Adaptive multi-level explicit congestion notification

A. Durresi*

Department of Computer and Information Science,
Indiana University Purdue University Indianapolis,
Indianapolis, IN 46202, USA
E-mail: durresi@cs.iupui.edu
*Corresponding author

M. Sridharan

Department of Computer Science and Engineering,
The Ohio State University,
Columbus, OH 43021, USA
E-mail: sridhara@cse.ohio-state.edu

R. Jain

Department of Computer Science and Engineering,
Washington University in St. Louis,
St. Louis, MO 63130, USA
E-mail: jain@cse.wustl.edu

Abstract: We propose Adaptive Multi-level ECN (AMECN), a new TCP congestion scheme, as an extension to Multi-level Explicit Congestion Notification (MECN). AMECN allows network operators to achieve high throughput and low delays. However, AMECN performance depends its parameter settings and the level of congestion, hence, no guarantees can be given about delay. To achieve a predictable average delay with AMECN, constant tuning of the parameters to adjust to current traffic conditions is needed. We show how to tune AMECN parameters. The analysis of our simulations shows that Adaptive MECN performs better than Adaptive RED.

Keywords: adaptive congestion management; multilevel ECN; TCP congestion control; QoS.

Reference to this paper should be made as follows: Durresi, A., Sridharan, M. and Jain, R. (2007) 'Adaptive multi-level explicit congestion notification', *Int. J. High Performance Computing and Networking*, Vol. 5, Nos. 1/2, pp.3–11.

Biographical notes: Arjan Durresi is an Associate Professor of Computer Science at Indiana University Purdue University Indianapolis. His current research interests include network architectures, heterogeneous wireless networks, security, QoS routing protocols, traffic management, optical and satellite networks, multimedia networking, performance testing and bioinformatics. He has authored more than 100 papers in refereed journals and international conference proceedings. He is on the editorial boards of *Ad Hoc Networks Journal* (Elsevier), *Journal of Ubiquitous Computing and Intelligence and Informatica – International Journal of Computing and Informatics*.

Mukundan Sridharan is currently a Research Assistant for the Dependable Distributed and Networked Systems Lab and is pursuing his PhD Degree in The Ohio State University. His research interest includes wireless and sensor networks, internet measurements, video conferencing and congestion control.

Raj Jain is a Professor of Computer Science and Engineering at Washington University in St. Louis. He is a fellow of IEEE, a fellow of ACM. He has 14 patents, more than 40 journal and magazine papers, and more than 60 conference papers. His papers have been widely referenced and he is known for his research on congestion control and avoidance, traffic modelling, performance analysis and error analysis.

1 Introduction

End-to-end congestion control schemes continue to be one of the main pillars for internet robustness, as shown by Floyd and Fall (1999). Nevertheless, congestion remains the main obstacle to QoS on the internet. Although a number of schemes have been proposed for network congestion control, the search for new schemes continues (Ramakrishnan and Floyd, 1999; Floyd and Jacobson, 1993; Clark and Fang, 1993; Feng et al., 1999; Kalyanaraman et al., 2000; Floyd and Fall, 1997, 1999; Mathis et al., 1996, 1997, 1997; Chiu and Jain, 1989; Floyd and Henderson, 1999; Ramakrishnan and Jain, 1990; Jain et al., 1994, 1994b; Athuraliya et al., 2001; Hollot et al., 2001; Katabi et al., 2002; Wang et al., 2004; Wei et al., 2006; King et al., 2005; Low et al., 2005; Wu and Rao, 2005). A survey of various congestion control schemes proposed for use in routers can be found in Low et al. (2002) and Medina et al. (2005).

The research in this area has been going on for at least two decades. There are two reasons for this. First, there are requirements for congestion control schemes that make it difficult to get a satisfactory solution. Second, there are several network policies that affect the design of a congestion scheme. Thus, a scheme developed for one network, traffic pattern, or service requirements may not work on another network, traffic pattern, or service requirements.

The proposed solutions expand over a wide spectrum of improvements. At one end of this spectrum there are simpler, more incremental and more easily employable changes to the current TCP. Examples of such proposed solutions are RED (Floyd and Jacobson, 1993) and Explicit Congestion Notification (ECN) (Ramakrishnan and Floyd, 1999). At the other end of the spectrum, there are solutions with more powerful changes that result in new transport protocols with higher performance but with less chance to be deployed in a large scale on the Internet at least in the immediate future. An example of such solution is XCP (Katabi et al., 2002). Other proposals, such as REM (Athuraliya et al., 2001), Proportional Integral Controller (Hollot et al., 2001), HighSpeed TCP (Floyd, 2003), Quick Start TCP (Floyd et al., 2006) reside along the simplicity-deployability spectrum. At the end the choice among all these solutions depends on the tradeoff between performance and practical use that will better fit the internet. Because of the size and multidimensional complexity of the internet, the robustness in heterogeneity is valued over efficiency of performance, which leads to favour evolution compared to revolution of changes. For this reason, in our solution we propose minimal changes to ECN and try to derive the maximum performance improvements out of them.

