2016 IEEE 41st Conference on Local Computer Networks

Even Lower Latency, Even Better Fairness: Logistic Growth Congestion Control in Datacenters

Peyman Teymoori, David Hayes, Michael Welzl, Stein Gjessing
Department of Informatics
University of Oslo
Email: {peymant|davihay|michawe|steing}@ifi.uio.no

***Abstract*—Datacenter transport has attracted much recent interest, however, most proposed improvements require changing the datacenter fabric, which hinders their applicability and deployability over commodity hardware. In this paper, we present a novel congestion controller, Logistic Growth Control (LGC), for datacenters which does not require changes to the datacenter fabric. LGC uses a similar ECN marking as in DCTCP, but adapts to congestion using the logistic growth function. This function has been proven to have nice characteristics including stability, convergence, fairness, and scalability, which are very appealing for congestion control. As a result, our LGC mechanism operates in the datacenter network in a more stable and fair manner, leading to less queuing and latency. LGC also behaves better than DCTCP, and it converges to the fair share of the bottleneck link capacity irrespective of the Round-Trip-Time (RTT). We discuss the stability and fairness of LGC using a fluid model, and show its performance improvement with simulations.**

I. Introduction

Standard transport protocols, such as the Transmission Control Protocol (TCP), do not perform well in datacenters [1]. Optimizing both the end systems and network fabric to the datacenter environment (e.g. [2]) has many benefits, but often proves difficult or impossible to deploy (see [3]), and the network fabric needs special approaches such as [4] to be able to run the new algorithm. In addition, in spite of the performance improvements achieved by custom-built datacenter fabrics, the price differences of commodity versus non-commodity networking hardware, especially at a large scale, has been a strong motivation towards using commodity hardware [5], [6].

Another approach in datacenters has been to focus on easily deployable solutions. These include purely end-system modifications, such as Data Center TCP (DCTCP) [1]) as well as scheduling transmissions [7], [8], though the later has scalability concerns due to their complexity or the use of a centralized arbiter [9]. Hence, like DCTCP [1], we too focus on an easily deployable mechanism. Such a mechanism operates purely at the transport layer of end systems and does not require changes to network equipments such as switches and routers.

Our work is inspired by logistic growth in nature [10] (see section II-A). We build upon earlier work that has found logistic growth to be a generally useful function for congestion control [11], [12], [13], and present the design and simulation-based evaluation of a new congestion controller for datacenters that is based on logistic growth. Cornerstones of our design are:

- Similar to DCTCP, we let packets be ECN-marked when the instantaneous queue length exceeds a threshold (which can be achieved using a special configuration of the common RED Active Queue Management (AQM) mechanism, and hence needs no hardware changes). However, different from DCTCP where this threshold is a function of the Bandwidth×Delay Product (BDP) [1], which can be very large in modern datacenters [14], our threshold is always set to only one packet, irrespective of the BDP.
- We utilize a similar method of echoing ECN (acks and delayed acks) as DCTCP. However, how sources react to ECN signals is governed by our new congestion controller.
- We do not let the queue grow, neither do we let the queue length oscillate a lot. We achieve this by using a more stable congestion controller that is based on the logistic growth function. This function has stability properties when used in congestion control (see Section IV-C), and lets us attain fairness among flows irrespective of the RTT, which is not the case for TCP and DCTCP.

Through presenting an analytical model, we discuss LGC properties, and through simulation in OMNeT++ [15], we show that it works under various scenarios. Simulation results show that it can reduce the queue length and latency, and it attains fairness among flows with heterogeneous RTTs.

Section II presents a congestion controller model based on the logistic growth function. In Section III we develop our congestion controller based on this model and then analytically evaluate it in Section IV. We show how it can be further improved in Section V before evaluating the performance of our logistic growth based congestion controller through simulations in Section VI. We finish with an overview of related work in Section VII and our conclusions in Section VIII.

II. Congestion Controller Model

A. Logistic Growth

The Logistic Growth (LG) function is often used to specify the growth of a population/species over time. It is described by the differential equation

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

© 2016, Peyman Teymoori. Under license to IEEE.
DOI 10.1109/LCN.2016.12

IEEE computer society

TABLE I
NOTATIONS

Symbol	Description
S	the number of competing flows/species
x_i	the normalized rate of flow i
$x_i(0)$	the initial rate/population of flow/species i
r_i	the growth rate of flow i
a_{ij}	the competitive effect of species i on species j
$l(t)$ or $l[n]$	$\frac{S-1}{S}$ times the total load on the bottleneck link
$\hat{l}_i(t)$ or $\hat{l}_i[n]$	the approximation of $l(t)$ and $l[n]$ respectively
$c(t)$ or $c[n]$	the ECN-marking process
$\hat{x}_i(t)$ or $\hat{x}_i[n]$	an approximation of the fair sending rate of source i
K	the carrying capacity: the source's maximum sending rate (the normalization factor)
k	the ratio of the capacity of the bottleneck link to K
var(t)	is used to represent a continuous time variable
var$[n]$	is used to represent discrete variable at epoch n

where N is the size of a population, K the so-called "carrying capacity" (the value that the equation converges to), and r the maximum per capita growth rate for a population. We use the dot notation for time differentiation, i.e. $\dot{N} = \frac{dN}{dt}$. Table I summarizes the notation we use throughout the paper.

The idea of growth constrained by a capacity limit has direct parallels with network congestion control. It allows us to draw upon the large body of analytic research work on this model and its enhancements (e.g. [16]) which has proven nice characteristics including stability, convergence, fairness, and scalability. These properties make LG very appealing for congestion control [13].

We normalize (1) by defining $x(t) = N(t)/K$, which yields

$$\dot{x} = rx(t)\,(1 - x(t)) \tag{2}$$

with the solution:

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

where $x(0)$ denotes the initial population. Clearly, $\lim_{t\to\infty} x(t) = 1$. In the rest of the paper, we use LG in this normalized form.

