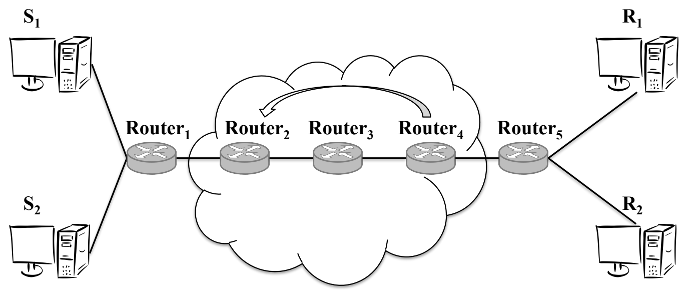


Fig. 5: Aggregate congestion control for two flows (S_1 to R_1 and S_2 to R_2), "flat" view.

ECN signals. A relay can obtain the total amount of congestion on a path as follows: it monitors and averages the fraction of ACKs from the receiver that carry "ECN Echo (ECE)" marks. By subtracting the amount of upstream congestion (the average fraction of incoming packets towards the same destination that already are ECN-marked) from this value, the relay obtains the amount of downstream congestion. Using this result, it can make an informed decision about the outgoing links to choose. This will allow us to go even beyond the in-network resource pooling method in [15], which can only utilize each load balancer's immediate outgoing congestion levels as input.

Last but not least, we can also address **R5** with puzzle piece **D**: if we can obtain the currently available capacity from a link layer, then, e.g., in Figure 1, IPCP L_1 in 1-DIF #2 could tell N_2 how fast it can transmit data towards N_3. This rate is a slightly longer-term estimate than MAC algorithms typically deal with; in [32], we have documented preliminary investigation steps in support of an idea to develop a Machine Learning (ML) model which predicts the usable rate in a WiFi network. While this rate may seem easy to obtain in a downstream single-Access Point scenario, this is not always the case, and an ML model also has use cases for upstream data transfers or wireless mesh scenarios, where the usable rate is much less obvious.

The rate that our future WiFi ML model will give us may oscillate wildly—and directly using the resulting value as the sending rate on a longer network path is probably not advisable. Again, LGC helps, because it converges towards a definable carrying capacity (y), and thus it seems obvious to feed the ML-derived WiFi rate into this variable as a way of interconnecting WiFi links with a larger network.

D. Fan-out: a problem eliminated by recursion

Since congestion control would normally be used at every layer (DIF) in RINA, at some lower DIFs it would operate on aggregates of the flows between IPCPs at upper DIFs. Such aggregation can be quite beneficial, e.g. by avoiding unnecessary competition between flows—we have already discussed some of these benefits in the context of Internet congestion control coupling [18, 19, 21].

Here, we show how the recursive nature of RINA automatically eliminates a potential problem that such per-aggregate congestion control inside the network might have in a "flat"

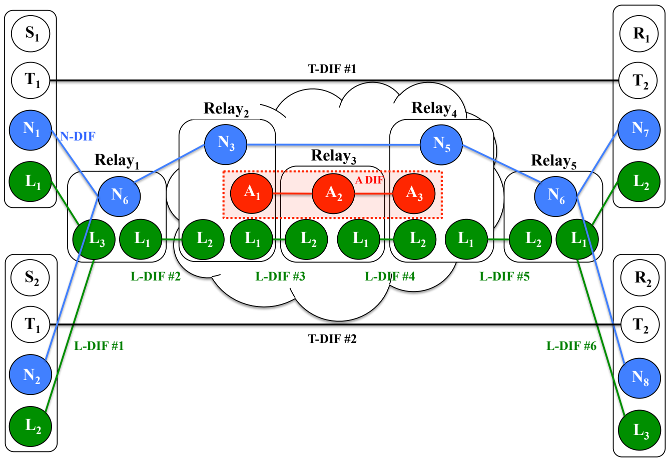


Fig. 6: Aggregate congestion control with RINA.

design. Consider Figure 5: here, the bent arrow indicates a per-aggregate congestion control loop where feedback is sent back from router 4 to router 2 such that router 2 could somehow adjust the transmission rate of incoming traffic (e.g., reduce it by dropping packets). This might work to react to congestion between, e.g., routers 3 and 4—but what about congestion happening between router 5 and receiver R_1, e.g. because a new flow is started? This case, which we call "fan-out", seems tricky: the aggregate control loop should only reduce traffic from sender S_1, but such per-flow operation is hardly scalable.

In Figure 6, we look at the same case in a RINA network. Again, we use layers in a way that mimics the Internet, at least to some degree: we have six "link layers" (L-DIFs), a "network layer" (N-DIF) and two T-DIFs representing the "transport layer". For simplicity, we assume no congestion control (empty policies) in the N-DIF but end-to-end congestion control in the T-DIFs, just like in the Internet without AQM algorithms. We also decided that relay 3 does not need to participate in the N-DIF; translated to Internet terms, it may be a device that operates purely on aggregates (like, e.g., MPLS switches) and does not need to have an IP address. To consider the aggregate control loop from relay 2 to relay 4, we add an "aggregation" DIF (A-DIF).

