J. Parallel Distrib. Comput. 70 (2010) 547–557

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Distributed bandwidth allocation based on alternating evolution algorithm☆

Xiaomeng Huang*, Yongwei Wu, Guangwen Yang, Weiming Zheng, Jinlei Jiang

Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China

ARTICLE INFO

Article history:
Received 17 June 2009
Received in revised form
8 November 2009
Accepted 9 February 2010
Available online 24 February 2010

Keywords:
Resource allocation
Congestion control
Alternating evolution
Logistic Model

ABSTRACT

In a network, end nodes have to compete for bandwidth through some distributed congestion control algorithms. It is a great challenge to ensure the efficiency and fairness of the distributed control algorithms. TCP congestion control algorithms do not perform well in terms of their efficiency and fairness in high speed networks. In this paper, we propose a novel asymptotic evolution algorithm based on the Logistic Model to allocate limited bandwidth resource. The algorithm introduces an explicit bandwidth pre-allocation factor. The factor is carried by the packet and is computed in routers based on the information of the router capacity, the aggregate load, and the instantaneous queue length; therefore the algorithm does not require the routers to keep the per-flow state. According to this pre-allocation bandwidth factor, the senders asymptotically adjust their sending rate and the bandwidth factor changes asymptotically along with the variation of the aggregate load and the queue length in the routers; therefore the sending rate and the pre-allocation bandwidth factor form alternating evolution and eventually reach a steady state.

Theoretical analysis and simulation experiments were conducted to compare our algorithm with related ones. The results show that our algorithm not only provides fast convergence to efficiency and fairness, but also keeps a strong robustness against crossing traffic.

© 2010 Elsevier Inc. All rights reserved.

1. Introduction

The poor performance of TCP [1,28,14,33,21] can be quantitatively measured by convergence to efficiency and fairness [4,22]: the time taken for the transport control system to transit from its initial state to a steady one. When a new flow joins the network, it is expected that the new flow would instantly grab the available bandwidth of the network, which emphasizes convergence to efficiency. When a flow joins a fully occupied network, it is expected that the flow would quickly achieve fair bandwidth allocation. At this time, convergence to fairness is accentuated.

Supposing that the throughput of a TCP flow in steady state is P, some literature (e.g., [4,37,15]) prove that for the classical Additive Increase Multiplicative Decrease (AIMD) algorithm used in TCP, the time for convergence to efficiency and fairness is both $O(P)$, which implies that the AIMD algorithm linearly converges to efficiency and fairness. Since the value of P is huge in high speed networks, the AIMD algorithm takes a long time to converge to efficiency and fairness; therefore designers attempt to improve the convergence by applying an additional slow-start algorithm in its starting phase. However, this does not help improve the convergence speed in congestion avoidance phase.

The above issue motivated some novel transport protocols, such as HSTCP [9,7], STCP [17], BIC-TCP [36], XCP [16], EMKC [37], VCP [35], EVLF-TCP [13], JetMax [38,23], RCP [39,40]. HSTCP, STCP and BIC-TCP improve convergence by using an aggressive increasing and a conservative decreasing algorithm. Such an algorithm is always with a higher loss ratio than that of the TCP algorithm. Meanwhile, this method makes the round trip time (RTT) unfairness problem of HSTCP, STCP and BIC-TCP severer than that of TCP. EMKC, VCP, EVLF-TCP and JetMax allocate network resources effectively at the end node by explicitly feeding back the state information (e.g., the loss ratio, load factor, and virtual load factor) of the router. However, EMKC, VCP and EVLF-TCP only improve the convergence to efficiency from $O(P)$ to $O(\ln P)$ and the convergence to fairness is still $O(P)$. On the contrary, XCP and RCP improve the convergence by letting the router directly allocate the bandwidth for each flow, so that constant convergence and exponential convergence can be achieved respectively in a network of simple topology. However since XCP and RCP are too sensitive to network load, they are instable for heavy crossing traffic and XCP cannot achieve max–min fairness in multi-congested gateway networks [23].

In this paper, we propose a novel distributed alternating evolution algorithm with fast convergence and global asymptotic stability. The algorithm is based on some characteristics of the

☆ An earlier version of this paper was presented at the fifteenth IEEE International Conference on Network Protocols and was published in its proceedings.

* Corresponding author.
E-mail addresses: hxm@tsinghua.edu.cn, huangxiaomeng@gmail.com (X. Huang), wuyw@tsinghua.edu.n (Y. Wu), ygw@tsinghua.edu.cn (G. Yang), zwm-dcs@tsinghua.edu.cn (W. Zheng), jjlei@tsinghua.edu.cn (J. Jiang).

