Adaptive Multi-level Explicit Congestion Notification

Mukundan Sridharan, Arjan Durresi, Raj Jain
Dept. of Computer and Information Science,

Raj Jain is now at Washington University in Saint Louis,
jain@cse.wustl.edu http://www.cse.wustl.edu/~jain/

***Abstract*— In this paper we extend Multi-level ECN, a new TCP congestion scheme, which we have proposed previously. The Multi-level Explicit Congestion Notification (MECN) algorithm allows network operators to achieve high throughput with corresponding low delays. But MECN average queue, is sensitive to its parameter settings and its the level of congestion, hence no guarantees can be given about delay. Delay being a major component of the quality of service network operators would naturally like to have a rough estimate of the average delays in their congested routers. To achieve a predictable average delays with MECN would require constant tuning of the parameters to adjust to current traffic conditions. The goal of this paper is to solve the parameter tuning problem of the MECN. We compare the performance of the Adaptive MECN system with the Adaptive RED system using simulations,using ns-2 simulator. Based on simulations we find that Adaptive MECN performs better than Adaptive RED.**

I. INTRODUCTION

End-to-end congestion control schemes continues to be one of the main pilasters in the robustness of the Internet [4]. Congestion remains the main obstacle to Quality of Serivce (QoS) on the Internet. Although a number of schemes have ben proposed for network congestion control, the search for new schemes continues. [5] gives a survey of different congestion contol schemes. But the winner for the time being seems to be RED/ECN class of algoriths and ECN was make a standard by the IETF in 2001 [6]. Hence it becomes imperative that we explore the possibilties of utilising the ECN framework to the fullest. In [1] proposed a new scheme called the Multi-level Explicit Congestion Notification (MECN), which works with the frame work 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 [7], MECN's average queue is also sensitive to parameter setting and the level of congestion. This average queuing delay is a very importent for QoS applications. So setting the parameters of MECN is very importent and maintain a constant delay at the routers, is a must, to give any QoS guareentees to the end users. In this paper we propose a Adpative version of MECN, which sets its parameters automatically and adapts its maximum marking probability to main a constant queueing delay. We compare the performace of AMECN, with ARED and MECN and show that it performace better than both the schemes. In Section II, we give a brief introduction to the MECN protocol. In Section III, we introduce the Adaptive Multilevel ECN protcol and give some guidelines on setting the parameters. We prove using simulations using the ns [8] simulator that AMECN performs better than MECN and Adaptive RED in section IV. In Section V, we present the conclusions of our research.

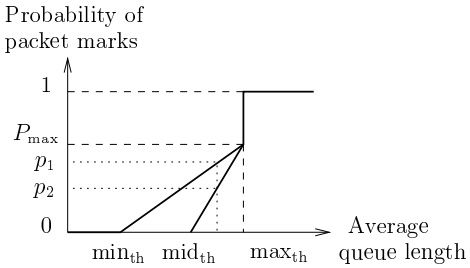

Fig. 1. Probabilities of marking packets for the new scheme

II. BRIEF INTRODUCTION TO MECN

A. Marking bits at the router

MECN [1] uses the two bits that is being specified for the use of ECN [6], in the IP header (bits 6 and 7 in the TOS octet in Ipv4, or the Traffic class octet in Ipv6), to indicate four different levels of congestion, instead of the binary feedback provided by ECN. The non ECN-capable packets are identified by '00', and it uses the other combinations to indicate three different levels of congestion and with packet-drop, four different levels of congestion is indicated and appropriate action could be taken by the source TCP depending on the level of congestion. The MECN packet marking/dropping policy is shown in Figure 1. If the size of the average queue is in between $\min_{\text{th}}$ and $\min_{\text{th}}$, there is incipient congestion and the ECN bits are marked as '10' with a probability p_1. If the average queue is in between mid_{th} and $\max_{\text{th}}$, there is moderate congestion and the ECN bits are marked as '11' with a probability p_2. If the average queue is above the maxthresh all packets are marked.

B. 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 2-level marking in the traditional ECN, we make use of 3 combination of the 2 bits 8, 9 in the reserved field of the TCP header and the other combination used by the source has to indicate that the congestion window reduced.

