Implementation of the Network Control Protocol using ECN Bits (eNCP)

Debessay Fesehaye and Klara Nahrstedt
Dept. of Computer Science, University of Illinois at Urbana-Champaign, USA. {dkassa2,klara}@illinois.edu

***Abstract*—Many studies have shown that the transmission control protocol (TCP) which is the major transport protocol in the Internet today is finding it increasingly difficult to cope with the growth of communication network capacities and applications. TCP's inability to properly utilize network links is one of the problems. Besides, TCP takes a long time to achieve fairness between flows. Many of the new modification of TCP inherit these main problems of TCP.**

Clean-slate protocols such as the explicit Congestion control Protocol (XCP) which get congestion feedback from routers can fully utilize the links and reduce queueing delays in the path of the flows. But XCP, in addition to having many router computation overheads also takes many rounds to allocate fair shares to flows. To solve the drawbacks of XCP and TCP, we have previously presented a network control protocol (NCP). NCP allocates fair share to flows in one round resulting in increased average file completion time (AFCT) as short flows are not blocked by big file transfer flows. Although smaller than that of XCP, NCP uses a 32 bit additional header in every packet to carry a congestion feedback from the routers in the path of every flow to each source of the flow. Apart from the inconvenience of modifying the TCP/IP packet formats, the additional header may accumulate to cause some increase in AFCT.

In this paper we present an efficient implementation scheme (algorithm) of the Network Congestion Control (NCP) protocol using Explicit Congestion Notification (ECN) bits (eNCP). Using this implementation scheme, NCP does not need any additional packet header, avoiding the per packet overhead. In addition to the convenience of not having to change the TCP/IP packet header format, numerical results show that a significant amount of data overhead can be saved using eNCP. Not having to add a layer can also make NCP easily deployable and backward compatible.

I. Introduction

Transport protocols which control the rate of data transfer in communication networks play an important in avoiding network congestion. The transmission control protocol (TCP) is the major transport protocol in the Internet today. In spite of avoiding congestion in the early stages on the Internet, TCP is finding it increasingly difficult to cope with the growth of communication network capacities and applications as shown in the literature [1]. TCP's inability to properly utilize network links is one of the problems of TCP. Any delay or packet loss can cause TCP to dramatically reduce its congestion window and hence under-utilize the link. This is mainly because TCP doesn't have a direct way of finding the available capacity in the path of the flows. So it takes many rounds for TCP to find the correct rate at which flows need to send data. This means that flows take unnecessarily long time to finish. Besides, TCP takes long time to achieve fairness between flows. For instance flows which are sending at a higher rate and others sending at a lower rate are both told to reduce their sending rates (congestion window size) by half on event of congestion. This means that the flows which are sending at a higher rate keep sending at a higher rate. This results in long (big file transfer) flows starving short flows, which are the majority of flows in the Internet today [1].

To solve the problems of TCP, there have been some works in the literature. Many of them make some modification of TCP resulting in protocols which inherit the main problems of TCP presented above. On the other hand there are clean-slate protocols such as the explicit Congestion control Protocol (XCP) [6]. The design of XCP gets congestion feedback from routers to fully utilize the links and reduce queueing delays at the routers in the path of the flows. However XCP inherits some of the unfairness problems of TCP as it takes many rounds to allocate a fair share to flows. Besides, causes many router computation overheads.

Recently we presented a network control protocol (NCP) which avoid the weaknesses of TCP and XCP and which can generalize XCP. NCP can allocate fair share to flows in one round. This implies that big file transfer flows which have already opened their window size big and hence are sending at a higher rate do not cause short flows, the majority of the Internet flows to take longer than necessary to finish. NCP uses a 32 bit additional header in every packet to carry a congestion feedback from the routers in the path of every flow to each source of the flow. In addition to the requirement of a change in the TCP/IP header format, the additional NCP header may cause some overhead which can accumulate to cause unnecessary delays in file transfers.

In this short paper we present an implementation scheme (algorithm) of our Network Congestion Control (NCP) protocol using ECN bits and we call the resulting protocol eNCP. This implementation avoids the extra packet header overhead which NCP uses. To the best of our knowledge there is no such an implementation algorithm to NCP-like algorithms. As described by some exemplary numerical results, some data overhead can be avoided using eNCP in addition to the simplicity of the eNCP implementation.

The rest of the paper is organized in such a way that we first present the implementation scheme in Section II. We then present preliminary numerical results showing some gains of the new implementation scheme in Section III. Finally we give a brief summary and description of work in progress in Section IV.

