HOLNET: A Holistic Traffic Control Framework for Datacenter Networks

Zhijun Wang[a] Akshit Singhal[a] Yunxiang Wu[b] Chuwen Zhang[c] Hao Che[a] Hong Jiang[a] Bin Liu[c] and C. Lagoa[d]
[a]University of Texas at Arlington, USA [b]Purple Mountain Laboratories, China
[c]Tsinghua University, China [d]Pennsylvania State University, USA

***Abstract*—In this paper, we put forward a HOListic traffic control framework for datacenter NETworks (HOLNET). HOLNET reformulates the network utility maximization (NUM) framework into a HOLNET NUM framework that fully harnesses the potential of the existing NUM-based solutions to allow large families of traffic control protocols of various degrees of sophistication to be developed, i.e., host-based, single or multiple Class-of-Service (CoS) enabled, single or multi-path congestion control, with or without in-network load balancing. Unlike the existing solutions that are largely empirical and point by design, HOLNET is a principled, systematic framework. All the protocols in a family developed under HOLNET share a common, user-defined global optimization objective and fairness criterion. As a result, the protocols in a family can be fairly compared and carefully selected to fully explore the performance, scalability and design complexity tradeoffs. Case studies, based on both a single and a multi-path host-based solutions, demonstrate the viability and flexibility in HOLNET design space exploration. To further test the backward compatibility and performance with respect to some existing lightweight solutions, we develop HOLNET-UTA, an integrated congestion control and load balancing protocol, achieving TCP-fair resource allocation. HOLNET-UTA is found by simulation to improve the average flow completion time (FCT) by more than 20%, compared to DRILL with DCTCP.**

I. Introduction

Traffic control in today's loosely controlled public Internet has been largely limited to end-to-end TCP congestion control [1] and static, equal-cost-multipath (ECMP) load balancing [2]. In contrast, in a well controlled environment like a datacenter network, more sophisticated traffic control solutions become viable. This has led to the proliferation of a wide spectrum of congestion control solutions for datacenter networks in recent years, ranging from congestion control protocols with minimum involvement of in-network nodes, like DCTCP [3] and D^{2}TCP [4] based on Explicit Congestion Notification (ECN), all the way to those using the link load information for the control, e.g., FCP [5], or even clean-slate solutions requiring network architectural redesign, e.g., NUMFabric [6]. Meanwhile, various dynamic load balancing solutions have emerged, exploring path diversity to further improve datacenter network resource utilization. Again, the input for the control ranges from purely local, e.g., DRILL [7], all the way to the path load, e.g., CONGA [8].

However, there are two major drawbacks of the existing traffic control solutions for datacenter networks. First, the solutions are mostly point by design, focusing on one aspect of the traffic control only, e.g., single-path or multi-path congestion control, or in-network load balancing, but not both in an integrated fashion, with various performance targets and input requirements. More often than not, a point solution has to coexist with some other point solutions in practice, e.g., a newly developed congestion control protocol coexisting with a load balancing solution and/or a legacy TCP congestion control protocol. This, however, may adversely impact the effectiveness of all protocols involved. Indeed, it has been widely recognized [8]–[13] that independently developed traffic control solutions, especially at different layers, may adversely interact with one another, leading to suboptimal, unexpected performance, or even network instability. Moreover, as point solutions, they cannot easily adapt to software/hardware capability changes and hence, cannot fully explore the performance, scalability and design complexity tradeoffs.

Second, most of the existing solutions are empirical by design, without provable convergence, stability, and optimality properties. Although optimization-based traffic control protocols underpinned by the Network Utility Maximization (NUM) framework exist, they are all point solutions with limited scope, applicable to elastic, rate adaptive traffic only. Notable examples are the two earlier traffic engineering (i.e., load balancing) solutions, i.e., MATE [14] and TeXCP [15], and three more recent datacenter network solutions, i.e., NUMFabric [6] and FCP [5] for congestion control, and LocalFlow [16] for load balancing. The difficulty lies in the fact that the current NUM framework is incapable of providing performance guarantee for inelastic flows with diverse performance requirements (see Section II for details).

To address the above drawbacks of the existing solutions, in this paper, we put forward a HOListic traffic control framework for datacenter NETworks (HOLNET). The approach taken by HOLNET is principled and systematic. At the core of HOLNET is the introduction of the notion of *center-of-utility fairness* and the reformulation of the NUM framework into what we call, HOLNET NUM. HOLNET NUM fully harnesses the potential of the existing NUM solutions, allowing a whole new spectrum of traffic control protocols to be developed. The protocols developed under HOLNET can provide (soft) minimum flow rate guarantee and center-of-utility fair sharing of network resources for diverse applications. Hence, HOLNET makes a key contribution to bridge the gap between the existing solutions to NUM and their practical application to the design of traffic control

978-1-7281-6992-7/20/$31.00 ©2020 IEEE

protocols in support of diverse applications. More specifically, HOLNET possesses the following salient features,

- It allows large families of congestion control and load balancing protocols of various degrees of sophistication to be developed, with all the protocols in a family achieving a common global objective and fairness criterion. Protocols from a given family can be selected to fully explore the performance, scalability and design complexity tradeoffs and adapt to various possible software/hardware capability changes.
- All the protocols developed under HOLNET enjoy provable convergence, stability and optimality properties by design.
- It allows fully integrated host-based (multipath) congestion control and in-network load balancing protocols to be developed, making fair resource allocation in the presence of both congestion control and load balancing a reality;
- With proper design, HOLNET leads to protocols that are backward compatible with TCP and hence are TCP friendly by design.

In this paper, the flexibility for the HOLNET protocol design space exploration is demonstrated by the design of two protocols, i.e., an end-to-end multi-Class-of-Service (CoS), single-path congestion control protocol and an end-to-end single-CoS multipath congestion control. To further test the backward compatibility, scalability, extensibility, and the performance of HOLNET-based solutions, we introduce a family of protocols, called HOLNET with Utility-of-TCP-based flow rate Allocation (HOLNET-UTA) and we study one member of this family, a congestion control protocol with in-network load balancing. HOLNET-UTA is TCP-friendly by design and backward compatible with TCP Reno [1]. It is found by simulation to reduce average FCT by more than 20% compared to DRILL [7] with DCTCP [3].

II. Background, Related Work and Motivations

Traffic Control Design Space: The existing traffic control solutions for datacenter networks can be broadly classified into two categories, i.e., host-based transport congestion control and host-based multipath or in-network load balancing. A partial list of the existing solutions (including some earlier ones which are not datacenter specific) is given in Table I (note that this list is not meant to be exhaustive, but to illustrate the overall traffic control design space). For each solution listed, the performance target(s) and required input information for the control are also given.

First, we note that the listed solutions are point by design with respect to the following aspects: (a) except for some host-based solutions, such as MPTCP [32] and NDP [33], congestion control and load balancing solutions are developed independently; (b) the required input for the control can be quite different from one solution to another, ranging from purely local, all the way to path-utilization-based ones; and (c) the performance targets are also quite diverse, e.g. average flow completion time (FCT), throughput (TP), flow deadline (FD), NUM, and utility min-max. Moreover, some solutions,

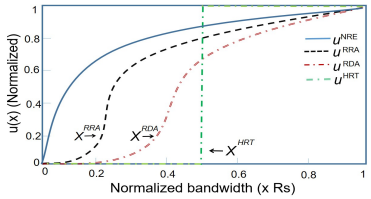


Fig. 1. Normalized user utilities for four CoSes.

e.g., pFabric [19], PIAS [24], and D3 [31], are clean-slate and require architectural redesign. Clearly, it is difficult to combine such solutions to fully explore performance, scalability, and design complexity tradeoffs and to allow adaptation to software/hardware capability changes for datacenter networks at large. Moreover, as mentioned earlier, such solutions, when coexist with one another, may adversely interact with one another, leading to suboptimal, unexpected performance, or even network instability [8]–[13].

Second, most solutions listed in Table I are empirical by design. Although some solutions are NUM-based, e.g., [5], [6], [16], they are again point solutions with limited scope, applicable to elastic traffic only and deal with either congestion control or load balancing, but not both. Since HOLNET is also rooted in NUM, in what follows, we first give a background overview of NUM. Then we identify its three major drawbacks, which motivates our work and also helps to explain why the existing NUM-based solutions are point by design and has limited scope.

