LGCC: A Novel High-Throughput and Low Delay Paradigm Shift in Multi-Hop Congestion Control

Peyman Teymoori, Michael Welzl, David Hayes

Abstract—Technological advancements have provided wireless links with very high data rate capacity for 5G/6G mobile networks and WiFi 6, which will be widely deployed by 2025. However, the capacity can have substantial fluctuations, violating the assumption at the transport layer that the capacity is (almost) steady. In this paper, we present a general and efficient, yet deployable solution to this problem through a novel design empowered with a rich theory, allowing a significantly improved experience in using new technologies, especially mobile cellular services. We employ the well-known theory of food-chain models in biology, where a bottleneck link can be modeled as prey, while flows are predators. We extend this model to a chain of predators and preys to form a multi-hop congestion controller, called LGCC. Through simulation evaluation with real-life 5G traces we show the effectiveness of LGCC, compared with the state-of-the-art ABC (Accel-Brake Control). Our results show an order of magnitude bottleneck queuing delay decrease, with only a small decrease in throughput because LGCC tries to never exceed link capacities. LGCC's design can additionally open a new paradigm in stable multi-hop congestion control and flow aggregation.

Index Terms—Multi-hop Congestion Control, Logistic Growth, Food Chain, 5G/6G.

1 Introduction

In the Internet design, congestion control has been deployed at the transport layer or above, and it is supposed to operate in an "end-to-end" manner, where "end" represents the application host [1]. The commonly used transport protocol, TCP, reacts to packet drops as the main congestion signal from the network. This design greatly impedes the consideration of link-specific properties. Today, wireless links are prevalent, and the capacities that they expose to TCP are often time-varying, especially with new link layer technologies.

Notable technological advancements in communication have made it possible to exploit millimeter waves (mmWaves) [2] to benefit from the very high data rates it can provide. The mmWave spectrum has also been considered for next generation wireless networking such as cellular networks (e.g., 5G/6G). mmWave communication is at odds with TCP's assumption of a relatively stable end-to-end path with a static capacity at the bottleneck. It promises extremely low latency, yet it will often expose a drastically fluctuating network capacity. TCP should be able to react to these fluctuations—but the reaction speed of the end-to-end control loop is inevitably in the order of end-to-end round-trip times (RTTs). Such a control loop cannot ensure low latency and good utilization in the face of capacity changes that occur at sub-RTT timescales [3].

Even with today's 4G networks, it is already common to shorten the control loop by installing Performance Enhancing Proxies (PEPs)—devices which interfere with TCP connections, e.g., by dividing them in two (acting as a receiver towards a sender and acting as a sender towards a receiver) [4]. Splitting an end-to-end connection like this (or with an application-layer proxy, which is currently under discussion in the IETF for the QUIC protocol [5]) has shown benefits, and it may even be the only feasible way to attain efficient communication over mmWave links [6], [7]. Most importantly perhaps, connection splitting eliminates the need to operate "normal" TCP congestion control throughout the path, allowing the deployment of a more suitable congestion control mechanism on the wireless segment only.

What, then, should a mechanism for the wireless segment look like? Rather than suggesting a quick fix for one specific link layer technology, we argue that it is necessary to design a *general* congestion control mechanism which:

- can swiftly adapt to changing physical capacity.
- will stably operate when multiple instances of the mechanism are used in sequence (e.g., when transferring data from one wireless segment to another), as a chain of control loops.
- has a low deployment barrier, e.g., by not requiring to update every router along the path.

To the best of our knowledge, this paper is the first attempt at devising a congestion control mechanism that satisfies all of these requirements.

Chains of congestion control loops have been shown as a key feature in new network architectural patters such as the Recursive InterNetwork Architecture (RINA) [1]. In our previous work [8], we investigated various forms of feedback for a recursive congestion control as in RINA; the main finding was that faster more direct feedback is better, suggesting that chained congestion control should relay congestion signals back to the source without delaying them; this constitutes our main design principle in this paper. Similar to earlier explicit feedback methods such as XCP or RCP [9], [10], we opt for an approach that does

- *P. Teymoori and M. Welzl are with the Department of Informatics, University of Oslo, Norway. E-mail: {peymant|michawe}@ifi.uio.no*
- *D. Hayes did this work while at Simula Metropolitan Center for Digital Engineering, Oslo, Norway. E-mail: david.hayes@ieee.org*

Manuscript received 2022.

not only rely on physical queue growth for feedback. These mechanisms were found not to be deployable in practice as they require to change all routers on an end-to-end path. Our situation is more favorable, as we aim to apply our mechanism on path segments only. By basing the feedback method on the Datacenter TCP (DCTCP) [11] style of ECN usage (interpreting ECN marks from multiple packets as a multi-bit congestion signal), we made it even easier to deploy.

Specifically, the method that we propose, called "Logistic Growth Chain Control" (LGCC):

- can be deployed by only updating end systems, and introducing some devices that support it at loop intersection points (a key difference from explicit rate feedback schemes like XCP or RCP); within each control loop, commodity hardware can be used, provided that this hardware supports Random Early Detection (RED) AQM marking with Explicit Congestion Notification (ECN).
- is general enough to be adopted in different use cases, and can be easily enhanced by feeding it more information from the network. For example, it is possible to further accelerate convergence if there are more bits available in the packet header than ECN. Because we use an *additive* signal akin to *cost* as defined in the Network Utility Maximisation (NUM) framework [12], it would also be possible for routers to mark packets before a physical queue even grows (as proposed in [13], [14], for instance).
- is globally asymptotically stable.
- is fair, and can attain different types of fairness.
- is fast in terms of convergence due to the use of the LG function. Since we shorten control loops, flows can converge faster in case of capacity decrease/increase.
- responds quickly to capacity changes since the capacity can be measured more directly than by inferring it based on queue-related metrics such as ECN and delay. LGCC uses the concept of *carrying capacity*, the maximum available resource, denoted by K. In a 5G case, K is time varying, so we add a time parameter, i.e. $K(t)$. There are already useful signals at lower layers that can feed into $K(t)$—e.g., various MAC solutions calculate a physical rate using rate adaptation techniques, making the current physical rate a promising candidate as input to $K(t)$. $K(t)$ is then promptly and explicitly sent back to the sender inside the next ACK packet[1]. This enables the sender to know the current bottleneck capacity in a duration much shorter than the RTT. Consequently, sources do not need to rely on large, delay increasing, buffers to avoid fluctuations. By not relying solely on buffers and contrasting with methods like ABC [15], which necessitates a minimal persistent queue for its stable operation, an insignificant reduction in throughput can occur.

LGCC is based upon an early version of our mechanism, the biology-inspired Logistic Growth Control (LGC) [16], combined with our *additive* ECN feedback theory [17] that enables us to take advantage of higher marking probabilities. In contrast to prior work, we now assume multiple consecutive control loops between each source and destination, one per administrative domain. This allows us to react much faster to congestion than with an end-to-end loop, but it is a more complex design approach which inevitably raises stability concerns. A stability analysis is, thus, a key contribution of this paper.

In Section 2, we survey related work. Section 3 discusses our motivations behind this paper, and Section 4 introduces the theory behind LGCC. We establish the theory of LGCC in Section 6, and in Section 8, we analyze stability and fairness properties of LGCC. Section 9 presents how LGCC is implemented, and its simulation results are presented in Section 10. Finally, Section 11 concludes the paper.

2 Related Work

The advent of high capacity wireless links (g.e. mmWave) has raised serious challenges for traditional congestion control [18]. In a series of work [3], [7], [19], [20], [21], [22], it was shown that the full potential of mmWave cannot be achieved at higher layers, especially at the transport layer.

One of the main solutions of the above problem has been to introduce some intermediate boxes helping the end-to-end congestion controller, e.g. by breaking it into several shorter loops. For example, in [6], a TCP proxy architecture was presented without any modification at the sender side. The proxy is installed in the Radio Access Network (RAN), and by collecting statistics from the 5G base station (gNB), it estimates the downlink Bandwidth-Delay Product (BDP) and performs flow window management using the advertised window field in ACKs. Another method, called X-TCP [23], was proposed to adjust the congestion window of TCP at the mobile equipment side. It estimates the current physical layer rate using the Signal to Interference plus Noise Ratio (SINR), thus keeping the buffer small.

Explicit feedback control protocols (e.g. XCP [9] and RCP [10]) can overcome some of the above challenges. However, they have deployment limitations in the Internet because they need drastic changes to all packet headers, routers, and endpoints. The ABC mechanism [15], [24] has targeted the above problems by providing a deployable method, compared with prior explicit schemes, since it utilizes the existing Explicit Congestion Notification (ECN) [25] infrastructure. This makes the mechanism the closest competitor of the one that we present in this paper. ABC uses a Multiplicative-and-Additive-Increase/Multiplicative-Decrease (MAIMD) scheme to ensure fairness among flows. ABC adjusts the sender's window upon receiving an ACK from the receiver: it is increased by one if the ACK is marked as *accelerate*[2], and in case of a *brake* mark, it is decreased by one; this allows the sender to have a larger window size dynamic within one RTT to follow the link capacity dynamics faster. In the network, ABC routers mark packets with accelerate or brake proportional to the queue length. Additionally, ABC has a fall-back mechanism to coexist with non-ABC traffic.

1. Our implementation uses the receiver window size field, and its value is interpreted as bytes per second at the sender of the previous control loop.

2. To solve the problem that "AQM schemes do not signal increases" [15].

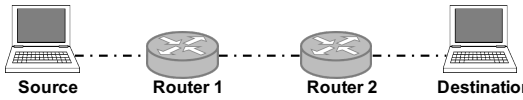


Fig. 1. A simple topology used for a preliminary comparison of DCTCP, LGC, ABC, and LGCC. The links have a capacity of 500 Mbps, and Source's end-to-end RTT is 6 ms. The link capacity between Router 2 and Destination is time-varying; it drops to one third in $[2, 3]$s of the simulation. Other parameters are set such that 100% utilization with minimal delay is achieved for each method.

New network architectures such as the Recursive InterNetwork Architecture (RINA) [26] represent a different approach to many aspects of networking, including congestion control: here, network mechanisms are designed from scratch, without compromise.[3] In RINA, it is natural to break an end-to-end path into several shorter control loops, each with its own customized congestion controller. In addition, a larger control loop can operate over several shorter ones; this is the recursive property of RINA, which can provide many benefits [28]. What we propose in this paper can also be adopted by RINA.

Since the advent of hop-by-hop congestion controllers, as the extreme case of multi-hop controllers, their stability has been of paramount concern because some unstable behavior was observed in some situations [29], [30]. This had led to the "fear of instability". However, as demystified by [30], the cause of instability was the unsuccessful use of non-discriminatory on-off type controls. In addition, research work such as [29], [30], [31], [32], [33] has shown that per-hop controllers accelerate the response to load changes in the path, and that certain conditions can be obtained under which the chain of control loops is stable. For example, in case of Split-TCP (where a middlebox spoofs IP addresses to act like a receiver towards the sender and act like a sender towards the receiver), the system can be stable under certain conditions [31], [34].

The assumption of having links with time-varying capacity also complicates stability conditions of the congestion controller. Analyses such as [35] show system stability with feedback delay can be achieved in the joint congestion control and scheduling for multi-hop wireless networks. In [36], a family of optimization-based distributed traffic control laws was developed to enable load balancing and/or rate adaptation in response to capacity variations in the network. Instead of proving system stability around a single equilibrium point, the authors in [37] propose a primal-dual congestion control algorithm which is proven to be trajectory stable when link capacities are time-varying; this guarantees the system is stable around a time varying reference trajectory.

3 Motivation

In this section, we motivate the need for a deployable multi-hop congestion controller with preliminary simulation and evaluation; this shows why the controller needs to be fast, in both rate increase and decrease, and also in terms of convergence rate, with the direct knowledge of link capacity and its changes as well as employing ECN to increase deployability.

