One Bit Is Enough: a Framework for Deploying Explicit Feedback Congestion Control Protocols

Nedeljko Vasić*, Srinidhi Kuntimaddi† and Dejan Kostić*
*EPFL, Lausanne, Switzerland, first.last@epfl.ch
†IIT, Guwahati, India, kuntimaddi@iitg.ernet.in (work done during an Internship at EPFL)

***Abstract*—**
In this paper we describe UNO, a framework for fine-grain explicit feedback congestion control protocols that uses only 1 or 2 existing ECN bits, thus making algorithms that use more than 2 bits for encoding the load factor and the RTT immediately deployable. UNO accomplishes this task by changing the way load and RTT information is encoded in packets in a way that is similar to some existing schemes for encoding bottleneck link load. UNO leverages the values present in the IP identification field and trades-off a small amount of time (to send several packets) for space to emulate the existence of several extra bits within the IP header. The results from extensive ns2 simulations over various bandwidth and delay scenarios are encouraging. By using only one ECN bit we achieve substantially lower convergence times and better link utilization than the existing deployable protocols, with similar low queue size and negligible packet loss. With 2 ECN bits, we achieve very good fairness for flows with different RTTs, while keeping all the good characteristics of the 1-bit protocol and providing functionality that did not previously exist.

I. Introduction

As the bandwidth-delay products (BDP) of Internet links keep increasing, issues with TCP's AIMD controller in these environments are becoming more pressing. For example, TCP's additive increase phase increases the congestion window by only one packet per RTT. This behavior leads to poor convergence times and low link utilization as the flow takes a long time to ramp up to available bandwidth after a packet loss. In addition, TCP's throughput is inversely proportional to the RTT, resulting in fairness issues. In an attempt to solve these and other problems, recent research efforts have pushed in two different directions. Strict end-to-end schemes [8], [19] bring significant benefits, but do not solve the problem in the long run as there are limits to using strictly packet loss [8] and/or delay [5] as congestion signals. The second direction includes using feedback from the network. The simplest form which has progressed the furthest (and is deployed in some routers) is TCP+AQM/ECN. ECN-enabled routers signal congestion sooner than loss-based schemes by setting an appropriate bit in the IP header. This approach reduces loss rate and queue sizes in the network, but does not ensure good link utilization in high BDP environments [11].

Recent efforts in this field can be classified into two groups based on the granularity of congestion feedback they use: i) algorithms with explicit-feedback information and ii) algorithms with approximate-feedback information. The best representative of the first group is eXplicit Congestion Control (XCP) [11]. XCP augments the IP header to enable the sender to transmit its current congestion window size and estimated RTT. XCP router then estimates the fair rate for each flow and sends it back to the sender as explicit feedback. XCP performs very well, regardless of link capacity, round trip time (RTT), and number of flows. However, its deployment is hampered mostly by the need to change the IP header. Variable-structure congestion Control Protocol (VCP) authors [20] recognized this problem and proposed an approximate feedback scheme that uses the existing 2 ECN bits to transmit a few levels of load information to senders. However, VCP does not propagate per flow RTT information. This constraint, as well as the course-grained load information, is responsible for relatively long VCP convergence times. In addition, VCP suffers from fairness issues for flows that have very different RTTs. Most recent work in this space, MLCP [14], alleviates most of VCP's performance issues by i) encoding more load levels by using 4 bits, ii) transmitting the per-flow RTTs, and iii) redesigning the controller for faster convergence. However, this scheme again requires use of the entire TOS field in the IP header which reduces its deployment potential. Existing proposals for reducing the number of header bits that are required for new congestion control protocols [3], [18] do not collect and transmit RTT-related information.

As we move toward the next-generation Internet, it is important to start experimenting with congestion control protocols for high BDP networks as soon as possible. Even if one has an opportunity to design a completely new IP header (and include arbitrary information), it is important to gain experience in large-scale deployments across today's Internet. We believe this is most likely to occur if the changes that are required to the IP header and the router implementations are minimal.

In this paper we describe UNO, a framework for fine-grain load-factor based (explicit feedback) congestion control protocols that uses only of 1 or 2 existing ECN bits [15]. UNO accomplishes this task by changing the way load and RTT information is encoded in packets. UNO takes advantage of the values present in the IP identification field and trades-off a small amount of time (to send several packets) for space to emulate the existence of several extra bits (up to 16) within the IP header.

Specifically, we leverage the fact that the IP identification field (IPID) in packets originating from every host is: i) monotonically increasing, or ii) chosen uniformly at random. For example, to transmit load information using UNO, every router examines the few least significant IPID bits in packets it is forwarding. When the load matches those bits, the router sets the ECN bit. As in the previous schemes, the receiver reflects the load back to the sender in ACK packets. After several packets, the sender is aware of the load at the bottleneck router in the end-to-end path, and it feeds this information to its controller to determine the new congestion window. The design of the controller can vary. In this paper, we have used simple modifications to the VCP's congestion controller with considerable success. In high BDP environments that are the target environment of this protocol, there will be several consecutive packets in flight. It is therefore highly probable that our encoding scheme introduces a minimal delay to transmit the extra load and RTT information to the routers and all the way back to the sender.