II. eNCP: NCP Implementation Using ECN Bits

The Network Control Protocol (NCP) [3], [4] is a new cross-layer congestion control and routing protocol. As discussed in [3], [4], it can outperform the XCP [6] and other congestion

control protocols. The implementation of NCP needs an additional shim throughput (cwnd) header between the IP and TCP layers as shown in Figure 1. In this section we show

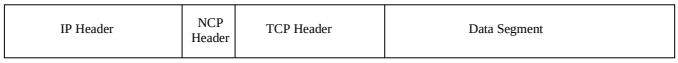

Fig. 1. Packet headers

an implementation of NCP using ECN bits of one or more than one packets as shown in [8], [9]. This helps NCP work without any additional packet header and makes it easier to implement.

A. Derivation of the Feedback

As shown in [3], [4] let's denote the link capacity, link buffer size, control interval and current (time t) sending rate of a flow carried in packet j of the L packets (per control interval) at a bottleneck link with $C, Q, d, R_j(t)$, respectively. The rate $R(t+d)$ by a router at time $t+d$ as derived in [3] is given by

$$R(t+d) = \frac{Cd - Q}{\sum_j^L (1/R_j(t)}). \tag{1}$$

As packet j carrying inter packet interval length $\ell_j = 1/R_j(t)$ of a flow crosses a router with the rate $R(t+d)$, it sets its ℓ_j as

$$\ell_j = \frac{1}{R_j(t)} := max\left(\frac{1}{R_j(t)}, \frac{1}{R(t+d)}\right). \tag{2}$$

In the next section we will show how the value of ℓ_j is encoded using ECN bits. The receiver copies the encoded ECN bit values of the data packets to their corresponding ACK packets and sends them back to the sender (source). The source currently sending at the rate of $R_j(t)$ sets its rate $R_j(t+d)$ for the next round to the new value $R_j(t)$ it receives from the ACK packets. It also calculates its congestion window size w_j using its round trip time (RTT) RTT_j, round as

$$w_j(t+d) = RTT_j R_j(t+d). \tag{3}$$

For our implementation of NCP using ECN bits, the feedback which has to be carried back from the network to sender j is $\ell_j = 1/R_j(t) = RTT_j/w_j(t)$ which is the inter-packet interval length carried by packet j of a the flows.

B. Encoding the Feedback

The feedback value ℓ_j can be encoded in the ECN bits of single packet like in the VCP [7], two packets like in [8] or even more packets like in the UNO [13] case. The schemes in [7], [8], [13] encode a load factor which gives some information to a TCP-like source to adjust its sending rate. However since the load factor alone doesn't give enough feedback to the source, such resulting load-based congestion control algorithms take a long time to converge to the desired sending rates. This results in increased AFCT.

In this section of the paper we present a scheme which encodes the actual rate at which sources send data to fully utilize the bottleneck links in their paths without causing significant queue at the routers. Even though using more packets to encode the rate gives more information to a source, the source may have to wait for the ACKs of all the packets which encode the rate to adjust its sending rate. In this report we demonstrate our encoding and implementation scheme using three packets. Our scheme can also take advantage of using more packets to encode the rate. This can be done by having the first few packets encode the most common (higher) rate (lower inter-packet interval) values.

Figure 2 shows the IPv4 header along with the ECN bits. Details of the ECN bit and other fields can be obtained in [10], [12]. The IP header is one of the headers a packet carries as shown in Figure 1. Taking each packet independently, the two ECN bits of each packet can encode 4 values (00, 01, 10, 11) where 00 encodes a default value of ℓ_j at the sender. The value of ℓ_j is such that $0 < \ell_j \leq \ell_u$ where ℓ_u is the upper bound of ℓ_j. We will discuss ℓ_u below.

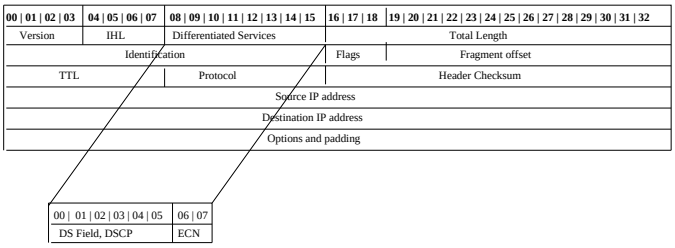

Fig. 2. IP Header (IPv4) with the ECN Bits

Each packet can encode three distinct values (other than the default value) independently. Three packets can then encode 9 values of ℓ_j. The default unmarked ECN value of all the packets can as well encode an additional value. So the three packets can encode a total of 9 distinct values. As the cwnd w_j and sending rate R_j of flows increase, the inter packet interval length shrinks. If the maximum round trip time (RTT) of a flow is $400ms$, and the cwnd is increasing exponentially as 16, 32, 64, 128, 256, ℓ_j can take the values $25, 12.5, 6.25, 3.125, 1.5625ms$. Our choice of 16 as a minimum congestion window size value is motivated by recent recommendations [2] of an increase of the initial cwnd by 10. The use of 400ms as a maximum RTT is motivated by some Internet measurement results [14] which shows that about $75\% \sim 90\%$ of flows have RTTs less than 200 ms. Another work on RTT measurements of TCP [11] shows that about 90% of flows in the experiment have an RTT of less than 400 ms. Here $\ell_u = 400/16 = 25ms$.