NUM and NUM Solutions: NUM can be formally stated as follows,

$$max\{V(\mathbf{x}) = \sum_{i=1}^{n} w_i u_i(x_i)\}, \tag{1}$$

subject to link bandwidth constraints (i.e., the total flow rate for flows sharing a link must not exceed the link bandwidth), where n is the number of active flows, $u_i(x_i)$ and w_i are the normalized user utility function of the allocated flow rate x_i and the weight for flow i, respectively, and $V(\mathbf{x})$ is the sum of the weighted user utilities for all active flows. Here $u_i(x_i)$'s are meant to be used to characterize Quality of Experience (QoE) in terms of bandwidth demand for users of diverse applications. Note that QoE or how a user feels about a service is, in general, a complex function of many factors, e.g., TP, FCT, packet delay and delay jitter, and even price paid for using the service. Meeting bandwidth demand of a user may be viewed as the first order approximation of meeting QoE.

For example, in his seminal work [46] back in 1995, Shenker discussed protocol design challenges for the future Internet in the context of NUM and defined four broad CoSes to meet diverse application requirements, in terms of four different types of user utilities, as illustrated in Figure 1, i.e., u^{NRE}, u^{RDA}, u^{RRA}, and u^{HRT}, for Non-Realtime Elastic (NRE), Realtime Delay Adaptive (RDA), Realtime Rate Adaptive (RRA), and Hard RealTime (HRT) CoSes, respectively. R_s in Figure 1 is defined as the saturated flow rate that maximizes the user utility, e.g., the highest encoding rate

TABLE I
EXISTING DESIGN SPACE

Solution	Transport	Host-based LB	In-network LB	Input Information	Performance Target (QoS)
DCTCP [3]	$\surd$	$\times$	$\times$	ECN	FCT&TP
FCP [5]	$\surd$	$\times$	$\times$	budget/link price	NUM
DX [17]	$\surd$	$\times$	$\times$	latency	queuing Delay
HULL [18]/DCTCP	$\surd$	$\times$	$\times$	ECN/QL	mean & tail PL
D^2TCP [4]	$\surd$	$\times$	$\times$	ECN/FDL	FD ($\surd$)
pFabric [19] (clean-slate)	$\surd$	$\times$	$\times$	FD	FCT ($\surd$)
ExpressPass [20] (clean-slate)	$\surd$	$\times$	$\times$	credit-based signaling	FCT ($\surd$)
pHost [21]	$\surd$	$\times$	spraying	token	FCT, TP
NUMFabric [6]	$\surd$	$\times$	$\times$	Weights	NUM
RC3 [22]	$\surd$	$\times$	$\times$	double loop	FCT
XCP [23]	$\surd$	$\times$	$\times$	ECN	TP
PIAS [24] (clean-slate)	DCTCP/TCP	$\times$	$\times$	AQM	FCT
PASE [25] (clean-slate)	DCTCP/D^2TCP	$\times$	$\times$	FD/AQM	FCT ($\surd$)
L^2DCT [26]	$\surd$	$\times$	$\times$	ECN	FCT
DCQCN [27]/PFC [28], [29]	$\surd$	$\times$	$\times$	ECN, Pause	PL
TIMELY [30]/PFC (opt)	$\surd$	$\times$	$\times$	RTT, Pause (opt)	packet TL/TP
Karuna [9]/DCTCP	$\surd$	$\times$	$\times$	ECN/RTT	FD/FCT ($\surd$)
D3 [31] (clean-slate)	$\surd$	$\times$	ECMP	min-rate	FD ($\surd$)
MPTCP [32]	$\surd$	$\surd$	$\times$	source inferred	TP
NDP [33]	$\surd$	$\surd$	$\times$	Packet trimming	FCT/TP
PDQ [34]/RCP [35] (clean-slate)	$\surd$	$\surd$	ECMP	FD,Pause	FCT,FD ($\surd$)
Presto [36]	$\times$	$\surd$	$\times$	path weights	TP
DRB [37]	TCP	$\surd$	$\times$	ECN	FCT/TL
FlowBender [38]	TCP	$\surd$(rerouting)	ECMP	ECN	FCT/TL
HERMES [39]	$\times$	$\surd$	$\times$	ECN/RTT/Timeout	FCT
Clove [40]/ECMP	$\times$	$\surd$	$\surd$	ECN/path-utilization	FCT
HULA [41]	$\times$	$\times$	$\surd$	per-hop utilization	FCT/TP
LocalFlow [16]	TCP	$\times$	$\surd$	local	NUM
RBS [42]	$\times$	$\times$	$\surd$ path-ECMP	Flow paths	FCT/TP
MATE [14]	$\times$	$\times$	$\surd$ edge-based	path utilization/delay	Min-Cost
TeXCP [15]	$\times$	$\times$	$\surd$ edge-based	path utilization	utility Min-Max
Flare [43]	$\times$	$\times$	$\surd$	local	TP
LetFlow [44]	$\times$	$\times$	$\surd$	none	FCT
CONGA [8]	$\times$	$\times$	$\surd$ edge-based	path congestion	FCT
Expeditus [45]	$\times$	$\times$	$\surd$	local/path combined	FCT/TL
DRILL [7]	$\times$	$\times$	$\surd$	local	FCT
ECMP [2]	$\times$	$\times$	$\surd$	none	TP

FCT: Flow Completion Time; TP: Throughput; TL: Tail Latency; PL: Packet Latency; FD: Flow Deadline; RTT: Round Trip Time
AQM: Active Queue Management; NUM: Network Utility Maximization; ECN: Explicit Congestion Notification

for an application of RRA CoS or maximum link bandwidth for an application of NRE CoS. In general, any flow i with rate x_i, is associated with a given user utility function, $U_i(x_i)$, and the corresponding normalized user utility, $u_i(x_i)$, where,

$$u_i(x_i) = U_i(x_i)/U_i(R_{s,i}), \tag{2}$$

and $R_{s,i}$ is the saturated rate for flow i. Clearly, $u_i(x_i)$ is a more convenient measure of the degree of user satisfaction than $U_i(x_i)$[1].

To design the Internet protocols on the basis of the above NUM, the first thing one must do is to find distributed solutions to NUM. The first breakthrough came along in 1998 when Kelly, et. al. [47] showed that the utility function of logarithm form, i.e., $u_i(x_i) = \log(x_i) \in u^{NRE}$, for $\forall i$, results in a solution in the form of a distributed flow rate adaptation algorithm that resembles the distributed TCP congestion control, achieving the so called proportional fairness. This encouraging result has stimulated the research interests in finding more general distributed solutions to NUM. In particular, Low [48], [49] proposes a distributed primal-dual congestion control solution to NUM with arbitrary concave user utilities in u^{NRE} CoS. This solution requires that the flow rate at the source of a flow and a variable, called price (a function of the link load), at each and every link along the flow path are iteratively updated and exchanged. This solution provided the theoretical underpinning for all the aforementioned NUM-based datacenter network traffic control protocols of the NRE CoS, including NUMFabric [6] and FCP [5] for flow rate control, and LocalFlow [16] for load balancing.

Another line of research based on Sliding Mode in control theory [50] has culminated in the finding of distributed solutions spanning a large design space, allowing for flow multipath and in-network flow load balancing with minimum information exchange, and admitting flow rate constraints and both concave (e.g., u^{NRE}) [51], [52] and nonconcave [53] (e.g., u^{RRA}, u^{RDA} and u^{HRT}) user utilities.

Although promising, so far, we have seen no application of

[1]NUM is traditionally defined in terms of unnormalized user utility functions $U_i(x_i)$'s, i.e., $max\{V(\mathbf{x}) = \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{U_i(x_i)}\}$, subject to link bandwidth constraints, which is a special case of the current definition, when $w_i = U_i(R_{s,i})$, for $i = 1, ..., n$.

the above NUM solutions as the theoretical underpinning for the design of traffic control protocols for inelastic datacenter applications. The root cause lies in the fact that NUM in the current formulation is not suitable to support multiple CoSes for three main reasons.

Three Drawbacks of NUM: First, NUM cannot provide user utility guarantee. NUM is purely a "social-welfare" based framework, striving to maximize the sum of user utilities, with no regards to application-specific requirements, nor meaningful fairness among users of distinct CoSes. Under this framework, the achievable individual user utilities, regardless of which CoSes they belong to, are strong functions of traffic load and traffic mix, which, however, may change over time. For example, for a single-link network with bandwidth, C, the sum of user utilities, $nu(C/n)$, for n NRE flows with the same concave utility function, $u(x)$, where $u(x) \in u^{NRE}$, is an ever increasing function of n [46]. This means that if a RRA, RDA, or HRT flow shares the same link bandwidth with these NRE flows and as n increases, to maximize $V(\mathbf{x})$, its share of link bandwidth will eventually diminish, resulting in unbearably low user utility. This means that a flow of a realtime CoS may experience unpredictable and poor performance, as the traffic load fluctuates and/or flow mix pattern changes.

