Abstract—As an explicit congestion control protocol, API-RCP (Adaptive PI rate control protocol) was originally proposed for best-effort traffic control in the IP networks with high bandwidth-delay product. The original API-RCP adopts a truncated network model for IP router in order to simplify the control system design while still achieving various designed performance measures at steady-state. A further benefit can be realized if one can specify a desirable transient system response (e.g., a shorter settling time). We have achieved this by using a pole-placement technique on a non-truncated closed-loop network model along with proper assignments of the damping ratio and the undamped natural frequency. Our OPNET simulations demonstrate a satisfactory transient network performance as designed. We also make performance comparison between API-RCP and TCP/RED. In the sequel, we can verify the effectiveness of the original truncated model using theoretical analysis instead of numerous simulations.

Index Items—Explicit Congestion Control, Pole Placement Technique, Transient Response, TCP/RED

1. INTRODUCTION

Current Internet mainly uses TCP (Transmission Control Protocol) control mechanism to prevent congestion [1]. As the network bandwidth increases, the TCP control mechanism is prone to system instability, regardless of the AQM (Active Queue Management) schemes (e.g., RED (Random Early Detection) [2], SRED (Stabilized RED) [3]) [4]. In wireless networks, when packets are lost due to wireless effects (e.g., channel fading) rather than congestion, TCP unnecessarily cuts the source sending rate by half, resulting in inefficient channel utilization and violation of QoS (Quality of Service) [4,5]. On the other hand, TCP cannot provide a relatively smooth flow throughput or guarantee a small jitter for streaming media traffic [6].

XCP (eXplicit Congestion Control Protocol) [4] was proposed to overcome the problems of TCP by explicitly feedback the network congestion information [7]. By calculating an advertised source sending rate based on link capacity and router queue size, XCP attempts to obtain high link utilization in high bandwidth-delay product networks, while maintaining a small queue size in the routers. Nevertheless, two potential problems of XCP have lured attention recently. (1) A poor choice of XCP parameters may lead to arbitrarily low link utilization and drift the flow rates far from their max-min allocations in a multiple-bottleneck network [7, 8]. MaxNet [9] and JetMax [10] are some enhancement that have been proposed to achieve max-min fairness of XCP; (2) The wireless capacity estimation error may introduce a large steady-state error and therefore fluctuations in XCP flow throughput [8, 11-13]. The method proposed in QFCP (Quick Flow Control Protocol) [13] is one effort to reduce capacity estimation error in a wireless network with high link utilization. To avoid these two potential problems of XCP, API-RCP (Adaptive Proportional-Integral Rate Control Protocol) [14] calculates an advertised source sending rate (carried in each data packet) based on the router queue size only [5, 15]. All the API-RCP flows with different round-trip times receive the same rate in the same link, thus guaranteeing the fairness of bandwidth allocation. Every packet carries a field for the lowest rate along the path to prevent the whole network from congestion.

Lyapunov stability criterion from the time-domain analysis and Nyquist stability criterion from the frequency-domain have been widely used to achieve system stability in network congestion control design or analysis (e.g., [4, 14, 16]). However, it will be very useful if we can also specify the transient response of the control system as done in the pole placement technique under the frequency domain [17].

The original API-RCP uses an address queue (or a hash table) to estimate the number of active flows, but this estimation method becomes impractical due to extensive CPU computation required for millions of flows. The SRED zombie list [3] is a much more CPU cost-effective estimation method with many industrial applications (e.g., NetFlow [18]). However, it suffers from its own estimation error. We therefore propose a double monitoring scheme to address the large estimation error of SRED zombie list that may cause PI controller to self-tune unnecessarily thus deteriorating network performance.

The original API-RCP has adopted a truncated model for an IP router in order to simplify the control system design [14]. The validity of truncated model has been supported by numerous of OPNET simulations that are run based on a real IP network (i.e., the true (non-truncated) model) in [14]. However, there is an interest to determine mathematically the effect of a true model for IP router in the explicit PI congestion controller design. Therefore, in this paper, we first developed a non-truncated model on which we apply our pole-placement technique to control the transient response of our congestion control system.