0743-7315/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2010.02.004

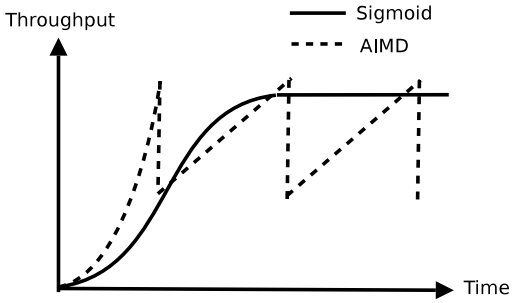


Fig. 1. Comparing the sigmoid curve with the AIMD curve.

"Logistic Model" in population ecology [30,24], which is widely used in many fields of modeling and forecasting [2,10]. This alternating evolution algorithm puts forward an the explicit bandwidth pre-allocation mechanism. Besides, we performed theoretical analysis of stability and convergence to determine the impact of control parameters on the algorithm performance. We also conducted simulation experiments on the NS2 simulator to confirm the favorable performance of the algorithm.

The remainder of this paper is organized as follows. Section 2 discusses our design rationale by addressing the ideal congestion control and its relationship with the Logistic Model. Section 3 presents our distributed asymptotic evolution algorithm. Section 4 analyzes the global asymptotic stability and convergence of the algorithm. Section 5 introduces the implementation of the corresponding transport protocol. Simulation results are given in Section 6. We conclude the paper in Section 7.

2. Design rationale

It is still an open issue to choose an operation point of congestion control. For most of implicit loss-based approaches (e.g., HSTCP, STCP, and BICTCP), full utilization if achievable, often comes at the cost of severe oscillations and potentially large queuing delay. On the contrary, some explicit congestion feedback (e.g., XCP and VCP) choose zero queue length as the operation point of congestion control. Thus a near-zero queue length and near-zero packet loss rate are easy to achieve at the cost of a little spare bandwidth. The main difference is that the implicit approaches works in a slight overload state, and the explicit approaches works in a slight underload state. In our opinion, an efficient congestion controller should guarantee a small size of the buffer in steady state so that the controller can yield high utilization and achieve small queuing delay.

In addition, compared with the sawtooth shape of the AIMD mechanism, we believe that the type of sigmoid curve generated by the source algorithm is better suited for congestion control in high speed networks. As shown in Fig. 1, the sigmoid control curve increases the sending rate gently at the initial phase, accelerates exponentially in the middle stage and finally approaches the upper bound of network capacity. The advantages of the sigmoid control curve are: (1) Not sending too many packets at the initial phase to avoid the burst traffic. (2) Exponentially increasing when the available capacity is sufficient. (3) Avoiding congestion as far as possible when the network load is heavy. (4) Allocating network resource effectively, and avoiding the waste of network resource caused by the oscillation of the AIMD mechanism.

Next, we introduce the foundation of the Logistic Model, a model studying the dynamics of populations in ecology. In the Logistic Model, the population number $x(t)$ in generations is expressed as:

$$\dot{x} = rx\left(1 - \frac{x}{K}\right). \tag{1}$$

Parameter r is the intrinsic rate of increase, which can be interpreted as the difference between the birth rate and the death rate of the population. Parameter K, called carrying capacity, is the upper bound of population growth. It is usually interpreted as the amount of resources expressed in the number of organisms that can be supported by the resources. The population growth ratio $\frac{dx}{xdt}$ declines with the population number x and reaches 0 when $x = K$. If the population number exceeds K, then the population growth ratio becomes negative and the population number declines. The curve of the Logistic Model is just a sigmoid curve.

Overall, it is easy to see that the Logistic Model consists of: (1) an intrinsic rate of increase r, and (2) an density-dependent factor $(1-x/K)$. If there is no resource limitation, the population number exponentially increases with an intrinsic rate r. However, as the resources are consumed gradually, the density-dependent factor becomes more and more dominant on the population growth rate, and finally forces the population number to achieve an equilibrium state. The Logistic Model provides a mature mathematical method to analyze how the populations share limited resources, so it is natural to adapt the Logistic Model to do congestion control.

We observed that it is difficult to enforce the congestion control using the Logistic Model directly. The problem is that when the network aggregate traffic exceeds the available bandwidth, the packets queue in the router buffer, which is not considered in population ecology models. Therefore, queues behavior should be controlled and a reasonable density-dependent factor is required to link the Logistic Model to the queuing model.

