Robust Explicit Congestion Controller Design
For High Bandwidth-Delay Product Network: A H_\infty Approach

Yang Hong and Oliver W.W. Yang
SITE, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5
E-mail: {yhong, yang}@site.uottawa.ca

Abstract—TCP becomes inefficient and prone to instability when its bandwidth-delay product increases. By sending explicit congestion information from routers to end hosts, both XCP and API-RCP have been proposed to solve the stability issue of TCP in high bandwidth-delay product networks. Since the estimation errors of the network parameters are unavoidable, we would like to design a robust controller to address this type of model uncertainty in the explicit congestion control system. We employ the H_\infty optimal criterion in our design, and select a proper weight function to solve the H_\infty sensitivity problem. By using the maximum modulus theorem from the robust control theory, we obtain an optimal internal model controller from which a robust H_\infty controller can be derived. OPNET simulations demonstrate that our H_\infty controller exhibits a good robustness to varying network parameters. Performance comparison between our robust explicit congestion controller and TCP/RED is provided.

Index Items—Explicit Congestion Control, Model Uncertainty, Robust Control, H_\infty optimal criterion, TCP/RED

1. INTRODUCTION
TCP (Transmission Control Protocol) uses AIMD (Additive Increase Multiplicative Decrease) control mechanism to prevent the Internet from congestion collapse. Since TCP simply treats a packet loss as network congestion, this treatment is incorrect when the packet loss is indeed due to wireless effects (e.g., channel fading) [1]. As the network bandwidth-delay product increases, TCP often leads to significant throughput fluctuation and arbitrarily low link utilization [2]. XCP (Xplicit Congestion Control Protocol) [1] was proposed to overcome the limitations of TCP by explicitly feedback the network congestion information [3]. By calculating an advertised source sending rate based on link capacity and router queue size, XCP attempts to achieve high link utilization, while maintaining a small queue size in a high bandwidth-delay product network. Unfortunately, two potential problems of XCP have been received much attention recently: (1) A poor choice of XCP parameters may cause flows to obtain an arbitrarily small fraction of their max-min fair allocations in a multiple-bottleneck network [2,4]. MaxNet [5] and JetMax [6] are some enhancement that have been made to achieve max-min fairness; (2) The wireless capacity estimation error may introduce a large steady-state error which perpetuates the oscillations in XCP flow throughput [2,3,7,8]. The method proposed in QFCP (Quick Flow Control Protocol) [9] is one effort to reduce the capacity estimation error when the wireless network exhibits a high link utilization.

To avoid the two potential XCP problems above, the API-RCP (Adaptive Proportional-Integral Rate Control Protocol) [10] calculates an advertised source sending rate (carried in each data packet) based on the router queue size only [4,8,11,12]. Every packet carries a field for the lowest rate along the path to prevent the whole network from congestion. Like XCP, the original API-RCP had applied the Nyquist stability criterion [13] to guarantee system stability. API-RCP obtains its PI rate controller parameters according to estimated network parameters (i.e., the active flow number and the average RTT (Round Trip Time) of the passing flows). Since the unavoidable estimation error also introduces the network model uncertainty to PI rate controller design, it will be very useful if we can design a robust explicit congestion controller to minimize the system sensitivity to model uncertainty. There are various techniques to address the model uncertainty. For example, the H_\infty approach [14] appears appealing as it works on a principal different from the classical control (e.g., PI-controller designed by Nyquist stability criterion [10]), and it exhibits more robustness to bounded network traffic changes.

The main contributions of this paper are: (1) employing robust control theory to model the uncertainty of the explicit congestion control system; (2) using H_\infty optimal criterion to design a robust controller and selecting a proper weight function to solve H_\infty sensitivity problem; (3) investigating the robustness of H_\infty controller to a bounded traffic change; (4) making performance comparison between robust H_\infty explicit congestion controller and TCP/RED.

![Figure 1: System Model of an IP Router with Explicit Congestion Control](image)

2. EXPLICIT CONGESTION CONTROL OVERVIEW

Fig. 1 depicts the system model of an IP router. Every IP router in the network participates in Internet congestion control by using an explicit congestion controller such as the H_\infty controller proposed in this investigation or an implicit congestion control such as the TCP/RED [15]. Each source sends its packets, with a controlled rate of q_i(t), to its destinations through a series of IP routers in the Internet.