Like before, the first case of congestion between relay 3 and relay 4 can be handled directly by the A-DIF. In the RINA diagram, it is clear that that the A-DIF is unaware of congestion from relay 5 to receiver R_1—it will only be visible in the N-DIF, with the effect (packet loss in case of a TCP-like control) being noticeable in T-DIF #1 above. In fact, the T-DIFs are unnecessary in the RINA configuration in Figure 6, and it would seem most natural to implement congestion control directly in the N-DIF instead—giving us congestion control at every layer.

To summarize: if congestion appears at a point that is out of scope of the A-DIF, an upper DIF will manage to reduce the rate of the correct flow (with the inevitable disadvantage of having a longer control loop), and if congestion appears within the scope of the A-DIF, the A-DIF will handle it (with a shorter control loop and all its benefits). This shows how congestion control should, indeed, be a "mechanism" in RINA—a function that exists at all layers. We can also see that, if a per-aggregate congestion control mechanism is introduced in the Internet, it should operate *in addition to* TCP's end-to-end congestion control. In a way, we can consider the A-DIF as a larger-scale multi-hop repetition of the function usually carried out by MAC at the link layer, which is also unaware of TCP connections but does not eliminate the need for TCP.

IV. Conclusion

We have seen that a whole host of Internet problems disappear as we develop congestion control in the most obvious way in RINA, almost as a by-product of necessary design decisions in some cases—for example, latency is minimized because we *cannot* reasonably rely on queuing delay and packet loss as feedback signals. With so many problems being addressed by this from-scratch design, one may wonder: is it inevitably the case that, to become more efficient, congestion control must move in the direction that we have outlined in this paper?

A recursive architecture is not only a different design approach, but it changes how we *think* about networks. This change of perspective indeed puts Internet developments in a different light. In [23], we have seen that a PEP-like behavior is natural in RINA. The IETF condemns PEPs as breaking the Internet architecture, yet they yield benefits and they are therefore deployed. It is even common to split TCP connections—despite the disadvantage of requiring full reliability per PEP (hence producing buffering delay). In RINA, there is no reason to always combine reliability and congestion control; it might be worth considering a PEP design that divides control loops without requiring full reliability at every PEP, too (this would require changing the TCP standard).

As we have discussed in Section II, TCP does not work well when its basic operation assumptions fail, e.g. when a link layer drops packets for reasons other than congestion or when a link's capacity changes. Many older research papers tried to solve such problems with ideas of cross-layering [16]—e.g., by using information from the link layer inside TCP—, but none of these ideas ever had any success in the IETF. One common argument for rejection is that TCP must work everywhere, and such a proposal would not work when the link layer that it is designed for is not the bottleneck.

From a RINA-congestion-control perspective, this is an obvious problem: we can only perform link-layer specific congestion control if we apply a control loop that spans this link layer and interconnect it with the layer above. For the Internet, this calls for an IP-layer mechanism that interacts with both the adjacent link layer and TCP. Of course, such design will be limited by what link layer designers allow to access or change on the one hand, and the rigidness of the TCP standard on the other. As we have discussed, AQM algorithms are the only part of the Internet's congestion control that is implemented at the network layer—and the quite recent DOCSIS-PIE AQM algorithm indeed uses such lower-level information [33]. Perhaps a PEP could do an even better job.

It seems worthwhile to nudge the Internet towards a more efficient mode of operation that is (as we argue, by necessity) more in line with RINA's design. In the OCARINA project however, we follow a vision of uncompromising, "real" RINA deployment, using methods such as tunneling, gateways, or even directly "switching-over" when a (presumably short) Internet path is found to fully support RINA. Readers interested in transitioning possibilities may want to consult our first publication on RINA deployment for measurements on "switching over" and further discussion of this matter [34].

Acknowledgment

This paper has benefited from suggestions by John Day and Eduard Grasa. The authors were part-funded by the Research Council of Norway under its "Toppforsk" programme through the "OCARINA" project (http://www.mn.uio.no/ifi/english/research/projects/ocarina/). The views expressed are solely those of the authors.

References

[1] M. Handley, "Why the Internet only just works," *BT Technology Journal*, vol. 24, no. 3, pp. 119–129, Jul 2006.

[2] J. Touch, Y. Wang, and V. Pingali, "A recursive network architecture," ISI, Tech. Rep. ISI-TR-626, 2006. [Online]. Available: https://www.strayalpha.com/pubs/isi-tr-626.pdf

[3] J. Day, *Patterns in Network Architecture: A Return to Fundamentals*. Prentice Hall, 2007.

[4] P. Zave and J. Rexford, "The Compositional Architecture of the Internet," *Commun. ACM*, vol. 62, no. 3, pp. 78–87, Feb. 2019.

[5] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," in *Proc. SIGCOMM 1994*. New York, NY, USA: ACM, 1994.

[6] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-host congestion control for tcp," *IEEE Communications Surveys Tutorials*, vol. 12, no. 3, pp. 304–342, Third 2010.