Second, in general, a solution to NUM with more than one distinct user utility or even a single user utility but more than one distinct saturated rate may not provide meaningful fair flow rate allocation. For instance, a widely studied family of concave α-fairness user utilities [54][2], is given as,

$$U_\alpha(\alpha, x) = x^{1-\alpha}/(1-\alpha), \quad for \quad \alpha \in (0, \infty). \tag{3}$$

Now, consider the user utility function, $U_\alpha(0.5, x) = 2\sqrt{x}$ shared by two NRE applications with two distinct saturated rates, i.e., $R_{s,1} = 400$ Mbps and $R_{s,2} = 100$ Mbps, respectively. If two flows, one from each application, share a 100 Mbps link, the optimal flow rate allocation for NUM with $w_i u_i(x_i) = U_\alpha(0.5, x_i)$ in Eq. (1) is to give 50 Mbps to each flow. It can be easily calculated from Eq. (2) that the degrees of user satisfaction for the two flows at this flow rate allocation are quite different, i.e., 0.354 and 0.707, respectively, which also change drastically as the available link bandwidth changes. Also it can be easily shown that using $u_i(x_i)$ with w_i=1 for i=1, 2 in Eq. (1) results in even worse rate allocation with user utilities, 0.224 and 0.894 for the first and second flows, respectively. Clearly, it would be even harder to predict how NUM would allocate the flow rates if flows with more than one distinct user utility share a link bandwidth.

Third, it is, in general, difficult, if not impossible, to accurately quantify, $u(x)$, as it is just one of many important factors that determine QoE.

Implications: The above stated mismatches between what NUM is intended to accomplish and what it is actually capable of doing explain why, to date, NUM has been used only as a tool to enforce a given fairness criterion among elastic flows sharing the same user utility and saturation rate. More specifically, all the existing NUM-based solutions [5], [6], [16] for datacenter networks are exclusively focused on flows sharing a single utility function of NRE CoS only, for which the fairness criterion can be established, e.g., a given user utility in the α-fairness family, rather than flows with diverse user utilities or CoSes.

In conclusion, the existing NUM-based solutions fall short of what NUM sets out to accomplish – supporting applications of diverse CoSes. In fact, the ability to accommodate *inelasticity* of applications should be a basic requirement of the future protocol design for datacenter applications, simply because almost all the predominant datacenter workloads today call for some sort of minimum user utility guarantee, in the form of a minimum flow rate or flow deadline, e.g., for user-facing applications, such as adaptive video streaming. This then begs the following fundamental question that motivates the current work: Can NUM be reformulated in such a way that its full potential, in terms of its existing, rich distributed solutions, e.g., [51]–[53], can be harnessed to enable large families of traffic control protocols for applications?

III. HOLNET

In this section, we attempt to provide a definitive answer to the above question. We first introduce HOLNET NUM that overcomes the three drawbacks of NUM. Then we characterize the design space of HOLNET solutions.

A. HOLNET NUM

As discussed above, NUM fails to provide: (a) user utility guarantee; (b) meaningful fairness in the presence of flows with different user utilities and/or saturated rates; and (c) faithful characterization of user utility. HOLNET NUM directly addresses (a) and (b) by providing minimum user utility guarantee and center-of-utility fairness, respectively, which together capture the essential characteristics of user utility, hence, to a great extent, achieving (c).

Minimum User Utility Guarantee: We note that a user utility is generally composed of two parts, i.e., a non-concave lower part and a concave higher part, e.g., the parts below and above the inflection points X^{RRA} (X^{RDA}) for RRA (RDA) as shown in Fig. 1. For instance, X^{RRA} may be the lowest level encoding rate for an adaptive video stream. NRE and HRT are two extreme cases, one without a lower part and the other without a higher part. While offering more bandwidth or user utility beyond the lower part may make a user happier, which however, is not critically important, satisfying the lower part of the user utility, called the *minimum user utility* in this paper, is likely to be essential to the user satisfaction of the service and hence, cannot be compromised.

[2]In particular, it is found that $u_\alpha(\infty, x)$ and $u_\alpha(\alpha \to 1, x) \to log(x)$ provide the flow rate max-min and proportional fairness, respectively [54].

The minimum user utility for a datacenter service is often spelled out by datacenter service providers in terms of a given service level objective (SLO), e.g., a tail-latency SLO for a user-facing interactive service. Since an SLO, in turn, can be translated into given flow rates or flow deadlines [31], many existing traffic control solutions aim at providing minimum flow rate or the flow deadline guarantee [19], [25], [34] without having to explicitly construct the user utility function. Likewise, HOLNET aims at providing minimum flow rate guarantee, assuming that the minimum flow rate required to achieve the minimum user utility is already known.

To this end, HOLNET NUM simply extends NUM to allow the minimum flow rate, e.g., θ, to be enforced as a flow rate constraint, i.e., $x \geq \theta$. Clearly, any feasible solution to HOLNET NUM provides the minimum flow rate, or equivalently, the minimum user utility guarantee, regardless of the traffic load and flow mix pattern.

Center-of-Utility Fairness: HOLNET NUM aims to enable a meaningful user-utility-aware fairness criterion for flow rate allocation among flows with different user utilities, called the *center-of-utility fairness*.

First, we define *center-of-utility*, x_i^c, for flow i, that captures the average user utility of flow i, as follows,

$$x_i^c = \frac{\int_0^{R_{s,i}} x_i \frac{dU_i(x_i)}{dx_i} dx_i}{U_i(R_{s,i})} = \int_0^{R_{s,i}} x_i \frac{du_i(x_i)}{dx_i} dx_i. \quad (4)$$

The concept for the center-of-utility is borrowed from mechanics in physics. More specifically, if we view the user utility density, $dU_i(x_i)/dx_i$, as the mass density at x_i for an uneven bar of length, $R_{s,i}$, and total mass, $U_i(R_{s,i})$, then the center of mass [55] of the bar is x_i^c, hence the name. x_i^c is the center, to which the user utility, $U_i(x_i)$, is concentrated. Hence, x_i^c is the exact measure of the average bandwidth demanded by the users of flows with user utility, $U_i(x_i)$.

We then propose to enable the so called *center-of-utility fair flow rate allocation*, i.e.,

$$\frac{x_i}{x_j} = \frac{x_i^c}{x_j^c}, \quad (5)$$

for any pair of flows i and j sharing a bottleneck link. As a result, HOLNET aims to achieve *center-of-utility fair flow rate allocation, provided that the minimum flow rates to sustain the minimum user utilities are satisfied.*

The rationale behind the use of the above fairness criterion can be best illustrated by example. Consider an adaptive audio flow and an adaptive video flow sharing a bottleneck link. It is clear that both the average bandwidth demands (i.e., the center-of-utilities) and the minimum encoding rates (i.e., the flow rates to sustain the minimum user utilities) for the audio and video applications are quite different, say, 50 Kbps versus 5 Mbps, and 10 Kbps versus 1 Mbps, respectively. With the center-of-utility fairness, the video flow will then be allocated 100 times of the additional link bandwidth than the audio flow, provided that the minimum encoding rates for the two flows, i.e., 10 Kbps and 1 Mbps, are satisfied. Clearly, this flow rate allocation solution captures the QoE in terms of user bandwidth demand well, i.e., the guaranteed minimum user bandwidth and additional bandwidth allocation in proportion to the relative bandwidth demands.

However, as we mentioned earlier, it is generally difficult to exactly quantify $U_i(x_i)$, which, in turn, makes it difficult to quantify x_i^c, as it is derived from $U_i(x_i)$. Fortunately, however, with the minimum user utility guaranteed, it is no longer essential to allocate the additional resource among flows perfectly in proportion to the relative bandwidth demands, which by itself, is difficult to define. So, in practice, x_i^c may be roughly estimated in terms of the order of the bandwidth demand of the underlying application without having to know the exact $U_i(x_i)$. For example, assume that an adaptive audio application has several encoding levels in the range of 10Kbps-100Kbps. It suffices to set x_i^c at around 50Kbps to capture the order of the overall bandwidth demand of this application.