The benefit of using UNO is twofold: i) it makes all algorithms that use more than 2 bits for encoding the load factor and the RTT immediately deployable (in conjunction with necessary router forwarding logic changes), and ii) it enables network protocol designers and operators to choose how many existing ECN bits (one or two) they want to use for congestion control. The 1-bit scheme is potentially preferable to 2-bit schemes as it is interoperable with the existing ECN implementations in routers.

Benefits of this framework are not only limited to congestion control protocols. It can also be useful in traffic management tasks which span congestion control, routing protocols and traffic engineering. Recently, a class of distributed algorithms that shape traffic across multiple paths (e.g., TRUMP [10]) has been proposed. Here, link prices are incrementally computed and each iteration step is equal to the longest round trip time (RTT) in the network. Since UNO can be used to convey

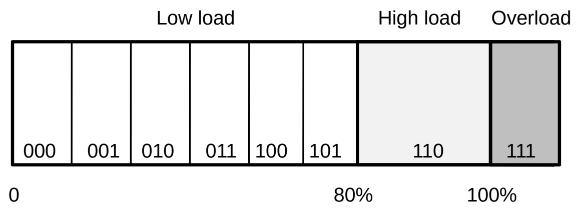

Fig. 1: Using 3 bits to encode load factor at a link

the RTT-related information, this class of algorithms can significantly benefit from our framework.

We evaluate UCP's performance by running extensive ns2 simulations over various bandwidth and delay scenarios. Our results are encouraging, as by using only one ECN bit we achieve substantially lower convergence times and better link utilization than VCP, with similar low queue size and negligible packet loss. In addition, the performance is comparable to that of a protocol that uses 3 bits to transfer exact link utilization in each packet. With 2 ECN bits, we achieve better fairness for flows with different RTTs than VCP, while keeping all the good characteristics of the 1-bit protocol.

The rest of the paper is organized as follows. In section II, we describe our encoding scheme in more details. We provide more information about protocol implementation across the participants within the network (senders, routers and receivers) in section III. In section IV, we evaluate the performance of the framework using extensive simulations. We review related work in section V, and summarize and discuss our work in section VI.

II. The UNO Framework

We assume that the sender, intermediate routers, and the receiver collaborate to achieve high link utilization, low convergence time and small queue sizes. The sender typically does most of the work, and runs a congestion controller that takes load, RTT and other information into account. Routers encode load (and other) information into IP packets using N bits by performing lightweight packet inspection and potentially stamping outgoing packets with the UNO bit(s). The receiver reflects the information that is being collected back to the sender via ACK packets.

The exact way of encoding load information is protocol-dependent. To demonstrate the benefits of using UNO however, in this paper we use a modified version of the scheme used by VCP [20]; Figure 1 shows that we encode load in the low-load region (0-80% link utilization) in 6 fine-grain levels.

In this section we first describe our approach for transmitting load information using only one UNO bit, thereby making algorithms with approximate-feedback information immediately deployable, in conjunction with the necessary router forwarding logic changes. We

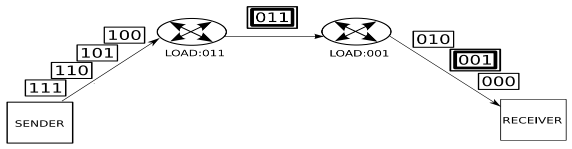

Fig. 2: Using 1 UNO bit to transmit link load information to the receiver

then describe how an additional bit can be used to transmit RTT-related information for protocols that wish to achieve better convergence and fairness.

A. Using one bit to transmit load factor

The primary goal of our framework is to improve the precision of load information that is collected by the routers on an end-to-end path. The technique we use leverages the temporal proximity of several packets that are in flight from sender to the receiver in high BDP environments. We observe that there exist 16-bit fields within the packet that are changing over time; one such field is the IP identification field (IPID). Our framework could use other fields, but we concentrate on the IPID as it is already in the IP header. This choice allows packet processing to remain on the fast path and avoids the need to perform deep packet inspection. Depending on a particular protocol stack, IPID in packets that are originating from every host are: i) monotonically increasing (e.g., on Windows Vista [12]), or ii) random (e.g., Solaris). Due to various security reasons, the overarching trend in modern protocol stacks is to generate IPID uniformly at random for each outgoing packet.

Given our initial focus on the 16-bit IPID field, our framework could be used to transfer load information that is encoded in several bits. Since recent work [14] shows that 3 bits are sufficient for achieving good convergence (and adding more bits brings only marginal benefits), we use 3 bits for the remainder of this paper to describe load information.