Fig. 1 plots (3) for different values of r, which affects the speed of convergence. In this diagram, there is one species with the normalized initial population of 0.1. We see that larger values of r result in a faster convergence to the carrying capacity 1. The LG function (3), however, models the population of only one species. When there is a competition among a number of species for a common resource, the competition is modeled using the Lotka-Volterra model, which can also parallel different network sources competing for shared network capacity.

B. Lotka-Volterra Competition Model

We consider the general Lotka-Volterra model of competition [17], where S species compete for a common limited

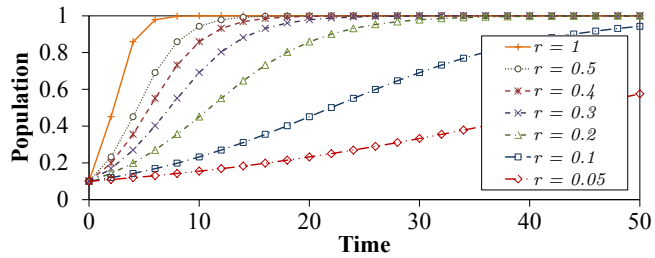


Fig. 1. Growth rate effects on the population growth of LG. $x(0) = 0.1$.

resource according to the Logistic Growth equation

$$\dot{x_i} = x_i r_i \left(1 - \sum_{j=1}^{S} a_{ij} x_j\right) \tag{4}$$

where $r_i > 0$ denotes the growth rate of species i. $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{S\times S}$ is called the community matrix [18], where the value of a_{ij} determines the competitive effect of species i on species j.

We define matrix $\mathbf{A}$ as

$$a_{ii} = \frac{2S-1}{S}, \qquad a_{ij} = \frac{S-1}{S} \forall i \neq j. \tag{5}$$

When $a_{ij} > 0\ \forall i \neq j$, all species i have a competitive effect on all species j. This parallels network flows competing with one another. However if a_{ij} and a_{ji} have different signs, the interactions are more complex with predation between species. This can be used for more complex network analysis, but is beyond the scope of this paper.

The definition of $\mathbf{A}$ directly affects the stability of the system represented by (4) [19]; in the following, we elaborate more on why we define $\mathbf{A}$ as (5).

1) Equilibrium: The equilibria of (4), denoted by $\mathbf{x}^*$ in the vector form with elements x_i^*, are determined by solving the system of equations $\dot{x}_i = 0\ \forall i$. This implies that either $x_i = 0$ or $1 - \sum_{j=1}^{S} a_{ij}x_j = 0$. If no species dies out, $x_i > 0$ (or for network all flows are sending), and using (5), the only equilibrium point we get is

$$x_i^* = \frac{1}{S}. \tag{6}$$

Thus all species, or flows in our case, converge to an equal share of the capacity, jointly converging to 1.

2) Stability: It can easily be shown that because in our case matrix $\mathbf{A}$ is symmetric with positive real elements, its eigenvalues are real and positive. According to (5), $a_{ii} > a_{ij}\ \forall i \neq j$, which makes all the pivots of $\mathbf{A}$ positive. This accordingly implies that all eigenvalues of $\mathbf{A}$ are positive. Thus the system represented by (4) is stable around the equilibrium (6) [20]. The symmetric property of $\mathbf{A}$ with $a_{ij} > 0$ results in another nice property of the model: the above equilibrium becomes globally stable because there can be found a positive definite Lyapunov function minimized at the equilibrium with positive values at other points [20], [16]. In the networking context, this means that all network flows competing for the bottleneck link capacity using (4) will

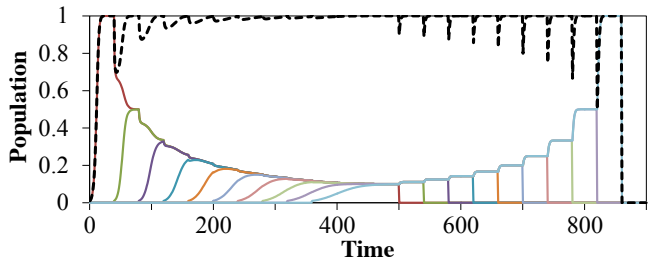


Fig. 2. Competition of 10 species (the solid lines) with $r_i = 0.5$ based on the LV system. The dashed line is the total population.

eventually converge to a stable equilibrium (6), irrespective of their initial sending rate.

3) Numerical evaluation: To illustrate how the system of equations presented in (4) behaves, a sample trajectory of 10 species is plotted in Fig. 2. Each species starts with an initial population of 0.01, but at different time instants. All the numbers are normalized with the carrying capacity equal to 1. It is clearly seen in Fig. 2 how each species' population grows until it receives its fair share of the total capacity, which is 0.1 once all 10 species have joined at around 400. As species *leave* the competition one by one, meaning that they are removed from the system, we can observe how the system converges to the new equilibrium. The stable adaption for varying numbers of species parallels varying number of network flows and how capacity is shared between them in the networking context.

III. Logistic Growth Control (LGC)

A. Congestion Controller

Applying (4) to the context of a distributed congestion controller that operates at discrete time intervals, we write (4) with (5) as follows:

$$x_i[n+1] = x_i[n] r_i \left(1 - \frac{2S-1}{S} x_i[n] - \frac{S-1}{S} \sum_{j, j \neq i} x_j[n]\right) + x_i[n]. \tag{7}$$

After simplifying the right-hand side of (7), we get

$$x_i[n+1] = x_i[n] r_i \left(1 - x_i[n] - l[n]\right) + x_i[n] \tag{8}$$

where $l[n] = \frac{S-1}{S} \sum_j x_j[n]$. At equilibrium $x_i[n+1] = x_i[n]$ in (7), yielding $x_i^* = \frac{1}{S} \ \forall i$ at that fixed point. This shows that the discrete equation has the same equilibrium point as (4).

Key to implementing LGC in a datacenter is being able to estimate $l[n]$. This requires that each source, i, is able to estimate the number of competing flows, S, as well as the combined sources transmission rate, $\sum_j x_j[n]$ (Remember these rates are normalized with respect to the source link's capacity). In the next section, we will explain how each source can have a good approximation of $l[n]$ without requiring to know the exact value of S.