Some researchers have attempted to improve the performance of congestion control using the Logistic Model. For example, Welzl [32] directly uses the Logistic Model to design CADPC. However, he does not consider the influence of queuing phenomena on congestion control as we discussed above.

3. Algorithm design

A transport protocol may be divided into a distributed link algorithm and a distributed source algorithm [29,8,6,19]. The link algorithm runs in the router to detect the congestion of the network and to generate congestion signals such as dropped packets, delay, explicit congestion notification, explicit packet loss rate, and explicit load factor. The source algorithm runs in the end node to adjust the sending rate according to the congestion signal. The main design issue of the congestion control algorithm is to select the appropriate congestion signal for the link algorithm and find the best way to respond to the signal in the source algorithm.

In general, explicit congestion feedback schemes directly communicate with the router to precisely inform the end node of the state of the network. This is achieved by sending special packets or by changing some fields in the packets as they travel through the routers. The use of explicit congestion feedback usually results in superior congestion control protocols that converge faster and have a lower packet loss rate than the protocols using implicit congestion feedback.

Similar to XCP and MaxNet [34], our novel mechanism is an explicit bandwidth pre-allocation mechanism. As shown in Fig. 2, each router maintains a pre-allocation rate factor r. The basic adjusting strategy of r is that when the link is underloaded, r gradually increases, otherwise r decreases gradually. The minimal r value corresponding to all links along the path is sent to the end node. After the end node receives the pre-allocation rate factor r, it treats the value of r as the upper limit of capacity that the network can provide, and then it makes the sending rate x approach the r value quickly. Driven by the alternating evolution of r and x, the end nodes and the routers can reach a steady state gradually.

Consider a network with a set L of links, and let C_l be the finite capacity of link l, $r_l(t)$ be the pre-allocation rate factor by the link l,

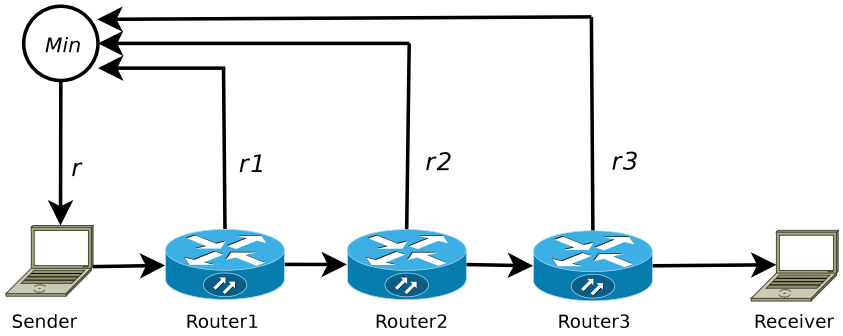


Fig. 2. Principle of bandwidth pre-allocation mechanism.

$q_l(t)$ be the instantaneous queue size of the link l, for $l \in L$. Let a router be a non-empty subset of L, and write P for the set of possible routes. If $l \in p$, then the link l lies on the route p. If $p \in l$, then the route p passes through the link l. Associate a route with a flow, and suppose that the rate $x_p(t)$ is allocated to user p.

We proposed the following distributed congestion control model, consisting of the the link algorithm and source algorithm.

3.1. Link algorithm

Consider a system of differential equations:

$$\dot{r}_l(t) = \beta r_l(t)\left(1 - \frac{\sum_{p\in l} x_p(t) + (q_l(t) - q_0)/T}{C_l}\right) \tag{2}$$

where β is a constant parameter, q_0 is the expected queue length in steady state, T is a time constant. The term $\sum_{p\in l} x_p(t)$ denotes the whole load on link l. In order to control the queue length, we treat the queuing packets as special species that also consumes part of the bandwidth resources. Then the available bandwidth should equal to the bottleneck bandwidth minus the aggregate traffic and the queuing traffic in the router buffer. The term $(1 - \frac{\sum_{p\in l} x_p(t)+(q_l(t)-q_0)/T}{C_l})$ represents the normalized available capacity.

Therefore when the available capacity is sufficient, r_l grows quickly, otherwise r_l grows slowly. When the available capacity is completely consumed, r_l achieves equilibrium. At this time q_l equals q_0, and $\sum_{p\in l} x_p(t)$ equals C_l exactly. The link algorithm mainly enables the pre-allocation rate factor that each link maintains to quickly respond to the instantaneous load and queue length. It is obvious that the link control algorithm is only decided by the aggregate state of flows passing through the link. It is no need to record per-flow state in routers to compute r_l.