An IP router also processes the packets from an uncontrolled traffic with an arrival rate of \lambda(t). Then the queueing dynamic of explicit congestion controlled router can be described as follows [10],

\[x(t) = \sum_{i=1}^{N} q_i(t) \tau_i(t) + u(t) - \mu(t) ,\]

where \tau_i is the varying forward time delay from the controlled source node i to the router. N is the number of active long-lived controlled flows. The parameter \mu is the link capacity (or the service rate) of the router, which may vary in the wireless networks [2,3,7,8], according to Shannon’s Capacity Theorem [16].

Most current explicit congestion controllers (e.g., XCP [1], MaxNet [5], JetMax [6]) calculate an advertised source sending rate \hat{q}_j(t) for all the passing flows based on both the link capacity and the router queue size. However, the
estimation error of wireless capacity may introduce a large steady-state error which causes the flow throughput [3, 7] to fluctuate. To avoid this potential problem of wireless networks, API-RCP calculates an advertised source sending rate $q_i(t)$ based on the instantaneous router queue size $x(t)$ only [4,8], i.e.,

$$
q_i(t) = K_p e(t) + K_{1,0} e(t) d \tau = K_p (x_0 - x(t)) + K_{1,0} e(t) d \tau
$$

where $x_0$ is the target queue size; $e(t)$ is queue deviation; $K_p$ and $K_1$ are the proportional and integral gains of the PI rate controller located at the router. The transfer function between $q_i(t)$ and $e(t)$ is a PI rate controller $C(s)=K_p+K_{1} s$. The parameter $q_i(t)$ is carried in a field that can be defined in the IP packets and the ACK packets. Only the smallest $q_i(t)$ along the path is kept and then used to inform the source to prevent the bottleneck router from congestion. The source node $i$ will regulate its sending rate $q_i(t)$ after receiving $q_i(t)$ [10], i.e.,

$$
q_i(t) = q_i(t - \tau_{bi}),
$$

where $\tau_{bi}$ is varying feedback path delay from the router to the controlled source node $i$ via the destination.

By letting $\tau = \tau_0 + \tau_{bi}$ be the varying round trip time of the controlled source node $i$, the queueing dynamics becomes

$$
\dot{x}(t) = \sum_{i=1}^{N} q_i(t - \tau - \tau_{bi}) + u(t) - \mu(t) = \sum_{i=1}^{N} q_i(t - \tau) + u(t) - \mu(t).
$$

By obtaining the average round trip time $\tau$ as $\tau = \frac{1}{N} \sum_{i=1}^{N} \tau_i / N$, we can approximate the queueing dynamic in (4) as

$$
\dot{x}(t) = N q(t - \tau) + u(t) - \mu(t).
$$

Then the explicit congestion control system model, the transfer function between the instantaneous queue size $x(t)$ and the current advertised source sending rate $q(t)$, is given as

$$
P(s) = \frac{X(s)}{Q(s)} = \frac{N e^{-\tau s}}{s}.
$$

Based on the control model $P(s)$, the phase margin specification from Nyquist stability criterion was employed to design a PI rate controller $C(s)$ in [10].

![Figure 2: An Explicit Feedback Congestion Control System with Multiplicative Uncertainty](image)

3. Controller Design Using $H_{\infty}$ Optimal Criterion

Fig. 2 models the multiplicative uncertainty in an explicit feedback congestion control system for our robust control design. We shall employ the $H_{\infty}$ optimal criterion to design a robust explicit congestion controller $C(s)$ (which is different from PI controller in [10]) that can regulate the source sending rate $q(t)$. Let $\Delta(s)$ be the multiplicative uncertainty of the control model $P(s)$. The $\Delta(s)$ block represents the estimation errors of the active flow number $N$ and the average round trip time $\tau$. The load disturbance $d(t)$ represents the uncontrolled traffic rate $\tau d(t)$.