Among the congestion control schemes, the 'de facto' standard and the most used are the RED/ECN class of algorithms. In ECN, a bit in the IP header is set when the routers are congested. It is shown in Ramakrishnan et al. (2001) that ECN performs better than RED and it was made standard by IETF in 2001. ECN is much more powerful than the simple packet drop indication used by existing routers and is more suitable for high distance-bandwidth networks. Hence it becomes imperative that we explore the possibilities of utilising the ECN framework to the fullest. We proposed in Durresi et al. (2001) a new scheme called the MECN, which works with the framework of ECN, but uses the two bits allocated for ECN, in the IP to indicate four different levels of congestion, to the source. But just like RED (Floyd and Jacobson, 1993), MECN's average queue is also sensitive to parameter setting and the level of congestion. The average queuing delay is a very important parameter for QoS applications. Therefore, setting the parameters of MECN is very critical in maintaining a constant delay at the routers, which is a must to guarantee a given QoS to the end users. In this paper we propose an Adaptive version of MECN, which sets its parameters automatically and adapts its maximum marking probability to keep the average queuing delay constant. We compare the performance of Adaptive MECN (AMECN), to that of Adaptive Random Early Detection (ARED) and MECN and show that the first outperforms the other two schemes. In Section 2, we give a brief introduction to the MECN protocol. In Section 3, we introduce the Adaptive Multilevel ECN protocol and give some guidelines on setting the parameters. We prove using simulations using the *ns* Network Simulator (2007), that AMECN performs better than MECN and Adaptive RED in Section 4. In Section 5, we present the conclusions of our research.

2 Brief introduction to MECN

2.1 Marking bits at the router

The current proposal for ECN (Ramakrishnan et al., 2001) uses two bits in the IP header (bits 6 and 7 in the TOS octet in IPv4, or the Traffic class octet in IPv6) to indicate congestion. The first bit is called ECT (ECN-Capable Transport) bit. This bit is set to 1 in the packet by the traffic source if the source and receiver are ECN capable. The second bit is called the CE (congestion Experienced) bit. If the ECT bit is set in a packet, the router can set the CE bit in order to indicate congestion. The two bits specified for the purpose of ECN can be used more efficiently to indicate congestion, since using two bits we can indicate four different levels. If non ECN-capable packets are identified by the bit combination of '00', we have three other combinations to indicate three levels of congestion. In our scheme the bit combination '01' – indicates no congestion, '10' – indicates incipient congestion and '11' – indicates moderate congestion. Packet drop occurs only if there is severe congestion in the router and when the buffer over flows. So with packet-drop we have four different levels of congestion indication and appropriate action could be taken by the source TCP depending on the level of congestion. The four levels of congestion are summarised in Table 1. The marking of CE, ECT bits is done using a multilevel RED scheme. The RED scheme has been modified to include another threshold

called the mid_{th}, in addition to the min_{th} and max_{th}. If the size of the average queue is in between min_{th} and min_{th}, there is incipient congestion and the CE, ECT bits are marked as 10 with a maximum probability of P1max. If the average queue is in between mid_{th} and max_{th}, there is moderate congestion and the CE, ECT bits are marked as 11 with a maximum probability P2max. If the average queue is above the max_{th} all packets are dropped. The packet dropping policy of RED is shown in Figure 1. The modified packet marking/dropping policy of MECN (Durresi et al., 2001) is shown in Figure 2. We would like to stress that the major advantage of MECN compared to other congestion management schemes is that it conveys more accurate feedback information about the network congestion status than the current ECN. We have designed, as shown in Section 2.3, a TCP source reaction that takes advantage of the extra information provided about congestion. This is the reason why MECN responds better to congestion by allowing the system to reach faster the stability point, which results in better network performance as shown in later in our results in this paper.

Table 1 Router response to congestion: probabilistic marking of CE and ECT bits and packet dropping

Congestion state	*CEbit*	*ECTbit*
No congestion	0	1
Incipient congestion	1	0
Moderate congestion	1	1
Severe congestion	Packet	Drop

Figure 1 Probabilities of marking packets in RED

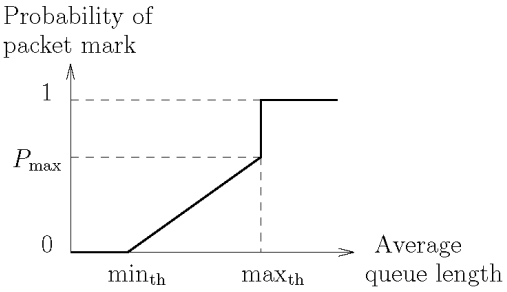
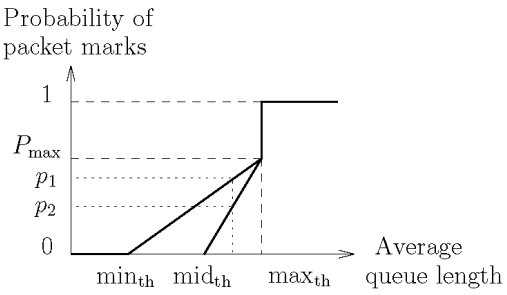


Figure 2 Probabilities of marking packets for MECN



2.2 Feedback from receiver to sender

The receiver reflects the bit marking in the IP header, to the TCP ACK. Since we have three levels of marking instead of two-level marking in the traditional ECN, we make use of three combination of the 2 bits 8, 9 (CWR, ECE) in the reserved field of the TCP header, which are specified for ECN. Right now the bit combination '00' indicates no congestion and '01' indicates congestion. And in piggybacked acknowledgements, '10' and '11' indicated non-congestion and congestion, with the receiver source indicating that the congestion window has been reduced. In our scheme, if the source has to indicate that the congestion window has been reduced then, the congestion information has to wait for the next packet. In this case the congestion information is ignored. But this will not cause any major problems to the scheme because, if the congestion is persistent then a lot of packets are going to get marked and the received source will eventually get the congestion information. So in the new scheme, '00' will indicate congestion window reduced, '01' will indicate no congestion, '10' will indicate mild congestion and '11' will indicate heavy congestion. The packet drop is recognised using traditional ways, by timeouts or duplicate ACKs. The marking in the ACKs CWR, ECE bits is shown in Table 2.