[7] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-friendly High-speed TCP Variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008.

[8] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control," *ACM Queue*, vol. 14, no. 5, pp. 50:20–50:53, Oct. 2016.

[9] N. Dukkipati, T. Refice, Y. Cheng, J. Chu, T. Herbert, A. Agarwal, A. Jain, and N. Sutin, "An Argument for Increasing TCP's Initial Congestion Window," *ACM SIGCOMM Computer Communications Review*, vol. 40, pp. 27–33, 2010.

[10] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proc. SIGCOMM'17*, 2017.

[11] B. Briscoe, "Flow Rate Fairness: Dismantling a Religion," *ACM SIGCOMM CCR*, vol. 37, no. 2, 2007.

[12] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, Mar 1998.

[13] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses," RFC 6824, Jan. 2013.

[14] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving Datacenter Performance and Robustness with Multipath TCP," in *Proc. SIGCOMM'11*, 2011.

[15] I. Psaras, L. Saino, and G. Pavlou, "Revisiting Resource Pooling: The Case for In-Network Resource Sharing," in *Proc. ACM HotNets-XIII*, 2014.

[16] S. V. J. Rani, M. Suryakanth, and S. Kaushik, "Cross layer based schemes for improving the performance of TCP in wireless networks," in *2017 International Conference on Computational Intelligence in Data Science(ICCIDS)*, June 2017, pp. 1–6.

[17] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end Arguments in System Design," *ACM Trans. Comput. Syst.*, vol. 2, no. 4, pp. 277–288, Nov. 1984.

[18] H. Balakrishnan, H. Rahul, and S. Seshan, "An Integrated Congestion Manager Architecture for Internet Hosts," in *Proc. ACM SIGCOMM*, 1999.

[19] S. Islam, M. Welzl, S. Gjessing, and N. Khademi, "Coupled congestion control for RTP media," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, Aug. 2014.

[20] S. Islam, M. Welzl, D. Hayes, and S. Gjessing, "Managing real-time media flows through a flow state exchange," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, April 2016, pp. 112–120.

[21] S. Islam, M. Welzl, K. A. Hiorth, D. Hayes, G. Armitage, and S. Gjessing, "ctrlTCP: Reducing Latency through Coupled, Heterogeneous Multi-Flow TCP Congestion Control," in *IEEE Infocom'18 GI Workshop*, Honolulu, USA, Apr. 2018.

[22] S. Islam and M. Welzl, "Start me up: Determining and sharing TCP's initial congestion window," in *Proceedings of the 2016 Applied Networking Research Workshop*, ser. ANRW '16. New York, NY, USA: ACM, 2016, pp. 52–54. [Online]. Available: http://doi.acm.org/10.1145/2959424.2959440

[23] P. Teymoori, M. Welzl, S. Gjessing, E. Grasa, R. Riggio, K. Rausch, and D. Siracusa, "Congestion control in the Recursive InterNetworking Architecture (RINA)," in *Proc. IEEE ICC 2016*, May 2016.

[24] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations," RFC 3135, Jun. 2001.

[25] D. A. Hayes, P. Teymoori, and M. Welzl, "Feedback in Recursive Congestion Control," in *Computer Performance Engineering: 13th European Workshop, EPEW 2016*. Springer, 2016.

[26] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proc. ACM SIGCOMM*. New York, NY, USA: ACM, 2002, pp. 89–102.

[27] N. Dukkipati, "Rate Control Protocol (RCP): Congestion Control to Make Flows Complete Quickly," Ph.D. dissertation, Stanford, CA, USA, 2008.

[28] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)," in *Proc. ACM SIGCOMM*, 2010, pp. 63–74.

[29] S. H. Low, "A duality model of TCP and queue management algorithms," *IEEE/ACM Transactions on Networking (ToN)*, vol. 11, no. 4, pp. 525–536, 2003.

[30] P. Teymoori, D. A. Hayes, M. Welzl, and S. Gjessing, "Estimating an additive path cost with explicit congestion notification," University of Oslo, Tech. Rep. 487, 2019. [Online]. Available: http://tiny.cc/gvki8y

[31] P. Teymoori, D. Hayes, M. Welzl, and S. Gjessing, "Even Lower Latency, Even Better Fairness: Logistic Growth Congestion Control in Datacenters," in *Proc. LCN'16*, Nov 2016.

[32] K. A. Hiorth and M. Welzl, "Design considerations for RINA congestion control over WiFi links," in *ICIN 2019 Workshop (RINA 2019)*, Paris, France, Feb. 2019.

[33] G. White and R. Pan, "Active Queue Management (AQM) Based on Proportional Integral Controller Enhanced PIE) for Data-Over-Cable Service Interface Specifications (DOCSIS) Cable Modems," RFC 8034, pp. 1–17, Feb. 2017.

[34] K. Ciko and M. Welzl, "First Contact: Can Switching to RINA save the Internet?" in *ICIN 2019 RINA Workshop*, Feb. 2019.