3.2. Source algorithm

Consider a system of differential equations:

$$\dot{x}_p(t) = \alpha x_p(t)(\ln r_p(t) - \ln x_p(t)) \tag{3}$$

where

$$r_p(t) = \min\{r_l(t) | l \in p\}. \tag{4}$$

α is a constant parameter. In a real network, the path p often consists of multiple links. Because each link l maintains a pre-allocation rate factor r_l, in order to obtain the most congested node in the network, we can only choose the minimum value r_p among all pre-allocation rate factors.

For the flow p, the r_p value is the maximum capacity that the network can provide. Generally, flow p enters the network with a low initial rate. The pre-allocation rate factor r_p received by the end-system will be larger than x_p, so that x_p exponentially approaches r_p according to Eq. (3). When x_p equals r_p, the end node reaches equilibrium. We use the logarithm function in the source algorithm to keep the time for convergence to fairness as $O(\ln \ln P)$ (see Section 4).

In general, the link algorithm and source algorithm have the intrinsic rate of increase and the density-dependent factor that are similar to those of the Logistic Model. The key control variable x and r in the link algorithm and source algorithm are coupled so that the x and r can alternatively evolve to the steady state. So we call the complete control model consisting of (2) (3) and (4) the "Alternating Evolution Control Model", and its corresponding transport protocol the "Alternating Evolution Control Protocol" (AECP).

XCP uses the direct bandwidth allocation method in the router to instantly get the target assignment and it requires the exchange of congestion window in the packet header as well as the RTT signal, which makes the network more vulnerable to router attacks. Different from XCP, our AECP adopts an exploratory assignment strategy in the router, which continuously adjusts the r factor without any auxiliary information from the end node. We name this mechanism as bandwidth pre-allocation. r and x achieve the final target assignment through joint evolution of the link and source algorithm.

The fluid-flow queuing model proposed in [11,26] contains another equation of the flow rate. For the bottleneck link, given the aggregate arrival rate and link bandwidth, we can calculate instantaneous queue length $q_l(t)$ as follows:

$$\dot{q}_l(t) = \sum_{p\in l} x_p(t) - C_l \quad (\text{if } q_l(t) > 0). \tag{5}$$

This equation shows that a queue builds up when the aggregate arrival rate exceeds the link bandwidth.

Notice that the pre-allocation rate factor perceived by the sender also has a time delay τ_p, i.e. $r_p = r_p(t - \tau_p)$; therefore the whole control model of the network may be expressed as:

$$\begin{cases} \dot{x}_p(t) = \alpha x_p(t) \cdot (\ln r_p(t - \tau_p) - \ln x_p(t)) \\ \dot{r}_l(t) = \beta r_l(t) \cdot \left(1 - \dfrac{\sum_{p\in l} x_p(t) + (q_l(t) - q_0)/T}{C_l}\right) \\ \dot{q}_l(t) = \sum_{p\in l} x_p(t) - C_l \quad (\text{if } q_l(t) > 0) \\ r_p(t) = \min\{r_l(t) | l \in p\}. \end{cases} \tag{6}$$

4. Performance analysis

In this section we study the global asymptotic stability [5,12, 20,31] of the system described by the differential equations (6) and determine the time to convergence. We analyze the impact of control parameters on the AECP's performance and provide a guideline to determine the appropriate parameters of AECP.

4.1. Stability

To prove that AECP can reach a fair and stable state, we derived the following theorem:

Theorem 1. *The system described by the differential equations (6) is globally asymptotically stable independent of bottleneck capacity, number of flows and round trip time.*

Proof. Suppose $u_p(t) = \ln x_p(t)$, $v_l(t) = \ln r_l(t)$, and then the Eq. (6) can be rewritten as:

$$\begin{cases} \dot{u}_p(t) = \alpha \cdot (v_p(t-\tau_p) - u_p(t)) \\ \dot{v}_l(t) = \beta \cdot \left(1 - \dfrac{\sum_{p\in l} \exp\{u_p(t)\} + (q_l(t) - q_0)/T}{C_l}\right) \\ \dot{q}_l(t) = \sum_{p\in l} \exp\{u_p(t)\} - C_l \quad (\text{if } q_l(t) > 0) \\ v_p(t) = \min\{v_l(t) | l \in p\}. \end{cases} \tag{7}$$

Since $v_l(t)$ is computed by $u_p(t)$ and $q_l(t)$, and $q_l(t)$ is also computed by $u_p(t)$, we define a mapping function f from $u_p(t)$ to $v_l(t)$ which satisfies:

$$\dot{v}_l(t) = \beta f(u_p(t)). \tag{8}$$

Therefore, the Eq. (7) can be simplified as:

$$\begin{cases} \dot{u}_p(t) = \alpha \cdot (v_p(t-\tau_p) - u_p(t)) \\ \dot{v}_l(t) = \beta f(u_p(t)) \\ v_p(t) = \min\{v_l(t) | l \in p\}. \end{cases} \tag{9}$$