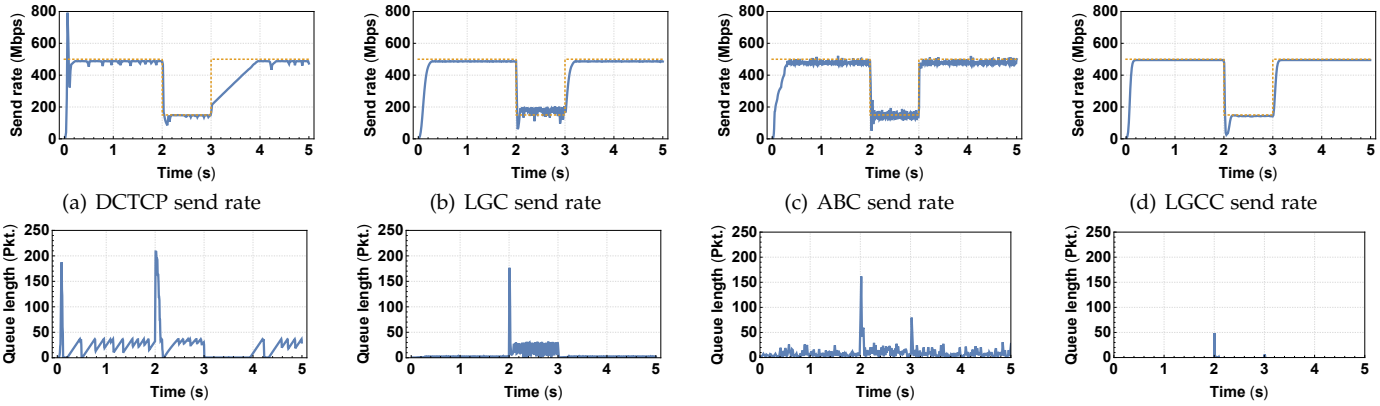
In previous sections, we discussed about limitations of end-to-end congestion control and why shortening control loops can respond faster to congestion and capacity change. ABC is a very recent state-of-the-art mechanism, which, according to its authors, operates at the Pareto frontier when compared with other congestion controllers such as TCP Cubic, Vegas, XCP, BBR, and PCC [15]. Although ABC is empowered by ABC routers on the path to signal a rate increase/decrease via ECN, it still operates in an end-to-end manner. This means that ECN marks first arrive at the destination, and then they are echoed back to the source. Considering the topology illustrated in Fig. 1, in Fig. 2 we show a preliminary simulation[4] comparison of LGCC with ECN-based mechanisms: LGC (our previous end-to-end approach [16]), DCTCP [11], and ABC. We observe that DCTCP produces the largest queue because it cannot incorporate link capacity fluctuations, and because it is additive-increase, it is very slow in filling the newly increased capacity. LGC can incorporate link capacity in its rate dynamics, but it cannot track capacity changes since it is end-to-end, and starts fluctuating when the capacity drops, leading to a larger queue. ABC only incorporates link capacity at ABC routers through queue drain rate, but still the ABC sender can track changes faster because the ABC router motivates the source in both increase and decrease through ECN marks. LGCC can, however, outperform all of the others in terms of rate fluctuation, convergence speed, and keeping the queue small because of a multi-hop design where Router 1 and Router 2 act as connection splitters, shortening the control loop, and incorporating the link capacity and its changes directly in their rate dynamic.

In terms of convergence rate, we need a fast method for flows to reach their target rate. Although ABC flows can follow the underlying link capacity quickly with the help of ABC routers, they do not get a fair share of the capacity promptly as they fill the capacity. The main problem is that when a new flow joins, to attain fairness among flows, ABC's MAIMD adds/subtracts less than 1 packet per RTT to/from the congestion window (*cwnd*) of the flows, and the increase/decrease becomes smaller as flows converge.

We evaluate ABC's convergence rate through a simple analysis: assume n flows with equal RTTs and congestion window w_i, competing for a bottleneck capacity. ABC's congestion window of flow i increases by $1+1/w_i$ per accelerate and decreases $-1+1/w_i$ per brake signal. Therefore, during an RTT where the proportion of accelerates is f, the window increases by $fw_i(1 + 1/w_i) = f(w_i + 1)$ and decreases by $(1 - f)w_i(-1 + 1/w_i) = (1 - f)(-w_i + 1)$. Adding these two changes to w_i to obtain the new window size, and considering the bottleneck capacity is completely filled, we

3. Compromises will usually be necessary when trying to interconnect the whole architecture with the Internet. This can, for example, be done by tunneling, using gateways or with direct switch-over as explained in [27] for relatively short communication paths.

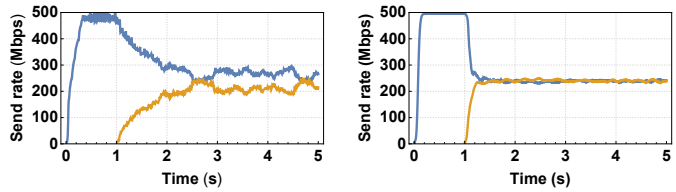
4. It is performed in OMNeT++; we used the code we had implemented in [16] for DCTCP and LGC. We re-implemented ABC, and validated it vs. the results in [15].



(a) DCTCP send rate (b) LGC send rate (c) ABC send rate (d) LGCC send rate

(e) DCTCP bottleneck link (f) LGC bottleneck link (g) ABC bottleneck link (h) LGCC bottleneck link

Fig. 2. Performance comparison of 1 flow of DCTCP, LGC, ABC, and LGCC. The dotted line in the top row illustrates the link capacity, and the blue lines show the senders' rate using each of the aforementioned methods.



(a) ABC (b) LGCC

Fig. 3. Convergence comparison of two flows

have

$$\begin{aligned} w_i^{(\text{new})} &= w_i + f(w_i+1) + (1-f)(-w_i+1) \\ &= 2fw_i + 1 \end{aligned}$$

$$\sum_{i=1}^{n} w_i^{(\text{new})} = BDP = 2f\left(\sum_{i=1}^{n} w_i\right) + n$$

Let f^* denote the equilibrium of f. Solving for f^* yields

$$f^* = \frac{\text{BDP} - n}{2\sum_{i=1}^{n} w_i} = \frac{1}{2}\frac{\text{BDP} - n}{\text{BDP}} < \frac{1}{2} \ .$$

As an example, if $n = 2$ and BDP $= 200$ packets, then $f^* = 0.495$. If $w_1 = 198$ and $w_2 = 2$, in the next RTT, $w_1 = 2fw_1 + 1 = 197.02$, and $w_2 = 2fw_2 + 1 = 2.98$. Since $(2f-1) < 0$, the only term that increases w_2 is 1. This means that in each RTT, the w_2 increase is less than 1, and it becomes smaller as w_1 and w_2 get closer. For example, after 50 RTTs, $w_2 = 40$ and $w_1 = 160$, and after 400 RTTs, they are approximately equal, which can be a very long time. Simulation results also confirm this. In case of LGCC, after 25 RTTs we have $w_2 = 40$ and $w_1 = 160$, and after 60 RTTs, $w_1 \approx w_2$. It should be noted that RTTs are shorter in LGCC because of shorter control loops. Fig. 3 shows a simulation comparison on the convergence of ABC and LGCC in OMNeT++. The first flow fills the capacity, and then the second flow joins. We can observe that it takes a long time for the ABC flows to converge. However, this is shorter than half a second for LGCC. In the following sections, we elaborate on the design of LGCC and show how it incorporates link capacity, capacity changes, and ECN in its rate dynamic.

4 Logistic Growth (LG)

The base of LGCC is the Logistic Growth (LG) function. LG is usually used to model the growth of a species population over time. This function is defined by the following differential equation

$$\dot{N} = \frac{\gamma N(t)(K - N(t))}{K} \tag{1}$$

where N is the population size, K the "carrying capacity" determining the maximum value of N, and γ is growth rate. By the dot notation, we mean time differentiation, i.e. $\dot{N} = \frac{dN}{dt}$. We summarize the notation in Table 1. The idea of growth constrained by a capacity limit in LG has inspired the use of this function in network congestion control [16], [38], [39].

The LG function (and its dynamics (1)) is usually normalized by defining $x(t) = N(t)/K$, which yields

$$\dot{x} = \gamma x(t)\big(1 - x(t)\big) \tag{2}$$

with the solution

$$x(t) = \frac{1}{1 + \big(\frac{1}{x(0)} - 1\big)e^{-\gamma t}} \tag{3}$$

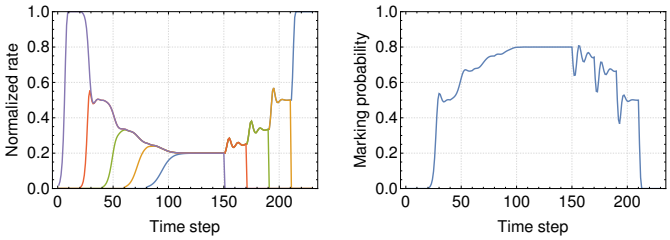
where $x(0)$ denotes the initial population. Clearly, $\lim_{t\to\infty} x(t) = 1$.

4.1 Lotka-Volterra Model

The LG function specifies how one species population grows over time. The Lotka-Volterra model consists of a set of equations examining the effect of inter-species competition where two or more species compete for some limited resource. The growth rate of a species is then not only bounded by the carrying capacity, but also by the growth rate of a competing species. If a generalized Lotka-Volterra model for n interacting species has a non-trivial equilibrium, $x_i^* > 0$, then there can be found a Lyapunov function that is zero at the equilibrium and negative at other points [40]. This implies that the model is globally stable. In Section 6, we will use this relationship to model the interconnected loops of a path like individual species, using this relationship to interconnect them.

TABLE 1
Notations

Symbol	Description
K	Carrying capacity
R	The set of sources
L	The set of links
Q	The set of queues
A	The set of aggregating points/nodes
x_r	Send rate of source r
γ	Growth rate
λ_l	Lagrange multiplier of link l
p_l	Marking probability of link l
$U_r(.)$	Utility function of source r
R_l	The set of the sources crossing link l
L_r	The set of the links source r crosses
O_r	The number of loops of source r to reach its destination
$L_{r,i}$	The set of the links source r crosses in loop i
q_r	The total cost source r is charged by links L_r
q_r	The total marking probability of source r traversing links L_r
$p_{r,i}$	The marking probability received by flow r in loop i
c_l	Capacity of link l
b_l	Buffer size of link l
y_l	The sum of rates of sources crossing link l
$.^*$	Optimum value of . that optimizes some function
$\varrho_r(x_r)$	Step size function of source r
$\sigma_l(p_l)$	Step size function of link l
β	Step size parameter
γ	Step size parameter
ϕ	Logarithm base parameter



(a) Rate (b) Marking probability

Fig. 4. 5 LGC competing flows with $\gamma = 1$

5 Logistic Growth Control (LGC)

LGC [16] was presented as a congestion controller using the Lotka-Volterra competition model [41]. In this controller, the rate of source r is updated by

$$\dot{x}_r = x_r\gamma_r\big(1 - x_r - \hat{p}_r\big) \tag{4}$$

or in the discrete form,

$$x_r[n+1] = x_r[n]\gamma_r\big(1 - x_r[n] - \hat{p}_r[n]\big) + x_r[n] \tag{5}$$

where γ_r is a constant value and $\hat{p}_r$ is the (end-to-end) measured marking probability at source r. This yields a normalized rate which sources can multiply by their link capacity to obtain a send rate in bits per second. Since LGC needs the link capacity, it was proposed for data centers in which link capacities are known/fixed. If there is only one bottleneck on the path, it is proven that $\hat{p}_r$ converges to $\frac{S-1}{S}$ where S is the number of competing flows, and hence, $x_r^* = \frac{1}{S}$.

To illustrate how LGC works, we numerically evaluated (5): we assume that there were 5 flows competing for a bottleneck capacity, and excess traffic was marked proportional to the capacity. Formally speaking,

$$p_r = \Big[p_r + \big(\sum_{i=1}^{5} x_i - 1\big)\Big]_0^1 \tag{6}$$

where $[.]_0^1 = \max(\min(.,1),0)$. Fig. 4 shows how the 5 flows compete when they join one by one. The first flow starts at time step 0, and then after every 20 time steps, one new flow joins. We see that the normalized rate of flows converges to 0.2, and their observed marking probability converges to 0.8. At step 150, flows leave one by one every 20 steps until there is only one flow left. We can also see that the convergence speed of pure logistic growth decays as more flows enter the system—flow 5 is quite slow at obtaining its bandwidth share. We have fixed this by making γ adaptive in [16], and we apply the same method in this paper, in Section 9.1.