Figure 2 depicts how a protocol that uses only 1 UNO bit collects load information from the bottleneck router. Here, several packets are in flight across two routers that wish to mark outgoing packets with their utilization levels of $(011)_2$ and $(001)_2$, respectively. Toward this end, they each examine the 3 least significant IPID bits in the incoming packets. When the load on its outgoing interface matches those bits, the router simply sets the UNO bit. Receivers reflect the load (when the UNO bit is set) back to the sender in the ACK packet.

Given that our protocol disperses load information across several packets, they could be carrying different load that corresponds to the varying load information across the routers on the path. In our example, the left router sets the UNO bit in the packet marked as $(011)_2$ while the right router sets the UNO bit for the packet labeled with $(001)_2$. It is important that our framework handles this and other cases, including packet loss, and correctly conveys the highest load encountered on the path.

Having two or more routers with the same maximum load does not represent the problem. If routers have different load, as is the case in Figure 2, the sender would not know the correct maximum load until the ACK packet with the specific IPID contents arrives. Therefore, UNO controller waits until it receives 8 successive packets (2^3, for all possible 3-bit combinations) and picks the maximum reported load. Our protocol trivially handles packet loss - in this case the controller reacts exactly as it should to the packet loss event (there is no need to use “weaker” congestion information in the form of the router load).

One might be concerned that spreading the load information across 8 packets might unnecessarily delay the load information. In a high BDP environment that is precisely the target of our framework, there will be a considerable number of packets in flight. It is therefore highly likely that our encoding scheme introduces a small delay in transmitting the load information. Our experimental results confirm this intuition (the average observed delay is 0.07 ms on a 1-Gbps link with 1000-byte packets, which is typically a small fraction of the RTT).

When IPID is generated uniformly at random, 8 successive packets might not capture all possible 3-bit values. In this case, the sender checks the outbound packets and determines how many of them are carrying different 3-bit fields. It then waits to receive the ACKs for those packets before passing along the load information to the congestion controller. In expectation, 22 packets will capture all possible 3-bit load encodings (this is an instance of the coupon collector problem). Again, we do not expect this requirement to be an issue, and our experimental results confirm our intuition.

An astute reader might wonder about pathological cases in which, for example, a given host has exactly 8 synchronized flows, and the packets are being sent one at a time in round-robin fashion (so that the low-order 3 bits are not changing in the flows). With the modern protocol implementations moving to random IPID fields, we believe this is less of an issue. Further, per-host traffic is typically bursty. Finally, the corner cases can be further mitigated by using a slightly different technique for setting UNO bits. For example, the router would not wait for perfect match, but would set the UNO bit if its load is higher from the one encoded in the IPID. We leave this alternative encoding implementation as future work.

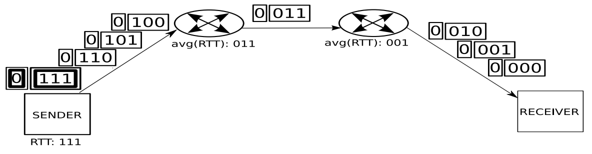

Fig. 3: Transmitting per-flow RTT to the routers using the second UNO bit

The receiver in some protocols might use delayed ACKs, e.g., it might send an acknowledgment after every other packet it receives. UNO handles this case easily by having the receiver report the maximum load seen in the last two packets.

B. Using the second bit to transmit RTTs

Although using just one UNO bit is sufficient for transmitting fine-grained load information, protocols that do not calculate and transmit average RTT time for flows passing through routers (e.g., VCP), have two drawbacks: i) they become unfair in the presence of flows that exhibit a large difference in their RTTs and ii) and they have to be conservative which negatively affects low latency flows. Therefore, there is an obvious need for transmitting RTT-related information [14]. To accomplish this task, our framework lets end hosts transmit their RTTs to routers, thus enabling routers to calculate fair shares. In addition, UNO enables routers to communicate the average RTT of flows passing through them back to end hosts, allowing their controllers to determine how to change their rates.

We require another UNO bit for this task. Relative to transmitting link load, this use case is more complex as information needs to be transmitted in two rounds. First, senders provide routers with their estimated per-flow RTTs. Second, routers convey the average observed RTTs to receivers.

Figure 3 shows the first round in which a sender provides routers along the path with its estimated RTT, using a 3-bit encoding scheme. In a general, N-bit encoding scheme, the sender looks into (N+1)st right-most bit. If it is zero it then checks the N right-most bits in the IP identification field. If there is a match with its estimated RTT, the sender sets the second (RTT) UNO bit. Routers recognize this information and update the average RTTs for flows passing through them.

Routers transmit average observed RTT to senders by checking the (N+1)st bit from right. If it is 1, they compare their N-bit representation of the average RTT with the N rightmost bits from the identification IP field. If the condition is satisfied, they set the second UNO bit. Receivers propagate this information back to senders via ACK packets as they do for the load.