In order to ensure the closed-loop system stability, robust control theory requires the controller $C(s)$ to satisfy the following constraint [14],

$$
\|\Delta(s) T(s)\|_\infty \leq \frac{\|\Delta(s)\|}{1+C(s) P(s)} < 1.
$$

where $T(s)$ is the complementary system sensitivity function with $T(s) + S(s) = 1$. The function $S(s)$ is the system sensitivity function which can be expressed as

$$
S(s) = \frac{1}{1+C(s)P(s)}.
$$

To meet the constraint in (7), we can design a robust controller $C(s)$ using the $H_{\infty}$ optimal criterion $\min\|W(s) S(s)\|_\infty$ [14], where $W(s)$ is a proper weight function to be selected. According to the robust control theory [14], a $H_{\infty}$ controller $C(s)$ to guarantee the system internal stability can be obtained from

$$
C(s) = \frac{Q(s)}{1 - Q(s) P(s)},
$$

where $Q(s)$ is a stable internal model controller. For explicit congestion control based on queue size, the target queue size $x_0$ is the system input, and can be treated as a step signal $x_0 \delta$ [14]. Then we can select the weight function as $W(s) = 1/s$.

To simplify the robust controller design, we linearize the explicit congestion control model described by (6) using a first-order Padé approximation of $e^{-\tau s}/(1+\tau s)$ [17], i.e.,

$$
P(s) = \frac{N e^{-\tau s}}{s} = \frac{N (1 - \tau s)}{s(1 + \tau s)}.
$$

Since the network model $P(s)$ has a zero $s=1/\tau$, based on the maximum modulus theorem, we can obtain

$$
\|W(s) S(s)\|_\infty \geq \|W(1/\tau)\| = \tau.
$$

By minimizing the left side of (12), we can obtain $H_{\infty}$ optimal criterion as

$$
\min\|W(s) S(s)\|_\infty = \tau.
$$

Substituting (8), (9), (10) and (11) into (13), we can obtain the optimal internal controller $Q_{opt}(s)$ as

$$
Q_{opt}(s) = \frac{W(s) - \tau}{W(s) P(s)} = \frac{s(1 + \tau)}{N}.
$$

One can see that the optimal controller $Q_{opt}(s)$ is improper and cannot be implemented in practice. We introduce a low-pass filter $J(s)$ to make it proper, thus obtaining the sub-optimal controller $Q(s)$ as

$$
Q(s) = Q_{opt}(s) J(s),
$$

$$
J(s) = \frac{\gamma s + 1}{(\beta s + 1)^{2}},
$$

where $\beta$ and $\gamma$ are two tuning parameters.

According to CAIDA(Cooperative Association for Internet Data Analysis)'s traffic workload overview, measurements at backbone routers, edge routers, and major exchange points show 90-95% of the bytes belonging to TCP traffic [18]. It means that the uncontrolled traffic $\tau d(t)$ is restricted to a small portion (e.g., <10%) or an upper bound $\nu$. We can treat $\nu$ as a step disturbance, i.e., $d(t)=\nu s$. The load disturbance $d(t)$ is equivalent to a part of the output disturbance, which can be expressed as $d(s) P(s)=\nu(N(1-\nu)/(\pi(1+\nu))$ [14]. Therefore, the sensitivity function $S(s)$ should include at least two zeros at $s=0$ to reject the load disturbance $d(t)$ [14]. Then we have
\[ \lim_{\tau \to 0} \frac{d}{ds}(S(s)) = \lim_{\tau \to 0} \frac{d}{ds}(1 - Q(s)P(s)) = \lim_{\tau \to 0} \frac{d}{ds}\left(1 - \frac{(s + 1)(1 - \tau)}{(\beta s + 1)^3}\right) = 0, \tag{17} \]

which leads to
\[ \gamma = 3\beta + \tau. \tag{18} \]

Substituting (14), (16) and (18) into (15), we can obtain
\[ Q(s) = \frac{s(1 + \alpha)(3\beta + \tau)s + 1}{N(\beta s + 1)^3}. \tag{19} \]

Obviously when \( \beta \) approaches zero, \( Q(s) \) is equal to \( Q_{\text{opt}}(s) \). Substituting (19) into (9), we can obtain robust \( H_c \) controller as
\[ C(s) = \frac{(3\beta + \tau)s^2 + (3\beta + 2\tau)s + 1}{N(3\beta^2 + 3\beta \tau + \tau^2)}. \tag{20} \]