Table 2 End host reflecting congestion information: marking of CWR and ECE bits

Congestion state	*CWRbit*	*ECEbit*
Congestion window reduced	0	0
No congestion	0	1
Incipient congestion	1	0
Moderate congestion	1	1

2.3 Response of TCP source

We believe that the marking of ECN should not be treated as the same way as a packet drop, since ECN indicates just the starting of congestion and not actual congestion and the buffers still have space. And now with multiple levels of congestion feedback, the TCP's response needs to be refined. We have implemented the following scheme: When there is a packet-drop the cwnd is reduced by $\beta_3 = 50\%$. This done for two reasons: First, a packet-drop means severe congestion and buffer overflow and some severe actions need to be taken. Second, to maintain backward compatibility with routers which do not implement ECN. For other levels of congestion, such a drastic step as reducing the cwnd as half is not necessary and might make the flow less vigorous. When there is no congestion, the cwnd is allowed to grow additively as usual. When the marking is '10' (incipient congestion), cwnd is decreased by $\beta_1\%$. When the marking is 11 (moderate congestion) the cwnd is decreased multiplicatively not by a factor of 50% (as for a packet drop), but by a factor $\beta_2\%$ less than 50% but more than $\beta_1\%$. In Table 3 are shown the TCP source responses and the value of βs we have implemented. Another method could be to decrease additively the window, when the marking is '10' (incipient congestion), instead of maintaining the window.

Table 3 TCP source response

Congestion state	*CWND change*
No congestion	Increase 'cwnd' additively
Incipient congestion	Decrease by $\beta_1 = 20\%$
Moderate congestion	Decrease by $\beta_2 = 40\%$
Severe congestion	Decrease by $\beta_3 = 50\%$

If the average queue length is less than mid_{th}, then the modified-TCP congestion windows corresponding to the marks '10' keep increasing by 1 every round-trip time in congestion avoidance mode, thus linearly increasing the sending rates of these flows. Consequently, the average queue length will keep increasing unless some marks '11' are received by the sources, which correspond to operating in the region where the average queue length is larger than mid_{th}. We can thus conclude that the steady-state average queue length is larger than mid_{th}.

3 Adaptive MECN

3.1 Motivation

In Adaptive MECN, the objective is to maintian the queue near the *targetqueue*. If the average queue does not vary and remains constant at *targetqueue*, then the probability of packet drop/mark will remain fixed. Let this probability be P_{target}. We set the *targetqueue* to be in between min_{th} and mid_{th}. Hence only the first probability curve will be active, in this region. Hence the probability P_{target}, is given by equation (1):

$$P_{\text{target}} = \frac{P_{\max}}{\max_{th} - \min_{th}} \times (\text{Averagequeue} - \min_{th}). \quad (1)$$

Since in the above equation, P_{target}, min_{th}, max_{th} are all constant, we can say that,

$$Averagequeue \ \alpha \frac{1}{P_{\max}}. \quad (2)$$

In any network, we do not have the control over the traffic and the average queue increases or decreases with the load (as shown in Section 4.2). But the aim is to have the *Averagequeue*, always equal to the *targetqueue*. Hence if the *Avgqueue*, is greater than *targetqueue*, at any instant, we need to increase $P_{\max}$ which would decrease the *Avgqueue* so that it becomes equal to *targetqueue* and if the *Avgqueue*, is less than *targetqueue*, at any instant, we need to decrease $P_{\max}$, to allow the queue, to grow, which would give a better throughput. Thus to keep a constant queue we need to adapt the $P_{\max}$.

Also we need to set the other parameters like w_q, max_{th}, mid_{th} and min_{th} automatically.

The above discussion, leads us to the conclusion on the requirement of AMECN algorithm; Adapt $P_{\max}$ in response to measured queue lengths and set w_q, max_{th}, mid_{th} and min_{th} automatically, based on the link speed and target queue.

3.2 Algorithm

The overall Adaptive MECN, which was implemented has the following features:

- $P_{\max}$ *is* adapted to keep the average queue size with a target range half way between min_{th} and max_{th}.
- $P_{\max}$ is adapted slowly, over time scales greater than a typical round-trip time and in small steps. The time scale is generally 5–10 times the typical round- trip time of the network.
- $P_{\max}$ is constrained to remain with the range of [0.01, 0.5].
- Instead of multiplicately increasing and decreasing $P_{\max}$, we use Additive-Increase Multiplicative-Decrease (AIMD) policy.

The algorithm for Adaptive MECN is given in Figure 3.