C. Deployment and interoperability with existing ECN implementations

For deployment of an 1-bit protocol, UNO sender would stamp outgoing packets with the ECT(1) [15] code symbol ($(10)_2$ value of the ECN bits). A router with a matching load would set the bit using ECT(0) ($(01)_2$). Using just one bit enables the sender and the receiver to detect UNO-enabled routers as they would flip the leftmost ECN bit from 1 to 0 when inserting the link load. Further, this mechanism allows for interoperability with existing ECN-enabled routers; these routers expect the senders to be using either the ECT(0) or ECT(1) combination to announce their willingness to use ECN, and they mark the onset of congestion by using the $(11)_2$ ECN bit combination. UNO-enabled receivers would then realize that there is a legacy ECN router in the path by observing the $(11)_2$. Finally, this scheme lets the senders probe the path for "old" routers without ECN or UNO that do not make any changes to the ECN bits. Some of the additional protocols developed on top of ECN, e.g., ECN Nonces [17], might require modification to work with UNO.

Two-bit UNO protocols would likely require widespread deployment as they are not directly interoperable with existing ECN implementations. In this scheme, UNO uses the existing ECN bits for its purposes ("leftmost" ECN bit for encoding load information, the other one for RTT-related data). UNO would still function in the presence of the routers that are not ECN/UNO-enabled.

III. Implementation

To take advantage of the additional fine-grain load and RTT-related information, we need an appropriate controller (at the sender) and router support. We start with the VCP [20] logic, and first modify it to work with 8 instead of 3 load levels (Figure 1). We refer to this protocol as UCP1. To incorporate RTT-related information, we borrow some features from the MLCP [14] controller and router design. We use the 2-bit UNO framework to encode and transmit RTT information with 3 bits. The resulting protocol is called UCP2. In the remainder of this section, we explain UCP1 and UCP2 by discussing the logic at the sender, the router, and the receiver.

A. Sender

The sender applies either MI, AI or MD, based on the load factor reported by the most congested router. Transition points are chosen as in VCP [20]:

Low load (Load factor [0%,80%)). In this regime, senders apply the MI factor according to the load value encoded over 6 uniformly-sized levels.

MI: $cwnd\left(t+rtt\right)=cwnd\left(t\right)*\left(1+\xi_s\left(\sigma\right)\right)$ where
$\xi_s\left(\sigma\right)=\left(1+\xi\left(\sigma\right)\right)^{\frac{rtt}{t_\rho}-1}$, $\xi\left(\sigma\right)=k*\frac{1-\sigma}{\sigma}$,
σ – load factor and $k=0.15$

To address the problem of RTT heterogeneity, senders estimate RTT and normalize it with the common factor t_ρ. Scaling parameters in this way emulates the behavior where all flows have the same RTT, which is equal to t_ρ. We discuss t_ρ in more details in the next section.

High load (Load factor $>$80%.) Once a system achieves a desirable utilization level, senders use the AIMD algorithm to converge to fair share. Within the [80%,100%) interval, the senders apply AI. If the system moves into overload, senders apply MD. The following equations describe these steps:

AI: $cwnd\left(t+rtt\right)=cwnd\left(t\right)+\alpha_s$, where
$\alpha_s=\alpha*\frac{rtt}{t_\rho}$ and $\alpha=3.0$
MD: $cwnd\left(t+\delta t\right)=cwnd\left(t\right)*\beta$, where $\beta=0.875$

We note that the AI parameter is scaled in the same fashion as MI parameter. However, since MD is an impulse-like operation that is not affected by the length of the RTT, it is not scaled. Since we have only one MD level, the β parameter does not depend on the load factor. We are aware that the presence of multiple MD levels enables the protocol to be more responsive to congestion according to the degree of utilization on the most congested link [14]. Nevertheless, we leave this to the future work and in this paper use the VCP-like controller to illustrate the main points of our framework.

B. Router

In a scheme with one UNO bit (UCP1), routers only compute the load factor and set the scaling factor t_ρ to 200 ms, as in VCP. In a 2-bit scheme, they also compute the average RTT for flows passing through them in a way similar to MLCP. We encode RTTs using 3 bits, so the t_ρ value is chosen from a set S of 8 elements (2^3 = 8), where: S = {80,200,400,600,800,1000,1200,1400}.

S is constructed according to the following observations: i) previous work shows that if a flow is within $2-2.5*t_\rho$, there is hardly any queue buildup, and ii) having this many RTT levels avoids any fluctuations in t_ρ due to minor variations in flows' RTT.