5.1 Deflated LGC

In [17], we obtained the utility function of LGC, and showed how LGC can be adopted by the NUM theory. This also helps change the equilibrium marking probability of LGC. In other words, we can deflate its marking probability, which can become very high when the number of competing flows is large. The send rate dynamic is

$$\dot{x}_r = x_r\gamma_r\Big(-\log_{\phi_1}(x_r) + \log_{\phi_2}(1-\hat{p}_r)\Big). \tag{7}$$

where ϕ_1 and ϕ_2 are the base parameters that can be configured to control the marking probability. If $\phi_1 = \phi_2$, then (7) reduces to (4). In case there is one bottleneck, the equilibrium marking probability can be obtained, which is given by

$$p_r^* = 1 - e^{\frac{\log(\phi_2)\log(\frac{1}{S})}{\log(\phi_1)}} \tag{8}$$

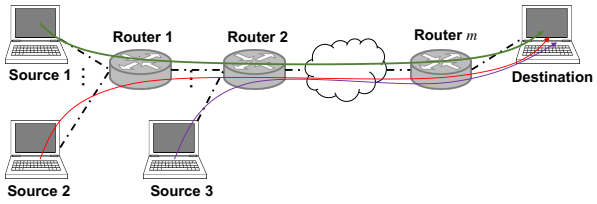
where S is the number of competing flows. In this case, if $\phi_1 > \phi_2$, the equilibrium marking probability becomes smaller. For example, if we set $\phi_1 = 10$ and $\phi_2 = 2$, then $p_r^* = 0.1897$ (this was 0.8 in Fig. 4(b)). However, the congestion controller behavior does not change—it exhibits the exact behavior shown in Fig. 4(a).

6 LG Chain Control (LGCC) Model

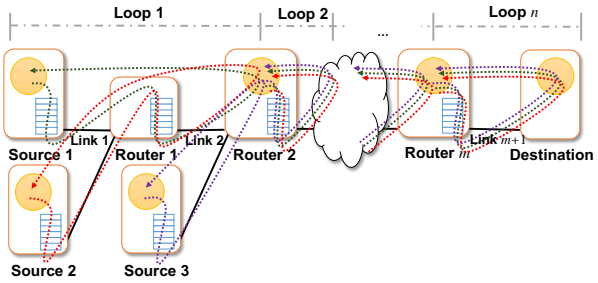
6.1 Network Model

In this section, we model a chain of LGCs and discuss its deployment considerations. Our main aim is to have a deployable solution with minimal required changes to current commodity hardware. ECN is a widely-deployed mechanism in commodity hardware with broad utility. We utilize this feature and assume that routers in the network support packet marking via RED. However, some of the nodes on the path towards the destination need to support LGCC, which we call *LGCC router*; these nodes could be domain border routers. In other words, LGCC does not need to be implemented as a hop-by-hop solution although that is also possible.

Fig. 5(a) illustrates an example network topology of a source and a destination with m routers on the path. This can be mapped into the diagram in Fig. 5(b) in which only



(a) A sample network topology



(b) Corresponding control loops

Fig. 5. An example of how loops are formed.

some of the routers deploy LGCC, meaning that an end-to-end congestion controller between the source and the destination is broken in several smaller congestion control loops. The other routers are assumed to have RED deployed, which ECN-marks packets randomly if the output queue is non-empty. Therefore, there is one LGC loop from the source to Router 2, another one from Router 2 to some other router, and finally, there is one from Router m to the destination. There is one LGC loop per source from sources 1 and 2 to Router 2, one loop from source 3 to Router 2, another one per source from Router 2 to some other router (continued in that way until Router m), and finally, there is one per source from Router m to the destination. This example already highlights the possibility of aggregating traffic to reduce unnecessary competition between flows, but we deem a full flow aggregation mechanism for heterogeneous traffic out of scope for the present paper. However, here we discuss the *possibility* of flow aggregation and its benefits with regards to LGCC, and provide a fairness analysis.

The number of LGCC routers and where they operate depends on the network design. For example, each loop can be formed over one domain, or if the medium is wireless, the loop can operate over only that medium specifically. We assume that it is a design issue and up to the network manager where to deploy LGCC routers; however, we will provide performance evaluations about different deployments in Section 10.

Fig. 6 shows a high level internal structure of an LGCC router. It has several "Receivers" to control the congestion control loops ending at it, and several "Senders" to start new ones towards their destination/next LGCC router. Senders have their own send queues with the aim of holding excess incoming packets if their current send rate is lower than the incoming rate. The role of "Schedulers" is to prioritize taking packets from receivers and pass them to the correct next sender. If packets are accumulated at receivers waiting for schedulers to pass them, a virtual queue management scheme can be deployed to ECN-mark packets. These marks are sent back towards the previous senders via ACKs from

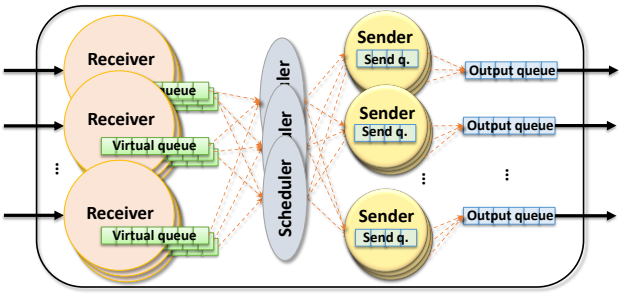


Fig. 6. Loop management architecture in LGCC routers

the receivers. In addition to the ECN echo each ACK carries back to the sender of the loop, the current send rate of the next loop is also copied to the receiver window size field of the ACK. This ensures that previous senders as well as the original source are updated very quickly, i.e., in at most half an end-to-end RTT depending on the bottleneck location, with the latest rate changes of the next loops to reduce the risk of instability. The value of this field can be interpreted as bit (or Byte) per second at the previous sender side; this determines the carrying capacity of the previous loop ending at this LGCC router. The sender of the last loop receives the buffer size from the destination as normal since the last hop sender needs to ensure that it does not overflow the destination. However, it feeds back its current rate inside the receiver window size[5] field of ACKs originating from it. In the next section, we will show how carrying capacity is used in rate calculations.

LGCC routers could be implemented as PEPs or a part of border routers. Here we assume that they explicitly break a connection, and hence, packets of one LGCC router are destined explicitly to another one, and so on. As such, it is not possible for one LGCC router to be somehow bypassed by changes in the traffic route. However, it is possible to have route changes between two LGCC routers. The goal of ECN is to capture traffic load changes on the path segment, even due to routing changes.

To elaborate the LGCC router design in Fig. 6, assume there are two loops arriving at an LGCC router from different links, and departing it from the same link. In case of per-flow chaining, which is our main focus in the paper, each incoming flow has its own dedicated receiver and sender in the router. Packets from the receivers are passed to their corresponding senders, which now compete for the same output queue; their packets might also be ECN-marked in the next loops as well. The senders accordingly adjust their send rate to the fair share of the capacity of the next link/loop, and their current send rate is echoed back to the senders of the previous loops using every ACK generated by the receivers. The previous senders will accordingly adjust their send rate in the same manner. In case of aggregated chaining, there is only one sender at the output link, and the scheduler passes packets from the receivers to the sender; with the assumption that there are

5. In practice this may have side effects, such as causing the source operating system (or middle boxes) to become suspicious of the ever-changing field. One possible alternative is to use TCP options. Use with IPSec is another challenge, that may require trusted routers. However, implementation specifics are beyond the scope of this paper.

only homogeneous greedy flows, sharing for aggregations is simply weighted with respect to the number of flows in each aggregated flow. If there are no other senders at this output link, the sender can send at the link capacity without queuing. However, the sender of the previous loops need to send at a lower rate, which can be done through ECN-marking packets at the receivers' virtual queue. We will demonstrate a simple policy to perform this kind of marking in Section 10.5.

In the above architecture, each LGCC loop operates within an autonomous administrative domain, e.g., from an ingress router to an egress router, where these two can also act as LGCC routers. Incorrect ECN configuration might lead to either longer delays or competing flow instability within that domain. However, since LGCC uses the same ECN as DCTCP, it should be able to coexist harmoniously with traditional TCP in the L4S (Low Latency, Low Loss, and Scalable Throughput) architecture [42]. A detailed evaluation is out of the scope of this paper.

6.2 Congestion Controller Model

Here, we introduce LGCC. First, we form a chain of (4) where the carrying capacity is not normalized to 1. Instead, it is determined by the send rate of next loop. In addition, we use notation $\hat{p}_{r,i}$ to reflect the marking probability of only loop i that is measured by its sender. Assume a flow broken into n loops from a source to a destination. The rate dynamics at the sender of each loop are given by

$$\dot{x}_{r,1} = x_{r,1}(t)\gamma_{r,1}\left(1 - \frac{x_{r,1}(t)}{k_{r,1}(t)} - \hat{p}_{r,1}(t)\right) \ , \tag{9a}$$

$$\dot{x}_{r,2} = x_{r,2}(t)\gamma_{r,2}\left(1 - \frac{x_{r,2}(t)}{k_{r,2}(t)} - \hat{p}_{r,2}(t)\right) \ , \tag{9b}$$

$$\vdots$$

$$\dot{x}_{r,n} = x_{r,n}(t)\gamma_{r,n}\left(1 - \frac{x_{r,n}(t)}{k_{r,n}(t)} - \hat{p}_{r,n}(t)\right) \ , \tag{9c}$$

where

$$k_{r,1}(t) = \min\Big(x_{r,2}(t), c_l(t)\Big), \quad \forall l \in L_{r,1} \ , \tag{10a}$$

$$k_{r,2}(t) = \min\Big(x_{r,3}(t), c_l(t)\Big), \quad \forall l \in L_{r,2} \ , \tag{10b}$$

$$\vdots$$

$$k_{r,n}(t) = \min\Big(x_{r,n+1}(t), c_l(t)\Big), \quad \forall l \in L_{r,n} \ , \tag{10c}$$

and $x_{r,n+1}(t)$ and c_l denote the destination's processing rate and link capacity, respectively. LGC uses the notion of carrying capacity, and so does LGCC. Therefore, the carrying capacity K should be determined in each loop. This can be the minimum of the send rate of the next loop and the link capacity in the current loop, which is obtained by (10). $\hat{p}$ includes the price of the receiver's virtual queue to the next sender as well in case of aggregated chaining. Since we assume that the formation of each loop is under one authority/domain, link capacities may be known in that domain. In case a link capacity is changing over time, e.g. in a wireless medium, information about the current capacity and modulation is usually available at the output link of the LGCC router. This could be used as the *current* link/carrying capacity. A similar technique has been done by ABC [15].

In case of aggregated chaining and if packets of two or more receivers are forwarded to the same sender in the next loop, i.e., the loops are aggregated into one flow in the next loop, we can enable schedulers to prioritize packets of certain receivers while passing to their next sender. Here, we assign weights to receivers and employ a weighted fair queuing discipline in the scheduler. This is under the assumption that flows are homogeneous and greedy. Other cases will be left as future work. In this case, the rate dynamic of loop i is given by

$$\dot{x}_{r,i} = x_{r,i}(t)\gamma_{r,i}\left(1 - \frac{x_{r,i}(t)}{k_{r,i}(t)} - \frac{\hat{p}_{r,i}(t)}{\omega_{r,i}}\right) \tag{11}$$

where $\omega_{r,i}$ denotes the normalized weight assigned to the send rate of flow r in loop i.

6.3 Benefits of the LGCC Design

The notion of carrying capacity is an important feature of the LGCC design. It helps the controller to have an accurate estimate of the upper capacity limit, which is a direct signal sent back to previous controllers, relieving them from having to wait for indirect congestion/cost signals such as queue marks to adjust their rates. In other words, each loop uses the send rate of the next loop as its own K, which leads to faster rate adjustment to available capacity at next loops.

LGC's increase/decrease behavior is non-linear. As opposed to controllers with, e.g. additive increase, how fast to increase/decrease depends on the difference between $x_{r,i}$ and $(1 - \hat{p}_{r,i})k_{r,i}$. This means that in case of a sudden increase in the medium capacity (assume a wireless link that is recovered from a rate drop due to an intermittent noise), capturing the available capacity can be fast.

6.4 Utility Function

For LGCC, we apply the same method that we used to obtain the LGC utility function in [17]. In equilibrium, we know that $q^*_{r,i} = U'(x_{r,i})$ where $q^*_{r,i} = -\log_{\phi_2}(1 - p^*_{r,i})$. Solving for p yields $p^*_{r,i} = 1 - \phi_2^{q^*_{r,i}}$. We also have $1 - \frac{x^*_{r,2}(t)}{k_{r,2}(t)} - p^*_{r,2}(t) = 0$. These give

$$U(x_{r,i}) = x_{r,i}\left(\frac{1}{\log \phi_1} - \log_{\phi_1}\left(\frac{x_{r,i}}{k_{r,i}}\right)\right) \ , \tag{12}$$

which is a strictly concave, twice differentiable function in the range $(0, k_{r,i}]$. These properties facilitate its stability and convergence.