Figure 3 The Adaptive MECN algorithm

Every $interval$ (0.5) seconds :
  if ($avg > target$ and $P_{max} <= 0.5$)
    increase P_{max}:
    $\alpha = 0.25 * \frac{avg - target}{target} * P_{max}$;
    $P_{max} = P_{max} + \alpha$;
  elseif ($avg < target$ and $P_{max} >= 0.01$)
    decrease P_{max}:
    $X = 0.17 * \frac{target}{target - min}$;
    $\beta = 1 - X * \frac{target - ave}{target}$;
    $P_{max} = P_{max} * \beta$;
Variables:
avg: average queue size
Fixed parameters:
$interval$: time; 0.5 seconds
$target$: target for avg;
  $[min_th + 0.4 * (max_th - min_th), min_th + 0.6 * (max_th - min_th)]$
α: increment; $0.25 * \frac{avg - target}{target} * P_{max}$
β: decrease factor; $1 - X * \frac{target - ave}{target}$
X: scaling factor; $0.17 * \frac{target}{target - min}$

The guideline of adapting $P_{\max}$ slowly and infrequently allows the dynamics of MECN – of adapting the packet-dropping probability in response to changes in the average queue size – to dominate on smaller time scales. The ad-pation of $P_{\max}$ is invoked only as needed over longer time scales. This time period is set as 0.5 seconds, which in comparable to RTT (around five times the RTT, since average RTT of terrestrial networks is approximately 100 ms).

The robustness of Adaptive MECN comes from its slow and infrequent adjustment of $P_{\max}$. The price of this slow modification is that after a sharp change in the level of congestion, it could take sometime, before $P_{\max}$ adapts to its value. But also adapting α and β makes this process faster and decreases the response time of the system.

Hence AMECN has better sensitivity than its RED counterpart 'Adaptive RED'.

3.3 Setting the parameters

3.3.1 The range for P_{max}

The upper bound of 0.5 on $P_{\max}$ can be justified because, when operating under the gentle mode, this would mean that the packet drop rate varies from 0 to $P_{\max}$, when average queue varies from $\min_{th}$ to $\max_{th}$ (or mid_{th} to) and varies from $P_{\max}$ to 1.0, if queue changes from to $2 \times \max_{th}$.

For scenarios with very small drop rates, MECN will perform fairly robustly with $P_{\max}$ set to the lower bound 0.01, and no one is likely to object to an average queue size less than the target range.

3.3.2 Parameters α and β

It takes $0.49/\alpha$ intervals for $P_{\max}$ to increas form 0.01 to 0.5; this is 24.5 seconds, if α is set as 0.01 (as recommended in Floyd et al. (2001)). Similarly, it takes at least $\log 0.02/\beta$ intervals for $P_{\max}$ to decrease form 0.5 to 0.01; with the default values, which is 20.1 seconds. Therefore if there is a sharp change in the router load, then it may take as long as 24.5 seconds for the average queue to reach the target range. This time is really a long time in network. Hence we believe that α and β should also be adapted, according to the position of the average queue, with respect to the target queue. So the value of α and β are also recalculated every 0.5 seconds when the $P_{\max}$ calculation is done. Taking the recommendation form (Floyd et al., 2001), that ($3 > 0.83$, we scale the value of β from 0.83–1.0 when average queue, varies from 0 to target queue Thus use the formula given below to adapt β.

$$\beta = 1 - (0.17 \times (\text{target} - \text{avg}) / (\text{target} - \min)). \tag{3}$$

Setting α again the recommendation form Floyd et al. (2001) are incorporated which says $\alpha < 0.25 \times P_{\max}$. So we scale α such that it varies from 0 to $0.25 \times P_{\max}$, when average queue varies form target to 0.

Thus formula we use to adapt α is

$$\alpha = 0.25 \times (0.17 \times (\text{avg} - \text{target}) / \text{target}) \times P_{\max}. \tag{4}$$

3.3.3 Setting mid_{th}, $\max_{th}$

To reduce the need for other parameter-tuning, we also give some guidelines for setting the mid_{th}, $\max_{th}$ and w_q. The $\max_{th}$ is set to three times the $\min_{th}$ as recommended in Floyd (1997). In this case the target average queue size is centred around $2 \times \min_{th}$. We believe that, the target queue should be kept in the low congestion region (i.e., between $\min_{th}$ and mid_{th}), to maximise the throughput, but at the same time the mid_{th} should not be too far from the *targetqueue*, so that when the average queue rises above target, a quick response to congestion is achived, when the second probability curve, comes into action. This belief, led us to setting the mid_{th} slightly above the *targetqueue*. Thus mid_{th} was set at $2.25 \times \min_{th}$ (*targetqueue* $= 2 \times \min_{th}$).

The guidelines for setting w_q given in Floyd and Jacobson (1993), are used. From Floyd and Jacobson (1993), if the queue size changes from one value to another it takes $-1/\ln(1 - w_q)$ packet arrivals for the average queue to reach 63% of the way to the new value. Thus we refer to $-1/\ln(1 - w_q)$ as the time constant of the estimator for the average queue size. Following the approaches in Jacobson et al. (1999) and Ziegler et al. (2001), in automatic mode we set w_q as a function of the link bandwidth. For MECN in automatic mode, we set w_q to give a time constant for the average queue size estimator of one second. Thus we set

$$\omega_{q_r} = 1 - \exp\left(\frac{-1}{C}\right) \tag{5}$$

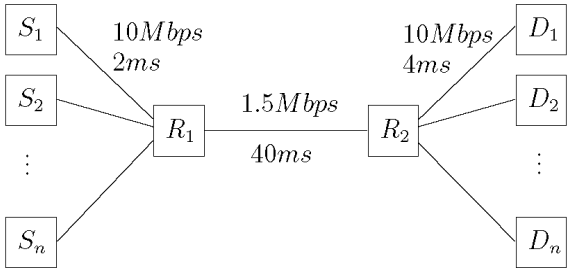
where C is the link capacity in packets/second, computed for packets of the specified default size.