Next we use Lyapunov Stability Theory [20,12,18,25,27,3] to prove the global asymptotic stability of AECP. Firstly, we construct the following positive definite function:

$$\mathcal{U}(u_p(t), v_l(t) | p \in P, l \in L) = \beta \cdot \sum_{l\in L} \int_0^{-f(u_p(t))} y\mathrm{d}y + \alpha \cdot \sum_{p\in P} \frac{(v_p(t-\tau_p) - u_p(t))^2}{2}. \tag{10}$$

Observe that:

$$\frac{\partial}{\partial u_p(t)} \mathcal{U} = -\alpha(v_p(t-\tau_p) - u_p(t)) - \beta f(u_p(t)) \tag{11}$$

$$\left.\frac{\partial}{\partial v_l(t)} \mathcal{U}\right|_{v_l(t)=v_p(t)} = \alpha(v_p(t-\tau_p) - u_p(t)) \tag{12}$$

$$\left.\frac{\partial}{\partial v_l(t)} \mathcal{U}\right|_{v_l(t)\neq v_p(t)} = 0. \tag{13}$$

Furthermore,

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\mathcal{U} &= \sum_{p\in P} \frac{\partial}{\partial u_p(t)} \mathcal{U} \cdot \frac{\mathrm{d}}{\mathrm{d}t} u_p(t) + \sum_{l\in L} \frac{\partial}{\partial v_l(t)} \mathcal{U} \cdot \frac{\mathrm{d}}{\mathrm{d}t} v_l(t) \\ &= \sum_{p\in P} [-\alpha(v_p(t-\tau_p) - u_p(t)) - \beta f(u_p(t))] \\ &\quad \times [\alpha \cdot (v_p(t-\tau_p) - u_p(t))] \\ &\quad + \sum_{p\in P} \alpha\beta (v_p(t-\tau_p) - u_p(t)) \cdot f(u_p(t)) \\ &= -[\alpha(v_p(t-\tau_p) - u_p(t))]^2. \end{aligned} \tag{14}$$

Clearly, the function $\dot{\mathcal{U}}$ is negative by definition. It is a Lyapunov function for the system of differential equations (7). According to the Lyapunov stability theory, the system is globally asymptotically stable. □

Consider that N long-lived flows share the capacity C, and suppose the equilibrium point of the differential equations (6) is $M(x_1^*, x_2^*, \ldots, x_N^*, r^*, q^*)$. Then we have,

$$\begin{cases} \alpha x_i^* \cdot (\ln r^* - \ln x_i^*) = 0 \\ \beta r^* \cdot \left(1 - \dfrac{N x_i^* + (q^* - q_0)/T}{C}\right) = 0 \\ N x_i^* - C = 0. \end{cases} \tag{15}$$

Since $x_p^* = 0$ is not a stable point, we have,

$$\begin{cases} x_i^* = r^* = \dfrac{C}{N} \\ q^* = q_0. \end{cases} \tag{16}$$

In the steady state, each flow gets the same rate and the queue length equals q_0, which indicates that AECP guarantees reasonable fairness and full link utilization.

4.2. Convergence

In this section, we show that AECP converges to efficiency and fairness exponentially. Since it is hard to solve the differential equations (6) in networks of complex topology, we only consider the simplest network topology in which there are N long-flows and a single bottleneck link.

4.2.1. Convergence to efficiency

To better understand the time AECP requires to reach a certain level of efficiency, we define:

Definition 1. For a given positive constant θ ($0 < \theta \leq 1$) and a bottleneck link with finite capacity C, a resource allocation ($x_1, x_2, \ldots, x_N$) is θ *efficiency*, if:

$$f(t) = \frac{\sum_{i=1}^{N} x_i(t)}{C} \geq \theta. \tag{17}$$

Thus the time for convergence to efficiency is the interval during which the link utilization increases from the minimal utilization to θ first, i.e. $f(t_\theta) = \theta$.