Such \( H_c \) controller \( C(s) \) is equivalent to a classical PID controller with a noise filter in the form of [19]
\[ C(s) = K_P + \frac{1}{K_I s} + \frac{K_D}{T_F s + 1}, \tag{21} \]

where \( K_P \) is proportional gain, \( K_I \) is integral gain, \( K_D \) is derivative gain and \( T_F \) is noise filter constant. By comparing (20) and (21), we can obtain the following controller parameters
\[ T_F = \frac{\beta^2}{3\beta^2 + 3\beta \tau + \tau^2} \quad K_P = \frac{3\beta + 2\tau - T_F}{N(3\beta^2 + 3\beta \tau + \tau^2)} \quad K_I = \frac{1}{N(3\beta^2 + 3\beta \tau + \tau^2)} \quad K_D = \frac{(3\beta + \tau)(-3 - 3\beta + 2\tau)T_F - T_F^2}{N(3\beta^2 + 3\beta \tau + \tau^2)}. \tag{22} \]

We can now tune the parameter \( \beta \) to achieve desirable robustness of the explicit congestion control system. Based on numerous experiments, a rule of thumb for the range of \( \beta \) is to suggest it be \( 2\tau \leq \beta \leq 3\tau \), where \( \tau \) is average round trip time.

For a real-time implementation, we use the SRED zombie list [20] to estimate \( N \) (the number of active long-lived flows), and the moving average filter [10, 15] to estimate average round trip time \( \tau \) [12]. Then we use (22) to calculate robust controller parameters based on estimated \( \hat{N} \) and \( \hat{\tau} \).

We use \( \alpha = \hat{N}\hat{\tau} \) and the average queue size \( x_{\text{avg}} \) to monitor the traffic load change. Like RED, we use an exponentially weighted moving average filter [15] to obtain \( x_{\text{avg}} \) as
\[ x_{\text{avg}}(t) = (1 - w_q)x_{\text{avg}}(t-1) + w_q x(t) \tag{23} \]

where \( w_q \) is the filter weight. The monitoring intervals of \( \alpha \) and \( x_{\text{avg}} \) are \( |\alpha - \alpha_0| < 0.5 \text{min}\{\alpha, \alpha_0\} \) and \( |x_{\text{avg}} - x_0| < 0.5x_0 \) respectively. Only when both two monitoring parameters \( \alpha \) and \( x_{\text{avg}} \) go outside their monitoring intervals, PI rate controller self-tunes.

### 4. PERFORMANCE EVALUATION OF ROBUST CONTROLLER

We run OPNET® [22] simulation to justify the robustness of our explicit congestion controller upon bounded load disturbance (i.e., bounded traffic change) when we disable the self-tuning mechanism. We make transient network performance comparison between our robust explicit congestion controller and TCP/RED in a real IP network. We consider two types of TCP Reno sources that use the Fast-Retransmit and Fast-Recovery mechanisms in the simulations: 1) the long-lived sources (i.e., controlled FTP sources for best-effort service traffic), which always have IP packets to send as long as their congestion windows allow; and 2) the short-lived sources (i.e., HTTP sources), which enter the network after a random period of think-time, send a file of a random length, and then wait for another think-time period [20]. The destination’s advertised window size is set sufficiently large so that the throughputs of the TCP connections are not restricted at the destination. Fig. 3 depicts the network simulation topology.

In all the simulations, all IP packets have the same size of 1024 bytes. The maximum window size allowed for all the TCP (or robust explicit congestion controlled) sources is 20,000 packets. The buffer size \( B \) of the IP Router 1 (or the bottleneck router) is determined based on the different bandwidth-delay product [15], while the buffer size of the Router 2 is big enough so that no packets will be lost. For robust explicit congestion controller, the target queue size is specified to be 30% of the buffer size, i.e., \( x_0 = 0.3B \) [10]. There are \( N = 25 \) controlled FTP source nodes and one uncontrolled guaranteed FTP source node transmitting the IP packets through the router. At the mean time, 50 HTTP sources send the data through the router throughout the simulation period and their RTPDs (Round Trip Propagation Delays) are uniformly distributed between 80 ms and 240 ms. The uncontrolled flow throughput within different active period are shown in Fig. 4(c) and 5(i). The tuning parameter \( \beta \) for robust controller is set to be \( \beta = 2.5\tau \). The filter weight for zombie list \( w_F \) is 0.002 [20].