7 Congestion Control Algorithms

The LGCC dynamics (9) can be converted into a Network Utility Maximization (NUM) problem. This enables us to analyze LGCC better and find deployable solutions, especially on commodity routers. In [17], we obtained the utility function of LGC, and showed how LGC can be adopted by the NUM theory. Here, we use the utility function behind

LGCC dynamics, i.e., (12), and then, we divide the optimization logic between sources and routers. Since sources and LGCC routers use the same module to send packets, by *sender dynamics* we mean the controller logic in both of them, and by *router dynamics* we mean commodity routers that do not break connections. Our assumption about these routers is that they can either implement our specific packet marking scheme or at least, they support RED, which is a widely-deployed mechanism.

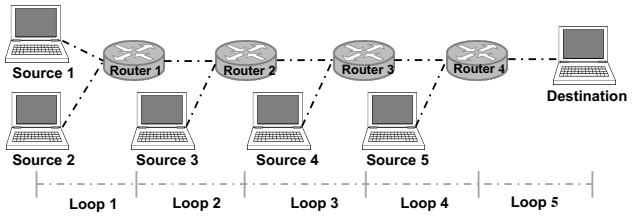


Fig. 7. 5 loops chained towards the destination

7.1 Primal Algorithm

In this algorithm, the optimization logic is deployed at the sender side. The routers only return a network cost to senders, which is a direct mapping of the current total traffic of a link to a non-negative real value. It is given by

$$\dot{x}_{r,i} = x_{r,i}\gamma_{r,i}\left(U'(x_{r,i}) + \log_{\phi_2}(1-\hat{p}_{r,i})\right) \tag{13}$$

where $U(.)$ is the utility function of senders [17]. Substituting the derivative of (12) in (13) yields

$$\dot{x}_{r,i} = x_{r,i}\gamma_{r,i}\left(-\log_{\phi_1}\left(\frac{x_{r,i}}{k_{r,i}}\right) + \log_{\phi_2}\left(1-\hat{p}_{r,i}\right)\right) . \tag{14}$$

Since our assumption is that routers support RED, the network cost can be sent back to senders via ECN marks. This is denoted in (13) by $\hat{p}_{r,i}$. In the next subsection, we introduce an ECN marking algorithm for this purpose.

7.2 Dual Algorithm

In the dual algorithm, the optimization logic is deployed in routers, and senders directly map the cost they get from the network into a send rate. First, we discuss a general router which could run this algorithm, and then discuss how routers with RED AQM can employ it.

We use the fact that at equilibrium, $q^*_{r,i} = U'(x^*_{r,i})$ where $q^*_{r,i}$ is the cost sender i gets. As we already showed in [17], in case of employing RED, $q_{r,i} = -\log_{\phi_2}(1-\hat{p}_{r,i})$. Combined, these two yield

$$x_{r,i} = k_{r,i}\; e^{\left(\log(1-\hat{p}_{r,i})\log(\phi_1)/\log(\phi_2)\right)} \tag{15}$$

meaning that the sender can directly calculate its optimal send rate using $\hat{p}_{r,i}$. However, routers need to iterate on their cost until they converge to the optimal value. The first dual algorithm, called **Dual1** is given by

$$\dot{p}_l = \beta_l\Big[y_l - C\Big]^+_{p_l} \tag{16}$$

where p_l and y_l denote the marking probability of link l (at the router's output queue to the link) and the total traffic sent over link l in the last interval. β_l denotes the step size. At output queues, $C = c_l$ where c_l is the link capacity, and at receivers' virtual input queues, $C = x_{r,i+1}$, i.e. the send rate of the next loop, and y_l is equal to the rate of the traffic sent to this queue from the sender of loop i in the last interval.

As we showed in our previous work, we can use RED with a special configuration (see Appendix E of [17]) as the dual algorithm. This enables us to have a more deployable solution. The cost/marking probability dynamics at routers' output queues (links) in a discrete form is given by

$$p_l[n] = \left[\frac{b_l[n]}{\max_{\text{th}}}\right]^1_0 \tag{17}$$

where b_l, $\max_{\text{th}}$, and n denote the queue length, the maximum threshold parameter of RED over which all the packets are marked, and the interval number. Other RED parameters are $\min_{\text{th}} = 1$, $w_q = 1$, and $\max_{\text{p}} = 1$. This algorithm implies that the current backlogged size can be divided by the maximum threshold to have a marking probability for the current interval.

One of the benefits of the dual algorithms of LGCC is that, due to the notion of carrying capacity, senders can sometimes *jump* to a send rate if, for example, the carrying capacity suddenly changes. In case the sender's loop is not shared with other loops, jumping to higher rates does not affect the network. Otherwise, if some links of a loop is shared with other loops, the primal algorithm can be used instead of (15); the use of a primal-dual algorithm can also handle this case, which will be explained in the next subsection.

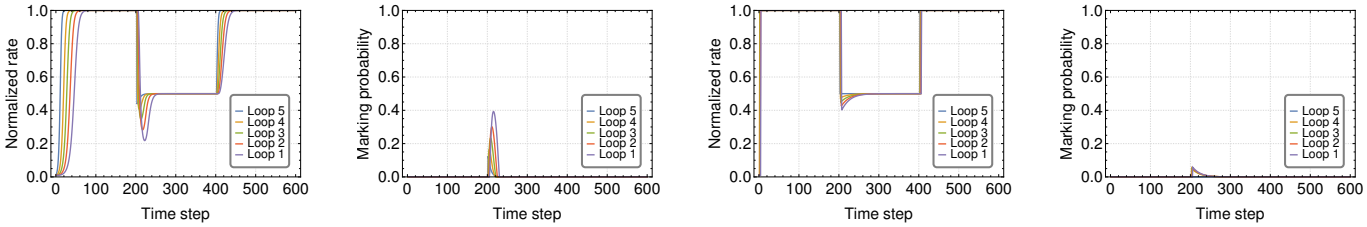
7.3 Primal-Dual Algorithm

This type of algorithm divides the optimization logic between senders and routers. The combinations (14)–(16) and (14)–(17) form two sets of primal-dual algorithms.

7.4 Numerical Evaluation

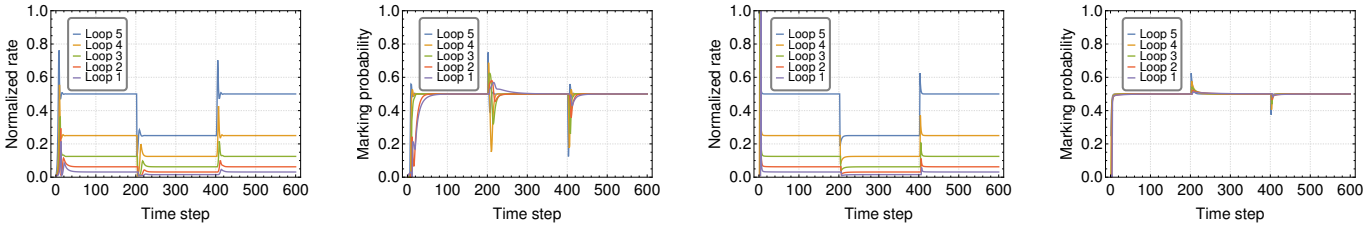
In order to have a notion on how much (9) is effective and can accelerate convergence of a chain of loops due to the usage of the carrying capacity, we perform a numerical evaluation here on the topology shown in Fig. 7. We consider two scenarios: 1) a chain of 5 loops where inside each loop, there is no other competing loop (i.e. Source 2 to Source 5 do not send any packets), and 2) a chain of 5 loops where inside each loop, there is another competing loop, which will be aggregated into one flow in the next loop (i.e. all sources send packets, which will be aggregated at the next router). Therefore, in the last loop, only 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 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. Apparently, in both scenarios, the rate of loop i, $x_{r,i}$, is the carrying



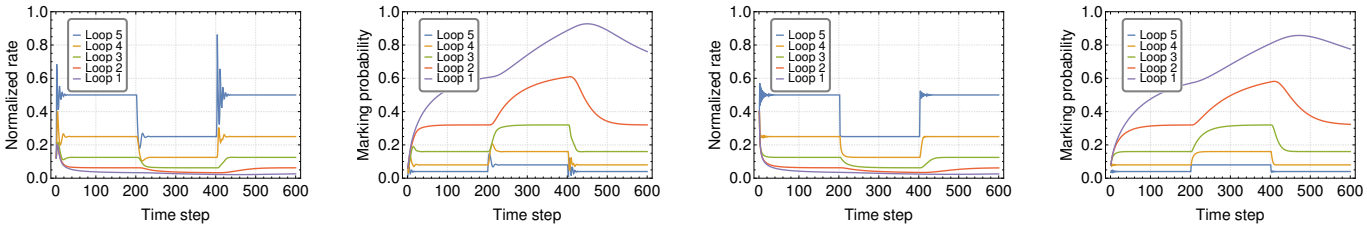
(a) Normalized rate of flows (b) Marking probability of flows (c) Normalized rate of flows (d) Marking probability of flows

Fig. 8. Chaining 5 loops of LGCC. (a) and (b): Primal-Dual, and (c) and (d): Dual.



(a) Normalized rate of flows (b) Marking probability of flows (c) Normalized rate of flows (d) Marking probability of flows

Fig. 9. Chaining 5 loops of LGCC where there are two competing flows in each loop, aggregated into one the next loop. (a) and (b): Primal-Dual, and (c) and (d): Dual.



(a) Normalized rate of flows (b) Marking probability of flows (c) Normalized rate of flows (d) Marking probability of flows

Fig. 10. Chaining 5 loops of the logarithmic controller where there are two competing flows in each loop, aggregated into one the next loop. (a) and (b): Primal-Dual, and (c) and (d): Dual.

capacity of loop $i-1$, but $p^*_{r,i}$ is equal to 0 and 0.5 in scenarios 1 and 2, respectively. 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.

We evaluated the Primal-Dual and Dual algorithms under the above scenarios. In addition, to see how a chain of controllers that do not employ the notion of carrying capacity behaves, we used a controller with a logarithmic utility function, i.e. $U(x) = \log(x)$. Hence, the rate dynamics in loop i are given by

$$\dot{x}_{r,i} = \gamma_{r,i}\left(\frac{1}{\hat{p}_{r,i}}\right) . \tag{18}$$

The Dual algorithm (16) was used for the algorithms, and the controllers were also tuned for fastest convergence. Fig. 8 and Fig. 9 illustrate the results of scenario 1 and scenario 2, respectively, and Fig. 10 shows the results of the controller with a logarithmic utility function. From the figures we observe that LGCC can converge faster when the capacity drops to 0.5 in loop 5 at step 200, and also at step 400, when it changes back to 1. It might seem that a Dual method is superior to the other one. As mentioned before, if links in a loop is not shared by other flows, a Dual method can be used to *jump* to the rate routers imply; however, it is not true in a general topology.

8 Analytical Evaluation

In this section, we analytically evaluate properties of LGCC such as equilibrium, stability, and fairness.

8.1 Equilibrium

Depending on where the bottleneck in the network is, LGCC can converge to different values. For this analysis, we assume that the bottleneck is always in some loop n, and then, the `min` operator in (10) always kicks in, and for the sake of simplicity, we assume that schedulers in Fig. 6 employs a fair queuing discipline. Hence, assuming $x_{r,i+1}(t) \leq c_l(t)$, $\forall l \in L_{r,i}, \forall t$:

$$x^*_{r,i}(t) = x^*_{r,i+1}(1 - p^*_{r,i}) \tag{19}$$

and when the marking probability is deflated

$$x^*_{r,i} = x^*_{r,i+1}\ e^{\left(\log(1-p^*_{r,i})\log(\phi_1)/\log(\phi_2)\right)} \tag{20}$$

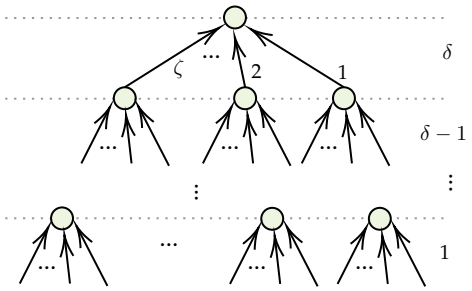


Fig. 11. Aggregation tree

Employing a fair queuing discipline in the scheduler implies that $x^*_{r,i} = x^*_{s,i}$ for competing flows r and s in loop i. Using the fact that $\sum_r x^* r, i = x^*_{r,i+1}$, we get

$$e^{\left(\log(1-p^*_{r,i})\log(\phi_1)/\log(\phi_2)\right)} = \frac{1}{S} \tag{21}$$

where S is the number of competing flows. Solving for $p^*_{r,i}$ yields

$$p^*_{r,i} = 1 - e^{\left(\log(1/S)\log(\phi_2)/\log(\phi_1)\right)} \ . \tag{22}$$

8.2 Stability Analysis

Here, we discuss the stability of LGCC. The following theorem will prove its global stability.