Based on this definition, we derive the following theorem:

Theorem 2. *Considering N synchronous AECP flows competing for the bottleneck bandwidth C with the initial throughput of each flow as x_0 ($x_0 \ll C/N$), the time for convergence to efficiency satisfies the following equations:*

$$\max\left(\frac{\ln\left(\frac{C}{Nx_0} \cdot \frac{\theta}{1-\theta}\right)}{\beta}, \frac{\ln\left(\frac{\ln\frac{Nx_0}{C}}{\ln\theta}\right)}{\alpha}\right) < t_\theta < \frac{\ln\left(\frac{C}{Nx_0} \cdot \frac{\theta}{1-\theta}\right)}{\beta} + \frac{\ln\left(\frac{\ln\frac{Nx_0}{C}}{\ln\theta}\right)}{\alpha}. \tag{18}$$

Proof. Since there is no persistent packet queuing before the utilization reaches θ, $q(t) = 0$. Suppose all flows are synchronous, the system can be simplified as:

$$\begin{cases} \dot{x}_i(t) = \alpha x_i(t) \cdot (\ln r(t) - \ln x_i(t)) \\ \dot{r}(t) = \beta r_l(t) \cdot \left(1 - \dfrac{N x_i(t) - q_0/T}{C}\right). \end{cases} \tag{19}$$

Consider two extreme cases: (1) Let $x_i(t)$ equal to $r(t)$ directly in the end node and $r(t)$ be adjusted based on the link algorithm, in which case, we denote that the time for convergence to efficiency is t_1; (2) Let $r(t)$ equal to C/N, and let $x_i(t)$ be adjusted based

on the source algorithm, in which case, we denote the time for convergence to efficiency is t_2. Certainly we have $\max(t_1, t_2) < t_\theta < t_1 + t_2$.

In the first case, the control law can be expressed as:

$$\begin{cases} x_i(t) = r(t) \\ \dot{r}(t) = \beta r_l(t) \cdot \left(1 - \dfrac{Nx_i(t) - q_0/T}{C}\right). \end{cases} \tag{20}$$

Furthermore,

$$\dot{x}_i(t) = \beta x_i(t) \cdot \left(1 - \frac{Nx_i(t) - q_0/T}{C}\right). \tag{21}$$

Solve the above equation and we have,

$$x(t) = \frac{\frac{C(1-q_0/CT)}{N}}{1 + \left(\frac{C(1-q_0/CT)}{Nx_0} - 1\right)\exp\{-b(1-q_0/CT)t\}}. \tag{22}$$

In high speed networks, $C \gg q_0/T$, $C \gg Nx_0$. Thus,

$$x(t) \approx \frac{\frac{C}{N}}{1 + \frac{C}{Nx_0}\exp\{-bt\}}. \tag{23}$$

Based on Definition 1, $f(t_1) = Nx_i(t_1)/C = \theta$. Therefore we can solve t_1 as follows:

$$\begin{aligned} t_1 &= \frac{\ln\left[\left(\frac{C}{Nx_0} - 1\right) \cdot \left(\frac{\theta}{1-\theta}\right)\right]}{\beta} \\ &\approx \frac{\ln\left(\frac{C}{Nx_0} \cdot \frac{\theta}{1-\theta}\right)}{\beta}. \end{aligned} \tag{24}$$

In the second case, the control law can be expressed as:

$$\dot{x}_i(t) = \alpha x_i(t)\left(\ln\frac{C}{N} - \ln x_i(t)\right). \tag{25}$$

Since $x(t_2) = \theta C/N$, we can solve t_2 as follows:

$$t_2 = \frac{\ln\left(\frac{\ln\frac{Nx_0}{C}}{\ln\theta}\right)}{\alpha}. \tag{26}$$

Finally,

$$\begin{aligned} &\max\left(\frac{\ln\left(\frac{C}{Nx_0} \cdot \frac{\theta}{1-\theta}\right)}{\beta}, \frac{\ln\left(\frac{\ln\frac{Nx_0}{C}}{\ln\theta}\right)}{\alpha}\right) < t_\theta \\ &< \frac{\ln\left(\frac{C}{Nx_0} \cdot \frac{\theta}{1-\theta}\right)}{\beta} + \frac{\ln\left(\frac{\ln\frac{Nx_0}{C}}{\ln\theta}\right)}{\alpha}. \quad \square \end{aligned} \tag{27}$$

Clearly, the time for AECP convergence to efficiency is $O(\ln\frac{C}{N})$ and it is of exponential complexity.

4.2.2. Convergence to fairness

To study the time that AECP requires to reach a certain level of fairness, we define:

Definition 2. For a given positive constant ε ($0 < \varepsilon \leq 1$), a resource allocation $(x_1, x_2, \ldots, x_N)$ exhibits ε *fairness*, if:

$$g(t) = \frac{\min_{i=1}^{N} x_i(t)}{\max_{j=1}^{N} x_j(t)} \geq \varepsilon. \tag{28}$$

Thus the time for convergence to fairness is the interval during which g increases from the maximally unfair state to ε *fairness*. i.e. $g(t_\varepsilon) = \varepsilon$.