![Network Simulation Topology for Performance Evaluation](image)

#### Table 1. RTPD Configuration For Robustness Test

<table>
<thead>
<tr>
<th>Source ID</th>
<th>RTPD (ms)</th>
<th>Simulation Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP 1-3/4-5</td>
<td>80</td>
<td>0≤t≤5400/0≤t≤320</td>
</tr>
<tr>
<td>FTP 6-8/9-10</td>
<td>120</td>
<td>0≤t≤5400/0≤t≤320</td>
</tr>
<tr>
<td>FTP 11-13/14-15</td>
<td>160</td>
<td>0≤t≤5400/0≤t≤320</td>
</tr>
<tr>
<td>FTP 16-18</td>
<td>200</td>
<td>160≤t≤320</td>
</tr>
<tr>
<td>FTP 19-21</td>
<td>240</td>
<td>160≤t≤320</td>
</tr>
<tr>
<td>uncontrolled FTP</td>
<td>120</td>
<td>0≤t≤5400</td>
</tr>
</tbody>
</table>

#### 4.1 Robustness to Bounded Traffic Changes

The bottleneck link capacity (i.e. service rate of IP Router 1) is set to 100 Mbps (i.e., 12,207 packets/sec). The buffer size is 2000 packets. For robust controller, the target queue size is set at 600 packets. Table 1 provides the RTPD's and the active periods of different FTP flows; their values reflect a bounded traffic change (i.e., ±6/15=±40% traffic change). Along with Fig. 4 below on the transient behaviors, we shall
discuss/demonstrate the robustness of our explicit congestion control system.

As seen from Table 1, there are 15 controlled FTP flows between time $t=0$ and 160s. Our robust $H_\infty$ controller self-tunes based on the estimated number of active flows and the average RTT. Beyond time $t=80s$, the self-tuning mechanism is disabled to force our controller to remain unchanged (as shown in Fig. 4(b)). At time $t=80s$, one can see that as the uncontrolled FTP flow rate increases, the queue increases sharply initially, but our robust controller is able to clamp the queue size back to the target queue size (600 packets) very fast. This robustness is again demonstrated at other time points of $t=160s$ and $320s$. At time $t=160s$, 6 controlled FTP flows join the network. This represents a $40\% (6/15)$ increase on the initial design of our robust $H_\infty$ controller. At time $t=320s$, 12 controlled FTP flows leave the network, which represents a $40\% (6/15)$ decrease on the initial design of 15 controlled flows. Upon the fluctuation of $40\%$ traffic change, our robust $H_\infty$ controller can always drive the system back to its steady state very quickly, thus demonstrating a good robustness to the bounded disturbance or a bounded network traffic change.

4.2 Performance Comparison of Robust Controller and TCP/RED

In this experiment, we increase the bottleneck link bandwidth (i.e. service rate of IP Router 1) to 1 Gbps (i.e., 122,070 packets/sec). The buffer size is 20,000 packets. For robust controller, the target queue size is 6000 packets. We select the RED parameters recommended by [15]: the maximum value for drop probability $p_{max}$ is 0.1; the maximum queue threshold $max_q$ is 14000; the minimum queue threshold $min_q$ is 3000, the filter weight $w_f$ is 0.002 [15]. Table 2 provides the RTPDs and the active periods for the controlled FTP sources and uncontrolled FTP source. Due to space limitation, we will make performance comparison with other congestion control algorithms in our future work.