Now what is left to be done is to recast NUM in Eq. (1) in such a way that it will indeed lead to the fair flow rate allocation defined in Eq. (5). To this end, we define HOLNET NUM as follows,

$$max\ \{V(\mathbf{x}) = \sum_{i=1}^{n} w_i U_c(x_i)\}, \quad (6)$$

subject to both link bandwidth constraints and flow rate constraints for flows with minimum user utility requirements, where $U_c(x)$ is a concave user utility, called the *base utility*, shared by all the flows[3]. In other words, our design goal of HOLNET NUM is no longer to maximize the sum of the user utilities, but to achieve center-of-utility fair flow rate allocation via careful design of w_i and $U_c(x)$ as follows.

Consider flows sharing a bottleneck link. With the Lagrangian multiplier technique [56], it can be easily shown that to maximize the sum of $w_i U_c(x_i)$ for all the flows sharing this link, we must have,

$$\frac{w_i}{w_j} = \frac{dU_c(x_j)/dx_j}{dU_c(x_i)/dx_i}, \qquad \forall i, j, \quad (7)$$

for any pair of flows i and j bottlenecked at this link. Now let $U_c(x) = U_\alpha(\alpha, x)$ defined in Eq. (3), we have,

$$w_i/w_j = (x_i/x_j)^\alpha, \qquad \forall i, j. \quad (8)$$

Here, the weight, w_i, for any user utility, $U_i(x_i)$, can be calculated against the weight, $w_0 = 1$, for the base utility, $U_0(x_0) = U_c(x_0)$, with a base center-of-utility, x_0^c, calculated from Eq. (4). Here, w_0 is set to 1, without loss of generality. By substituting Eq. (5) into Eq. (8), we have,

$$w_i = (x_i^c/x_0^c)^\alpha, \qquad \forall i. \quad (9)$$

Then, substituting the above w_i and $U_c(x_i) = U_\alpha(\alpha, x_i)$ into Eq. (6), we arrive at a new NUM, called HOLNET

[3]While on the surface, using a single concave utility with weights in NUM is not new [6], HOLNET uses it in a completely different way than the traditional one, i.e., realizing the center-of-utility fair flow rate allocation, rather than achieving some network-centric performance target, e.g., minimizing average FCT [6]

NUM, achieving center-of-utility fair flow rate allocation with minimum user utility guarantee. It can be easily shown that (the proof is not given due to the page limit) HOLNET NUM also works for the case where each flow x_i can be split into L subflows, $x_{i,l}$, for $l = 1, ..., L$, with each subflow taking different paths to the same destination.

A nice property of HOLNET NUM is that it only deals with a single concave user utility. As such, all the existing solutions to NUM with concave user utilities can be applied to HOLNET NUM, hence harnessing the potential of NUM, to be described in the next section.

Finally, we note that in HOLNET, the user utility information for a given flow i, including the minimum user utility and center-of-utility fairness, is incorporated in HOLNET NUM only through a pair of parameters, θ_i and w_i. This means that HOLNET can also be used to enable user-utility-agnostic traffic control solutions, as long as this pair is properly defined. For example, this pair may be tied to a pricing model to achieve price-proportional flow rate allocation and fairness.

B. HOLNET Design Space

In this section, we first formally state the problem and its solutions. Then we outline the design space of the solutions.

Consider a network with a set of nodes (i.e., switches) B. Let L_b be the set of all links l connected to node b; $L_{b_{s_i}}$ be the set of all outgoing links l of a source node S_i; and $x_{i,l}$ be the subflow rate of flow i through link l. Namely, flow i is split into n_i subflows, which are mapped to different first-hop nodes and we have,

$$x_i = \sum_{l \in L_{b_{si}}}^{n_i} x_{i,l}, \quad \forall i. \tag{10}$$

This setup allows for host-based multipath or multi-next-hop congestion control and load balancing. Now HOLNET NUM is to achieve the global objective given in Eq. (6), subject to the network link bandwidth constraints and the following flow rate constraints,

$$\theta_i \leq x_i \leq \Theta_i, \quad \forall i. \tag{11}$$

Different values of lower bound θ_i and upper bound Θ_i can be used to provide minimum user utility guarantee for different CoSes. For example, for NRE, $\theta_i = 0$ and $\Theta_i = \infty$; for HRT, $\theta_i = \Theta_i > 0$; for RRA and RDA, $\theta_i > 0$, and $\Theta_i = \infty$.

HOLNET is underpinned by the family of optimal, distributed traffic control solutions to NUM with concave user utilities given by Su, et. al. [52], rather than the family of primal-dual solutions [48], [49], which underpin all the aforementioned datacenter NUM-based protocols. The former one is adopted because it is most sophisticated one that (a) accommodates flow rate constraints; (b) covers a large design space, including integrated (multi-path) congestion control and load balancing; and (c) allows for the exploration of performance, scalability and design complexity tradeoffs and adaption to hardware/software changes. In other words, HOLNET NUM fully harnesses the potential of the most sophisticated NUM solutions for the design of a whole new set of families of traffic control protocols.

TABLE II
VALUES OF $r_{i,b}^{in}$ AND $r_{i,(b),l}^{out}$ (IT REPRESENTS BOTH $r_{i,l}^{out}$ AND $r_{i,b,l}^{out}$)

Congested			
next hop	**local**	$r_{i,b}^{in}$	$r_{i,(b),l}^{out}$
yes	yes	r_{max}	r_{min}
yes	no	r_{max}	r_{min}
no	yes	r_{max}	1
no	no	1	1

In its most general form, these families of solutions are applicable to a multidomain environment. However, due to the page limitation, in this paper, we limit the description of these families of solutions in the context of a single domain. The families of solutions are generally composed of a set of user-utility-based, multi-next-hop-enabled congestion controllers running at hosts (i.e., servers) and a set of in-network load balancers running at network nodes.

Multi-next-hop-enabled congestion controller: The families of optimal congestion controllers for flow i at source host S_i can be generally written as [52],

$$\dot{x}_{i,l} = z_i(t, x_i, cg_l, r_{i,l}^{out})[f_i(x_i) - (1 - \overline{cg}_l r_i r_{i,l}^{out})], \tag{12}$$

where

$$f_i(x_i) = 1 - e^{-\partial U_i(x_i)/\partial x_i}, \tag{13}$$

where $U_i(x_i)$ can be any concave function and for HOLNET NUM, $U_i(x_i) = w_i U_c(x_i)$, where each $U_c(x_i)$ defines a family of congestion controllers; cg_l is a local congestion indicator, taking value 1 if the outgoing link l is congested and 0 otherwise; $\overline{cg}_l$ is the logic negation of cg_l; $z_i(.)$ is a user-defined piecewise continuous positive scalar function; and r_i is determined by the CoS as follows,

$$r_i = \begin{cases} r_{max}^{CoS} & if \quad x_i < \theta_i \\ 1 & if \quad \theta_i \leq x_i \leq \Theta_i \\ r_{min}^{CoS} & if \quad x_i \geq \Theta_i, \end{cases} \tag{14}$$

an extension to the results in [52]. Here $r_{max}^{CoS} > 1$ and $r_{min}^{CoS} < 1$ are tunable constants; and $r_{i,l}^{out}$ is determined by the congestion information fed back from the first hop node (i.e., a top-of-rack (ToR) switch in the context of a datacenter network), a per-flow-based single-hop signaling. It is calculated based on Table II, where $r_{max} > 1$ and $r_{min} < 1$ are design parameters.

Now, we highlight some salient features of the above families of congestion controllers. First, a congestion controller for any given flow i is dependent on the user utility function, $U_i(x_i)$, or in the case of HOLNET, θ_i and $w_i U_c(x_i)$, for flow i only (see Eq. (13)). As we shall see shortly, the in-network load balancers are user-utility agnostic. This means that flows belonging to a new CoS with a new user utility function can be added by simply activating a new congestion controller with its minimum flow rate and weight value properly set at runtime. Second, the only nonlocal information is the congestion feedback, $r_{i,l}^{out}$, from the first-hop nodes, making the congestion control highly

scalable. Third, with multi-next-hop flow load balancing capability, these families of congestion controllers enable host-based load balancing features, as well as both host and in-network-based load balancing features by working seamlessly with in-network load balancers, to be introduced shortly. Fourth, each base utility, $U_c(x)$, defines a family of congestion controllers. Each family is composed of congestion controllers of various degrees of sophistication. Different congestion controllers may take as input different information for the control, by engineering $z_i(.)$ as a function of various additional information, ranging from the local information, all the way to the entire path information for performance enhancement, without changing the global design objective and fairness criterion. Last but not least, the above families of congestion controllers degenerate to families of transport congestion controllers [51] by simply setting $r_{i,l}^{out}$ to 1, if the in-network nodes are not involved. cg_l now represents the congestion indicator for path l from the source to the destination host, which may be (a) inferred by the source host, resulting in highly scalable end-to-end solutions, similar to the end-to-end TCP congestion control; (b) acquired explicitly using ECN [57], leading to lightweight ECN-based solutions, similar to DCTCP and D^{2}TCP; or (c) explicit link load or even path information.