Theorem 8.1. *The primal-dual algorithm* (14)–(16) *is globally, asymptotically stable.*

Proof. See Appendix A.1. □

Although Theorem 8.1 proves that the system of equations (14)–(16) is globally stable, we also analyze its local stability; this provides us a relationship between the step size parameters γ and β, and we can use this as a guideline for setting those parameters.

Local stability refers to the analysis of the system behavior close to an equilibrium. We use the dual algorithm (16) as the cost generator at routers. This indeed includes a step size parameter, β, which can affect the system stability.

We use an aggregation tree topology that is illustrated in Fig. 11 with variable depth (the number of loops of the chains – denoted by δ) and breadth (the number of competing flows in each loop – denoted by ζ). The circles represent LGCC routers at which incoming flows compete. For the sake of simplicity, we assume that the flows in each loop only compete at the input queue of the next loop, and they are all aggregated into one flow in the next loop.

Theorem 8.2. *The system of equations* (14)–(16) *is locally stable if and only if*

$$\beta_d \leq \frac{1}{4K}\gamma_d\,\zeta^{(\delta-d-2)} \quad \textit{for} \quad 1 \leq d \leq \delta \ . \tag{23}$$

Proof. See Appendix A.2. □

The above theorem implies that the stability in each loop i only depends on the number of competing flows in that loop; this means that the value of γ_i and β_i determine the stability, and as we get closer to the bottleneck link, i.e. $i = 1$ in this case, γ should be smaller than β.

8.3 Fairness Analysis

Fairness properties of a congestion controller determine how fairly the capacity is shared among a number of competing flows and how much each flow can expect to receive. Since LGCC chains controllers, we focus on two types of fairness: intra-loop and inter-loop.

8.3.1 Intra-Loop Fairness

Here, we focus on only one loop. We consider a number of flows competing for links capacity in an arbitrary topology. First, we define a new type of fairness.

Definition 1 (Logarithmic Proportional Fairness)**.** *Every rate change from the equilibrium proportional to the logarithm of the equilibrium is non-positive. Formally speaking,*

$$\sum_r -\log(x^*_r)(x_r - x^*_r) \leq 0 \ . \tag{24}$$

This is obtained from the fact that if the feasible set is convex and $U(.)$ has a globally maximized at $\mathbf{x}^*$, then for any feasible allocation vector $\mathbf{x}$, we have $\nabla(U)|_{\mathbf{x}^*}(\mathbf{x}-\mathbf{x}^*) \leq 0$ meaning that the gradient of U, evaluated at $\mathbf{x}^*$, times the change is not positive. The above definition also implies that if there are n sequential links with the normalized capacity of 1, 1 n-hop flow with rate x_1 crossing all the links, and n 1-hop flows crossing each link with rate x_2, then x^*_1 is the multiplication of the equilibrium rate of other 1-hop flows in its path. In other words,

$$x^*_1 = \prod_{i=1}^{n} x^*_2 \ .$$

8.3.2 Inter-Loop Fairness

To focus on only inter-loop fairness, here, we assume that all routers support LGCC. This implies that loops are formed in a hop-by-hop fashion. In this case, schedulers determine the fairness type. We consider a number of flows in a loop arriving at an LGCC router and analyze two types of chaining: 1) the flows are aggregated into one flow in the next loop, and 2) a separate flow is created in the next loop for each flow. We employ a weighted fair queuing discipline in LGCC routers. LGCC is able to attain max-min fairness in type 1) if weights are set equal to the number of aggregated flows, and proportional fairness in type 2) if weights are set in a special manner. The following theorems prove these characteristics.

Theorem 8.3. *In case of aggregated chaining, if weights are set equal to the number of aggregated flows, LGCC attains max-min fairness.*

Proof. See Appendix B.1. □

Theorem 8.4. *Let $p^u_{r,i}$ denote the upstream marking probability of flow r, i.e. the marking probability at the output queues of LGCC routers 1 to i. In case of per-flow chaining, if weights are set inversely proportional to $-\log(1-p^u_{r,i})$ at the scheduler in LGCC router $i+1$ for all routers i on the path, then LGCC attains proportional fairness.*

Proof. See Appendix B.2. □

Since in LGCC we assume that each loop operates with its own marking probability, accessing $p^u_{r,i}$ at downstream

routers would need a special bit in the LGCC header to convey the marks received so far at the output queues of LGCC routers.

9 LGCC Algorithm and Implementation

Algorithm 1 shows how each sender adjusts its rate. It can be either a Primal-Dual or a Dual algorithm. At startup, send rate x is set to the initial value x_{init}; this can be interpreted as the initial window size of TCP. However, the unit of x is bits per second. We estimate the loop marking probability, $\hat{p}$, using an exponential smoothing average with parameter α. As a rate-based method, we define intervals to sample ECN-marks, and at the end of each interval, $\hat{p}$ is updated. The carrying capacity is also obtained: it can be the minimum link capacity in the loop, or the last send rate of the next loop, which is echoed back to the previous loop via the receiver window size field; the value of this field is interpreted as a rate in bits (or to save space, Bytes) per second. Then, depending on the algorithm, and in case of Primal-Dual, (14) is run. Before the rate calculation, from line 11 to 18, the adaptive step size algorithm is run. The goal is to be more aggressive if the calculated rate from a Dual algorithm is *far* from the current send rate; this will be discussed in the next subsection. Then, the send rate is set, ensuring it is not lower than the initial send rate. In case of Dual, the send rate is directly calculated from the measured marking probability, $\hat{p}$. Details of the parameter setting are presented in the next section.

In both of the algorithms, we assume that routers run (17), which can be realized through a special configuration of RED. At the end of the loop, the receiver module of the next LGCC router, echoes back ECN marks.

The following subsections elaborate on some aspects of this algorithm that go beyond the previously discussed basic LGCC dynamics: adapting γ, obtaining the right capacity value to use (in line 9, we assume that the minimum of all link capacities is known), and how often to trigger "OnIntervalEnds".

Algorithm 1 LGCC congestion control for flow r in loop i

1: **procedure** INIT
2: $x \leftarrow x_{\text{init}}$
3: $\hat{p} \leftarrow 0$
4: $\gamma \leftarrow \gamma_{\text{init}}$
5: **end procedure**
6: **procedure** ONINTERVALENDS
7: $m \leftarrow$ ECN-marked percentage of ACKs
8: $\hat{p} \leftarrow (1-\alpha)\,\hat{p} + \alpha\, m$
9: $k \leftarrow \min\left(x_{r,i+1}, c_l\right) \quad \forall l \in L_{r,i}$
10: **if** Primal-Dual **then**
11: $\hat{x} \leftarrow k\; e^{\left(\log(1-\hat{p})\,\log(\phi_1)/\log(\phi_2)\right)}$
12: **if** $|x-\hat{x}| \le g_1$ **then**
13: $\gamma \leftarrow \gamma_{\text{conv}}$
14: **else if** $|x-\hat{x}| \ge g_2$ **then**
15: $\gamma \leftarrow \gamma_{\text{init}}$
16: **else**
17: $\gamma \leftarrow |x-\hat{x}|\frac{\gamma_{\text{init}}-\gamma_{\text{conv}}}{g_2-g_1} + \gamma_{\text{conv}}$
18: **end if**
19: $x \leftarrow x\gamma\left(\log_{\phi_2}\left(1-\hat{p}\right) - \log_{\phi_1}\left(\frac{x}{k}\right)\right) + x$
20: $x \leftarrow \max(x, x_{\text{init}})$
21: **else if** Dual **then**
22: $x \leftarrow k\; e^{\left(\log(1-\hat{p})\,\log(\phi_1)/\log(\phi_2)\right)}$
23: **end if**
24: **end procedure**

9.1 Adaptive Step Size

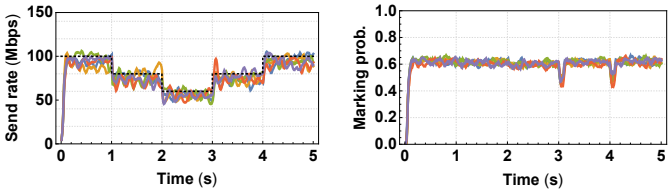
To improve the performance of LGCC via faster convergence, we use the same technique that we used in LGC [16] to adjust the step size of the Primal-Dual algorithm, i.e. γ. In LGCC, $\hat{p}$ and accordingly, (15) provides an indication of what the target rate can be, and how *far* the current send rate is from that target. In the Primal-Dual algorithm, when the current rate is far from the target, we use a larger value of γ. When the rates are close, then we use a small value for γ to have a smoother behavior. The algorithm (line 11 to 18 in Algorithm 1) determine if the difference of x and $\hat{x}$ are larger than the threshold g_2. If so, γ is set to a large value, i.e. γ_{init}; if the difference is smaller than the threshold g_1, then γ_{conv} is used; otherwise, it is set linearly based on the proportion of the difference. The range $[\gamma_{\text{conv}}, \gamma_{\text{init}}]$ is chosen such that it does not affect stability.

9.2 Carrying Capacity

As shown in (10), LGCC senders need to know the carrying capacity of their loop to obtain their rate; this could be either the minimum link capacity on their path in the loop, or the current send rate of the next sender in the next loop of the chain. In [16], we showed that if an LGC sender cannot get the carrying capacity, it still can operate (as we see in Fig. 2(b)). However, the queue length fluctuates more. When each loop is formed under one domain or over a specific medium, e.g. a wireless link, obtaining the carrying capacity is sometimes straightforward. In wireless links with time-varying capacity, the current send rate (depending on the noise level and modulation) is the carrying capacity. This is how the current transmission rate is obtained in some work such as [23]. ABC also uses a method based on the output queue drain rate to estimate the current link capacity. Hence, we assume that if there is a link with time-carrying capacity, an LGCC router is run on that node to start a new loop over that link; in case of wired links, LGCC routers can be run on interior/exterior routers of each authority domain, e.g. ISPs.

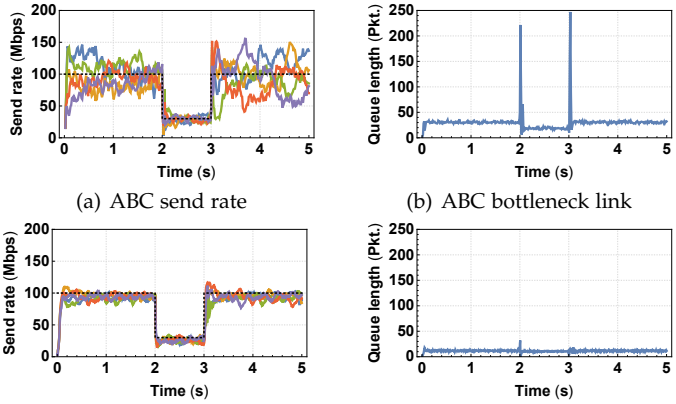
9.3 Rate Update Time Intervals

The rate is regularly updated by each sender per interval. The LGCC dynamics, (9), imply that the fair rate only depends on $\hat{p}$ in each loop, and as long as all sources *see* the same $\hat{p}$, they converge to the same rate. This means that the steady state behavior of the controller is not sensitive to the update interval: updating it more or less often will naturally make the control converge faster or slower, but the point of convergence is not affected (but updating it extremely fast could lead to oscillations). There are several options for the update frequency such as using the smallest RTT value seen over a certain time interval (an estimate of the RTT without queuing delay). If a flow's smallest RTT is still a long time interval, especially for a large-RTT flow, we can use shorter intervals. Based on our observation in



(a) Send rate at senders (b) Marking prob. at bottleneck

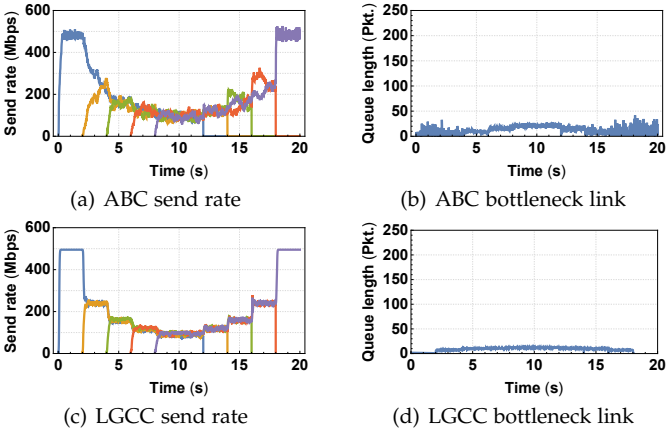
Fig. 12. Send rates and marking probability of 5 flows when the carrying capacity changes.