The main contributions of this paper are: (i) using the pole placement technique to specify the transient response of API-RCP; (ii) incorporation of the zombie list and double monitoring method to make the estimation of the active flow number CPU cost-effective and practical; (iii) OPNET performance evaluations to validate our pole placement design technique based on the non-truncated model. In addition, (iv) we have developed a theoretical analysis instead of a pure simulation in [14] leading to a proof that the original truncated model is effective for control system design.

2. THE NON-TRUNCATED NETWORK MODEL

Fig. 1 depicts the system model of an IP router using the API-RCP. IP packets generated at a source node are delivered to their destination through a sequence of IP routers in the Internet. Each API-RCP router calculates an advertised source window size (i.e., source sending rate $q_i'(t)$) for all the passing flows based on the instantaneous router queue size $x_i(t)$, as shown in (1) [14].

$$q_i(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau = K_p (x_0 - x(t)) + K_i \int_0^t (x_0 - x(t)) d\tau \ (1)$$

where $x_0$ is the target queue size; $e(t)$ is queue deviation; $K_p$ and $K_i$ are the proportional gain and integral gain of PI rate controller located at the router. The transfer function between $q_i'(t)$ and $e(t)$ is the PI rate controller $C(s)=K_p+K_i/s$. The parameter $q_i'(t)$ is carried in a field that can be defined in the
IP packets and the ACK packets, and only the smallest \( q'(t) \) 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 \( q'(t) \). Thus we can obtain

\[
q_i(t) = q_i(t - t_{bi}) ,
\]

(2)

where \( t_{bi} \) is the varying feedback path delay from the router to the controlled source node \( i \) via the destination. There are also uncontrolled traffic flows (both local and from the upstream router) with an aggregate input rate \( \dot{v}(t) \) into the router. Thus the router dynamic can be described as follows [14],

\[
\dot{x}(t) = \sum_{i=1}^{N} q_i(t - t_{bi} - t_{fi}) + \dot{v}(t) - \mu(t) ,
\]

(3)

where \( \mu(t) \) is the varying feedback path delay from the controlled source node \( i \) to the router. The parameter \( \mu \) is the link capacity (or the service rate) of the router, which may vary in the wireless networks [8, 11], according to Shannon’s Capacity Theorem [19]. By letting \( \tau = t_{bi} + t_{fi} \) the varying round trip time of the controlled source node \( i \), the queuing dynamic becomes

\[
\dot{x}(t) = \sum_{i=1}^{N} q_i(t - t_{bi} - t_{fi}) + \dot{v}(t) - \mu(t) = \sum_{i=1}^{N} q_i(t - \tau_i) + \dot{v}(t) - \mu(t) .
\]

(4)

To simplify the controller design, we can approximate the standardized round trip time by the average value \( \tau \) of all \( \tau_i \), i.e., \( \tau = \sum_{i=1}^{N} \tau_i / N \). We can update the router dynamic as

\[
\dot{x}(t) = \sum_{i=1}^{N} q_i(t - \tau) + \dot{v}(t) - \mu \approx Nq(t - \tau) + \dot{v}(t) - \mu(t) .
\]

(5)

Therefore, the transfer function between the instantaneous queue size \( x(t) \) and the current advertised source sending rate \( q'(t) \) (or \( q'(t) \)) is given as

\[
P(s) = \frac{X(s)}{Q(s)} = \frac{Ne^{-\tau s}}{s} \cdot \frac{\nu(s)}{s - \mu(s)} .
\]

(6)

where \( X(s), Q(s) \) and \( \mu(s) \) denote the Laplace transforms of \( x(t), q'(t) \) and \( \mu(t) \) respectively. Eq. (6) is what we refer to as the non-truncated network model.