Based on this definition, we derive the following theorem:

Theorem 3. *Consider that N synchronous AECP flows have completely shared the bottleneck capacity C at steady state, and suppose that a new flow enters into the network with initial throughput* x_0 ($x_0 \ll C/N$), *then after*

$$t_\varepsilon = \frac{\ln\ln\frac{C\varepsilon}{Nx_0}}{\alpha} \tag{29}$$

the network achieves ε *fairness.*

Proof. Let $x_{N+1}(t)$ be the new flow, and $x_j(t)$ ($j = 1, 2, \ldots, N$) be an existing one. Then the system can be described by:

$$\begin{cases} \dot{x}_{N+1}(t) = \alpha x_{N+1}(t) \cdot (\ln r(t) - \ln x_{N+1}(t)) \\ \dot{x}_j(t) = \alpha x_j(t) \cdot (\ln r(t) - \ln x_j(t)). \end{cases} \tag{30}$$

Based on Definition 2, $g(t) = x_j(t)/x_{N+1}(t)$. Then we derive:

$$\dot{g(t)} = -\alpha g(t)\ln g(t) \tag{31}$$

thus, we have

$$\ln\ln\dot{g}(t) = -\alpha. \tag{32}$$

Consider the initial condition $g(0) = \frac{C/N}{x_0} = \frac{C}{Nx_0}$, $g(t_\varepsilon) = \frac{1}{\varepsilon}$. We have:

$$t_\varepsilon = \frac{\ln\ln\frac{C\varepsilon}{Nx_0}}{\alpha}. \quad \square \tag{33}$$

Clearly, the time for AECP convergence to fairness is $O(\ln\ln\frac{C}{N})$ and it is of ultra exponential complexity.

4.3. Setting the parameters

In this section, we discuss the choice of parameters used by AECP and implement the alternating evolution model using the linearizing method of control theory proposed in [20]. We consider the network in which there are N synchronous long-flows and a single bottleneck link and omit the influence of delay.

Let the rate of each flow be $x(t)$, the pre-allocation rate factor be $r(t)$, and the queue length be $q(t)$. For simplicity, we use x, r, q to denote $x(t)$, $r(t)$ and $q(t)$. Suppose

$$\begin{cases} F(x, r, q) = \alpha x \cdot (\ln r - \ln x) \\ G(x, r, q) = \beta r \cdot \left(1 - \dfrac{Nx + (q - q_0)/T}{C}\right) \\ H(x, r, q) = Nx - C. \end{cases} \tag{34}$$

Suppose the equilibrium point is $M(x^*, r^*, q^*)$, where $x^* = r^* = C/N$, $q^* = q_0$, thus we linearize the Eq. (34) at point M, and we have

$$\begin{aligned} &\left.\frac{\partial F}{\partial x}\right|_{x=x^*} = -\alpha, \quad \left.\frac{\partial F}{\partial r}\right|_{r=r^*} = \alpha, \quad \left.\frac{\partial F}{\partial q}\right|_{q=q^*} = 0 \\ &\left.\frac{\partial G}{\partial x}\right|_{x=x^*} = -\beta, \quad \left.\frac{\partial G}{\partial r}\right|_{r=r^*} = 0, \quad \left.\frac{\partial G}{\partial q}\right|_{q=q^*} = -\frac{\beta}{NT} \\ &\left.\frac{\partial H}{\partial x}\right|_{x=x^*} = N, \quad \left.\frac{\partial H}{\partial r}\right|_{r=r^*} = 0, \quad \left.\frac{\partial H}{\partial q}\right|_{q=q^*} = 0. \end{aligned} \tag{35}$$

Suppose $\delta x = x - x^*$, $\delta r = r - r^*$, $\delta q = q - q^*$. Then we have:

$$\begin{cases} \delta\dot{x} = \alpha\delta r - \alpha\delta x \\ \delta\dot{r} = -\beta\delta x - \dfrac{\beta}{NT}\delta q \\ \delta\dot{q} = N\delta x. \end{cases} \tag{36}$$

Furthermore,

$$\begin{cases} \delta\dot{x} = \alpha\delta r - \alpha\delta x \\ \delta\dot{r} = -\dfrac{\beta}{N}\delta\dot{q} - \dfrac{\beta}{NT}\delta q \\ \delta\dot{q} = N\delta x. \end{cases} \tag{37}$$