(a) ABC send rate (b) ABC bottleneck link

(c) LGCC send rate (d) LGCC bottleneck link

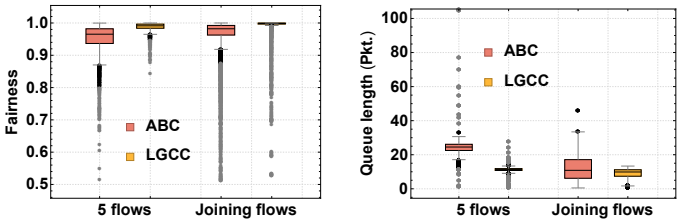
Fig. 13. Performance comparison of ABC and LGCC in a dumbbell topology: 5 flows start at the same time, and the capacity of the bottleneck link drops to 150 Mbps in $[2, 3]$s.



(a) ABC send rate (b) ABC bottleneck link

(c) LGCC send rate (d) LGCC bottleneck link

Fig. 14. Performance comparison of ABC and LGCC in a dumbbell topology: 5 flows join one-by-one at every two seconds, and leave after 12 seconds of transmission.



(a) Fairness (b) Queue length

Fig. 15. Box-Whisker plot of short-term Jain's fairness index and queue length of ABC and LGCC in the two different scenarios. Black and gray dots are near and far outliers, respectively.

simulations (performed in [16], the performance is robust if a flow can receive at least 4 ACKs per interval. Hence, another option is to use small, equal-sized intervals that can fit receiving a few ACKs.

10 Simulation and Experiment Results

10.1 Simulation Setup

In this section, we present simulation results of LGCC over various topologies and scenarios, compared with ABC as the state-of-the-art. Simulations were performed in the INET framework[6] of OMNeT++. We set the general parameters as: $\gamma_{\text{init}} = 0.3$, $\gamma_{\text{conv}} = 0.05$, $\phi_1 = 10$, $\phi_2 = 5$, $\alpha = 0.05$, $g_1 = 0.05\,\hat{x}$, $g_2 = 0.015\,\hat{x}$.

10.2 Flow Competition and Fairness Evaluation of LGCC

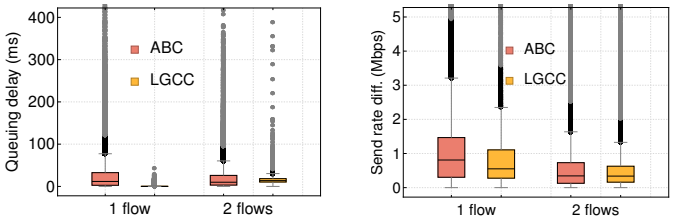
Here, we evaluate LGCC using when 5 sources (flows) competing for the bottleneck in a dumbbell topology, sending packets to 5 different destinations. Links have a capacity of 500 Mbps with 1 ms propagation delay. Router 1 and Router 2 that connect sources to destinations are LGCC routers, making 3 loops, and the capacity of the link between them changes every 1 second: it is 500 Mbps in the first simulation second, and drops to 400 Mbps at second 1 and to 300 Mbps at second 2. Then, it changes back to 400 Mbps and 500 Mbps. Fig. 12(a) shows how the rate of the 5 flows in the middle loop changes over time, and the dashed, black line is the fair share of the capacity. We see that sources closely track the capacity change while competing with each other. Fig. 12(b) plots the marking probability of the flows in the second loop, i.e. flows starting from Router 1 to Router 2. These flows compete at the output queue of Router 1. We used RED as the Dual algorithm with $\max_{\text{th}} = 75$ KB, and $\phi_1 = 10$ and $\phi_2 = 3$. The Equilibrium Marking Probability (EMP) is 0.6. Referring to the figure, we observe that EMP only depends on the number of competing flows, not the current link capacity. There are two drops for a short period in MP because the carrying capacity suddenly increases, and it takes a short period for the flows to converge by adjusting their rate.

10.3 Comparison with ABC

We compared LGCC with ABC. As the first topology, we used a dumbbell topology with 5 sources, sending packets to 5 different destinations. Fig. 13 shows the result of the first scenario, in which the flows start together, and in $[2, 3]$s, the capacity drops to 150 Mbps. The links have a capacity of 500 Mbps, and the link propagation delay is 1 ms. The queue length of the bottleneck router is also shown. Both ABC and LGCC have a special router at the bottleneck link. From the figures we see that LGCC operates faster and more smoothly in terms of convergence, which also results in a smaller queue at the bottleneck.

In the next scenario over the previous topology, flows join one-by-one every two seconds, and they last for 12 seconds. Fig. 14 illustrates how the two protocols operate; again, LGCC converges faster and behaves more smoothly than ABC with a smaller queue length.

6. The code is available at https://cutt.ly/2yV7eQK



(a) Queing delay (b) Rate difference

Fig. 16. Box-Whisker plot of rate difference and queuing delay of ABC and LGCC.

We ran the above two scenarios 10 times, each with a different random seed and a random start of flows. Fig. 15 shows how LGCC and ABC operate. Short-term fairness was calculated over the average values of time windows of 24 ms. This also confirms that LGCC operates more smoothly, and flows converge fast. The queue length is accordingly smaller because of the faster reactions to changes in the number of flows.

In the next scenario, we ran ABC and LGCC over three real LTE link capacity traces of the AT&T, TMobile, and Verizon operators. These are the traces used by [15] to evaluate ABC. The topology includes a destination, connected through an LTE link whose capacity changes over time (the capacity follows the pre-measured trace) and a source 3 hops away from the destination. Fig. 16 illustrates the difference between the send rate of Source and the rate in the trace file at any time; the smaller the difference, the faster the controller at convergence. In case of 2 flows, each flow should get half of the current capacity, and the difference is calculated with this. Referring to the figure, we observe that LGCC flows can track capacity changes faster, leading to a smaller queue at the LTE link. As the number of flows increases, the difference between ABC and LGCC becomes smaller because each flow should receive a smaller share of the capacity. The link utilization of the protocols are 93% for both ABC and LGCC.

10.4 5G Use Case

In the next scenario, we used a 5G trace collected from a major Irish operator in [43], which is the first publicly available dataset containing throughput information for 5G networks. The dataset contains traces of two different mobility patterns: *static* and *car*. We used the traces of the static pattern because results of [43] have shown that 5G coverage in streets was not complete, so the car traces mostly included LTE/4G. The trace files had some dramatic capacity changes dropping to values of around 0.3% of the maximum achieved throughput. We did not consider cases with connection timeouts or with dis-connectivity since our focus was on devising a congestion control method.

We applied the traces in OMNeT++ to dictate link speeds over time, where the maximum capacity was 333 Mbps. The topology was the one in Fig. 5(a) with $m = 5$ except there was only one source, i.e. Source 1, and there were 5 routers, making 6 hops to Destination. Link capacities were 500 Mbps with a propagation delay of 5 ms, yielding an end-to-end RTT 60 ms. There was one flow from Source 1 to Destination transferring a large file. We used this application as an example of various high-bandwidth applications 5G is envisioning such as holographic video call[7]. We set ABC's δ parameter to 300 ms; ABC recommends setting this to at least two third of the base RTT to guarantee stability. The ABC's minimum queuing delay threshold parameter were set to 1 ms. In general, we tried to set the ABC's parameters such that it is more responsive to capacity changes, has a shorter queue, while stability is not affected.

We ran ABC and LGCC over this network with various link configurations. First, we assumed that only the link between Router 5 and Destination is 5G with capacities collected by [43]; the results are shown by "ABC-R5" and "LGCC-R5" in Fig. 17, and there is only one ABC/LGCC router deployed, which was on Router 5. Second, we assumed that the link between Source 1 and Router 1 is 5G, with results shown by "ABC-S" and "LGCC-S"; in this case, the ABC queue is installed on Source 1 while the LGCC router logic should be installed on Router 1. "LGCC6-R5" shows the results of 6 LGCC loops, operating hop-by-hop where the last hop was 5G. In the third scenario both of the above links were 5G, shown by "ABC-2B" and "LGCC-2B". We used different combinations of traces as capacities, combined all collected data, and measured queuing delay illustrated as box-whisker plot, excluding outliers, in Fig. 17 with a logarithmic y-axis.

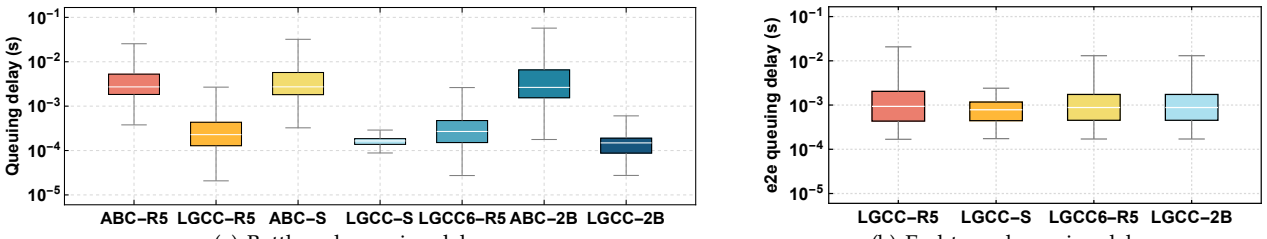
The queuing delay measurements of LGCC were performed in two categories: 1) the bottleneck link queuing delay, measuring only the delay at the bottleneck, and 2) the end-to-end queuing delay, which measures the delay of every queue (including all the senders' queues at LGCC routers) along the path to destination. The aim of the latter measurement is to include effects of all congestion control loops since they might momentarily delay forwarding packets if their current send rate is smaller than the send rate of their previous loop. In case of ABC, the end-to-end queuing delay is equal to the bottleneck queuing delay.

From the figure, we observe that LGCC can reduce queuing delay significantly. The achievable utilization[8] values of ABC and LGCC for the scenarios shown in Fig. 17 from left to right were 93.9%, 85.8%, 93.9%, 90.2%, 87.8%, 94%, and 90.3%, respectively[9]. In total, LGCC's utilization was a bit smaller than that of ABC because LGCC tries not to exceed the carrying capacity, which is the current link bandwidth; however, ABC needs a small standing queue for the sake of stability, which helps the link in sending more packets when its capacity increases, but at the cost of an order of magnitude increased delay. LGCC-R5 has the lowest utilization and larger delay values than the other LGCC configurations because it comprises a long loop from the sender to the last router, where the bottleneck is; this also implies the benefits of shortening control loops. The end-to-end queuing delay of LGCC-R5, in Fig. 17(b), shows larger variations compared with the delay of ABC-R5 because in LGCC-R5, the sender at Router 5 might delay forwarding

7. This has been demonstrated by Ericsson and Vodafone Germany https://www.ericsson.com/en/news/2018/11/3d-holographic-calls-with-5g

8. Defined as application layer throughput divided by the average link physical rate in percent.

9. We did not plot the channel utilization since the utilization values of the two methods were close with little variation.



(a) Bottleneck queuing delay (b) End-to-end queuing delay

Fig. 17. Queuing delay represented by a Box-Whisker plot over a 5G use case. Figure 5(a) topology with: i) last hop is 5G—ABC-R5 and LGCC-R5; ii) first hop is 5G—ABC-S and LGCC-S (LGCC R1); iii) last hop is 5G—LGCC6-R5 (LGCC all routers); iv) both first and last hops are 5G: ABC-2B and LGCC-2B (LGCC R1 and R5). Boxes span from the 0.25 quantile to the 0.75 quantile, where outliers are not shown for the sake of visibility.

packets due to a lower send rate than the incoming rate, which does not happen in ABC. However, the median value is smaller in LGCC.