Fig. 5 shows the transient behavior of our robust $H_\infty$ controller (depicted in black) and TCP/RED (depicted in red) in the bottleneck Router 1, estimated active controlled FTP flow number, estimated average RTT, monitoring parameter $\alpha$, average queue size $x_{avg}$, the robust $H_\infty$ controller gains. Fig. 5 shows the real-time self-tuning process of $H_\infty$ controller and the real-time estimation of the network parameters.

| Table 2. RTPD Configuration For Performance Comparison |
|-----------------|-----------------|-----------------|
| Source ID | RTPD (ms) | Simulation Time $t$ (sec) |
| 1-5 | 80 | $0 \leq t \leq 400$ |
| 6-10 | 120 | $0 \leq t \leq 400$ |
| 11-15 | 160 | $0 \leq t \leq 400$ |
| 16-20 | 200 | $160 \leq t \leq 320$ |
| 21-25 | 240 | $160 \leq t \leq 320$ |

uncontrolled FTP 120 $0 \leq t \leq 400$

Figure 4: Transient Behavior of Robust $H_\infty$ Explicit Congestion Control System upon Bounded Traffic Changes

Figure 5: Transient Behavior of the Congestion Control System Using the Robust $H_\infty$ Controller or the TCP/RED Controller
At time $t=0$ s, the 15 robust controlled FTP source nodes $(i=1,2,\ldots,15)$ start to transmit the packets into the network. The bottleneck router starts to estimate the flow number $N$ and average RTT $\tau$ and then calculates $H_\infty$ controller gains (shown in Fig. 5(g) and 5(h)) using Eq. (22) based on estimated $N$ and $\tau$ (shown in Fig. 5(c) and 5(d)). The controlled source node $i$ starts to increase its sending rate till it reaches a steady state of about 63 Mbps at the time around 16 s (as shown in Fig. 5(k)). The instantaneous queue size increases and approaches its target queue size (6000 packets). It starts to clamp its steady state around 6000 packets (depicted in black) at about $t=16$s (see Fig. 5(a)). At time $t=80$ s, the uncontrolled FTP source increases its sending rate from 50 Mbps to 150 Mbps (see Fig. 5(j)). The instantaneous queue size increases till reaches its peak at the time about 81 s (see Fig. 5(a)). Then it decreases to its target queue size (6000 packets). The controlled source node 1 starts to decrease its sending rate to a lower rate of around 57 Mbps at the time about 84s and remains steady after (see Fig. 5(k)).

At time $t=160$ s, another 10 controlled FTP sources $(i=16,17,\ldots,25)$ start to send the packets to the destination through the router. The queue size increases suddenly. The robust controlled router detects this dramatic change in traffic load, and then recalculates the controller gains (as shown in Fig. 5(g) and 5(h)) after noticing that both $\alpha$ and $x_{avg}$ fall outside their specified intervals (as shown in Fig. 5(e) and 5(f)). The proportional and integral gains decrease to reduce the flow throughputs of all 15 controlled FTP source nodes $(i=1,2,\ldots,15)$ (see Fig. 5(k)), thus clamping instantaneous queue size back to target queue size (6000 packets) again.

When the traffic changes happen at time $t=240$s and $t=320$s, robust $H_\infty$ controller can drive the queue size back to its target value very quickly. All the robust controlled FTP source nodes with different RTTs are assigned almost the same sending rates and exhibit a relatively flow throughput. However, a TCP/RED source with a shorter RTT is allocated with a higher bandwidth, e.g., the source sending rate of the source 1 with shortest RTPD=80 ms is almost two times larger than that of the source 21 with longest RTPD=240 ms (as shown in Fig. 5(k) and 5(l)). In the mean time, TCP/RED exhibits great fluctuants in both router queue size and single flow throughput.

The bottleneck link utilization of TCP/RED (depicted in red) fluctuates between 40% and 100% through the simulation period. Robust $H_\infty$ controller (depicted in black) maintains almost full link utilization most of time, as shown in Fig. 5(b).

### 5. Conclusions

We have discussed current explicit congestion control mechanism for high bandwidth-delay product network. We have employed the robust control theory to model the uncertainty of the explicit congestion control system. We selected a proper weight function to solve $H_\infty$ sensitivity problem. Then we used $H_\infty$ optimal criterion to design a robust explicit congestion controller. OPNET simulation has demonstrated that our $H_\infty$ controller exhibits a good robustness to a bounded network traffic change. Our robust $H_\infty$ controller not only utilizes network bandwidth efficiently, but also exhibits fast convergence to the system steady state and provides a relatively smooth flow throughput upon a traffic surge.

### References