If the maximum cwnd is $256pkts$ and the slow start threshold is $128pkts$, then more ECN bits can be made to encode values smaller than $1.5625ms$. So by making some simple approximations, the three packets can be made to encode the intervals $(0, 0.25], (0.25, 0.5], (0.5, 0.75], (0.75, 1], (1, 1.5], (1.5, 3.0], (3, 6], (6, 13], (13, 25]$ in ms. Here the ECN bits of packet i can be made to encode the triplet i where the intervals are sorted in ascending order. The advantage of this type of encoding where packets with small IPID's encode

smaller intervals is that the very small interval ℓ_j indicating higher bandwidth in the path is quickly detected by the ACK of the first or second packet. If the value of ℓ_j is higher and is encoded in packet 3 then, the feedback is delayed by two packets. This delay can however be compensated by spreading the increase in inter-packet interval delay ℓ_j among the remaining packets the flow is sending in that same RTT. With more packets many more tighter intervals can be encoded for better accuracy and convergence of eNCP. For example 6 packets can encode 18 intervals. The intervals we used above are just examples. Other intervals and better mappings can be used based on some experimental or analytic results.

The eNCP encoding scheme is presented in Figure 3. The sender (source) and the routers in the path of the flow share the same mapping function $m(IPID, ECNvalue)$ which maps the ECN bits and IPID combination of the packets to an interval I of ℓ_j. The source uses the mapping function to encode the highest rate (lowest ℓ_j) which is in the range $(0, 0.25]$. Each router uses the inverse mapping function of the ECN bit value and IPID combination in the packets to decode the ℓ_j value in each packet to use in Equation 1. It then compares its ℓ_j value with the one it decoded from the packet. If its value is greater than the value in the packet, then it over-writes the value in the packet (Equation 2). So for the two routers in Figure 3, because their ℓ_j is greater than the one in packet they receive, they over-write the ECN bits of the packets according the mapping function.

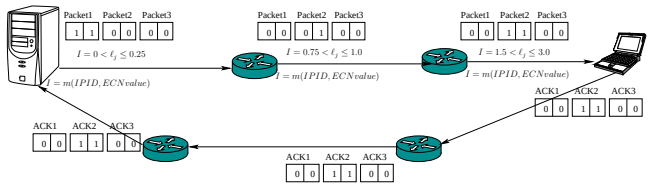

Fig. 3. The eNCP Encoding Scheme

C. Router Operations

When a source sends packets, it sets its ECN bits according to the encoding scheme discussed in Section II-B above. When a router which had computed its fair rate using Equation 1 receives a packet, it checks the packet IPID (IP identification field) and the ECN bit values. It then decodes the ℓ_j value using a mapping or a look-up table as discussed in the previous section. The router identifies the packets as either packet 1, packet 2 or packet 3 using a modulo 3 operation of the packet's IPID. If n packets are used for encoding the rate the router identifies each packet as its IPID modulo n. The router then updates the denominator sum component of Equation 1 as

$$S_m := S_m + \ell_j \tag{4}$$

and increments the number L_m of marked packets for the calculation of $R(t+d)$ of the next round if the ECN bits of the packet are set. If the ECN bits of the packet are not marked, the router just increases the count L of the total number of packets passing through the link by 1. The router also sets

$$\ell_j := max\left(\ell_j, \frac{1}{R(t+d)}\right). \tag{5}$$

If the updated ℓ_j cannot fit into packet j due to encoding mismatch, then it unmarks the ECN bits of packet j as it will encode the higher value of ℓ_j in the packets of the flow with higher IPID values that follow.

Now since $L_u = 2/3L \approx L - L_m$ packets are not marked (unmarked), the total denominator sum of Equation 1

$$S = \sum_j^L 1/R_j(t) = S_m + L_u \frac{S_m}{L_m}. \tag{6}$$

We can estimate the value of the control interval d as follows. First it has to be noted that

$$S = \sum_j^L 1/R_j(t) = \sum_j^L RTT_j/w_j(t) = \sum_i^N RTT_j$$

where N is the total number of concurrent (active) flows at a router. Initially we set the value of $d = 200ms$ and obtain the value $N = S/d$. For the subsequent control intervals we update d in terms N in one round and N in terms of d in the next round.