We estimate the load factor over time interval t_ρ, which needs to satisfy two conflicting conditions. First, it has to be larger than the RTT of most flows in order to avoid burstiness caused by flows's responses. Second, it should be small enough for good responsiveness and small queue size. Internet measurements shows that majority of flows has RTT less than 200 ms [20]. Thus, due to the lack of explicit latency information when only one UNO bit is used, we set $t_\rho=200ms$, as VCP. During this time interval, each router estimates a load factor σ for each its output link l as:

$$\sigma=\frac{\left(\lambda l+kq*ql\right)}{\left(\gamma l*Cl*t_\rho\right)}$$

where λ_l is the amount of traffic during the period, q_l is the persistent queue length during this time. Coefficient k controls how fast the persistent queue length drains (0.25), and C_l is link capacity. Techniques that we used for measuring the input traffic and the persistent queue length are "borrowed" from VCP and MLCP.

However, using fixed tq value leads to high sensitivity on very low (e.g. 1ms) or very large (higher than 800 ms) values of RTT in scaling AI and MD parameters. Protocols willing/able to use two UNO bits may obtain the average RTT of flows and avoid this problem. In that case, packets that carry information about sender's estimated RTT are used to update average RTT for flows as: $\overline{m}=a*RTT+\left(1-a\right)*\overline{m}$, where $a=0.02$, as in MLCP.

C. Receiver

The receiver just copies the identification field and the UNO bits to the ACK packet. If it is necessary for deployment, the receiver could send the load and the RTT back to the sender similar to the forward propagation mechanism.

IV. Evaluation

We use extensive ns2 simulations to show the benefits of our framework using UCP1 and UCP2, protocols that use 1 and 2 ECN bits, respectively. We compare them with a 1-bit scheme such as VCP across a wide range of network scenarios, including varying the link capacities in the range [100Kbps, 1Gbps], round trip times in the range [1 ms, 1.5 s], numbers of long-lived FTP-like flows in the range [1, 1000]. The bottleneck buffer size is set to the bandwidth-delay product, or two packets per-flow, whichever is larger. The data packet size is 1000 bytes, while the ACK packet size is 40 bytes. Due to space reasons, we only present the most important results.

To demonstrate that spreading the load and RTTs across several packets does not jeopardize performance, we include a protocol we call VCP3 in the comparison. VCP3 uses 3 bits to represent load information (as do UCP1 and UCP2), but includes link load in every packet by using 3 bits in the IP header.

We have implemented our protocols with both monotonically increasing and random IPIDs. As the results

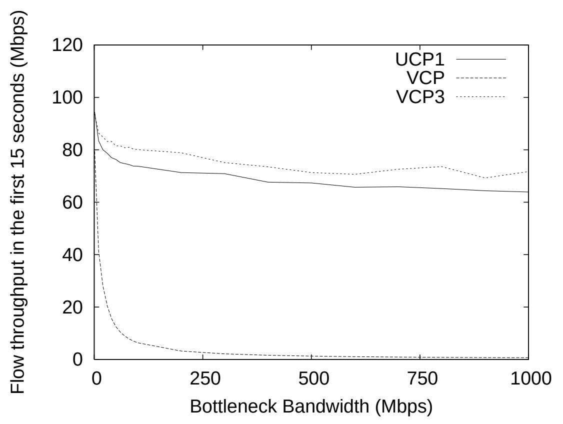

Fig. 4: Additional load levels help UCP1 converge quickly.

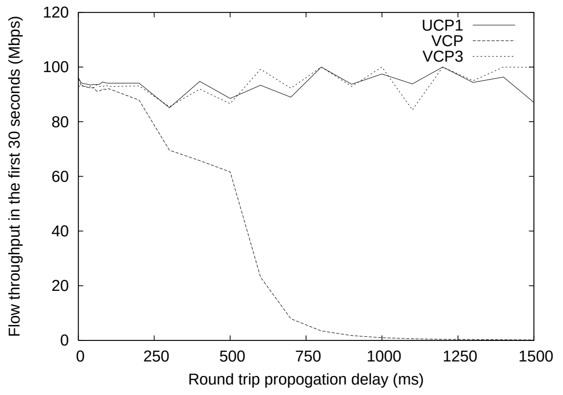

Fig. 5: Throughput after 30 sec while varying RTTs.

are very similar, we show the graphs obtained with the monotonically increasing IPIDs.

A. Convergence to High Utilization

Figure 4 shows the bottleneck utilization at time = 15 s for UCP1, VCP and VCP3. Due to finer-grain load information, UCP1 achieves slightly below 80% utilization across link capacities ranging from 1 Mbps to 1 Gbps, whereas, VCP utilization falls significantly as link capacity is increased. In addition, UCP1's utilization is similar to a protocol that always transmits 3 bits of link load.

We then fix the bottleneck capacity to 45 Mbps and vary the round-trip propagation delay from 1 ms to 1.5 s. Figure 5 shows the results obtained in this setup after 30 seconds. We see that UCP1 again achieves better throughput than VCP, and is practically identical to VCP3.

Having demonstrated UCP1's good convergence, for the remainder of the experiments we run for at least 100 s to ensure that all protocols reach stable state where they achieve high utilization, low persistent queue length, low packet drop rate and fairness.