Figure 1: System Model of the IP Router with API-RCP

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 [20]. It means that the uncontrolled traffic \( \dot{v}(t) \) is restricted to a small portion (e.g., <10%), and therefore can be upper-bounded by a threshold \( \nu \). Thus the Laplace transform of this disturbance can be approximated by a step \( a \nu / s \), which has been validated by OPNET simulations later on. Since it is not easy to design the PI controller based on (6) directly, the second term in (6) is dropped/truncated to take on a simpler form as below,

\[
P_0(s) = \frac{X(s)}{Q(s)} = Ne^{-\tau s} / s .
\]

(7)

By adopting the truncated model based on (7) as a nominal control plant in [14], we had run numerous of OPNET simulations to verify its validity and effectiveness.

Unlike the previous work, we are adopting the true plant \( P(s) \) in (6) for our control design. By substituting (1) into (5), we can obtain the dynamic of the closed-loop API-RCP control system as

\[
x(t) = N(K_p x_0 + x \dot{\tau} + K_i \int x \ddot{\tau} + x(t - \tau)) + \dot{v}(t) - \mu(t) ,
\]

(8)

where \( \dot{v}(t) \) is the varying feedback path delay from the router to the controlled source node \( i \). The Laplace transform can be expressed as

\[
x(s) = N(K_p s + K_i / s) e^{-\tau s} + \dot{v}(t - \tau) + \mu(t) .
\]

Thus we can obtain the transfer function (or the true model) of the closed-loop explicit congestion control system as

\[
G_c(s) = \frac{X(s)}{s x_0 + N(K_p + K_i / s) e^{-\tau s}} = \frac{N(K_p s + K_i / s) e^{-\tau s} + \dot{v}(t - \tau) + \mu(t)}{s^2 + N(K_p s + K_i / s) e^{-\tau s}} .
\]

(9)

Here \( G_c(s) \) can also be expressed as \( G_c(s) = G(s)(1 + G(s)) \) [17] with \( G(s) = c(s)p(s) \), where \( c(s) \) is the PI rate controller and \( p(s) \) is the explicit congestion control plant.

3. CONTROLLER DESIGN BASED ON POLE PLACEMENT

Unlike previous approach of using phase margin [14], we shall use the pole placement method to design our API-RCP controller (or the PI rate controller) based on the above non-truncated network model.

By making the approximation of \( e^{-\tau s} \approx 1 - s \tau \) when \( |s \tau| \leq 0.6 \) (i.e., \( 0.6 \leq 0.6 \tau \)), we can rewrite (9) as

\[
G_c(s) = \frac{N(K_p s + K_i)(1 - s \tau) + \dot{v}(t - \tau) + \mu(t)}{s^2 + (K_p s + K_i)(1 + s \tau)} .
\]

(10)

The characteristic equation of the closed-loop explicit congestion control system \( G_c(s) \) is

\[
s^2 + (K_p s + K_i)(1 + s \tau) = (1 - K_p N \zeta s^2 + (K_p N - K_i N) s + K_i N) = 0 ,
\]

which can be simplified as

\[
s^2 + K_p N - K_i N \zeta s^2 + K_i N = 0 .
\]

(11)

Let \( s_1 \) and \( s_2 \) be the roots of (11), which are also the poles of the second-order closed-loop control system. To guarantee the stability of the closed-loop control system, \( s_1 \) and \( s_2 \) will be placed in the left-half \( s \) plane [17]. The closed-loop transfer function of the second-order control system can be expressed in the standard form as

\[
G_c(s) = \frac{\zeta a_0^2}{s^2 + 2 \zeta a_0 s + a_0^2} ,
\]

(12)

where \( \zeta \) and \( a_0 \) are the damping ratio and the undamped natural frequency of the control system respectively, and can be used to describe the dynamic behavior of the second-order control system [17]. Control engineering practice and numerous experimental results show that a proper choice of proper damping ratio \( \zeta \) and undamped natural frequency \( a_0 \) (e.g., \( 0.4 \leq \zeta \leq 0.8 \) and \( 0.2 \leq 0.6 \tau \)) can achieve a desire transient response of the API-RCP control system. Based on