B. Estimating $l[n]$

In an unmodified IP-based datacenter fabric, exactly calculating $l[n]$ is not possible. The only signal from the network is the ECN flag in the packet header; a set flag indicates the queue length has exceeded a particular threshold, and a clear flag means it hasn't. However, we show that it is possible to accurately estimate $l[n]$ using ECN in the datacenter environment when for source i, $\hat{l}_i[n] = c_i[n]$, where $c_i[n]$ is the proportion of congestion indicating packets (ECN marked) in epoch n. Thus equation (8) is implemented as:

$$x_i[n+1] = x_i[n] r_i \big(1 - x_i[n] - \hat{l}_i[n]\big) + x_i[n] \,. \tag{9}$$

When a source starts transmission, it does not have any information to calculate $\hat{l}[0]$ so we start with $\hat{l}[0] = 0$. Section IV-D discusses the accuracy of this estimate in detail.

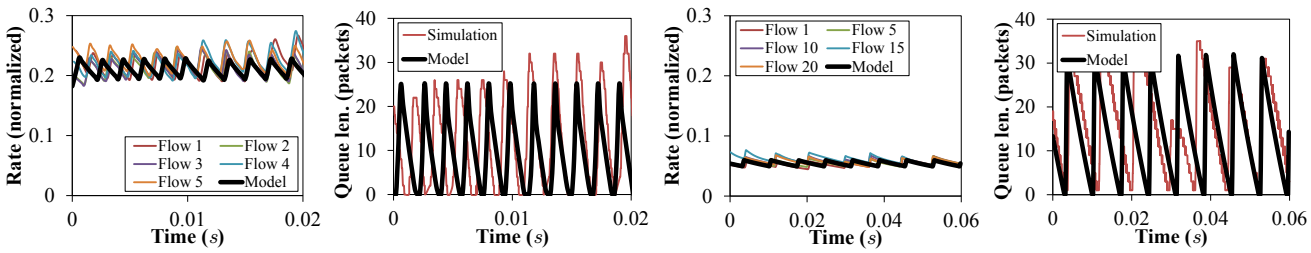
C. The Link Capacity

In (8) the capacity is normalized. Therefore, to send at a specific rate, every source should send at $x_i K$ where K is the maximum send rate of the source. In a general network topology, the link capacities in a path might be different or unknown. However, all the link capacities and the topology are known in datacenters, and as a general rule, we assume that K represents the smallest link capacity in a path. For example, in a datacenter with 10Gbps links between servers and TOR switches, and 40Gbps links between other switches, we use 10Gbps as the carrying capacity, K, in the calculations. We will show that if the 40Gbps link is congested, LCG is also able to react appropriately to the congestion, i.e. congestion in the backbone. This suggests that LGC may also be able to be applied outside datacenters to networks where the bottleneck capacity is not known.

D. Rate Update Time Intervals

The rate is regularly updated by each source using (9). As we see in (9), the fair rate only depends on $\hat{l}_i$, and as long as all sources *see* the same $\hat{l}$ (see Section IV-D), 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). However, using a flow's smallest RTT can still be a long time interval for a large-RTT flow, making it sluggish.

In LGC, we use equal-sized intervals. Based on our observation in simulations, performance is robust if a flow can receive at least 4 acks per interval. Although there are many active flows in a datacenter, the number of competing flows for the same link is small [21], [9]. This means a rate update interval as short as the minimum RTT in a datacenter will still allow flows around 4 acks per interval (or the equivalent information using DCTCP's delayed ack mechanism).



(a) Rate of 5 flows

(b) Queue length of the bottleneck link when 5 flows are competing

(c) Rate of 20 flows. 5 sample flows are only shown.

(d) Queue length of the bottleneck link when 20 flows are competing

Fig. 3. Validation of the fluid model against simulation. The bottleneck link capacity is 10 Gbps, and the RTT is 200 μs

IV. Analytical Evaluation of LGC

We use a fluid model of (9) to investigate its stability and the accuracy of $\hat{l}$.

A. The Fluid Model

We model (9) as a continuous time fluid process:

$$\dot{x_i} = x_i(t) r_i \left(1 - x_i(t) - \hat{l}_i(t)\right) \tag{10}$$

$$\dot{\hat{l}}_i = c\left(t - \tau_i\right) - c\left(t - 2\tau_i\right) \tag{11}$$

$$c(t) = 1_{\{q(t)>0\}} \tag{12}$$

$$q(t) = -k + \sum_{i=1}^{S} x_i(t) \tag{13}$$

where $c(t)$ is the ECN-marking process, τ_i is the time that takes for sender i to receive the ECN signal from the bottleneck link, and k denotes the ratio of the bottleneck link capacity to K. Unless otherwise specified we use $k = 1$ in calculations.

B. Model Validation

We validate the fluid model using simulations of the LGC mechanism performed in OMNeT++. Two scenarios are considered: 5 flows and 20 flows. Fig. 3 shows the comparative results for the source send rates and bottleneck queue size. We had two scenarios with 5 and 20 flows. In both scenarios the bottleneck link capacity is 10 Gbps with a source RTT of 200 μs. We see that in both cases, the rate of sources and the queue length match the model.

C. Stability and Equilibrium

The stability of LGC cannot be directly investigated by (9) because there is a delay in getting ECN signals from the bottleneck link router, i.e. $\hat{l}(t) = c(t - \tau)$. As it is illustrated in Fig. 3(a) and Fig. 3(c), this causes some periodic behavior. However, assuming that $c(t - \tau) \approx c(t - 2\tau)$ and considering the system model (10)-(13), we are able to 1) directly prove the stability of (9) based on the discussion in II-B2, and 2) calculate the equilibrium of LCG. Setting the right-hand side (RHS) of (10), (11), and (13) to zero and trying to solve the equations yields

$$1 - x^* - \hat{l}^* = 0 \ , \tag{14}$$

$$c^* = \hat{l}^* \ , \tag{15}$$

$$S(x^* + x^* r(1 - x^* - \hat{l}^*)) = k \ . \tag{16}$$

Assuming $k = 1$, the above system has a solution at equilibrium of $x^* = \frac{1}{S}$ with $\hat{l}^* = \frac{S-1}{S}$. This also reveals that process $c(t)$ approximates $\frac{S-1}{S}$ at equilibrium, i.e. $\frac{S-1}{S}$ percent of packets are ECN-marked. Since the number of competing flows is usually small in datacenters [21], [9], we do not expect $\frac{S-1}{S}$ becomes very close to 1. In Section V-A, we will discuss how we can have $c(t - \tau) \approx c(t - 2\tau)$, i.e. how to get approximately the same ECN-marked percentage in consecutive rate update intervals.