B. Dynamics

To obtain more insight into convergence behavior of UCP1 we run an experiment where 5 flows with RTTs in range [40 ms,80 ms] compete for the single bottleneck link with a bandwidth of 45 Mbps. The flows are introduced into the system with start times separated by 100 seconds. Figure 6 shows that UCP1 converges quicker than VCP to fair bandwidth allocation. There are two reasons for that. First, our controller uses a greater AI coefficient which slightly increases the queue length but significantly improves the convergence time. Second, we do not have to limit AI parameters. Moreover, we observe UCP1's performance in the presence of UDP flows. Namely, after 20 s from the beginning of the experiment, we introduced a UDP flow with a streaming rate of 40 Mbps. Figure 7 proves that UCP1 performs well in the presence of UDP flows, and quickly adjusts its congestion window. Once the UDP flow is gone, the MI phase rapidly brings the UCP1 flow to high utilization, with a low queue size.

C. Fairness

Next, we look at the RTT-induced fairness. Figure 8 presents the results of an experiment with 30 FTP flows with RTTs ranging from 40 ms to 156 ms. Since UCP2 collects and transmits mean RTT, there is no need for parameter limiting for very high or very small RTT. Thus, this protocol achieves a near-optimal distribution of bottleneck capacity among the flows. On the other hand, VCP limits parameters for flows with small RTT (less than 200 ms) and with large RTT (higher than 500 ms). Therefore, it cannot achieve sufficient fairness even in the case of flows with small RTT variation. UCP2 achieves better fairness than VCP3 (and UCP1 that behaves similarly), because the latter protocol does not collect average RTTs. This experiment highlights the need for using RTTs in congestion protocols for high BDP networks.

V. Related Work

There has been a large body of work on congestion control protocols. Here we highlight four categories that are most relevant to UNO.

Pure end-to-end schemes. TCP Vegas [4] and FastTCP [19] attempt to improve TCP performance in steady state by using latency-related information. The Probe Control Protocol (PCP) [2] emulates network-based control by relying on probe packets to estimate the available bandwidth on the bottleneck link. Nevertheless, pure end-to-end schemes cannot achieve good fairness and high utilization while achieving low queues and low loss rate in high BDP networks [11].

Congestion notification schemes. In these schemes routers try to predict when congestion is about to happen