---

1 Such control plant approximation is also widely adopted in the classical control system design such as [21-23].
(11) and (12), we can obtain [14]
\[ K_p N - K_t N \tau = \frac{2 \zeta o_n}{1 - K_p N \tau}, \quad K_t N = \frac{o_n^2}{1 - K_p N \tau}. \quad (13) \]
Then based on (13), we can obtain the parameters of PI rate controller \( K_p \) and \( K_t \) as follows,
\[ K_p = \frac{o_n^2 \tau + 2 \zeta o_n}{N(o_n^2 \tau^2 + 2 \zeta o_n \tau + 1)}, \quad K_t = \frac{o_n^2}{N(o_n^2 \tau^2 + 2 \zeta o_n \tau + 1)}. \quad (14) \]
Although the analysis for the pole placement design technique is complicated, it is still simple and easy to obtain the PI controller parameters \( K_p \) and \( K_t \) using (14) by specifying proper \( \xi \) and \( o_n \) values (e.g., \( \xi = 0.6 \) and \( o_n = 0.4/\tau \)) in the real-time implementation.

3.1 Double monitoring scheme for traffic load change

The Original API-RCP [14] uses a monitoring parameter \( \alpha = \tilde{N} \tilde{\tau} \) to monitor the traffic load change, where \( \tilde{N} \) is the estimated number of active long-lived flows and \( \tilde{\tau} \) is the estimated average round trip time. Once a large traffic change causes \( \alpha \) to go outside the interval \([\alpha - \alpha_{\min}, \alpha_{\max}]\), the PI rate controller self-tunes and resets to \( \alpha = \alpha_{\min} \). The original API-RCP [14] uses an address queue (or a hash table) to estimate the number of active flows, which is relatively accurate but computationally intensive. Therefore, it is impractical for an ordinary CPU when there are hundreds of thousands of flows.

SRED [3] has introduced a simple zombie list to make the active flow estimation cost-effective for a CPU. However, this approach has a severe shortcoming. When the number of active flows remains unchanged (or only changes slightly), a large estimation error on \( N \) might occur, and shift the monitoring parameter \( \alpha \) outside the interval. This will cause the PI rate controller to self-tune unnecessarily and result in the fluctuation of system performance such as queue size and flow throughput.

Since an increase of \( N \) would increase the queue size by injecting more arrival packets into the router while decreasing \( N \) would reduce the queue size, we specify another interval \([x_{\min}, x_{\max}]\) to monitor the average queue size \( \bar{x}_{\text{avg}} \) at the same time, i.e., \( x_{\min} \leq \bar{x}_{\text{avg}} \leq x_{\max} \) where \( x_{\max} \) is target queue size. Therefore, PI rate controller would self-tune only when both two monitoring parameters \( \alpha \) and \( x_{\text{avg}} \) fall outside their monitoring intervals.

Like RED, we use an exponentially weighted moving average filter [2] to obtain \( x_{\text{avg}} \), i.e.,
\[ x_{\text{avg}}(t) = (1-w) x_{\text{avg}}(t-1) + w x_p(t), \quad (15) \]
where \( w \) is the filter weight.

4. SIMULATIONS AND PERFORMANCE EVALUATION

We would like to compare the transient network performance between API-RCP and TCP/RED in a real IP network to validate the pole-placement design technique for API-RCP. 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 [3]. The destination’s window size is set sufficiently large so that the throughputs of the TCP connections are not constrained at the destination. Fig. 2 depicts the network topology that our OPNET® [24] simulations are based on.