D. Accuracy of $\hat{l}_i(t)$

We discuss in more detail why $\hat{l}_i(t)$ is a good approximation of $l_i(t)$ especially when $\sum_j x_j(t)$ is close to 1, i.e. the sum of source rates is close to the bottleneck link capacity. Formally speaking, we claim that

$$\lim_{t_{\max}\to\infty} \left(\frac{1}{t_{\max}} \int_0^{t_{\max}} c(t)dt \right) = \frac{S-1}{S}. \tag{17}$$

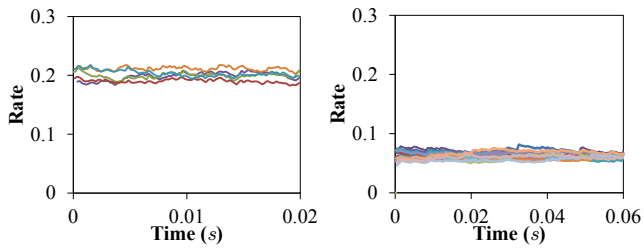
However, without requiring to solve (17), we can look at the RHS of (17) as how c behaves around the equilibrium. We solve (10) to obtain the relationship between system variables; this yields

$$\begin{aligned} x_1^* + \hat{l}_1^* &= 1 \ , \\ x_2^* + \hat{l}_2^* &= 1 \ , \\ &\vdots \\ x_S^* + \hat{l}_S^* &= 1 \ . \end{aligned} \tag{18}$$

Let us assume that sources experience the same proportion of ECN-marked packets (or very close to the same). We will discuss how this can be assured in Section V-A. This yields $\hat{l}_i^* = \hat{l}^* \ \forall i$. From (18) we obtain

$$x_1^* = x_2^* = ... = x_S^* = 1 - \hat{l}^* \ . \tag{19}$$

Since sources keep increasing their sending rate until the bottleneck link is 100% utilized, we have at equilibrium



(a) Rate of 5 flows (b) Rate of 20 flows

Fig. 4. Source send rates with exponentially-distributed packet pacing.

$\sum_{i=1}^{S} x_i^* = 1$ and since all sources transmit at the same rate, $x_i^* = \frac{1}{S}$. Thus (19) yields $\hat{l}^* = \frac{S-1}{S}$ which confirms that $l^* = \hat{l}^*$. This implies that the ECN marking process $c(t)$ also approximates $\frac{S-1}{S}$.

1) Validation Results: We validated the above argument using the simulation scenario of Section IV-B; the average value of $\hat{l}$ in case of 5 flows with three digit precision is 0.796 with a standard deviation of 0.124, and in case of 20 flows, it is 0.937 with a standard deviation of 0.113. We see that in both cases, the measured values by sources are very close to $\frac{S-1}{S}$, i.e. 0.8 in the first case, and 0.95 in the second one.

E. Different Bottleneck Link Capacity

Here we discuss what happens if the bottleneck link capacity is not equal to the maximum sending rate of the sources. Referring to (14)-(16), in general at equilibrium the normalized source send rate and load indicator are:

$$x^* = \begin{cases} \frac{k}{S} & k < S \\ 1 & k \geq S \end{cases} \quad \text{and} \quad \hat{l}^* = \begin{cases} \frac{S-k}{S} & k < S \\ 0 & k \geq S \end{cases}$$

If $k > S$, the sources combined maximum send rate is less than the bottleneck link's capacity, giving $x_i^* = 1$ and $\hat{l}^* = 0$. We ran a simulation experiment where there are 10 competing nodes connected to a 10 Gbps link (K=10 Gbps), and the bottleneck link capacity took the values $\{10, 20, 30, 40, 50, 60\}$ Gbps resulting in respective values of $k = \{1, 2, 3, 4, 5, 6\}$. We also added 40% of the link capacity Poisson traffic as the background traffic. In all six cases each nodes' normalized send rate was very close to $x_i^* = \frac{k}{S}$; confirming the above argument.

V. Further Improvements

A. Exponential Delay

As we discussed in Section IV-C, having $c(t-\tau) \approx c(t-2\tau)$ helps improve the stable behavior of LGC. Although a moving average with a large-enough window over ECN signals can reflect the same value, we are able to improve it further by pacing packets with an exponentially-distributed delay between them. The advantage of rate-based transmission with exponentially-distributed inter-packet delay is that, due to the PASTA property (Poisson Arrivals See Time Averages) [22], sources in the limit observe the same average congestion marking. When measured over the epoch n we have $\hat{l}_i(n) \approx \hat{l}_j(n) \approx \frac{S-1}{S}\ \forall i, j$.

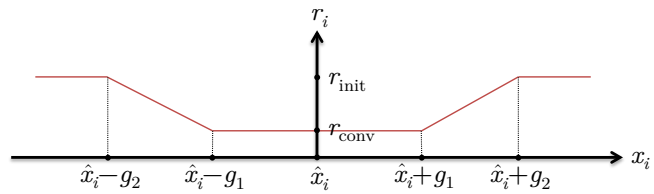


Fig. 5. Adapting r depending on the expected fair share and current rate.