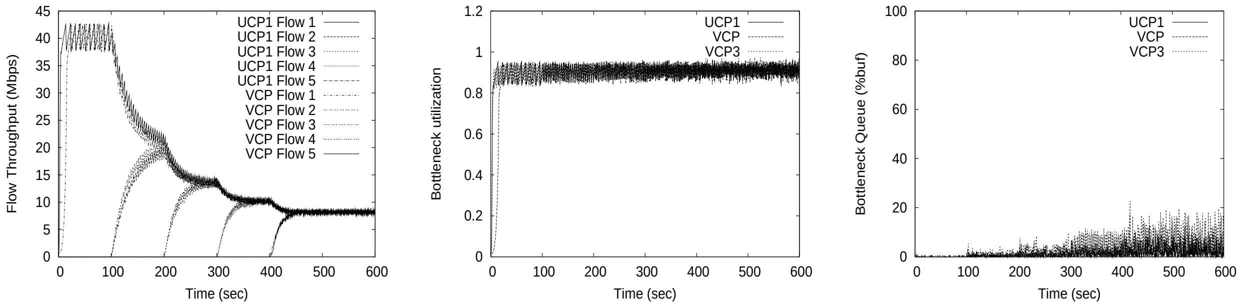

Fig. 6: Convergence with staggered flows

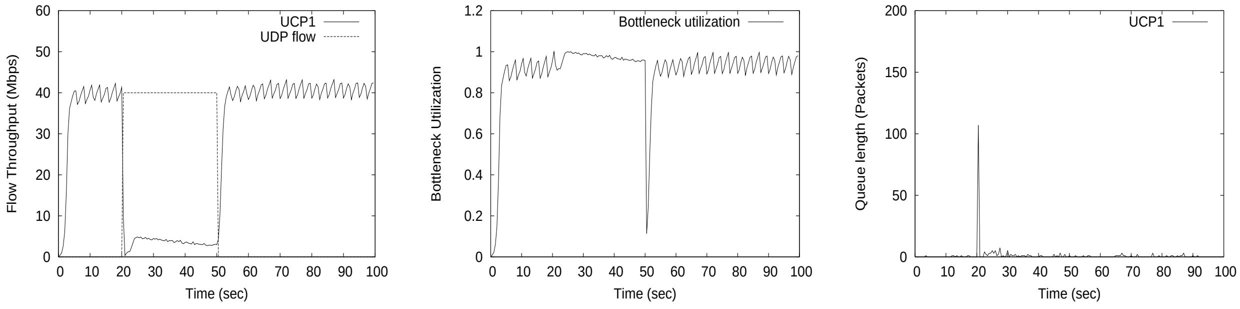

Fig. 7: Competing with a UDP flow

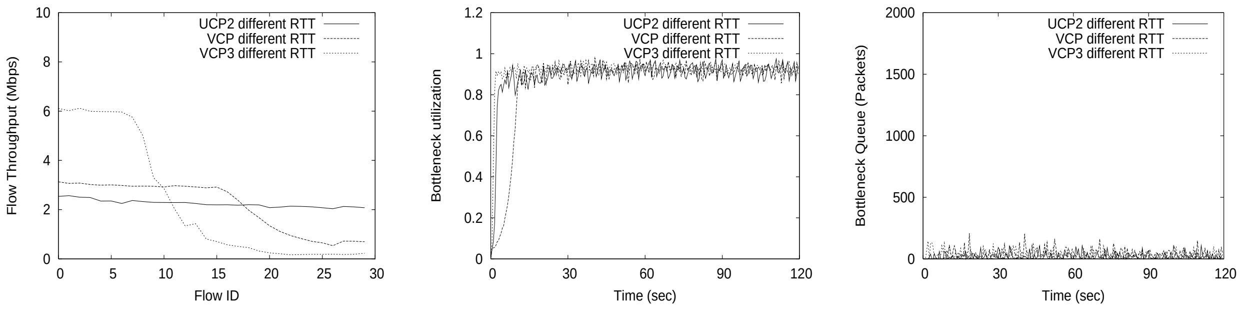

Fig. 8: Computing and transmitting average RTT improves fairness among flows with different RTTs.

and then signal senders to reduce their sending rates by marking packets (ECN) or implicitly, by dropping packets. In Randomly Early Detection (RED) [9], a router monitors its own queue length, and when it detects impending congestion it notifies the senders. However, it has been shown that congestion notification schemes such as TCP + AQM/ECN cannot achieve high utilization in high BDP networks.

Explicit feedback (load factor based) schemes. This group is best represented by XCP [11]. XCP senders transmit their current congestion window size and estimated RTT. The XCP router then estimates the fair rate for each flow and sends it back to the senders as explicit feedback. XCP performs very well, regardless of link capacity, round trip time (RTT), and number of flows. Like XCP, RCP [7] also uses explicit feedback from routers, but does not calculate a rate for each flow passing the router. Namely, RCP maintains only one common rate for all flows and aggressively gives bandwidth to new flows. Variable-structure congestion Control Protocol (VCP) [20] uses load factor (relative ratio of demand and capacity) as a signal of congestion in routers and sends 2-bit explicit feedback to a sender. VCP leverages the existing two ECN bits, whereas XCP uses multiple bits to encode the congestion-related information which can hinder its adoption. However, 2-bit information is not sufficient to ensure good convergence and avoid high-variance RTT issues. MLCP [14] can be considered as a compromise between the previous two schemes (XCP and VCP), as it uses 4-bit feedback and achieves near-optimal performance. In addition, it propagates the mean RTT of flows passing through routers (by using additional 3 bits in the IP header), thus enabling its controller to achieve better fairness. Finally, MLCP's controller employs an additional Inversely-proportional Increase

(II) phase to enable smooth rate variation. As the MLCP code is not yet available, we used a modified VCP controller to demonstrate the benefits of our framework, including the importance of collecting and transmitting the mean RTT at the routers. Our work is mostly orthogonal to these efforts, as UNO makes protocols in this category immediately deployable, with negligible impact on performance. UNO can also be leveraged by protocols such as MCP [13] for communicating load factor on the bottleneck link to improve VCP's performance. Additionally, MCP requires 2 more bits for enabling senders to urge all flows to operate in the fairing mode, which can be easily conveyed by UNO framework.

It has recently come to our attention that there have been proposals for **reducing the number of header bits that are required for new protocols and services**. For example, in [1] authors address the IP traceback problem using a probabilistic packet marking (PPM) approach that requires only 1 bit in the header. More recently, deterministic packet marking for congestion price estimation [18] makes use of the identification IP field along with the TTL field in order to calculate a partial sum of the path price. However, the authors are mainly concerned with estimating the error in determining price and never quantify the effect of such an error. Moreover, having only the sum of link prices along the path is not sufficient to enforce fairness among multiple flows. To address the problem of fairness, each source needs to know the maximum link price along a path which is not supported by the previous proposal. Adaptive Deterministic Packet Marking (ADPM) [3] paper shows how the identification IP field might be used to assist in conveying the binary representation of the price and allow the maximum price on a flow's path to be estimated. Unlike UNO, this work hashes the IP identification field. Deterministic Packet Marking for Max-Min Flow Control [16] also leverage the ECN bits and IPID field to convey the load factor information. In contrast to our framework, it uses both ECN bits to encode the load factor, thus it is not able to convey RTT-related information, and, it does not explore the impact of waiting for block of K IP packets. Here, the authors are only concerned with the error introduced by quantization techniques and packet loss. Relative to these efforts, our work makes several contributions. Most importantly, our framework provides a way to collect and transmit the RTT-related information to the controller, which has been shown to be necessary for enforcing fairness among flows with varying RTTs [14]. Further, we have implemented the framework and a sample controller in the ns2 simulator. Finally, we show the benefits of using our framework using extensive simulations.