In our eNCP scheme, the senders, the receivers and the routers first agree on the encoding scheme and initialization parameters. An encoding scheme other than the one we use above and something like the once presented in [8], [13] can as well be used here.

D. Decoding the ECN Bits

The interval of ℓ_j encoded by the ECN bits of the packets needs to be decoded by the routers in the path of the flow and by the sender. Each router needs to decode ℓ_j for the calculation of the rate given by Equation 1. The routers and the sender decode the ECN bits of the packets to find the interval within which the value of ℓ_j lies. Once the routers and the sender get the interval, they perform binary search every RTT, to find the exact value of ℓ_j's. For instance if the ECN bits of packet 2 are set to 10 indicating the interval $(1, 1.5]$, then the sender first sets $\ell_j = (1 + 1.5)/2 = 1.25$. The sender then observes the value of ℓ_j it gets from the ACK packets after one RTT. If the interval is the same, the sender sets the $\ell_j = (1+1.25)/2$. This decrease in inter packet interval length ℓ_j implies an increase in the rate of the flow. If the interval increases to $(1.5, 3.0]$, the sender sets ℓ_j to $(1.25 + 1.5)/2$.

The router on the other hand monitors its queue size Q to perform the sort of binary search. A router decodes the ECN bits of each packet to get the intervals of the ℓ_j it needs for the calculation of the rate given by Equation 1. So if the interval decoded from the ECN bits of packet j is $(\ell_j^l, \ell_j^u]$, then the router first uses $\ell_j = \ell_j^a = (\ell_j^l + \ell_j^u)/2$ of each packet to obtain the rate. The router also records its queue length Q. If the queue length in the next round (control interval) at the router increases (compared with the previous queue length) by some

constant K_Q, the router uses $\ell_j = (\ell_j^a + \ell_j^u)/2$ of all packets for the calculation of the rate to increase the inter packet interval length and hence decrease the rate allocation. If the queue length decreases by some constant K_Q or remains the same, then the router uses $\ell_j = (\ell_j^l + \ell_j^a)/2$ for the calculation of the rate R. Such a distributed binary search helps the senders maximize their sending rates and the routers minimize their queue length and hence delay. Other interpretations of the interval can also be used. For instance the routers may set $\ell_j = \ell_j^u$ to obtain their rate, to be cautious in avoiding congestion (conservative) and vice-versa.

Our scheme tolerates packet losses as the information of the bottleneck link in the path of each flow is encoded in about one third of the packets a flow sends. The eNCP protocol resembles to the active queue management (AQM) protocols like RED (the random early detection) [5] in that both eNCP and AQM schemes mark the ECN bits to signal congestion. However AQM schemes do not encode the rate at which flows should send data to avoid congestion and to fully utilize link capacities.

E. Security Analysis of eNCP

To analyze the performance of eNCP under malicious attacks or ECN bit errors we consider the following threat model.

1) eNCP Threat Model: ECN Bit Flips: Under this threat model, an attacker flips the ECN bits of packets 1, 2 or 3. The worst threat is when an attacker flips the ECN bits of packet 3 of eNCP making the value of ℓ_j appear very large by for instance setting (marking) the two ECN bits of packet 3. This threat model can be dangerous as the source wrongly sets its inter-packet interval ℓ_j to be very long which also translates into a very small sending rate. This is because the source thinks that there is a very congested link in the path of its flows.

2) Remedy for the Threat Model (ECN Bit Flips): If an attacker flips the ECN bits of packet 2, the throughput of a flow is not degraded to the lowest possible value as packet 3 carries the lowest throughput (highest ℓ_j) values. To flip packet 3 an attacker has to check all incoming packets. This threat model is unlikely because the value of ℓ_j is encoded in one third of the total number of packets a flow sends. So flips of some packets can be detected and corrected by the other packets. This assumes that the attacker does not have enough resources to check and flip all of these packets.

III. Numerical Results

The original NCP needs a 32 bit field to record th ℓ_j values. So to transmit a file of size $\psi\, pkts$, the original NCP needs an overhead of $32\,\psi bits = 4\psi\, Bytes$. Figure 4 shows the byte over-head reduction by eNCP for a packet size of 1000 Bytes as a linear function of file size. As can be seen from the plot, the reduction in overhead increases as the file size increases.