Proceedings of 2004 Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'04), San Jose, CA, July 25-29, 2004

Raj Jain is now at Washington University in Saint Louis, jain@wustl.edu

C. Response of TCP Source

The MECN source reaction can be summarised as:

- 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 don t 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 β_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 β_2% less than 50% but more than 1.

III. Adaptive MECN

A. Motivation

In Adaptive MECN, the objective is to maintian the queue near the $targetqueue$. If the average queue doesnt 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,

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

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

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

In any network, we donot have the control over the traffic and the average queue increases or decreases with the load (as shown in Section IV-B). 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.

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; $0.17 * \frac{target}{target - min}$

Fig. 2. The Adaptive MECN algorithm

B. 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 minth and maxth.
- 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 a n additive-increase multiplicative-decrease (AIMD) policy.

The algorithm for Adaptive MECN is given in Figure 2.

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 adpation 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 5 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'.

C. Setting the Parameters

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 max_{th}) and varies from P_{max} to 1.0, if queue changes from max_{th} to 2* 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.

2) Parameters α and β: It takes 0.49/α 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 [2]). Similarly, it takes at least log 0.02/β 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 beleive 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 [2], that $\beta > 0.83$, we scale the value of β from 0.83 to 1.0 when average queue, varies from 0 to target queue

Thus use the formula given below to adapt β.

$$\beta = 1 - (0.17 * (target - avg)/(target - min)) \quad (3)$$

Setting α again the recommendation form [2] are incorporated which says $\alpha < 0.25$ * P_{max}. So we scale α such that it varies from 0 to 0.25 * P_{max}, when average queue varies form target to 0.

Thus formula we use to adapt α is

$$\alpha = 0.25 * ((avg - target)/target) * P_{max} \quad (4)$$

3) Setting mid_{th}, max_{th} and w_q: 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 [9]. In this case the target average queue size is centered around 2 * min_{th}. We beleive that, the target queue should be kept in the low congestion region (i.e between min_{th} and mid_{th}), to maximize 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 * min_{th} ($targetqueue$ = 2 * min_{th}).

The guidelines for setting w_q given in [7], are used. From [7], 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 [10], [11], 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

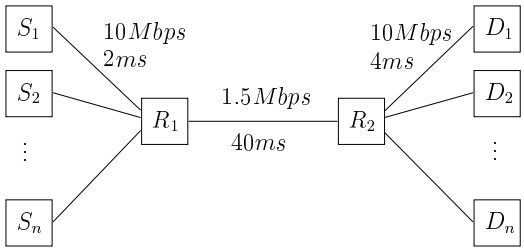

Fig. 3. Dumb-bell Network configuration for ns simulations

$$w_q = 1 - exp(\frac{-1}{C}) \quad (5)$$

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

IV. Simulations And Results

A. NS Simulation Configuration

This section illustrates the general simulation configuration we used for our simulations. Figure 3, shows the dumpbell configuration. A Number of sources S_1, S_2, S_3,..., S_n are connected to a router R_1 through 10Mbps, d ms delay links. Router R_1 is connected to R_2 through a 1.5 Mbps, 40ms delay link and a number of destinations D_1, D_2, D_3,..., D_n are connected to the router R_2 via 10Mbps 4ms 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.

B. 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 1.5Mbps. The buffer accomdates 40 packets. In all simulations w_q is set to 0.0027, minth is set to 5 packets, midth is set to 10 packets and maxth is set to 15 packets.

For the simulation in Figure 4, 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 4 illustrates non-adaptive MECN, with the average queue size changing as a function of the packet drop rate. The dark line shows the

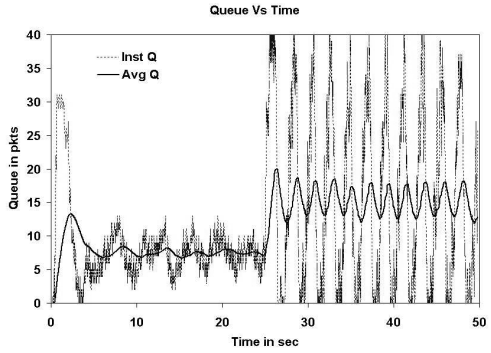

Fig. 4. MECN with increase in congestion

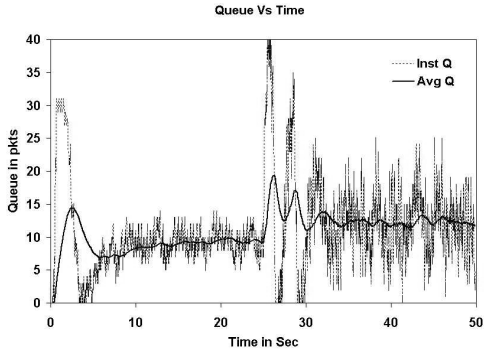

Fig. 5. AMECN with increase in congestion

average queue size as estimated by MECN, and the dotted line shows the instantaneous queue.