Fig. 4 shows the send rates achieved by the LGC sources over time when packets are paced with approximately exponentially-distributed inter-packet times. Fig. 4(a) and 4(b) can be compared with the results without pacing in Fig. 3(a) and 3(c). From these figures, we observe that rates are more stable with few large abrupt changes. It is worth noting that pacing at a high rate in data centers needs special treatment in terms of timers. The pacing performed by other methods such as TIMELY [9] confirms that the high transmission rate is not a barrier to pacing. In Section VI, we show how adding exponentially-distributed delays affects queue length.

B. Adapting the growth rate

The performance of LGC can be improved by tuning the growth rate parameter, r. Here, our goal is *to be more aggressive in changing the rate of a source that is getting a smaller or larger amount of resource than the fair share*.

Since $\hat{l}_i \approx \frac{S-1}{S}$, and $1 - \hat{l}_i \approx \frac{1}{S}$, we estimate the fair share as $\hat{x}_i = 1 - \hat{l}_i$. However, to avoid fluctuations we estimate this over a larger time interval:

$$\hat{x}_i[n] \triangleq 1 - \frac{1}{W} \sum_{w=1}^{W} c_i[n-w]$$

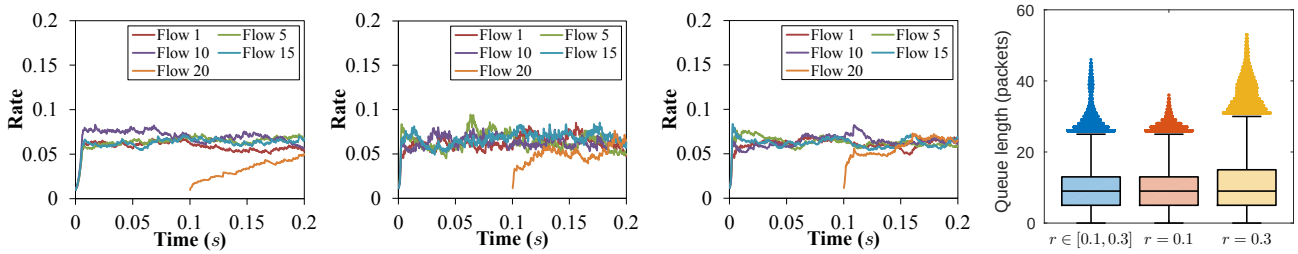
where W denotes the moving average window size.

In Fig. 1, we see how r affects aggressiveness. We would like to dynamically change r to decrease the convergence time of sources that are far from their fair share and stably maintain source rates when sources are close to their fair share. To achieve this we define a range of values where $r \in [r_{\text{conv}}, r_{\text{init}}]$. r_{init} denotes the initial value of r. Sources start by setting $r = r_{\text{init}}$, and they use r_{conv} when $x_i \approx \hat{x}_i$. Fig. 5 shows how LGC chooses the value for r_i for source i. Depending on the value of $\hat{x}_i$, we use a simple linear function which is bounded by r_{conv} and r_{init}. g_1 and g_2 denote the rate range used for changing between r_{conv} and r_{init}.

Fig. 6 illustrates three different scenarios: $r = 0.1$, $r = 0.3$, and adaptive r. $g_1 = 0.015$ and $g_2 = 0.45$. We see that using an adaptive r, as shown in Fig. 6(c), LGC takes advantage of the stability seen in Fig. 6(a) and faster convergence of Fig. 6(b) without any significant effects on the queue length distributions illustrated in Fig. 6(d) using box-and-whisker[1] plots.

Putting all together, Algorithm 1 shows how LGC congestion control works: first, it initializes its variables, and then,

[1]Boxes span the middle 50% of data, with whiskers extending up to 1.5 times the interquartile range. Outliers are offset so the density of points is clear.



(a) Rate of 5 sample flows out of 20 where $r = 0.1$.

(b) Rate of 5 sample flows out of 20 where $r = 0.3$.

(c) Rate of 5 sample flows out of 20 where $r \in [0.1, 0.3]$

(d) Queue length distribution for LGC.

Fig. 6. The effect of the growth parameter, r, on source aggressiveness. 19 sources start sending at $t = 0$ s, with the 20th source joining at $t = 0.1$ s.

Algorithm 1 LGC congection control

procedure INIT
  $x \leftarrow x_{\text{init}}$
  $l \leftarrow 0$
  $r \leftarrow r_{\text{init}}$
end procedure
procedure ONINTERVALENDS
  $l \leftarrow$ ECN-marked percentage of ACKs
  $\hat{x} \leftarrow 1 - \text{movingAvg}(l, W)$
  if $|x - \hat{x}| \leq g_1$ **then**
    $r \leftarrow r_{\text{conv}}$
  else if $|x - \hat{x}| \geq g_2$ **then**
    $r \leftarrow r_{\text{init}}$
  else
    $r \leftarrow |x - \hat{x}| \frac{r_{\text{init}} - r_{\text{conv}}}{g_2 - g_1} + r_{\text{conv}}$
  end if
  $x \leftarrow \max(xr(1 - x - l) + x, x_{\text{init}})$
end procedure

at the end of every rate update interval, based on l, the ECN-marked percentage of ACKs during that interval, new rate is calculated. It also adapts r if the sending rate is not *close* to the fair share approximation, i.e. $\hat{x}$. The source then should send at xK where K is the maximum sending rate. However, if there are more than one packet in the transmission queue, the source should not send them back-to-back; instead, it should add exponentially-distributed inter-packet delay between packets such that the sending rate is xK on average.

VI. COMPARATIVE PERFORMANCE RESULTS

In this section we present the comparative performance of LGC with DCTCP. We chose DCTCP for comparison because it is a well-known transport protocol for datacenters, and has the same goals as LGC. We implemented both LGC and DCTCP in the INET framework of OMNeT++ [15]. Our OMNeT++ DCTCP implementation was validated against its reference *NS*-2 implementation.