The 16-bit identification field (IPID) has been used for purposes other than congestion control. For instance, Chen et al. [6] have successfully used it for inferring network path and end-system characteristics such as: i) the internal traffic generated by a server, ii) the number of servers in a large scale system used for load balancing, and iii) the difference between one-way delays of two machines from a client machine.

VI. Conclusions and Future Work

We present UNO, a framework for immediate deployment of explicit feedback congestion control schemes that are highly suitable for high BDP networks. In particular, we show that "a single ECN bit can be enough" to transmit fine-grain link load information and achieve good convergence in these environments. This protocol has a distinct advantage of being interoperable with existing ECN implementations. In addition, we demonstrate how RTT-related information can be transferred using the remaining allocated ECN bit for improved fairness.

As part of our future work, we intend to integrate UNO with MLCP and XCP to evaluate its performance with these controllers. Further, we intend to investigate the performance of UNO-enabled protocols in the presence of unmodified and existing ECN-enabled routers.

References

[1] M. Adler. Tradeoffs in probabilistic packet marking for ip traceback. In *In Proceedings of 34th ACM Symposium on Theory of Computing (STOC*, pages 407–418, 2002.

[2] T. Anderson, A. Collins, A. Krishnamurthy, and J. Zahorjan. Pcp: efficient endpoint congestion control. In *NSDI*, 2006.

[3] L. L. H. Andrew, S. V. Hanly, S. Chan, and T. Cui. Adaptive Deterministic Packet Marking. *IEEE Communication letters*, 2006.

[4] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson. Tcp vegas: new techniques for congestion detection and avoidance. *SIGCOMM Comput. Commun. Rev.*, 24(4):24–35, 1994.

[5] H. Bullot and R. L. Cottrell. Evaluation of Advanced TCP Stacks on Fast Long-Distance Production Networks. http://www.slac.stanford.edu/grp/csc/net/talk03/tcp-slac-nov03.pdf, 2003.

[6] W. Chen, Y. Huang, B. F. Ribeiro, K. Suh, H. Zhang, E. de Souza e Silva, J. Kurose, and D. Towsley. Exploiting the IPID field to infer network path and end-system characteristics. In *Proceeding of the 2005 Passive and Active Measurement (PAM'05) Workshop*, March 2005.

[7] I. Dukkipati and N. Mckeown. RCP-AC: Congestion control to make flows complete quickly in any environment (Extended Abstract). In *IEEE INFOCOM*, 2006.

[8] S. Floyd. Highspeed tcp for large congestion windows, internet-draft draft-floyd-tcp-highspeed-00.txt. 2002.

[9] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[10] J. He, M. Suchara, M. Bresler, J. Rexford, and M. Chiang. Rethinking internet traffic management: from multiple decompositions to a practical protocol. In *CoNEXT '07: Proceedings of the 2007 ACM CoNEXT conference*, pages 1–12, New York, NY, USA, 2007. ACM.

[11] D. Katabi, M. Handley, and C. Rohrs. Internet congestion control for high bandwidth-delay product networks. In *Proceedings of ACM SIGCOMM*, August 2002.

[12] T. Newsham and J. Hoaglan. Windows Vista Network Attack Surface Analysis: A Broad Overview. *CanSecWest*, 2007.

[13] M. Podlesny and S. Gorinsky. Mcp: Few bits for fairing and small queues in the stable state. In *ISCC*, pages 1079–1084. IEEE, 2007.

[14] I. Qazi and T. Znati. On the design of load factor based congestion control protocols for next-generation networks. *INFOCOM*, April 2008.

[15] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. IETF RFC 3168. 2001.

[16] H.-K. Ryu and S. Chong. Deterministic packet marking for max-min flow control. *IEEE Communication letters*, 2005.

[17] N. Spring, D. Wetherall, and D. Ely. Robust Explicit Congestion Notification (ECN) Signaling with Nonces. IETF RFC 3540. 2003.

[18] R. W. Thommes and M. J. Coates. Deterministic packet marking for congestion price estimation. In *IEEE INFOCOM*, 2004.

[19] D. X. Wei, C. Jin, S. H. Low, and S. Hegde. Fast tcp: motivation, architecture, algorithms, performance. *IEEE/ACM Trans. Netw.*, 14(6):1246–1259, 2006.

[20] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman. One more bit is enough. *SIGCOMM Comput. Commun. Rev.*, 35(4):37–48, 2005.