During the all the following simulations, IP packets have a fixed size of 1024 bytes. The maximum window size allowed for all the TCP (or API-RCP) sources is 2000 packets. The bottleneck link bandwidth (i.e. service rate of IP Router 1) is 100Mbps (i.e., 12,207 packets/sec). The buffer size \( B \) of the IP Router 1 (or the bottleneck router) is determined based on the different bandwidth-delay product [2], while the buffer size of the Router 2 is big enough so that no packets will be lost. For API-RCP, the target queue size is specified to be 30% of the buffer size, i.e., \( x_{\max}=0.3B \) from our many simulation experiments (some of which can be found in [14]).

![Figure 2: Network Simulation Topology for Performance Evaluation](image)

4.1 Performance Comparison of API-RCP and TCP/RED

During the simulation, we use \( N=35 \) controlled FTP source nodes and one uncontrolled (guaranteed FTP) source node transmitting the IP packets through the router. At the same time, there are 50 HTTP sources sending their data through the router. Table 1 provides the RTPDs (Round Trip Propagation Delays) and the active periods (i.e. simulation time) for all these sources. The uncontrolled flow throughput within different active periods is shown in Fig. 3(b). The buffer size is 2000 packets. For API-RCP, we have specified the interval of \( \xi \) as \( 0.4 \leq \xi \leq 0.8 \) and nominal value of \( \zeta \) as 0.6, the interval of \( o_n \) as \( 0.2 \leq o_n \leq 0.6 \) and nominal value of \( o_n \) as 0.4/\( \tau \), where \( \tau \) is average round trip time. The target queue size is specified as 600 packets. We select the RED parameters recommended by [2]: the maximum value for drop probability \( p_{\text{max}} \) is 0.1; the maximum queue threshold \( \text{max}_{x_0} \) is 1400; the minimum queue threshold \( \text{min}_{x_0} \) is 300; the filter weight \( w_{\text{F}} \) is 0.002 [2]. The filter weight for zombie list \( w_{\text{F}} \) is 0.002 [3]. Due to space limitation, we will make performance comparison with other congestion control algorithms in our future work.

| Table 1. Round Trip Propagation Delay (RTPD) Configuration |
|-----------------|-----------------|
| Source ID       | RTPD(ms) | Simulation Time t (sec) |
| FTP 1–5         | 80       | 0≤t≤400                |
| FTP 6–10        | 120      | 0≤t≤400                |
| FTP 11–15       | 160      | 0≤t≤400                |
| FTP 16–25       | 200      | 160≤t≤5320              |
| FTP 26–35       | 240      | 160≤t≤5320              |
| HTTP 1–10       | 80       | 0≤t≤400                |
| HTTP 11–20      | 120      | 0≤t≤400                |
| HTTP 21–30      | 160      | 0≤t≤400                |
| HTTP 31–40      | 200      | 0≤t≤400                |
| HTTP 41–50      | 240      | 0≤t≤400                |
| uncontrolled FTP| 120      | 0≤t≤400                |

Fig. 3 compares various transient behaviors and performance measures between API-RCP (depicted in black)
and TCP/RED (depicted in red) in the IP Router 1. Using the relationship derived earlier, one can obtain the evolution of the monitoring parameter \( \alpha \) (Fig. 3(f)) and the controller gains \( K_P \) and \( K_I \) (Fig. 3(e)) as a function of the time evolution of estimated \( N \) (Fig. 3(d)) and estimated average \( RTT \) (Fig. 3(e)).

At time \( t = 160s \), another 20 controlled FTP sources \( (i=16,17,\ldots,35) \) start to send packets to their destinations through the router. The queue size again jumps quickly (Fig. 3(a)). Although our API-RCR router can detect this dramatic change in traffic load, it also notices that both \( x_{avg} \) (Fig. 3(b)) and \( \alpha \) (Fig. 3(f)) fall outside their specified intervals, so its PI rate controller self-tunes its proportional gain \( K_P \) and its integral gain \( K_I \) (Fig. 3(g)). The values of these two gain parameters are decreased dramatically, which cause all 15 controlled FTP source nodes \( (i=1,2,\ldots,15) \) to reduce their sending rates sharply (see Figs. 3(i) and 3(j)). Therefore the instantaneous queue size decreases and then approaches the target queue size (600 packets) very quickly (Fig. 3(a)).