In-network load balancer: At each node b, the outgoing data rate $x_{i,b,l}^{out}$ from node b through link l ($l = 1, 2, ..., n_{i,b}$) is given by,

$$\dot{x}_{i,b,l}^{out} = z_{i,b}(t, x_i, cg_l, r_{i,b}^{in}, r_{i,b,l}^{out})[-1 + \overline{cg}_l r_{i,b}^{in} r_{i,b,l}^{out}], \quad (15)$$

where $n_{i,b}$ is the number of next-hop nodes for flow i, which may be determined locally by node b; $z_{i,b}(.)$ is again a piecewise continuous positive scalar function. $r_{i,(b),l}^{out}$ and $r_{i,b}^{in}$ are determined by the local and next-hop congestion information, respectively, as given in Table II. Again, $r_{i,(b),l}^{out}$ enables per-flow-based congestion feedback. This family of load balancers can be directly applied to HOLNET NUM without further modification.

The above load balancers possess the following salient features. First, they are independent of the user utilities. Second, they are flow rate constraint unaware. These two features make the load balancing user-utility agnostic and hence, scalable.

To further improve the scalability, [52] introduces a percentage based destination-node-based load balancer that further reduces the total number of load balancers per node to be the number of edge nodes. Let $p_{i,b,l}$ be the percentage of all incoming traffic ($\sum_j x_{i,b,j}^{in}$) at node b routed to outgoing link l towards destination i. The control law for the outgoing traffic ($x_{i,b,l}^{out}$) through l is given by,

$$x_{i,b,l}^{out} = p_{i,b,l} \sum_{j \in L_b} x_{i,b,j}^{in} \quad (16)$$

and

$$\dot{p}_{i,b,l} = z_{i,b}(t, x_i, cg_l, r_{i,b}^{in}, r_{i,b,l}^{out}) \\ (\dot{x}_{i,b,l}^{out} \sum_{j \in L_b; j \neq l} p_{i,b,j} - p_{i,b,l} \sum_{j \in L_b; j \neq l} \dot{x}_{i,b,j}^{out}). \quad (17)$$

with

$$\dot{x}_{i,b,l}^{out} = -1 + \overline{cg}_l r_{i,b}^{in} r_{i,b,l}^{out} \quad (18)$$

HOLNET design space: The above HOLENET-NUM-based families of solutions span a large design space, as shown in Table III, from single-path rate adaptive congestion control to the most comprehensive ones involving both multi-CoS, multipath congestion control and in-network load balancing.

First, CC and MCC on the top of the list are the two simplest solutions among all, which are rate adaptive congestion control solutions without and with multipath, respectively. CC-CoS and MCC-CoS further enable services with minimum user-utility guarantee. All four types of solutions are host-based and can be end-to-end (as scalable as end-to-end TCP), ECN-based (as scalable as DCTCP), link-load-based, or even path-load-based, leading to a wide range of solutions of various degrees of sophistication, allowing for full exploration of the performance, scalability and design complexity tradeoffs.

The next four solutions take advantage of path diversity to further improve the flow performance via in-network load balancing. In addition to the congestion controllers running at hosts, these solutions require that a network node implement a set of load balancers with per-hop signaling for flow aggregates at some given granularities, e.g., ToR-to-ToR or destination-ToR based. These congestion controllers and load balancers use the local and next-hop congestion information as input for the control[4].

Practical Applicability of HOLNET: As stated at the beginning of the introduction section, sophisticated traffic control solutions, such as those with QoS features derivable from HOLNET (i.e., the types of protocols involving CoS features listed in Table III), clean-slate and flow-deadline-aware protocols listed in Table I become viable only in a well-controlled environment like a datacenter. This is simply because in a loosely controlled environment, such as the public Internet, selfish users can cheat the system by using protocols with better QoS features or more aggressive user utilities than needed. In contrast, in a well-controlled environment, what user utility, or equivalently, what HOLNET protocol may be used by a host (e.g., a server in a datacenter) in that environment is under the full control of the network operator/service provider in that environment. Moreover, to support users of the applications/services that require strict minimum user utility guarantee, some additional mechanisms, such as a call admission or a network resource monitory mechanism, must be in place, which is feasible only in a well-controlled environment.

[4]The proposed four solutions are the most lightweight and hence the most scalable optimal solutions possible. Indeed, it is shown by example [8] that load balancing with pure local information can result in worse performance than a traffic-oblivious solution, like ECMP. This means that to achieve optimal control, the input for the control must be at least per-hop based, which is the case for the current solution.

TABLE III
HOLNET DESIGN SPACE

Solution	Host-based	Host-based LB	In-network LB	Multi-CoS	Comments
CC	yes	no	no	no	Elastic Congestion Control (CC)
MCC	yes	yes	no	no	Multipath CC (MCC)
CC-CoS	yes	no	no	yes	CC and multi-CoS
MCC-CoS	yes	yes	no	yes	multipath CC-CoS
CC-LB	yes	no	yes	no	CC with in-network load balancing (LB)
MCC-LB	yes	yes	yes	no	Multi-next-hop CC-LB
CC-CoS-LB	yes	no	yes	yes	CC-CoS with LB
MCC-CoS-LB	yes	yes	yes	yes	MCC-CoS with LB

IV. CASE STUDIES

This section aims to demonstrate the viability and flexibility for HOLNET design space exploration. Specifically, we develop a family of CC-CoS congestion controllers and an MCC congestion controller and test them by ns-3 simulation.

A. A Family of CC/CC-CoS Controllers

To limit the exposure, we skip the subscription, i, for flow i in the rest of paper. The families of optimal CC/CC-CoS controllers using $U_c(x) = U_\alpha(\alpha, x)$ as the base utility with $z(.) = \gamma x^\alpha$ can be easily derived from Eq. (12) with $r_l^{out} = 1$, as follows,

$$\dot{x} = \begin{cases} \gamma x^\alpha (r - e^{-wx^{-\alpha}}) & if \quad cg = 0 \\ -\gamma x^\alpha e^{-wx^{-\alpha}} & if \quad cg = 1. \end{cases} \tag{19}$$

where r is the same as r_i defined in Eq. (14), and we set r_{max}^{CoS} at 3 for all the case studies in this paper. These controllers are then turned into window-based congestion control protocols, not shown here for the lack of space (the same for the rest of the case studies). In this case study, we only consider end-to-end control, meaning that the controllers use only source inferable information for the control, i.e., the three replicated acknowledgements (ACKs) and timeout, similar to TCP Reno. The only difference is that for the current controllers, the rate adjustment is the same for both timeout and three duplicated ACKs. These controllers are as scalable as TCP Reno.

We first apply a CC controller in the above families of controllers with base utility, $U_c(x) = U_\alpha(0.5, x)$ or $\alpha = 0.5$, to two NRE flows (i.e., $\theta = 0$ and $r = 1$) with the same user utility, $U_1(x) = U_2(x) = U_\alpha(0.5, x)$, which share a 100 Mbps link, as shown in Figure 2 (a). The saturated rates, R_s's, for the two are set at 400 Mbps and 100 Mbps, respectively. The center-of-utility ratio for the two flows can then be calculated (i.e., Eq. (4)), which turns out to be 4, resulting in the weight ratio of 2 (i.e., Eq. (8)). In other words, the optimal flow rate allocation should be 4:1, or 80 Mbps and 20 Mbps for flow 1 and flow 2, respectively, in order to achieve the center-of-utility fairness.