The graph in Figure 5 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 targer range, between 9 and 12 packets. The simulation with Adaptive MECN shown in Figure 5, have a slightly highter throughput than the one with MECN shown in Figure 4(96.3% instaed 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,

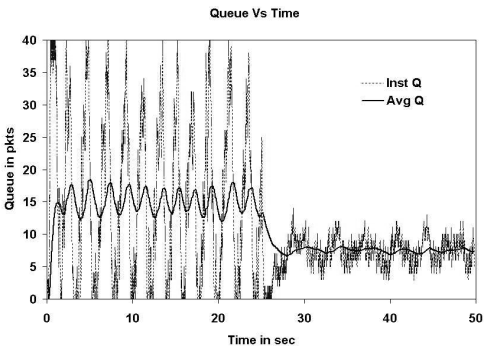

Fig. 6. MECN with decrease in congestion

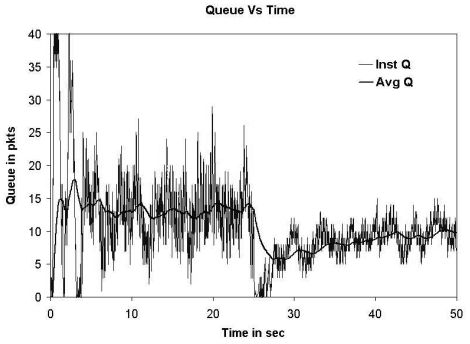

Fig. 7. AMECN with decrease in congestion

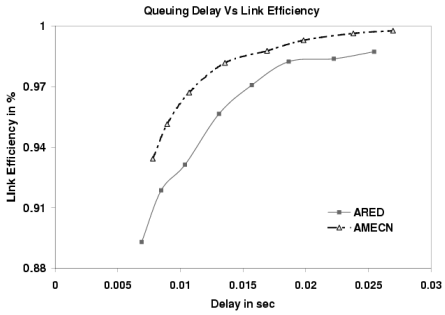

Fig. 8. Throughput Vs Average Delay for Dumb-bell Configuration

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 6 shows a similar simulation with 20 news flows starting at time 0 and stoping at time 25. The simulations with the MECN in Figure 6 shows the decrease in the average queue size as the level of congestion chagnes at time 25. Figure 7 shows the corresponding simulatin 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 simulatin with Adaptive MECN shown in Figure 7, has a slightly higher throughput to that of MECN shown in Figure 6(94.5% instead of 93.4%).

C. Comparison with Adaptive RED

1) Dumb-bell topology: The Adaptive MECN algorithm, is closely modelled after the Adaptive RED [2] 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 8 and 9 shows a set of simulations with a single congested link in a dumbbell topology shown in Figure 3, with 100 long-lived TCP flows. The flows have a RTT which