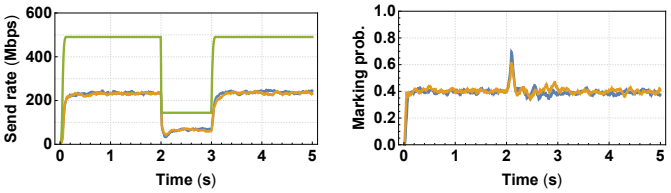
Any capacity change takes time to affect the source. We see that if the first link is 5G, shown by LGCC-S, any change is reflected very fast at the source, resulting in the highest utilization (90.2%). LGCC6-R5 which has 6 loops, can compensate for this without increasing queuing delay. ABC's utilization is approximately equal in all the cases because it is end-to-end. If both source's and destination's links were 5G, we see that LGCC and ABC achieve approximately equal utilization even though LGCC has only 3 loops; this is because the first loop, similar to LGCC-S, is very fast, compensating for the second loop that is longer, and the average utilization becomes comparable to that of ABC.

In order to evaluate the effect of increasing the number of loops, we also measured the utilization of two other cases where the last link is 5G: LGCC with 3 loops, where Router 1 and Router 5 are LGCC routers, and LGCC with 4 loops where Router 3 acts as an LGCC router as well. The utilization values were 85.8%, 86.4%, 86.6%, and 87.8% respectively, for the 2 loop (LGCC-R5), 3 loop, 4 loop and 6 loop (LGCC6-R5) scenarios. All the queuing delays are very close. This shows that adding more loops can increase the utilization, without increasing queueing delay, due to shortening control loops and faster convergence.

As discussed previously, the trace files included some dramatic capacity changes. So, if future 6G (and beyond) links have similar characteristics, as expected with sub-THz and Visual Light Communication (VLC) technology, LGCC is expected to operate similarly well. However, it should be noted that LGCC is only a congestion control method, expecting other underlying network mechanisms, such as multipath forwarding and load balancing, to manage cases where a link is down for a long time. LGCC could also be extended to work as a multipath congestion control scheme operating similarly to MPTCP [44], but with the additional advantages that LGCC routers provide.

10.5 Flow Aggregation

One of the potentials of LGCC is flow aggregation. Although a detailed evaluation of this needs a complete aggregation protocol (including scheduling algorithms and some form of signaling for fair sharing), which is out of the scope of this paper, however, using a simple topology we demonstrate how it looks in practice. Fig. 18 shows the results of two flows aggregated at an LGCC router into one flow on a



(a) Send rate at the first and the second loop (b) Marking prob. at first LGCC router

Fig. 18. Send rate and marking probability of 2 aggregated flows when the carrying capacity changes in a Dumbbell topology.

dumbbell topology. The link capacity is 500 Mbps, but the capacity at the second link drops to 150 Mbps in $[2, 3]$s. Since there is only one flow operating on the second link, there is no competition there. The LGCC router only applies a fair queuing discipline on the flows to be aggregated. The general LGCC router design shown in Fig. 6 and the two types of LGCC algorithms (Primal in (14) and Dual in (15)) allows different ways of implementing this scenario.

We adopted the Primal algorithm along with a marking policy at virtual queues of the LGCC router at the bottleneck. The marking policy operates as follows: it randomly marks every incoming packet with a fixed probability of 0.4 that is calculated by the scheduler using (22) to fairly divide the next sender's rate among these incoming loops. So, the flows receive half of the capacity, and in case of a drop, they follow the capacity change fast. The scheduling does not change the equilibrium marking probability *seen* by sources; the spikes in Fig. 18(b) only happen when on-the-fly packets arrive at the bottleneck link whose capacity is just dropped. This scenario, however, represents a simple case showing the possibility of aggregation, which can be of a huge potential in LGCC. For example, the use of a Dual algorithm at sources eliminates the waiting time for them to reach the maximum/equilibrium capacity, and newly-joined flows can instantly *hop on* the already established aggregated flow in the bottleneck. However, a more complicated scenario needs further design considerations, which are considered as future work.

11 Conclusion

In this paper, we presented a novel multi-hop congestion controller using the well-known Logistic Growth (LG) function in a food chain style. The idea of carrying capacity in LG parallels with the bottleneck link capacity, which in case of link layer capacity measurements, can provide an excellent

feedback to sources to adapt their send rate, especially for links with time-varying capacity. This combined with employing ECN enables us to deploy LGCC over path segments, and react faster to available capacity changes.

We analytically evaluated LGCC for stability and fairness; we proved that the system is globally stable, and provided conditions for the step size parameter setting. Through simulation study, we evaluated the performance of LGCC in terms of latency (queuing), utilization, and short-term fairness. We also used real-life traces of some LTE/4G/5G link capacities, and compared LGCC with ABC. Results confirm that LGCC can improve the delay and fairness performance significantly. LGCC greatly reduces queuing delay on the bottleneck link, minimizing the delay impact on other flows. It does this by moving queuing from the router's shared output queue, to a separated LGCC sender queue. LGCC also keeps a low e2e queuing delay, while maintaining a high utilization. In this paper we focused on delay. The balance between delay and utilization can be adjusted to meet QoS requirements, and does not impact the QoS of other flows sharing the bottleneck. In addition, LGCC can provide flow aggregation, which is of great benefits for congestion control.

As future work we will look at the effect of delay on control loop stability. Preliminary analysis indicates a generally stable system but suggests the corner case of connecting loops with slightly different equilibrium points needs further investigation. In addition, we will perform a deeper analysis of flow aggregation, devise a flow aggregation protocol, and provide a scalable solution considering heterogeneous flows, which might not be greedy; this will include the case where a flow has a bottleneck after it is split from an aggregation. Performing experiments on real use cases is a key interest of ours.

ACKNOWLEDGMENTS

P. Teymoori and M.Welzl 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] P. Teymoori, M. Welzl, S. Gjessing, E. Grasa, R. Riggio, K. Rausch, and D. Siracusa, "Congestion control in the Recursive Internetwork Architecture (RINA)," in *Proc. ICC*. IEEE, 2016, pp. 1–7.

[2] S. Rangan, T. S. Rappaport, and E. Erkip, "Millimeter-wave cellular wireless networks: Potentials and challenges," *Proc. IEEE*, vol. 102, no. 3, pp. 366–385, 2014.

[3] M. Zhang, M. Mezzavilla, R. Ford, S. Rangan, S. Panwar, E. Mellios, D. Kong, A. Nix, and M. Zorzi, "Transport layer performance in 5g mmwave cellular," in *Proc. INFOCOM workshops*. IEEE, 2016, pp. 730–735.

[4] D. A. Hayes, D. Ros, and Ö. Alay, "On the importance of TCP splitting proxies for future 5G mmWave communications," in *Proc. Local Comp. Netw. symposium*. IEEE, 2019, pp. 108–116.

[5] M. Kühlewind, Z. Sarker, T. Fossati, and L. Pardue, "Use Cases and Requirements for QUIC as a Substrate," Internet Engineering Task Force, Internet-Draft draft-kuehlewind-masque-quic-substrate-00, Mar 2020, work in Progress.

[6] M. Polese, M. Mezzavilla, M. Zhang, J. Zhu, S. Rangan, S. Panwar, and M. Zorzi, "milliproxy: A TCP proxy architecture for 5G mmWave cellular systems," in *Proc. Asilomar Signals, Systems, and Computers*. IEEE, 2017, pp. 951–957.

[7] R. Ford, M. Zhang, M. Mezzavilla, S. Dutta, S. Rangan, and M. Zorzi, "Achieving ultra-low latency in 5G millimeter wave cellular networks," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 196–203, 2017.

[8] D. A. Hayes, P. Teymoori, and M. Welzl, "Feedback in recursive congestion control," in *Proc. EPEW*. Springer, 2016, pp. 109–125.

[9] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 89–102, 2002.

[10] N. Dukkipati, "Rate Control Protocol (RCP): Congestion control to make flows complete quickly," Ph.D. dissertation, Stanford University, Stanford, CA, USA, 2008.

[11] 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*, New Delhi, India, 2010, pp. 63–74.

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

[13] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is more: Trading a little bandwidth for ultra-low latency in the data center," in *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. San Jose, CA: USENIX, 2012, pp. 253–266.

[14] K. Ciko, P. Teymoori, and M. Welzl, "Lgc-shq: Datacenter congestion control with queueless load-based ecn marking," *SIGCOMM Comput. Commun. Rev.*, vol. 52, no. 4, p. 2–11, dec 2022.

[15] P. Goyal, A. Agarwal, R. Netravali, M. Alizadeh, and H. Balakrishnan, "ABC: A simple explicit congestion controller for wireless networks," in *Proc. NSDI*. Santa Clara, CA: USENIX Association, Feb 2020, pp. 353–372.

[16] P. Teymoori, D. Hayes, M. Welzl, and S. Gjessing, "Even lower latency, even better fairness: Logistic growth congestion control in datacenters," in *Proc. Local Comp. Netw.* IEEE, Nov 2016, pp. 10–18.

[17] P. Teymoori, D. A. Hayes, M. Welzl, and S. Gjessing, "Estimating an additive path cost with explicit congestion notification," *IEEE Trans. Control Netw. Syst.*, vol. 8, no. 2, pp. 859–871, 2021.

[18] X. Wang, L. Kong, F. Kong, F. Qiu, M. Xia, S. Arnon, and G. Chen, "Millimeter wave communication: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1616–1653, 2018.

[19] P. Jimenez Mateo, "Analysis of TCP performance in 5G mm-wave mobile networks," Ph.D. dissertation, Universidad Carlos III de Madrid, Spain, 2017.

[20] M. Zhang, M. Mezzavilla, J. Zhu, S. Rangan, and S. Panwar, "TCP dynamics over mmwave links," in *Proc. SPAWC*. IEEE, 2017, pp. 1–6.

[21] M. Pieska and A. Kassler, "TCP performance over 5G mmWave links—tradeoff between capacity and latency," in *Proc. WiMob*. IEEE, 2017, pp. 385–394.

[22] M. Polese, R. Jana, and M. Zorzi, "TCP and MP-TCP in 5G mmWave networks," *IEEE Internet Comput.*, vol. 21, no. 5, pp. 12–19, 2017.

[23] T. Azzino, M. Drago, M. Polese, A. Zanella, and M. Zorzi, "X-TCP: a cross layer approach for TCP uplink flows in mmwave networks," in *Ad Hoc Networking Workshop (Med-Hoc-Net), 2017 16th Annual Mediterranean*. IEEE, 2017, pp. 1–6.

[24] P. Goyal, M. Alizadeh, and H. Balakrishnan, "Rethinking congestion control for cellular networks," in *Proc. HotNets*. ACM, 2017, pp. 29–35.

[25] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168 (Proposed Standard), Internet Engineering Task Force, Sep 2001, updated by RFCs 4301, 6040.

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

[27] K. Ciko and M. Welzl, "First contact:can switching to RINA save the Internet?" in *Proc. ICIN*, Feb 2019, pp. 37–42.

[28] M. Welzl, P. Teymoori, S. Gjessing, and S. Islam, "Follow the model: How recursive networking can solve the internet's congestion control problems," in *IEEE ICNC 2020, Big Island*, Feb 2020.

[29] V. Kulkarni, S. Bohacek, and M. Safonov, "Stability issues in hop-by-hop rate based congestion control," in *Proc. Annual Allerton Conference on Communication Control and Computing*, vol. 36. University of Illinois, 1998, pp. 79–88.

[30] P. P. Mishra and H. Kanakia, "A hop by hop rate-based congestion control scheme," in *ACM SIGCOMM CCR 22(4)*, 1992.
[31] S. Bohacek, "Stability of hop-by-hop congestion control," in *Proc. IEEE Decision and Control*, vol. 1, 2000, pp. 67–72.
[32] P. P. Mishra, H. Kanakia, and S. K. Tripathi, "On hop-by-hop rate-based congestion control," *IEEE/ACM Trans. Netw.*, vol. 4, no. 2, pp. 224–239, 1996.
[33] Y. Yi and S. Shakkottai, "Hop-by-hop congestion control over a wireless multi-hop network," *IEEE/ACM Trans. Netw.*, vol. 15, no. 1, pp. 133–144, 2007.
[34] F. Paganini, J. Doyle, and S. Low, "Scalable laws for stable network congestion control," in *Proc. IEEE Decision and Control*, 2001.
[35] F. Qiu and Y. Xue, "Robust joint congestion control and scheduling for time-varying multi-hop wireless networks with feedback delay," *IEEE Trans. Wireless Commun.*, vol. 13, no. 9, pp. 5211–5222, 2014.
[36] W. Su, C. M. Lagoa, and H. Che, "Optimization-based, qos-aware distributed traffic control laws for networks with time-varying link capacities," *Automatica*, vol. 72, pp. 158–165, 2016.
[37] G. Zhang, Y. Wu, and Y. Liu, "Stability and sensitivity for congestion control in wireless mesh networks with time varying link capacities," *Ad Hoc Networks*, vol. 5, no. 6, pp. 769–785, 2007.
[38] M. Welzl, *Scalable performance signalling and congestion avoidance*. Springer Science & Business Media, 2003.
[39] G. Hasegawa and M. Murata, "TCP symbiosis: Congestion control mechanisms of tcp based on lotka-volterra competition model," in *Proc. Interperf*. New York, NY, USA: ACM, 2006.
[40] B. S. Goh, "Global stability in many-species systems," *The American Naturalist*, vol. 111, no. 977, pp. 135–143, 1977.
[41] J. Vano, J. Wildenberg, M. Anderson, J. Noel, and J. Sprott, "Chaos in low-dimensional lotka–volterra models of competition," *Nonlinearity*, vol. 19, no. 10, p. 2391, 2006.
[42] B. Briscoe, K. D. Schepper, M. Bagnulo, and G. White, "Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture," RFC 9330, Jan 2023.
[43] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan, "Beyond throughput, the next generation: a 5g dataset with channel and context metrics," in *Proc. Multimedia Systems*. ACM, 2020, pp. 303–308.
[44] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, Implementation and Evaluation of Congestion Control for Multipath TCP," in *NSDI*, vol. 11, 2011, pp. 8–8.
[45] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 2001.
[46] B. Radunovic and J. Le Boudec, "A unified framework for max-min and min-max fairness with applications," *IEEE/ACM Trans. Netw.*, vol. 15, no. 5, pp. 1073–1083, 2007.