4 Simulations and results

4.1 NS simulation configuration

This section illustrates the general simulation configuration we used for our simulations. Figure 4, shows the dumpbell configuration. A Number of sources $S_1, S_2, S_3, \ldots, S_n$ are connected to a router R_1 through 10 Mbps, d ms delay links. Router R_1 is connected to R_2 through a 1.5 Mbps, 40 ms delay link and a number of destinations $D_1, D_2, D_3, \ldots, D_n$ are connected to the router R_2 via 10 Mbps 4 ms delay links. The link speeds are chosen so that congestion will happen only between routers R_1 and R_2 where our scheme is tested. An FTP application runs on each source. Reno-TCP is used as the transport agent. (The modifications were made to the Reno-TCP.) The packet size is 1000 bytes and the acknowledgement size is 40 bytes. The number of sources is varied to alter the congestion level. The RTT of the flows can be varied by varying the delay d between the source and router R_1.

Figure 4 Dumb-bell network configuration for ns simulations



4.2 Illustrating MECN's varying queue size and AMECN's stability

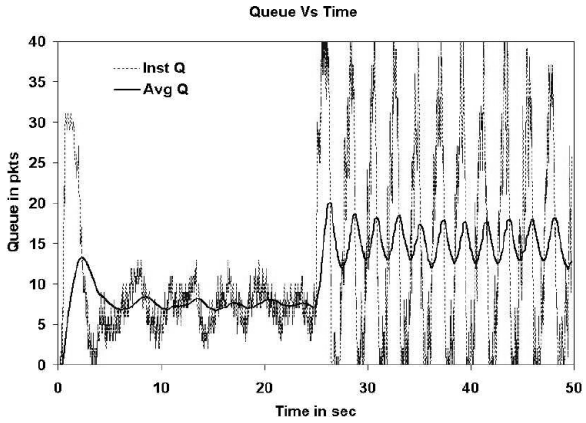
Here we investigate how MECN and Adaptive MECN respond to a rapid change in the congestion level. The simulations presented here illustrate MECN's dynamic

of the average queue size varying with the congestion level, resulting from MECN's fidex mapping from the average queue size to the packet dropping probability. For Adaptive MECN, these simulations focus on the transition period from one level of congestion to another.

These simulations use a simple dumbbell topology with a congested link of 1.5 Mbps. The buffer accomdates 40 packets. In all simulations w_q is set to 0.0027, $\min_{th}$ is set to 5 packets, mid_{th} is set to 10 packets and $\max_{th}$ is set to 15 packets.

For the simulation in Figure 5, the forward traffic consists of two long-lived TCP flows, and the reverse traffic consists of one long-lived TCP flow. At time 25, 20 new flows start, one every 0.1 seconds, each with a maximum window of 25 packets. This illustrate the effect of a sharp change in the congestion level. The graph in Figure 5 illustrates non-adaptive MECN, with the average queue size changing as a function of the packet drop rate. The dark line shows the average queue size as estimated by MECN, and the dotted line shows the instantaneous queue.

Figure 5 MECN with increase in congestion



The graph in Figure 6 shows the same simulation using Adaptive MECN. Adaptive MECN shows a similar sharp change in the average queue size at time 25. However, after roughly 15 seconds, Adaptive MECN has brought the average queue size back to the target range, between 9 and 12 packets. The simulation with Adaptive MECN shown in Figure 6, have a slightly higher throughput than the one with MECN shown in Figure 5 (96.3% instead of 94.5%), a slightly lower overall average queue size and a smaller packet drop rate. The simulations with Adaptive MECN illustrate that it is possible, but adapting $P_{\max}$, to control the relationship between the average queue size and the packet dropping probability and thus maintain a steady average queue size in the presence of traffic dynamics.

Figure 7 shows a similar simulation with 20 news flows starting at time 0 and stopping at time 25. The simulations with the MECN in Figure 7 shows the decrease in the average queue size as the level of congestion changes at time 25. Figure 8 shows the corresponding simulation for Adaptive MECN, which has a similar decrease in traffic at time 25, but with 15 seconds Adaptive MECN has brought the queue back to the target range. The simulation with Adaptive MECN shown in Figure 8, has a slightly higher throughput to that of MECN shown in Figure 7 (94.5% instead of 93.4%).