Results in this section are collected from a series of 10 simulation runs, each with unique random seeds. Sources randomly start transmission in an interval shorter than one RTT. Unless otherwise mentioned, we use the following parameters for LGC: $g_1 = 0.015$, $g_2 = 0.045$, $r_{\text{init}} = 0.3$, $r_{\text{conv}} = 0.1$, $w_l = 40$, and a rate update interval of $200\,\mu$s. We use the

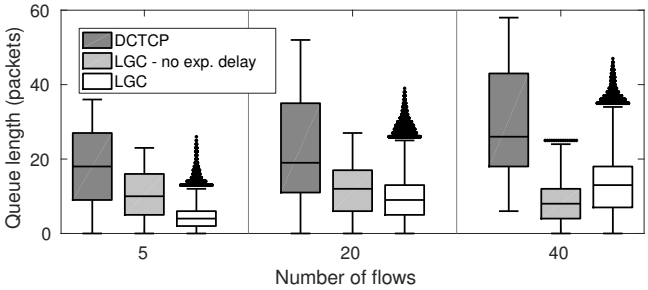


Fig. 7. Queue length distribution of the three methods.

recommended configurations of DCTCP in [1] for optimum operation.

A. Small-Scale Evaluation

We simulated an incast traffic pattern for evaluation. Incast is an important scenario to evaluate congestion control in datacenters [9]. This traffic pattern occurs when client sends a request to several servers in the same rack connected to the same switch, and the servers all respond at almost the same time exceeding the capacity of the link to the client. In our scenario, the link capacity between the rack devices and the Top Of Rack (TOR) switch is 10Gbps.

1) Queue Length: To evaluate the effect the different mechanisms have on the bottleneck queue length distribution we test 5, 20, and then 40 sources (clients) sending data to the same destination (server). Fig. 7 shows a box-and-whisker plot[2] for DCTCP, LGC without exponentially distributed packet pacing, and LGC. We observe that LGC reduces the queue length compared to DCTCP for a comparable average throughput as reported in Table II. The exponentially distributed packet pacing further reduces the queue length, though the benefit diminishes as the number of flows increase because flows get fewer acks per rate update intervals resulting in their load estimate $\hat{l}[n]$ becoming coarser. If LGC is to be used in scenarios where more than 20 competing flows is common, we suggest increasing the rate update interval to improve the precision of $\hat{l}[n]$. In the "LGC - no exp. delay" case the sources become synchronized when there are large numbers

[2]Boxes span the middle 50% of data, with whiskers extending up to 1.5 times the interquartile range. Outliers are offset so the density of points is clear.

TABLE II
THROUGHPUT COMPARISON UNDER DIFFERENT TOPOLOGIES

Topology	Throughput (Gbps)		
	DCTCP	LGC	LGC-no exp
Incast	9.64	9.50	9.52
Clos	9.56	9.45	-
Leaf-Spine	9.51	9.41	-

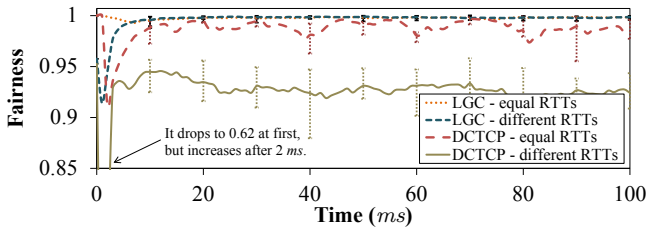


Fig. 8. Short-term fairness of 5 competing flows using LGC and DCTCP under equal and different RTTs using Jain's fairness index. Error bars span the range of values at 10 ms intervals.

of competing flows, resulting in load estimates of $\hat{l}[n] = 1$ for most rate update intervals, and $\hat{l}[n] = 0$ for a small number of intervals. The source synchronization, LG response to $\hat{l}$, and more deterministic packet arrival pattern result in lower queuing delays.

2) Fairness: We compare the short-term flow fairness of DCTCP and LGC using Jain's fairness index [23]. 5 sources send data to the same destination, with Jain's fairness index calculated over 1 ms intervals. Both LGC and DCTCP are tested in two scenarios: (i) all sources have homogeneous RTTs of 200 μs, (ii) the 5 sources have RTTs of 140, 170, 200, 230, and 260 μs respectively. Fig. 8 illustrates four simulation cases plotting the average of the 10 different runs. Error bars spanning the range of calculated fairness are plotted every 10 ms to indicate the variation in simulation runs.

First considering the case of homogeneous RTTs, we observe that LGC's fairness begins at close to 1, and remains very close to 1 throughout the simulation. DCTCP sources start using the same congestion window, which makes it fair at first. However, during first RTTs, sources do not have an accurate approximation of congestion (parameter α in [1]), and their start time is slightly different, which leads to a drop in fairness. As sources receive more acks, fairness improves, but it fluctuates in the range of 0.97 to 1. The fairness drop does not happen in LGC during first RTTs because sources pace packets with exponentially-distributed delays, which compensates for the initial inaccurate estimate and slight differences in start times.

In the heterogeneous RTT case both LGC and DCTCP sources start using the same congestion window size, but with the RTT difference this yields a fairness of around 0.96 at first. The LGC sources converge to their fair share (≈ 1.0) in just a few milliseconds, however, the DCTCP sources–after a drop in fairness due to not having an accurate approximation of congestion–attain only a fairness of about 0.93. This illustrates RTT independence of LGC's mechanism.

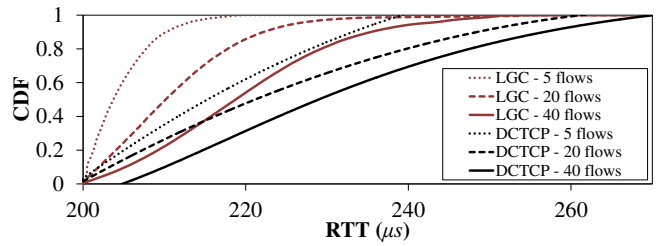


Fig. 9. Cumulative distribution function of the RTT distribution in an Incast scenario.