Figure 5 shows the gain in download speed (in seconds) of using the new implementation algorithm of NCP (eNCP) over the the old implementation of NCP. As shown in the plot

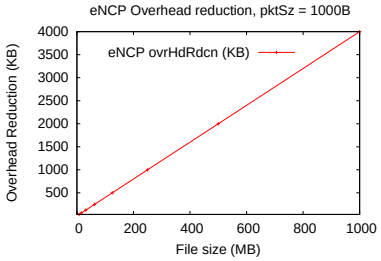

Fig. 4. Overhead reduction by eNCP

the gain in speed increases linearly as the file size increases. To transmit a file of size $\psi\, pkts$ NCP needs to transmit $\psi\, pkts + 32\psi\, bits = (10^3\psi + 4\psi)\, Bytes$ while eNCP needs to transmit only $10^3\psi\, Bytes$. Hence the download time of eNCP decreases accordingly.

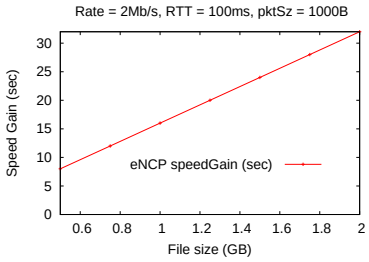

Fig. 5. Download speed gain of eNCP over NCP

IV. Summary and Ongoing Work

We have presented an efficient design of an implementation algorithm of a network congestion control protocol (NCP) using ECN bits (eNCP). Unlike the previous implementation algorithm of NCP and NCP-like algorithm, the design of eNCP encodes the rate information from the routers back to the sources without any additional packet header. This avoids the inconvenience of having to change the TCP/IP packet header format which new congestion control protocols like NCP require. Numerical results also show that some packet overhead can be saved by using this new implementation algorithm resulting in an increase in file download time. We are working on more simulation experiments to evaluate eNCP and compare its performance with other protocols. We will also implement eNCP in Linux end hosts and routers to prototype our design.

References

[1] Dukkipati, N., and McKeown, N. Why flow-completion time is the right metric for congestion control. *SIGCOMM Comput. Commun. Rev. 36* (January 2006), 59–62.

[2] Dukkipati, N., Refice, T., Cheng, Y., Chu, J., Herbert, T., Agarwal, A., Jain, A., and Sutin, N. An argument for increasing TCP's initial congestion window. *SIGCOMM Comput. Commun. Rev. 40* (June 2010), 26–33.

[3] Fesehaye, D. A Fast Congestion control Protocol (FCP) for Networks (the Internet). In *ITRE* (2006), pp. 131–135.

[4] Fesehaye, D., Nahrstedt, K., and Caesar, M. A Network Congestion control Protocol (NCP). In *CS/UIUC Technical Report* (Urbana, IL, USA, 2010).

[5] FLOYD, S., AND JACOBSON, V. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw. 1* (August 1993), 397–413.

[6] KATABI, D., HANDLEY, M., AND ROHRS, C. Congestion control for high bandwidth-delay product networks. *SIGCOMM Comput. Commun. Rev. 32* (August 2002), 89–102.

[7] LI, X., AND YOUSEFI'ZADEH, H. An implementation and experimental study of the variable-structure congestion control protocol (vcp). In *Military Communications Conference, 2007. MILCOM 2007. IEEE* (2007), pp. 1 –7.

[8] LI, X., AND YOUSEFI'ZADEH, H. DCP-EW: Distributed Congestion-Control Protocol for Encrypted Wireless Networks. In *WCNC* (2010), pp. 1–6.

[9] QAZI, I. A., AND ZNATI, T. On the design of load factor based congestion control protocols for next-generation networks. *Comput. Netw. 55* (January 2011), 45–60.

[10] RAMAKRISHNAN, K., FLOYD, S., AND BLACK, D. The Addition of Explicit Congestion Notification (ECN) to IP, 2001.

[11] SESSINI, P., AND MAHANTI, A. Observations on Round-Trip Times of TCP Connections. In *ZD Net* (2006).

[12] STEVENS, W. R. *TCP/IP illustrated (vol. 1): the protocols*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1993.

[13] VASIC, N., KUNTIMADDI, S., AND KOSTIC, D. One bit is enough: a framework for deploying explicit feedback congestion control protocols. In *Proceedings of the First international conference on COMmunication Systems And NETworks* (Piscataway, NJ, USA, 2009), COMSNETS'09, IEEE Press, pp. 503–511.

[14] XIA, Y., SUBRAMANIAN, L., STOICA, I., AND KALYANARAMAN, S. One more bit is enough. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2005), SIGCOMM '05, ACM, pp. 37–48.