Figure 6 AMECN with increase in congestion

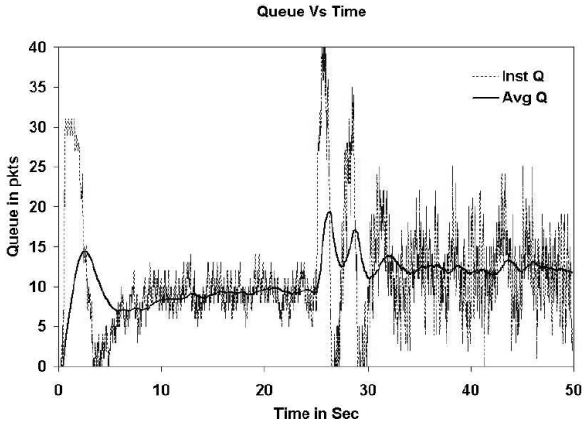


Figure 7 MECN with decrease in congestion

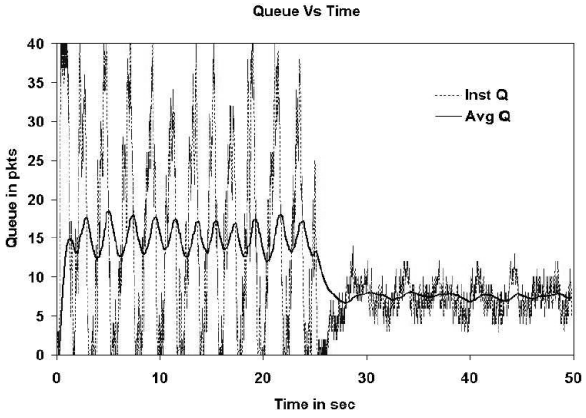
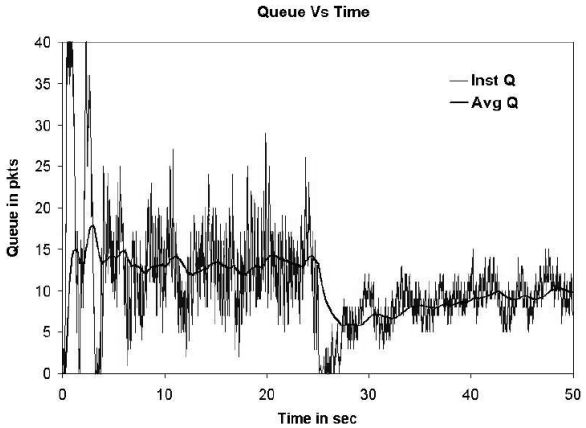


Figure 8 AMECN with decrease in congestion



4.3 Comparison with adaptive RED

4.3.1 Dumb-bell topology

The Adaptive MECN algorithm, is closely modelled after the Adaptive RED Floyd et al. (2001) algorithm and hence it become imperative that we compare the performance of AMECN with ARED. Adaptive RED, is the adaptive version of RED, where the $P_{\max}$ is adapted to keep the

average queue, with the target range. The difference between ARED and AMECN, is that in AMECN we use multiple level of congestion feedback and adapts also the parameters α and β, whereas in ARED we use binary congestion feedback and uses static α and β.

Figures 9 and 10 shows a set of simulations with a single congested link in a dumbbell topology shown in Figure 4, with 100 long-lived TCP flows. The flows have a RTT which varies from 100 ms to 150 ms and the simulations include web traffic and reverse path traffic. The congested link has a capacity of 7 Mb. Each point shown in the results is from a single simulation, with the *x*-axis showing the average queuing delay in packets over the second half of the 100-second simulation and the y-axis showing the link utilisation over the second half of the simulation. The simulations were carried out for both AMECN and ARED, for different target delays. Figure 9 shows the Link Efficiency vs. the Avergae Delay in the router, for both ARED and AMECN and Figure 10 shows the plot between the Target delays and the actual Measured Delay. We see that while both the schemes confirms very closely to the given target delay, AMECN gives better throughput for a given average delay. Hence AMECN gives higher throughput for a given targetdelay than ARED and a lesser delay for a given Link Efficiency.

Figure 9 Throughput vs. average delay for dumb-bell configuration

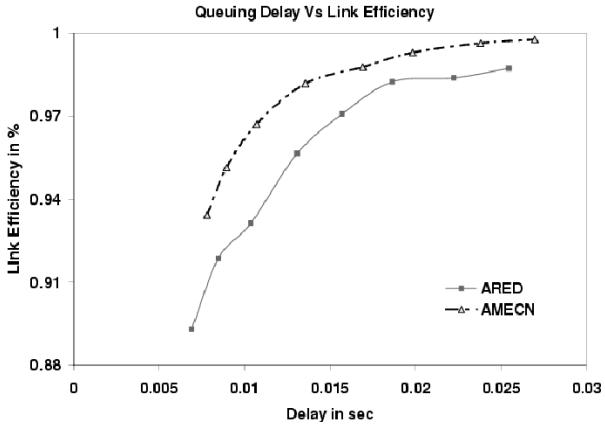
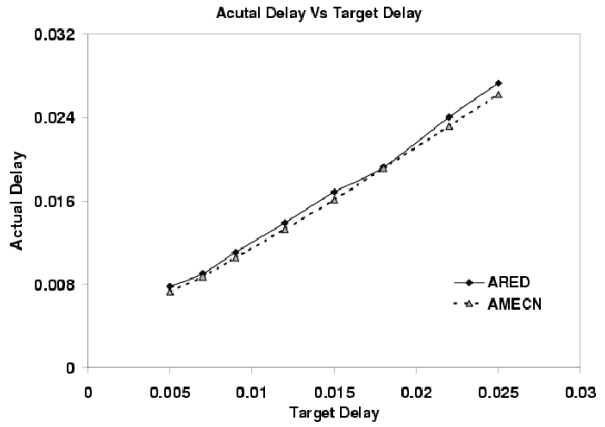