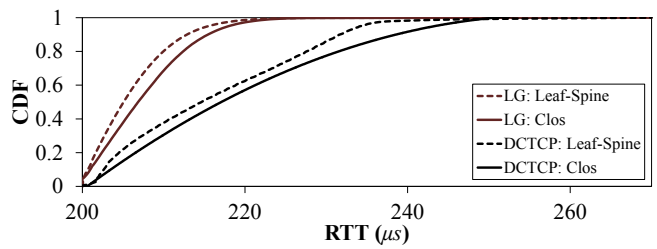


Fig. 10. Cumulative distribution function of the RTT distribution under Clos and leaf-spine topologies.

3) Round-Trip Time (RTT): One of the direct consequences of reducing the queue length is reducing the RTT. In the next scenario, we calculate the Cumulative Distribution Function (CDF) of the RTT distribution for both LGC and DCTCP with tests of 5, 20, and 40 competing flows using a base RTT of 200 μs. In all cases both LGC and DCTCP were configured optimally. Fig. 9 shows the results. LGC shows a significant reduction in RTT over DCTP in all three tests.

B. Large-Scale Evaluation

Here we investigate the comparative performance of LGC with respect to DCTCP under two large scale simulation scenarios:

(i) a classic Clos topology (8-ary fat-tree, 64 nodes in total) [6], [5], and
(ii) a similar leaf-spine topology to the one in [2], [3] (144 nodes, 9 leaf switches, 4 spine switches).

In the leaf-spine topology, all the links between hosts and TOR switches are 10Gbps, and the other links have 40Gbps capacity. In the Clos topology, all the links in the network have the same capacity, i.e. 10Gbps.

We use a *longest path uniform random* traffic pattern [9] in which a source sends data to a randomly-chosen destination with the longest path in the network. However, the set of destinations is smaller than the set of sources to ensure that congestion happens. Fig. 10 illustrates the CDF of RTT in both scenarios for LGC and DCTCP. It clearly illustrates that LGC can reduce network queue sizes in both scenarios, resulting in significantly shorter delays and RTTs for comparable average throughput (see Table II). The average number of competing flows in these scenarios was around 8, yielding a comparatively shorter RTT for both LGC and DCTCP than was achieved with the incast results shown in Fig. 9.

VII. Related Work

Datacenters are a challenging environment for efficient timely transmission of data with features including: multiple paths, small propagation delays and mixtures of long and short flows (see Zhang et al. [24] for a survey of transport control in datacenters). A key objective is to minimize flow latency, particularly flow completion times. This is a multifaceted issue [25], [26], of which congestion control is a significant component. Many proposed solutions require changes to multiple components within the datacenter, affecting the ease with which the different schemes can be deployed [26], [3].

It has been shown that traditional transport protocols such as TCP are not appropriate for datacenters (e.g. see [1]). For example, the TCP retransmission timeout mechanism performs poorly in incast scenarios, and the large queue occupancy of TCP in switches affects delay-sensitive short flows. The authors in [1] show that keeping queue occupancy at a low level is of paramount concern in datacenters.

Some approaches to reducing latency in datacenters use central arbiters. Fastpass [7] and TDMA Ethernet [27] are examples of this approach; however, they require centralized coordination which limits their scalability. Some other approaches such as DeTail [28] and pFabric [2] require changing switches to schedule flows using priorities in switches, but they cannot work on commodity hardware.

Our work focuses on easily deployable end-system based mechanisms. The closest datacenter specific work in this area to ours is Data Center TCP (DCTCP) [1], [29]. Apart from the end point transport mechanism, it requires only a low queue threshold based ECN packet marking from the bottleneck queue. A number of enhancements to DCTCP have been proposed in the literature that improve particular aspects [24]. HULL [30] adds packet pacing and virtual (phantom) queues at switches to reduce queuing and allow for earlier congestion notification at the expense of slightly lower throughput. D2TCP [31] enhances DCTCP to include the deadline aware advantages of D3 [32]. In general, these enhancements improve the mechanism but make it more difficult to deploy [3]. Our mechanism is able to improve many aspects of DCTCP while still remaining easily deployable.

Logistic growth as basis for congestion control was first proposed in [12] and later in [13]. It has been shown to be stable and reduce queue latency. We build on these ideas adapting the theory to the datacenter environment.

VIII. Conclusions and Further Work

In this paper, we have developed and evaluated a new congestion control algorithm for datacenters called Logistic Growth Control (LGC). The algorithm is inspired by logistic population growth and uses the logistic growth function to control packet transmissions over a congested path. We first analyzed LGC analytically using a fluid model to investigate its stability and accuracy. Based on this we were able to improve the algorithm and have implemented it in the INET framework of OMNeT++ [15]. Using this simulation framework, we have run extensive evaluations of LGC against another important datacenter transport protocol, DataCenter TCP (DCTCP) [1]. We simulated both a small-scale network with the important datacenter incast traffic pattern, and large-scale Clos and leaf-spine topologies. Our evaluations consistently show that communication latency in a datacenter is improved greatly by LGC, and also that LGC achieves much better fairness between flows than DCTCP.

LGC is relatively easy to model and understand, and has good convergence, RTT-independent fairness and stability properties. This makes it a promising candidate for more complex control scenarios where multiple congestion controls may be nested (e.g. to control flow aggregates between hypervisors in a Grid rather than changing the OS in the VMs) or chained together (as e.g. with middleboxes that act as TCP splitters). Recursive architectures such as RINA [33] and RNA [34] (see [35] for more details) generalize this approach and challenge traditional concepts of congestion control. We plan to further apply LGC in the datacenter environment and extend our work to more general recursive architectures.

Acknowledgment

This work has received funding from the European Union's FP7 research and innovation programme under grant agreement No. 619305 (PRISTINE). The views expressed are solely those of the author(s).

The authors would like to thank Bob Briscoe for his instructive feedback for this paper and its continuing work.

References

[1] 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.

[2] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pfabric: Minimal near-optimal datacenter transport," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 435–446, 2013.