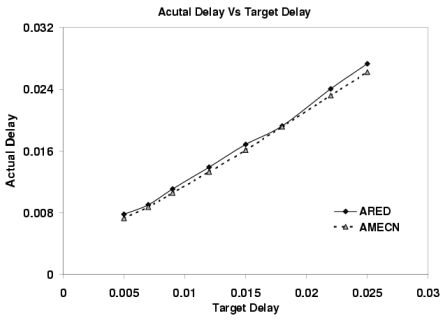

Fig. 9. Measured Delay Vs Target Delay for Dumb-bell Configuration

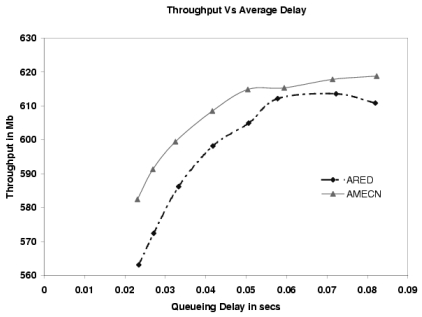

Fig. 11. Throughput Vs Average Delay For Multiple Congested Links

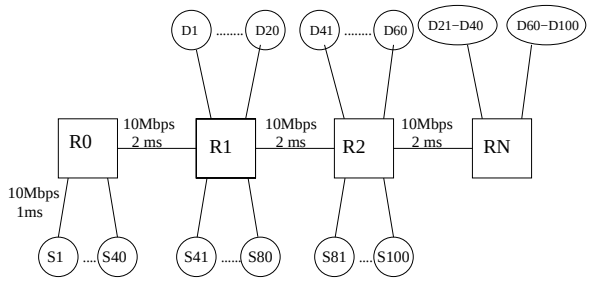

Fig. 10. Simulation Configuration for Multiple Congested Gateways

varies from 100ms to 150ms and the simulations include web traffic and reverse path traffic. The congested link has a capacity of 7Mb. Each point shown in the results is from a single simulation, with the x-axis showing the average queueing delay in packets over the second half of the 100-second simulation and the y-axis showing the link utilization over the second half of the simulation. The simulations were carried out for both AMECN and ARED, for different target delays. Figure 8 shows the Link Efficiency Vs the Avergae Delay in the router, for both ARED and AMECN and Figure 9 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 averge delay. Hence AMECN gives higher throughput for a given targetdelay than ARED and a lesser delay for a given Link Efficiency.

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 10. Its a typical parking lot configuration. Different flows in the network, travel for different lengths. There are 4 routers in the network, R_0 to R_3. At routers R_0 and R_1 20 flows enter the network and leave at R_3. In addition 20 flows exsist 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.

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 suming up the queueing delay of the 3 links.

Figure 11 shows the results of a set of simulation, for target queues for both AMECN and ARED. The target queues were set same on all 3 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.

V. Conclusions And Future Work

In this paper, we presented the Adaptive Multi-LevelExplicit Congestion Notification 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 utilization and delay. We show using simulations that AMECN has better delay and throughput performances than Adaptive RED. We are currently working on developing a control theory model for AMECN.

References

[1] A. Durresi, M. Sridharan, C. Liu, M. Goyal, and R. jain, "Traffic management using multilevel explicit congestion notification," in *Proc. of the 5th World MultiConference on Systemics, Cybernetics and Informatics SCI'2001, ABR over the Internet*, Orlando, FL, July 22-25 2001, pp. 12–17.

[2] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive red: An algorithm for increasing the robustness of red," 2001. [Online]. Available: citeseer.nj.nec.com/floyd01adaptive.html

[3] W. Feng, D. Kandlur, D. Saha, and K. Shin, "A self-configuring RED gateway," in *Proceedings of INFOCOM 99*, vol. 3, 1999, pp. 1320–1328. [Online]. Available: citeseer.nj.nec.com/feng99selfconfiguring.html

[4] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, 1999. [Online]. Available: citeseer.nj.nec.com/article/floyd99promoting.html

[5] A. Haider, H. Sirisena, K. Pawlikowski, and M. JFerguson, "Congestion control algorithms in high speed telecommunication networks." [Online]. Available: www.mang.canterbury.ac.nz/orsnz/conf2001/papers/Haider.pdf

[6] K. Ramakrishnan, S. Floyd, and D. Black, "A proposal to add explicit congestion notification (ECN) to IP," RFC 3168, September 2001.

[7] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993.

[8] "ns network simulator," http://www.isi.edu/nsnam/ns/.

[9] S. Floyd, "Red: Discussions of setting parameters," http://www.aciri.org/floyd/REDparameters.txt, November 1997.

[10] V. Jacobson, K. Nichols, and K. Poduri, "Red in a different light," 1999. [Online]. Available: citeseer.nj.nec.com/jacobson99red.html

[11] T. Ziegler, S. Fdida, and C. Brandauer, "Stability criteria for red with bulk-data tcp traffic," Technical report, August 2001. [Online]. Available: http://www-rp.lip6.fr/ sf/WebSF/PapersWeb/red.net2000.pdf