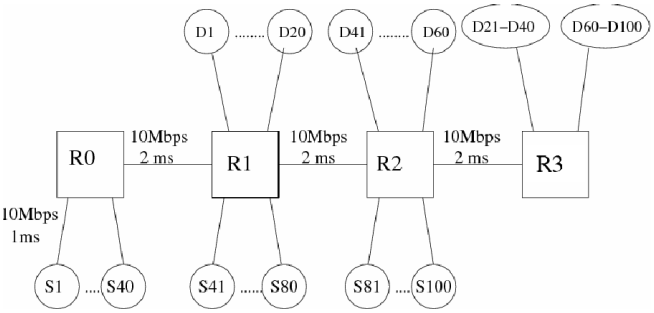
Figure 10 Measured delay vs. target delay for dumb-bell configuration



4.3.2 Multiple congested gateways

This simulation configuration is used to study the effect of the algorithm on Multiple Congested Gateways. The configuration is show in Figure 11. Its a typical parking lot configuration. Different flows in the network, travel for different lengths. There are four routers in the network, R_0–R_3. At routers R_0 and R_1 20 flows enter the network and leave at R_3. In addition 20 flows exist between each of these pairs of nodes R_0–R_1, R_1–R_2 and R_2–R_3. We intend to show that a system which uses AMECN on all routers has a better overall throughput than a system which uses ARED.

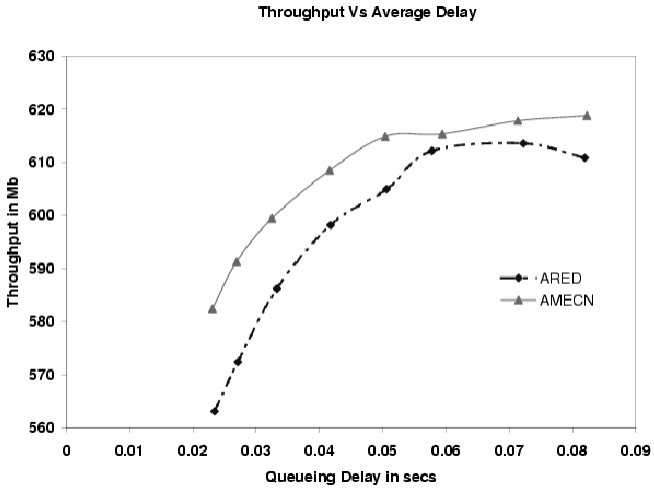
Figure 11 Simulation configuration for multiple congested gateways



The throughput is measured by measuring the throughput of all the individual flows and the then adding them up. The queuing delay is got by measuring the average queuing delay of each link over the simulation period and then summing up the queuing delay of the three links.

Figure 12 shows the results of a set of simulation, for target queues for both AMECN and ARED. The target queues were set same on all three links. The simulation was run for 100 secs and the results were averaged over the last 50 secs. As we can see the AMECN gives better overall throughput than ARED, even in the multiple congested case.

Figure 12 Throughput vs. average delay for multiple congested links



5 Conclusions and future work

In this paper, we presented the Adaptive MECN scheme, which adapts the MECN parameter $P_{\max}$ and automatically sets the MECN parameters w_q, mid_{th} and $\max_{th}$. The AMECN, maintains a buffer queue, which is set according to the delay requirements of the users. The choice of the target queue size, is a trade-off between the link utilisation and delay. We show with our simulations that AMECN has better delay and throughput performances than Adaptive RED. We are currently working on developing a control theory model for AMECN.

Acknowledgement

This research work has been partially supported by National Science Foundation Awards NSF-CNS-9980637 and NSF-CNS-0413187.

References

Athuraliya, S., Li, V.H., Low, S.H. and Yin Q. (2001) 'REM: active queue management', *IEEE Network*, May–June, Vol. 15, No. 3, pp.48–53.

Chiu, D. and Jain, R. (1989) 'Analysis of the increase/decrease algorithms for congestion avoidance in computer networks', *Journal of Computer Networks and ISDN Systems*, Vol. 17, No. 1, July, pp.1–14.

Clark, D.D. and Fang, W. (1993) 'Explicit allocation of best-effort packet delivery service', *IEEE/ACM Transactions on Networking*, Vol. 6, No. 4, pp.362–373.

Durresi, A., Sridharan, M., Liu, C., Goyal, M. and Jain R. (2001) 'Traffic management using multilevel explicit congestion notification', *Proceedings of the 5th World MultiConference on Systemics*, *Cybernetics and Informatics SCI'2001*, *ABR over the Internet*, July, pp.12–17.

Feng, W., Shin, K.S., Kandlur, D.D. and Saha, D. (2002) 'The BLUE active queue management algorithms', *IEEE/ACM Transactions on Networking*, Vol. 10, No. 4, pp.513–528.

Floyd, S. and Henderson, T. (1999) *The NewReno Modification to TCP's Fast Recovery Algorithm*, RFC 2582, April.

Floyd, S. and Jacobson, V. (1993) 'Random early detection gateways for congestion avoidance', *IEEE/ACM Transactions on Networking*, Vol. 1, No. 4, pp.397–413.

Floyd, S. (1997) *RED: Discussions of Setting Parameters*, http: //www. aciri.org/floyd/REDparameters.txt, November.

Floyd, S. (2003) 'HighSpeed TCP for large congestion windows', *ACM CCR*, RFC 3649, December.