[3] M. P. Grosvenor, M. Schwarzkopf, I. Gog, R. N. M. Watson, A. W. Moore, S. Hand, and J. Crowcroft, "Queues don't matter when you can jump them!" in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. Oakland, CA: USENIX Association, May 2015, pp. 1–14. [Online]. Available: https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/grosvenor

[4] A. Sivaraman, M. Budiu, A. Cheung, C. Kim, S. Licking, G. Varghese+, H. Balakrishnan, M. Alizadeh, and N. McKeown, "Packet transactions: High-level programming for line-rate switches," in *Proceedings of the 2016 ACM conference on SIGCOMM*. ACM, 2016.

[5] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63–74, 2008.

[6] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano *et al.*, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. ACM, 2015, pp. 183–197.

[7] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: A centralized zero-queue datacenter network," in *Proceedings of the 2014 ACM conference on SIGCOMM*. ACM, 2014, pp. 307–318.

[8] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing flows quickly with preemptive scheduling," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 127–138, Aug 2012. [Online]. Available: http://doi.acm.org/10.1145/2377677.2377710

[9] R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, "Timely: Rtt-based congestion control for the datacenter," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15. New York, NY, USA: ACM, 2015, pp. 537–550. [Online]. Available: http://doi.acm.org/10.1145/2785956.2787510

[10] P. F. Verhulst, "Notice sur la loi que la population poursuit dans son accroissement," *Correspondance mathématique et physique*, vol. 10, pp. 113–121, 1838. [Online]. Available: http://books.google.com/?id=8GsEAAAAYAAJ

[11] M. Welzl, *Scalable Performance Signalling and Congestion Avoidance*. Norwell, MA, USA: Kluwer Academic Publishers, 2002.

[12] ——, "Traceable congestion control," in *Proceedings of the 3rd International Conference on Quality of Future Internet Services and Internet Charging and QoS Technologies 2Nd International Conference on From QoS Provisioning to QoS Charging*, ser. QofIS'02/ICQT'02. Berlin, Heidelberg: Springer-Verlag, 2002, pp. 273–282. [Online]. Available: http://dl.acm.org/citation.cfm?id=1754656.1754692

[13] G. Hasegawa and M. Murata, "TCP symbiosis: Congestion control mechanisms of tcp based on lotka-volterra competition model," in *Proceedings from the 2006 Workshop on Interdisciplinary Systems Approach in Performance Evaluation and Design of Computer & Communications Sytems*, ser. Interperf '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1190326.1190338

[14] L. Jose, L. Yan, M. Alizadeh, G. Varghese, N. McKeown, and S. Katti, "High speed networks need proactive congestion control," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XIV. New York, NY, USA: ACM, 2015, pp. 14:1–14:7. [Online]. Available: http://doi.acm.org/10.1145/2834050.2834096

[15] OMNeT++, "The inet framework," https://omnetpp.org/, 2016.

[16] R. M. May, *Stability and complexity in model ecosystems*. Princeton University Press, 2001, vol. 6.

[17] 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.

[18] J. D. Murray, *Mathematical Biology I: An Introduction*. Springer, New York, NY, USA, 2002, ISBN: 978-0-387-95223-9.

[19] D. Luenberger, *Introduction to dynamic systems: theory, models, and applications*. Wiley, 1979.

[20] R. MacArthur, "Species packing and competitive equilibrium for many species," *Theoretical population biology*, vol. 1, no. 1, pp. 1–11, 1970.

[21] P. X. Gao, A. Narayan, G. Kumar, R. Agarwal, S. Ratnasamy, and S. Shenker, "phost: Distributed near-optimal datacenter transport over commodity network fabric," in *Proceedings of the CoNEXT*, 2015.

[22] R. W. Wolff, "Poisson Arrivals See Time Averages," *Operations Research*, vol. 30, no. 2, pp. 223–231, 1982.

[23] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems," DEC Research, Technical report TR-301, Sep 1984.

[24] J. Zhang, F. Ren, and C. Lin, "Survey on transport control in data center networks," *IEEE Network*, vol. 27, no. 4, pp. 22–26, Jul 2013.

[25] B. Briscoe, A. Brunstrom, A. Petlund, D. Hayes, D. Ros, I.-J. Tsang, S. Gjessing, G. Fairhurst, C. Griwodz, and M. Welzl, "Reducing internet latency: A survey of techniques and their merits," *Communications Surveys Tutorials, IEEE*, vol. PP, no. 99, pp. 1–1, 2014.

[26] R. Rojas-Cessa, Y. Kaymak, and Z. Dong, "Schemes for fast transmission of flows in data center networks," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1391–1422, 2015.

[27] B. C. Vattikonda, G. Porter, A. Vahdat, and A. C. Snoeren, "Practical tdma for datacenter ethernet," in *Proceedings of the 7th ACM European Conference on Computer Systems*, ser. EuroSys '12. New York, NY, USA: ACM, 2012, pp. 225–238. [Online]. Available: http://doi.acm.org/10.1145/2168836.2168859

[28] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing flows quickly with preemptive scheduling," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 127–138, Aug 2012. [Online]. Available: http://doi.acm.org/10.1145/2377677.2377710

[29] S. Bensley, L. Eggert, and D. Thaler, "Datacenter TCP (DCTCP): TCP Congestion Control for Datacenters," Working Draft, IETF, Internet-Draft draft-bensley-tcpm-dctcp, Feb 2014.

[30] 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. [Online]. Available: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/alizadeh

[31] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," in *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '12. New York, NY, USA: ACM, 2012, pp. 115–126. [Online]. Available: http://doi.acm.org/10.1145/2342356.2342388

[32] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM '11. New York, NY, USA: ACM, 2011, pp. 50–61. [Online]. Available: http://doi.acm.org/10.1145/2018436.2018443

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

[34] J. D. Touch and V. K. Pingali, "The RNA metaprotocol," in *Computer Communications and Networks, 2008. ICCCN'08. Proceedings of 17th International Conference on*. IEEE, 2008, pp. 1–6.

[35] P. Teymoori, M. Welzl, G. Stein, E. Grasa, R. Riggio, K. Rausch, and D. Siracuss, "Congestion control in the Recursive Internetwork Architecture (RINA)," in *IEEE International Conference on Communications (ICC), Next Generation Networking and Internet Symposium*, May 2016.