Figures 3 (a) and (b) depict the achieved normalized V(**x**) in Eq. (6) and flow rate allocations against their respectively optimal ones. As one can see, both converge to near optimal values quickly, resulting in the flow rate ratio of 3.61, close to, but lower than the optimal ratio of 4. This is because flow x_1, which has higher flow rate than flow x_2 generally senses more congestions and hence achieves lower than expected flow

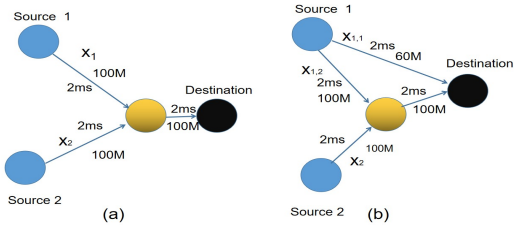
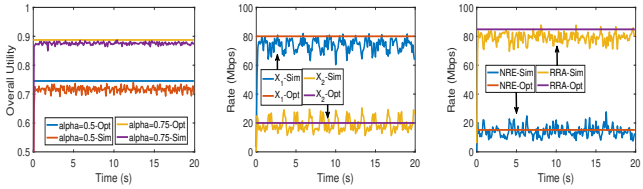


Fig. 2. Network topologies for (a) CC-CoS controllers; (b) MCC controller.



(a) (b) (c)

Fig. 3. (a) Overall utility; Rate allocation for (b) $\alpha = 0.5$ and (c) $\alpha = 0.75$.

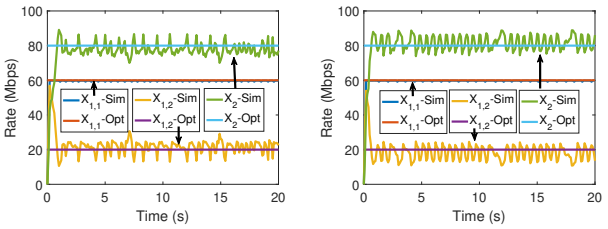
rate. Note that the aggregated flow rate is less than the link bandwidth of 100 Mbps, due to discrete time window-based control that cannot fully utilize the link bandwidth.

Now we apply a different family of the CC-CoS controllers, i.e., the α-utility with $\alpha = 0.75$ as the base utility. Consider an NRE flow, x_1, with $U_{NRE}(x_1) = \log(1 + x_1)$, and a RRA flow, x_2, with $U_{RRA}(x_2) = (x_2 - x_2^{RRA})^{1/3} + (x_2^{RRA})^{1/3}$ and $x_2^{RRA} = \theta = 20$ Mbps sharing a 100 Mbps link, as shown in Figure 2 (a), and flows have the same saturated rate $R_{s,1} = R_{s,2} = 100$ Mbps. The center-of-utility ratio for the two flows is then 5.59 (resulting in the weight ratio of 3.63) and hence the optimal flow rates for NRE and RRA flows should be 15.2 Mbps and 84.8 Mbps, respectively.

As one can see from Figures 3 (a) and (c), the simulated utility and flow rate allocation well match with the optimal ones. The flow ratio is 5.24, close to the optimal value, 5.59. Since the overall rate is higher than the minimum required rate 20 Mbps for the RRA flow, the minimum rate does not play a role for the rate allocation.

B. MCC: Multipath Congestion Control

Now we develop a scalable elastic, NRE (i.e., r=1), end-to-end multipath congestion controller. More specifically, we assume that all the flows have the same center-of-utility and saturated rate. This means that all the flows have the same weight, which can then be set to 1, without loss of generality. We let, $U_c(x) = \epsilon log(x) = \epsilon log(\sum_{l=1}^{L} x_l)$, where x_l is the



(a) (b)

Fig. 4. Rate Allocation: (a) MCC Controller; (b) MPTCP

flow rate of the subflow l, $x = \sum_{l=1}^{L} x_l$ is the flow rate and ϵ is a constant. We set, $z(.) = \gamma x_l$, where γ is a constant. The values for γ and ϵ are chosen such that the control law degenerates to the TCP AIMD (i.e, additive increase and multiplicative decrease) control in the case of a single-path flow. Then the optimal controller for subflow l can be readily derived from Eq. (12) with $r = 1$ and $r_l^{out} = 1$, as follows,

$$\dot{x_l} = \begin{cases} \gamma x_l(1 - e^{-\epsilon/x}) & if \quad cg = 0 \\ -\gamma x_l e^{-\epsilon/x} & if \quad cg = 1. \end{cases} \tag{20}$$

The above controller can be understood as follows. When $x >> \epsilon$, Eq. (20) can be approximated as $\dot{x_l} \approx \gamma \epsilon x_l / x$ at $cg = 0$ and $\dot{x_l} \approx -\gamma x_l$ at $cg = 1$. Namely, when congestion is detected, which is source inferred, flow decrease rate is similar to that of TCP congestion control. In the absence of congestion, on the other hand, the flow increase rate is inversely proportional to its overall flow rate x, meaning that a subflow increase rate becomes smaller if the overall flow rate becomes larger. This controller is similar to and as scalable as MPTCP [32], [58], allowing all flows to evenly share network bandwidths.

Consider a multipath flow with two subflows $x_{1,1}$ and $x_{1,2}$ and a single-path flow x_2 that share a network, shown in Figure 2 (b). Subflow, $x_{1,1}$, takes a path with bandwidth of 60 Mbps, and subflow $x_{1,2}$ and x_2 share a 100 Mbps link. With these network and flow path configurations, it can be easily shown that the optimal flow rate allocation for this HOLNET NUM problem is: $x_{1,1}$ =60 Mbps, $x_{1,2}$ = 20 Mbps and x_2 = 80 Mbps, i.e., to equalize the flow rates. The performance of the MCC controller is tested against this optimal flow rate allocation (for this and all the rest of the case studies, we only show flow rate allocation, not V(**x**), as the latter is just a means to achieve the former). It is also compared against MPTCP based on the ns-3 open source code [59].

Figure 4 presents the performance results. As one can see, overall both MCC controller and MPTCP are able to allocate flow rates evenly between the two flows. For the MCC controller, x_2 is slightly lower than the optimal one (73 vs 80 Mbps) while the other two subflows are nearly equal to their respective optimal ones. The reason is the same as the previous case. Namely, a flow with higher rate (i.e., x_2) generally senses more congestions than a flow with lower rate (i.e., $x_{1,2}$), thus resulting in more reduced rate, compared to the optimal one. Since MPTCP is empirically designed, its flow rate allocation cannot be easily interpreted with reference to the optimal one. Indeed, for MPTCP, x_2 actually achieves higher rate than the optimal one, whereas $x_{1,2}$ is lower than the optimal one.

V. HOLNET-UTA

While the previous section demonstrates both viability and flexibility of HOLNET in design space exploration, this section develops a pragmatic family of lightweight traffic controllers, called HOLNET with Utility-of-TCP-based flow rate Allocation (UTA), or HOLNET-UTA in short. HOLNET-UTA is TCP-friendly and backward compatible with TCP Reno. Since in HOLNET, both minimum user utility and the center-of-utility are factored into the congestion controller in terms of θ and w, in this study, we assume θ and w are given, and focus on the testing of traffic controller performance. Since HOLNET-UTA is lightweight, requiring only per-hop feedback from network nodes, it is only compared against some well-known lightweight transport and in-network load balancing solutions. Furthermore, since all the solutions to be compared against aim at achieving network-centric performance targets, such as FCT, we adopt the same performance targets.

First, to be TCP-friendly by design, HOLENET-UTA uses TCP utility as the base utility. We adopt the following TCP utility function and its z-function for TCP Reno, derived in [60]. Let ρx and βx be the multiplicative increase rate and multiplicative decrease rate, respectively. The TCP utility function in the SSP is then given by,

$$U_{tcp}(x) = x log(1 + \frac{\rho}{\beta}), \tag{21}$$

and in the congestion avoidance phase (CAP) is given by,

$$U_{tcp}(x) = (\frac{\mu}{\beta} + x)[log(\mu + \beta x) - 1] - x[log(\beta x) - 1], \tag{22}$$

where μ is the additive-increase rate. The z-function of the TCP control law is derived in [60] as,

$$z_(t, x, cg, z) = \begin{cases} (\rho + \beta)x & for\ SSP \\ \mu + \beta x & for\ CAP \end{cases} \tag{23}$$

With $U_c(x)$ being the TCP utility function in Eqs. (21) and (22) and $z(.)$ given above, Then the HOLNET-UTA family of congestion controllers are readily derived from Eq. (12) as,

$$\dot{x} = \begin{cases} (rr^{out} - (1 + \frac{\rho}{\beta})^{-w})(\rho + \beta)x & if \quad cg = 0 \\ -2^{1-w}\beta x & if \quad cg = 1, \end{cases} \tag{24}$$

for the SSP, and

$$\dot{x} = \begin{cases} [-(\frac{\beta x}{\mu + \beta x})^w + rr^{out}](\mu + \beta x) & if \quad cg = 0 \\ -(\frac{\beta x}{\mu + \beta x})^w(\mu + \beta x) & if \quad cg = 1, \end{cases} \tag{25}$$

for the CAP. Here r is given by Eq. (14) for flows with minimum user utility requirements. Otherwise, $r = 1$. The corresponding in-network load balancers are given by Eq. (17).