One can find more instances in Fig. 3(a) of how PI rate controller adapts quickly to the traffic changes at time \( t=240s \) and \( t=320s \). Also observe that all of the API-RCR controlled FTP source nodes obtain almost the same sending rates (compare Figs. 3(i), 3(j), 3(k) and 3(l) to each other), thus guaranteeing the fairness among the sources with the different \( RTT \) (Round Trip Time). On the other hand, when running TCP/RED, one can see from the same 4 figures that a source with a shorter \( RTT \) is allocated with a higher bandwidth, e.g., the source sending rate of the source 1 with \( RTPD=80ms \) (Fig. 3(i)) is almost twice that of the source 16 with \( RTPD=200ms \) (Fig. 3(k)). In the mean time, TCP/RED exhibits great fluctuations in both router queue size and every single flow throughput.

The nice features of our API-RCR can also be appreciated from the comparisons in other aspects. For example, Fig. 3(c) shows that bottleneck link utilization of TCP/RED router (depicted in red) fluctuates between 40% and 100% while that of the API-RCR (depicted in black) increases and then maintains almost 100% after \( t=13s \). There are only two sporadic locations of low utilization (each lasting about 2 seconds) due to drastic traffic change at time \( t=160s \) and \( t=320s \) (as shown in Fig. 3(c)).

### 4.2 The effectiveness of the truncated model

We have also performed simulations to compare our pole-placement design against the original phase-margin design. Fig. 4 shows one such comparison (in term of queue size) between the phase-margin design based on the truncated model in [14] (depicted in red) and the pole placement technique based on the non-truncated model (depicted in black). One can see the two performances are indeed very close.

From the viewpoint of the control theory and practice, the stability of a closed-loop control system is determined by its...
poles [17]. The available bandwidth ($\mu - \nu$) only has an impact on the zeros of the closed-loop congestion control system, and thus will not reduce the system stability margin [14], as shown in (9) and (10). In this regard, the truncated component ($\mu - \nu$) can be treated as “noise” in our model. Due to its robust characteristic, our PI controller can handle such “noise” effectively and drive the control system to a steady state successfully. This has been verified by OPNET simulations [14] and Linux implementations [25] using a real IP network (i.e., the non-truncated model). Therefore, it is wise and effective to adopt the truncated model (7) of API-RCP while neglecting the component ($\mu - \nu$) in the explicit congestion control system design in [14], according to the robust control theory [26]. In other words, the controller design based on truncated model will not reduce the system stability margin if the disturbance is bounded (e.g., the uncontrolled traffic should be restricted [20]). Performance of API-RCP based on multiple bottleneck network topology has been studied in [8].

![Figure 4: Instantaneous Queue Size of the Pole Placement Design vs the Phase-Margin Design](image)

5. CONCLUSIONS

We have revisited API-RCP design and employed a pole placement technique to specify the transient system response. The damping ratio $\zeta$ and the undamped natural frequency $\omega_0$ are appropriately assigned such that: 1) the satisfactory transient response of the explicit congestion control system is achieved; 2) all the poles lie in the left-half s-plane to guarantee the stability of the IP network; 3) the simple equations for obtaining the PI rate controller are derived. We used SRED zombie list (a CPU cost-effective flow estimation method) to estimate the active flow number. Then we proposed a double monitoring scheme to prevent the large estimation error of SRED zombie list from self-tuning PI rate controller unnecessarily.

We have also set up a true (non-truncated) network model, not only to facilitate our controller design, but also to allow us to perform a theoretical analysis to verify the validity of the truncated model adopted by the original API-RCP because the difference between the truncated and non-truncated model can be treated as “bounded noise” that can be handled by the robust characteristic of our PI controller.

REFERENCES