Peyman Teymoori received his Ph.D. degree in computer science from University of Tehran, in 2013. He has been a researcher fellow in the Department of Informatics, University of Oslo, Norway. His research interests include computer network protocols, recursive network architectures, and performance evaluation.

Michael Welzl is a full professor at the University of Oslo, Norway, since 2009. He received his Ph.D. and his habilitation from the University of Darmstadt / Germany in 2002 and 2007, respectively. Michael's main research focus is the transport layer; he is active in the IRTF and the IETF.

David Hayes David A. Hayes received his Ph.D. in Telecommunications Engineering from the University of Melbourne, Australia. He did this work while at Simula Metropolitan Center for Digital Engineering, Oslo, in the Center for Resilient Networks and Applications. David has recently been working on reliable consistent very high data rate mmWave communication and has research interests in network performance and protocol engineering.

APPENDIX A
STABILITY ANALYSIS

A.1 Proof of Theorem 8.1

Proof. We define a Lyapunov function based on the Primal-Dual algorithm in the form of

$$\mathcal{V}(x,p) = \sum_{r\in R}\sum_{i=1}^{O_r}\int_{x_r^*}^{x_r}\frac{1}{x_{r,i}\gamma_{r,i}}(\varphi - x_r^*)d\varphi + \sum_{l\in Q}\int_{p_l^*}^{p_l}\frac{1}{\sigma_l(\psi)}\big(\log_{\phi_2}(1-p_l^*) - \log_{\phi_2}(1-\psi)\big)d\psi \quad (25)$$

which is inspired from the Lyapunov function in [17] where the marking probability is mapped to cost using the function $-\log_\phi(1-p_l^*)$. Fixing the cost dynamics to the Dual1 algorithm and taking the time derivative of $\mathcal{V}(x,p)$ yields

$$\begin{aligned}
\frac{d\mathcal{V}}{dt} =& \sum_{r\in R}\sum_{i=1}^{O_r}\frac{\dot{x}_{r,i}}{x_{r,i}\gamma_{r,i}}(x_{r,i}-x_{r,i}^*) \\
&+\sum_{l\in L}\Big[y_l - c_l\Big]_{p_l}^{+}\big(\log_{\phi_2}(1-p_l^*) - \log_{\phi_2}(1-p_l)\big) \\
&+\sum_{l\in A}\Big[y_l - x_{r,i+1}\Big]_{p_l}^{+}\big(\log_{\phi_2}(1-p_l^*) - \log_{\phi_2}(1-p_l)\big) \\
\leq& \\
&\sum_{r\in R}\sum_{i=1}^{O_r}\big(U^{'}(x_{r,i}) + \log_{\phi_2}(1-p_{r,i})\big)(x_{r,i}-x_{r,i}^*) \\
&+\sum_{l\in L}(y_l - c_l)\big(\log_{\phi_2}(1-p_l^*) - \log_{\phi_2}(1-p_l)\big) \\
&+\sum_{l\in A}(y_l - x_{r,i+1})\big(\log_{\phi_2}(1-p_l^*) - \log_{\phi_2}(1-p_l)\big) \\
=& \sum_{r\in R}\big(\log_{\phi_2}(1-p_{r,i}) - \log_{\phi_2}(1-p_r^*)\big)(x_{r,i}-x_{r,i}^*) \quad (26a)\\
&+\sum_{l\in L}\big(\log_{\phi_2}(1-p_l^*) - \log_{\phi_2}(1-p_l)\big)(y_l - y_l^*) \quad (26b)\\
&+\sum_{l\in A}\big(\log_{\phi_2}(1-p_l^*) - \log_{\phi_2}(1-p_l)\big)(y_l - y_l^*) \quad (26c)\\
&+\sum_{l\in L}(y_l^* - c_l)\big(\log_{\phi_2}(1-p_l^*) - \log_{\phi_2}(1-p_l)\big) \quad (26d)\\
&+\sum_{l\in A}(y_l^* - x_{r,i+1})\big(\log_{\phi_2}(1-p_l^*) - \log_{\phi_2}(1-p_l)\big) \quad (26e)\\
&+\sum_{r\in R}\big(U^{'}(x_{r,i}) + \log_{\phi_2}(1-p_{r,i}^*)\big)(x_{r,i}-x_{r,i}^*) \quad (26f)\\
\leq& 0\ .
\end{aligned}$$

The sum of (26a), (26b), and (26c) is equal to zero. (26d) and (26e) are less than or equal to zero: if, for example, $y_l^* = c_l$, then (26d) is zero, and if $y_l^* < c_l$, then p_l^*, which makes $\log_\phi(1-p_l^*)$ zero, and since $-\log_\phi(1-p_l) \geq 0$, this term is non-positive. The same argument holds for (26e). In order for (26f) to be non-positive, if $x_r \leq x_r^*$, then we should have $U_r^{'}(x_r) \geq -\log_\phi(1-p_r^*)$, or vice versa. This means that, for example, in case of $x_r \leq x_r^*$, $x_{r,i+1}$ should be large enough to let the first term become non-negative. Since we connect each loop to the next one (i.e. the carrying capacity in each loop is only dependent on the rate of the next loop, not the previous one), loops converge from the last one, and for each loop, $x_{r,i+1}$ will increase until it meets the above condition and stabilizes the system. In case of $x_r \geq x_r^*$, the first term should be non-positive. Since the value of $x_{r,i+1}$ does not depend on $x_{r,i}$, this means that $x_{r,i+1}$ should be small enough for the term to become non-positive. Again, as the next loop converges first, and it does not depend on the previous one, this condition is met. This means that the Lyapunov function is negative, and it is zero at the equilibrium, making the system globally, asymptotically stable. □

A.2 Proof of Theorem 8.2

The following theorem establishes the necessary and sufficient conditions for local stability.

Theorem A.1 (see [45]). *The equilibrium $\mathbf{x}^*$ of the system of equations $\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x})$ is locally stable if and only if all the eigenvalues of the Jacobian of the system of equations evaluated at $\mathbf{x}^*$ have real parts less than zero.*

Now we prove Theorem 8.2:

Proof. First, we write the system equations and then, we obtain the equilibrium. We use (9) to obtain system dynamics. Consider some LGCC router i at depth d of the aggregation tree whose flows are aggregated into flow $x_{k,j}$ in loop j at depth $d+1$. The incoming flows to router i are governed by

$$\dot{x}_{1,i} = x_{1,i}(t)\gamma_d\left(1 - \frac{x_{1,i}(t)}{x_{k,j}(t)} - p_i(t)\right)\ , \quad (27a)$$

$$\dot{x}_{2,i} = x_{2,i}(t)\gamma_d\left(1 - \frac{x_{2,i}(t)}{x_{k,j}(t)} - p_i(t)\right)\ , \quad (27b)$$

$$\vdots$$

$$\dot{x}_{\zeta,i} = x_{b,i}(t)\gamma_d\left(1 - \frac{x_{b,i}(t)}{x_{k,j}(t)} - p_i(t)\right)\ , \quad (27c)$$

where

$$\dot{p_i} = \beta_d\Big[\big(\sum_{y=1}^{\zeta}x_{y,i}\big) - x_{k,j}\Big]\ . \quad (28)$$

In the loop at depth δ, $x_{k,j} = K$. The equilibrium of the above system is given by

$$p_i^* = \frac{\zeta-1}{\zeta}\ , \quad (29)$$

$$x_{.,i}^* = (1-p_i^*)x_{k,j}^*\ , \quad (30)$$

where $x_{k,j}^* = K$ for $d = \delta$. Now we obtain the Jacobian of the above system. In each loop, there are $\zeta+1$ variables: ζ equations representing flows' rate, and one equation adjusting the cost. This makes a Jacobian matrix of dimension $\frac{(1+\zeta)\left(-1+\zeta^\delta\right)}{-1+\zeta}$ (this is obtained by considering that the aggregation tree is a perfect m-ary tree, where the number of nodes determines the number of rate dynamics, and the number of non-leaf nodes also determines the number of LGCC routers). Although obtaining eigenvalues of a matrix is not trivial, since the above matrix has a special form, we can calculate its eigenvalues for an arbitrary ζ and δ. Each

element of the Jacobian matrix is given by $\frac{\partial \dot{\chi}}{\partial \upsilon}$ where χ and υ are any two system variables. For example, for $0 \leq b \leq \zeta$

$$\frac{\partial \dot{x}_{b,i}}{\partial x_{b,i}} = \gamma_d \left(1 - \frac{x_{b,i}}{x_{k,j}} - p_i\right) - \gamma_d \frac{x_{b,i}}{x_{k,j}} \ , \tag{31}$$

$$\frac{\partial \dot{x}_{b,i}}{\partial p_i} = \gamma_d x_{b,i} \ , \tag{32}$$

$$\frac{\partial \dot{p}_i}{\partial x_{b,i}} = \gamma_d \ , \tag{33}$$

and the partial derivatives are zero for other variables. This matrix has a number of eigenvalues; some of them are negative, and some of them are in the form of

$$\frac{1}{4}\left(- \gamma_d + \sqrt{-4K\zeta^{(d+2-\delta)}\gamma_d\beta_d + \gamma_d^2}\right) \ .$$

Constraining this eigenvalue to be negative and solving for a relationship between γ and δ yields (23). □

APPENDIX B
FAIRNESS ANALYSIS

B.1 Proof of Theorem 8.3

First, we provide a definition and theorem from [46] on max-min fairness.

Definition 2 (Bottleneck Link)**.** *Link l is a bottleneck for source r if and only if link l is saturated, and source r has the maximum rate among all sources crossing link l.*

Theorem B.1. *A feasible rate allocation vector to all sources is max-min fair if and only if every source has a bottleneck link.*

Now we prove Theorem 8.3:

Proof. At every LGCC router, there is a schedule per-output port. Taking into account that all the incoming flows that are forwarded towards the same output port are aggregated, and they are fairly scheduled, if this is LGCC router 1, then it is a bottleneck for source r; if the bottleneck is at some router i, then our weight assignment guarantees that again this is a bottleneck for source r, and hence, every source has a bottleneck, confirming that the policy attains max-min fairness. □

B.2 Proof of Theorem 8.4

Lemma B.2. *A proportionally-fair rate allocation vector assigns rates inversely proportional to the end-to-end cost the network charges each source r.*

Proof. With a logarithmic utility function, which attains proportional fairness, $x_r^* = \frac{1}{\sum_{l \in L_r} q_l}$. □

Now we prove Theorem 8.3:

Proof. According to the employed scheduling policy, as flow r crosses multiple routers, each router's cost is accumulated. Consider the last router with a non-zero cost: if it constrains x_r to $\frac{1}{\sum_{l \in L_r} q_l}$, then every flow r gets a share of the bandwidth inversely proportional to its cost. Since the sum of costs is less constrained at previous routers, it is the role of the last router to enforce fairness; previous loops of flow r are backpressured due to this constraint. This is similar to limiting the carrying capacity at the last router, not to increase more than a proportionally fair share. In [17], we proved that the function $-\log(1-p_{r,i}^u)$ provides the cost using ECN. Therefore, setting weights inversely proportional to this value guarantees proportional fairness. □