HOLNET-UTA covers a large part of the HOLNET design space, i.e., CC/CC-CoS and CC-LB/CC-CoS-LB, in Table III. It is minimalistic, meaning that it uses the minimal information feedback (i.e., source inferred or per-hop) and simplest possible queuing mechanism, i.e., a single FIFO queue per output port and enabling soft CoS features without call admission control. Hence, the HOLNET-UTA family is highly scalable.

Due to the limited space, we only derive an example CC-LB controller from the above family, i.e., by simply setting

$r = 1$ (i.e., $\theta = 0$) and $w = 1$ in Eqs. (24) and (25), and compare its performance against some well-known lightweight transport and in-network load balancing solutions using a widely adopted realistic workload, i.e., data-mining [61].

In our design, each source host runs a set of congestion controllers for its flows and each datacenter network node runs a set of load balancers for flow aggregates (i.e., Eq. (16)). Now we discuss some important design aspects in more detail.

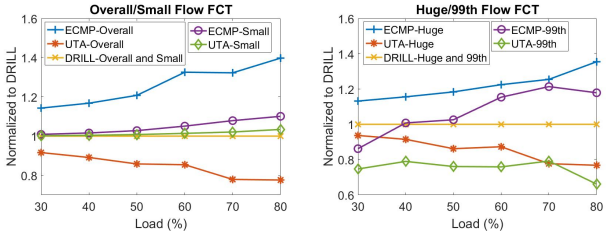
Congestion Detection: In our design, the congestion for any given output port is detected, when the queue level for the buffer corresponding to that port reaches a $p_q\%$ threshold. In our case, $p_q = 80$. This does not mean that when congestion is detected, the incoming packets are blocked. In fact, they are still allowed to enter the output buffer until buffer overflow, when the incoming packets are dropped.

Load balancing: In a two-tier leaf-spine datacenter network, to be used for our case study, load balancing is done only by the source host side leaf nodes. The load balancing is coarse-grained, with only one load balancer per destination-leaf flow aggregate, regardless of flow types. The flow aggregate rate allocation to the multi-path is based on the percentage-based load balancers given in Eq. (17) by setting $z(.) = 1$. Initially, the percentage of each outgoing port in the multi-path is assigned proportional to their bandwidth capacities. For example, with n outgoing ports, each has bandwidth B_i, for $i = 1, 2, ..., n$, the initial percentage assignments are set at, $p_i = B_i / \sum_{j=1}^{n} B_j$. After that, the percentages are updated periodically at a given time interval, which is set at RTT/4. At each update epoch, a leaf node checks if any outgoing port or its next hop is congested. If yes, then the percentage is updated according to Eqs. (16)-(18).

Packet loss recovery: Since HOLNET-UTA can sense congestion before buffer overflow, the packet drop rate is very low, particularly in the spine and sender side leaf nodes. So, we turn off the fast retransmission feature of TCP (i.e., retransmission upon receiving three duplicated ACKs) and solely rely on retransmission timeout for packet loss recovery. When a timeout occurs, for simplicity, the source host retransmits all the packets from the timed out packet (similar to go back N). In our solution, a timeout will not trigger congestion control. Instead, it solely relies on the local and the next hop feedback information for the control.

We compare the performance of UTA by simulation against DCTCP, combined with one of the two load balancing solutions, i.e., ECMP [2] or DRILL [7]. DRILL is found [7] to offer better performance than most of the existing schemes, including CONGA [8] that uses global information.

A widely adopted performance metric for load balancing is FCT. So we use average FCT as the main performance metric in the context of overall flows, small flows (size <100K), and huge flows (size >10M bytes), and the 99th FCT for overall flows. A 6x6 leaf-spine network topology with 24 hosts per rack is simulated. The bandwidth/propagation delay



(a) (b)

Fig. 5. Performance Comparison (Data-ming)

is set at 10Gps/10μs between a host and a leaf node, and at 20Gbps/30μs between a leaf and a spine node. The queue size in a leaf/spine node is 150/300 Kbytes. The ECN marking threshold is set at 65% of queue size, the typical value used in DCTCP [3]. The control parameters are set at $r_{min} = 0.1$ and $r_{max} = 1.5$. Flows arrive following a Poisson process. The network load is adjusted with the change of flow arrival rate. When a flow arrives, a host is randomly selected as the send host and then a host in a different rack is randomly selected as the destination host.

Figure 5 gives the results (normalized to DRILL) for the data-mining workload case (similar results are obtained for the web-search workload [3], which is not shown here for the lack of space). UTA outperforms both DRILL and ECMP for most cases studied. Particularly, for the heavy load case, UTA outperforms DRILL (ECMP) by more than 20% (60%) in terms of average FCT, average FCT for huge flows and 99th percentile and it is on par with DRILL for the small flow case. This is because with per-hop congestion feedbacks and integrated congestion control and load balancing, UTA can respond to network congestions much faster and allow better balanced load than both ECMP with DCTCP and DRILL with DCTCP, which are not integrated solutions.

VI. Conclusions

This paper presents HOLNET, a holistic traffic control framework for datacenter networks. HOLNET allows large families of traffic control protocols of various degrees of sophistication to be developed. Unlike the existing solutions that are largely empirical by design, HOLNET is a principled, systematic solution. Protocols in each family developed under HOLNET share a common, user-defined global optimization objective. As a result, the protocols in each family can be fairly compared and carefully selected to fully explore the performance, scalability and design complexity tradeoffs. As an example, we develop HOLNET-UTA, a family of integrated congestion controllers and load balancers, maximizing the sum of weight TCP utilities. The large-scale simulation demonstrates the backward compatibility and flexibility of HOLNET.

VII. Acknowledgments

We would like to thank our shepherd, Richard T. B. Ma, and the anonymous reviewers for their insightful feedbacks. This work is supported by the US NSF under Grant No. CCF XPS-1629625 and CCF SHF-1704504.

REFERENCES

[1] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno performance: a simple model and its empirical validation," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 133–145, 2000.

[2] C. E. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm," in *RFC 2992*.

[3] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," in *Proceedings of ACM SIGCOMM*, 2010.

[4] B. Vamana, J. Hasan, and T. Vijakumar, "Deadline-Aware Datacenter TCP (D^2TCP)," in *Proceedings f ACM SIGCOMM*, 2012.

[5] D. Han, R. Grandl, A. Akella, and S. Seshan, "Fcp: A flexible transport framework for accommodating diversity," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 135–146, 2013.

[6] K. Nagaraj, D. Bharadia, H. Mao, S. Chinchali, M. Alizadeh, and S. Katti, " NUMFabric: Fast and Flexible Bandwidth Allocation in Datacenters," in *Proceedings of the 14th ACM SIGCOMM*, 2014.

[7] S. Ghorbani, Z. Yang, P. B. Godfrey, Y. Ganjali, and A. Firoozshahian, "DRILL: Micro Load Balancing for Low-latency Data Center Networks," in *Proceedings of ACM SIGCOMM*, 2017.

[8] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese, " CONGA: Distributed Congestion-Aware Load Balancing for Datacenters," in *Proceedings of ACM SIGCOMM*, 2014.

[9] L. Chen, K. Chen, W. Bai, and M. Alizadeh, "Scheduling Mix-flows in Commodity Datacenters with Karuna," in *Proceedings of ACM SIGCOMM*, 2016.

[10] D. Acemoglu, R. Johari, and A. Ozdaglar, "Partially Optimal Routing," *IEEE Journal of Selected Areas of Communications*, vol. 25, pp. 1148–1160, 2007.

[11] L. Qui, Y. R. Yang, Y. Zhang, and S. Shenker, "On Relfish Routing in Internet-like Environments," in *Proceedings of ACM SIGCOMM*, 2003.

[12] F. Kelly and T. Voice, "Stability of end-to-end algorithms for joint routing and rate control," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 2, pp. 5–12, 2005.

[13] Y. Liu, H. Zhang, W. Gong, and D. F. Towsley, "On the interaction between overlay routing and underlay routing," in *Proceedings of IEEE INFOCOM*, 2005.

[14] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS Adaptive Traffic Engineering," in *Proceedings of IEEE INFOCOM*, 2001.