Floyd, S. and Fall, S. (1993) 'Random early detection gateways for congestion avoidance', *IEEE/ACM Transactions on Networking*, Vol. 1, No. 4, pp.397–413.

Floyd, S. and Fall, S. (1997) *Router Mechanisms to Support End-to-end Congestion Control*, cite-seer.nj.nec.com/floyd97 router.html.

Floyd, S. and Fall, S. (1999) 'Promoting the use of end-to-end congestion control in the internet', *IEEE/ACM Transactions on Networking*, Vol. 7, No. 4, pp.458–472.

Floyd, S., Gummadi, R. and Shenker, S. (2001) *Adaptive RED: An Algorithm for Increasing the Robustness of RED*, http://citeseer.nj.nec.com/floyd01adaptive.html.

Floyd, S., Allman, M., Jain, A. and Sarolahti, P. (2006) 'Quick-Start for TCP and IP', *IETF INTERNET-DRAFT draft-ietf-tsvwg-quickstart-07.txt*, October.

Hollot, C.V., Misra, V., Towsley, D. and Gong, W. (2001) 'On designing improved controllers for AQM routers supporting TCP flows', *Proceedings of IEEE Infocom 2001*, Anchorage, Alaska, USA, April 22–26, Vol. 3, pp.1726–1734.

Jacobson, V., Nichols, K. and Poduri, K. (1999) *RED in a Different Light*, http://citeseer.nj.nec.com/j acobson99red. html.

Jain, R., Kalyanaraman, S. and Viswanathan, R. (1994) 'Rate based schemes: mistakes to avoid', *ATM Forum/ 94-0882*, September, http://www.cs.wustl.edu/_jain/atmf/ftp/ af9409-mistakes.txt.

Kalyanaraman, S., Jain, R., Fahmy, S., Goyal, R. and Vandalore, B. (2000) 'ERICA switch algorithm for ABR traffic management in ATM networks', *IEEE/ACM Transactions on Networking*, Vol. 8, No. 1, pp.87–98.

Katabi, D., Handley, M. and Rohrs C. (2002) 'Internet congestion control for future high bandwidth-delay product environments', *Proceeding of ACM SIGCOMM 2002*, Pittsburgh, PA, USA, August, pp.69–102.

King, R., Riedi, R. and Baraniuk, R. (2005) 'TCP-Africa: an adaptive and fair rapid increase rule for scalable TCP', *Proceeding of IEEE Infocom 2005*, Barcelona, Spain, Vol. 3, March 13–17, pp.1838–1848.

Low, S., Andreq, L. and Wydrowski, B. (2005) 'Understanding XCP: Equilibrium and Fairness', *Proceeding of IEEE Infocom 2005*, Barcelona, Spain, Vol. 2, March 13–17, pp.1025–1036.

Low, S.H., Paganini, F. and Doyle, J.C. (2002) 'Internet congestion control', *IEEE Control Systems Magazine*, Vol. 22, pp.28–43.

Mathis, M., Semke, J. and Mahdavi, J. (1997) 'The macroscopic behavior of the TCP congestion avoidance algorithm', *ACM SIGCOMM Computer Communications Review*, Vol. 27, No. 3, pp.67–82.

Mathis, M., Mahdavi, J., Floyd, S. and Romanow, A. (1996) 'TCP selective acknowledgement options', *RFC 2018*, October, http://www.ietf.org/rfc/rfc2018.txt.

Medina, A., Allman, M. and Floyd, S. (2005) 'Measuring the evolution of transport protocols in the internet', *A CM CCR*, Vol. 35, No. 2, April, pp.37–52.

Network Simulator (2007) *NS-2 Network Simulator*, http://www.isi.edu/nsnam/ns/.

Ramakrishnan, K. and Floyd, S. (1999) 'A proposal to add explicit congestion notification (ECN) to IP', *RFC 2481*, January, http://www.faqs.org/rfcs/rfc2481.html.

Ramakrishnan, K., Floyd, S. and Black, D. (2001) 'A proposal to add explicit congestion notification (ECN) to IP', *IETF RFC 3168*, September, http://www.faqs.org/rfcs/rfc3168.html.

Ramakrishnan, K.K. and Jain, R. (1990) 'A binary feedback scheme for congestion avoidance in computer networks', *ACM Transcations on Computer Systems*, Vol. 8, No. 2, May, pp.158–181.

Wang, C., Li, B., Hou, Y.T., Sohraby, K. and Lin, Y. (2004) 'LRED: a robust active queue management scheme based on packet loss ratio', *Proceedings of IEEE Infocom 2004*, Hong Kong, March 7–11, Vol. 1, pp.1–12.

Wei, D.X., Jin, C., Low, S.H. and Hegde, S. (2006) 'FAST TCP: motivation, architecture, algorithms, performance', *IEEE/ACM Transactions on Networking*, Vol. 14, No. 6, pp.1246–1259.

Wu, Q. and Rao, N.S.V. (2005) 'A class of reliable UDP-based transport protocols based on stochastic approximation', *Proceeding of IEEE Infocom 2005*, Barcelona, Spain, Vol. 2, March 13–17, pp.1013–1024.

Ziegler, T., Fdida, S. and Brandauer, C. (2001) *Stability Criteria for RED with Bulk-data TCP Traffic*, http://www-rp.lip6.fr/sf/WebSF/PapersWeb/red.net2000.pdf.