[15] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the Tightrope : Responsive Yet Stable Traffic Engineering," in *Proceedings of ACM SIGCOMM*, 2005.

[16] S. Sen, D. Shue, S. Ihm, and M. J. Freedman, "Scalable, optimal flow routing in datacenters via local link balancing," in *Proceedings of ACM CoNEXT*, 2013.

[17] C. Lee, C. Park, K. Jang, S. Moon, and D. Han, "Accurate latency-based congestion feedback for datacenters," in *Proceedings of USENIX ATC*, 2015.

[18] M. Alizadeh, A. Kabbani, T. Edsall, and B. Prabhakar, "Less is More: Trading a little Bandwidth for Ultra-Low Latency in the Data Center," in *Proceedings of USENIX NSDI*, 2012.

[19] M. Alizadeh, S. Yang, M. Sharif, and S. Katti, "pFabric: Minimal Near-Optimal Datacenter Transport," in *Proceedings of ACM SIGCOMM*, 2013.

[20] I. Cho, K. Jang, and D. Han, "Credit-Scheduled Delay-Bounded Congestion Control for Datacenters," in *Proceedings of ACM SIGCOMM*, 2017.

[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 ACM CoNEXT*, 2015.

[22] R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker, "Recursively Cautious Congestion Control," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, 2014.

[23] D. Katabi, M. Handkwy, and C. Rohrs, " Congestion control for high bandwidth-delay product networks," in *Proceedings of ACM SIGCOMM*, 2002.

[24] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and W. Sun, "PIAS: Practical Information-Agnostic Flow Scheduling for Data Center Networks," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, 2014.

[25] A. Munir, G. Baig, S. M. Irteza, I. A. Qazi, A. X. Liu, and F. R. Dogar, "Friends, Not Foes: Synthesizing Existing Transport Strategies for Data Center Networks," in *Proceedings of ACM SIGCOMM.*, 2014.

[26] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. Ismail, M. S. Iqbal, and B. Khan, "Minimizing Flow Completion Times in Data Centers," in *Proceedings of IEEE INFOCOM*, 2013.

[27] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. Haj, and Y. Ming, "Congestion Control for Large-Scale RDMA Deployments," in *Proceedings of ACM SIGCOMM*, 2015.

[28] IEEE, "DCB. 802.1Qbb - Priority-based Flow Control," 2011. [Online]. Available: http://www.ieee802.org/1/pages/802.1bb.html

[29] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn, "RDMA over Commodity Ethernet at Scale," in *Proceedings of ACM SIGCOMM*, 2016.

[30] R. Mittal, U. C. Berkeley, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. G. Microsoft, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, "TIMELY : RTT-based Congestion Control for the Datacenter," in *Proceedings of ACM SIGCOMM*, 2015.

[31] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowstron, "Better Never than Late: Meeting Deadlines in Datacenter Networks," in *Proceedings of ACM SIGCOMM*, 2011.

[32] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving Datacenter Performance and Robustness with Multipath TCP," in *Proceedings of ACM SIGCOMM*, 2011.

[33] M. Handley, C. Raiciu, A. Agache, A. Voinescu, A. W. Noore, G. Antichi, and M. Wojcik, "Re-architecting datacenter networks and stacks for low latency and high performance," in *Proceedings of ACM SIGCOMM*, 2017.

[34] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing flows quickly with preemptive scheduling," in *Proceedings of ACM SIGCOMM*, 2012.

[35] N. Dukkipati and N. Mckeown, "Why Flow-Completion Time is the Right metric for Congestion Control and why this means we need new algorithms," *ACM SIGCOMM Computer Communication Review*, vol. 36, pp. 59–62, 2006.

[36] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akella, "Presto: Edge-based Load Balancing for Fast Datacenter Networks," in *Proceedings of ACM SIGCOMM*, 2015.

[37] J. Cao, R. Xia, P. Yang, C. Guo, G. Lu, L. Yuan, Y. Zheng, H. Wu, Y. Xiong, and D. Maltz, "Per-packet Load-balanced , Low-Latency Routing for Clos-based Data Center Networks Categories and Subject Descriptors," in *Proceedings of ACM CoNEXT*, 2013.

[38] A. Kabbani, B. Vamanan, J. Hasan, and F. Duchene, "FlowBender: Flow-level Adaptive Routing for Improved Latency and Throughput in Datacenter Networks," in *Proceedings of ACM CoNEXT*, 2014.

[39] H. Zhang, J. Zhang, W. Bai, K. Chen, and M. Chowdhury, "Resilient Datacenter Load Balancing in the Wild," in *Proceedings of ACM SIGCOMM*, 2017.

[40] N. Katta, A. Ghag, M. Hira, I. Keslassy, A. Bergman, C. Kim, and J. Rexford, "Clove: Congestion-Aware Load Balancing at the Virtual Edge," in *Proceedings of ACM CoNEXT*, 2017.

[41] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "HULA: Scalable Load Balancing Using Programmable Data Planes," in *Proceedings of ACM SOSR*, 2016.

[42] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella, "On the Impact of Packet Spraying in Data Center Networks," in *Proceedings of ACM INFOCOMM*, 2013.

[43] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic load balancing without packet reordering," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 2, p. 51, 2007.

[44] E. Vanini, R. Pan, M. Alizadehand, P. Taheri, and T. Edsall, "Let It Flow : Resilient Asymmetric Load Balancing with Flowlet Switching," in *Proceedings of ACM NSDI*, 2017.

[45] P. Wang, H. Xu, Z. Niu, D. Han, and Y. Xiong, "Expeditus : Congestion-aware Load Balancing in Clos Data Center Networks," in *Proceedings of ACM SoCC*, 2016.

[46] S. Shenker, "Fundamental design issues for the future internet," *IEEE Journal of Selected Areas in Communications*, vol. 13, pp. 1176–1188, 1995.

[47] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 1, pp. 237–252, 1998.

[48] S. H. Low and D. E. Lapsley, "Optimization Flow Control I: Basic Algorithm and Convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861–874, 1999.
[49] S. H. Low, "A Duality Model of TCP and Queue Management Algorithms," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 525–536, 2003.
[50] S. Korovin and V. Utkin, "Using sliding modes in static optimization and nonlinear programming," *Automatica*, vol. 10, no. 5, pp. 5250–532, 1974.
[51] B. A. Movsichoff, C. Lagoa, and H. Che, "End-to-End Optimal Algorithm for Integrated QoS, Traffic Engineering, and Failure Recovery," *ACM/IEEE Transactions on Networking*, vol. 15, no. 4, pp. 813–823, 2007.
[52] W. Su, C. Liu, C. Lagoa, H. Che, K. Xu, and Y. Cui, "A Family of Optimal, Distributed Taffic Control Laws in a Multidomain Environment," *IEEE Transactions on Control System Technology*, vol. 23, no. 4, pp. 1373–1386, 2015, The initial idea of this paper was documented in H. Che, W. Su, C. Lagoa, K. Xu, C. Liu, and Y. Cui, "An Integrated, Distributed Traffic Control Strategy for the Future Internet," *The ACM SIGCOMM INM Workshop, 2006.*
[53] M. Ashour, J. Wang, C. M. Lagoa, N. Aybat, and H. Che, "Non-Concave network utility maximization: A distributed optimization approach," in *Proceedings of IEEE INFOCOM*, 2017.
[54] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," pp. 556–567, 2000.
[55] "Center of mass," https://en.wikipedia.orgwikiCenter_of_mass.
[56] "Lagrange multiplier," https://en.wikipedia.orgwikiLagrange_multiplier.
[57] W. Bai, L. Chen, K. Chen, and H. Wu, "Enabling ECN in Multi-Service Multi-Queue Data Centers," in *Proceedings of ACM NSDI*, 2016.
[58] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proceedings of USENIX NSDI*, 2011.
[59] M. Kheirkhah, I. Wakeman, and G. Parisis, "Multipath-TCP in ns3," in *Proceedings of ACM Workshop on ns-3*, 2014.
[60] L. Ye, Z. Wang, H. Che, and C. M. Lagoa, "TERSE:A Unified End-to-End Traffic Control Mechanism to Enable Elastic, Delay Adaptive, and Rate Adaptive Services," *IEEE Journal on Selectied Areas in Communications*, vol. 29, no. 5, pp. 938–950, 2011.
[61] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A Scalable and Flexible Data Center Network," in *Proceedings of ACM SIGCOMM*, 2009.