As shown in Fig. 3, the closed-loop control system is composed of three components: the link algorithm acting as a PI controller component, in which the proportional constant is β/N, and the integral constant is T; the source algorithm acting as an inertia controller, in which the time constant is $1/\alpha$; the queue model acting as an integral controller. Therefore we can represent the following open-loop transfer function of the AECP transport system as:

$$\begin{aligned} L(s) &= \frac{\alpha}{s+\alpha} \cdot \frac{\beta}{N}\left(1 + \frac{1}{Ts}\right) \cdot \frac{N}{s} \\ &= \frac{\alpha\beta(s + \frac{1}{T})}{s^2(s+\alpha)}. \end{aligned} \tag{38}$$

In control theory, if the transfer function of an unadjusted system is

$$L_0(s) = \frac{K_0}{s(1 + T_0 s)} \tag{39}$$

then it can be adjusted by a PI controller whose transfer function is

$$L_c(s) = \frac{1 + \tau s}{T_i s} \tag{40}$$

to improve the performance of the system. Thus the final transfer function of the system is

$$L(s) = L_0(s)L_c(s) = \frac{K_0}{T_i}\frac{1 + \tau s}{s^2(1 + T_0 s)}. \tag{41}$$

In order to obtain the maximum stability margin and speed convergence as much as possible, the parameters in the PI controller takes

$$\tau = 4T_0, \qquad T_i = 8K_0T_0^2 \tag{42}$$

in general [12].

Comparing Eq. (38) with Eq. (41) and (42), we have

$$\beta = \alpha/2, \qquad T = \frac{4}{\alpha}. \tag{43}$$

From the analysis of convergence, we can see that the larger the α, the faster the convergence to efficiency and fairness. However we observed from simulations that the traffic also becomes more volatile when α gets large. To balance the convergence and oscillation, we set $\alpha = 2\ \mathrm{s}^{-1}$. Then we have $\beta = 1\ \mathrm{s}^{-1}$, $T = 2$ s.

5. Implementation

In order to practically apply this algorithm, it is necessary to derive discrete time equations from the differential equations (6).

Suppose T_1 and T_2 are the sampling time in the end node and in the router respectively and use the the discrete time, kT_1 and kT_2, to replace time t. Then we can make the following approximate transformation:

$$\begin{cases} \dfrac{\mathrm{d}\ln(x(t))}{\mathrm{d}t} \approx \dfrac{\ln x((k+1)T_1) - \ln x(kT_1)}{T_1} \\ \qquad = \dfrac{\ln x(k+1) - \ln x(k)}{T_1} \\ \dfrac{\mathrm{d}\ln(r(t))}{\mathrm{d}t} \approx \dfrac{\ln r((k+1)T_2) - \ln r(kT_2)}{T_1} \\ \qquad = \dfrac{\ln r(k+1) - \ln r(k)}{T_2} \end{cases} \tag{44}$$

where we use $x(k)$ and $r(k)$ to respectively denote $x(kT_1)$ and $r(kT_2)$ and omit the suffixes of the variables. Substituting the above equation into (6), we obtain:

$$\begin{cases} \dfrac{\ln x(k+1) - \ln x(k)}{T_1} = \alpha \cdot \left(\ln r\left(k - \dfrac{\tau}{T_1}\right) - \ln x(k)\right) \\ \dfrac{\ln r(k+1) - \ln r(k)}{T_2} = \beta \cdot \left(1 - \dfrac{\sum x(k) + (q(k) - q_0)/T}{C}\right). \end{cases} \tag{45}$$

Further we have,

$$\begin{cases} x(k+1) = x(k)^{1-\alpha T_1} \cdot r\left(k - \dfrac{\tau}{T_1}\right)^{\alpha T_1} \\ r(k+1) = r(k) \\ \qquad \times \exp\left\{\beta T_2 \cdot \left(1 - \dfrac{\sum x(k) + (q(k) - q_0)/T}{C}\right)\right\}. \end{cases} \tag{46}$$

In the implementation, we finally select $T_1 = \tau$ and $T_2 = 0.1$ s to guarantee the sampling precision. We also define q_0 be 100 packets in order to provide small queuing delay in high speed networks. This value is similar to some AQM algorithms [19]. Substituting the value of all parameters into the Eq. (46), we have the final congestion controller:

$$\begin{cases} x(k+1) = x(k)^{1-2\tau} \cdot r(k-1)^{2\tau} \\ r(k+1) = r(k) \\ \qquad \times \exp\left\{0.1\left(1 - \dfrac{\sum x(k) + (q(k) - 100)/2}{C}\right)\right\}. \end{cases} \tag{47}$$

From the above equation, we can see that the complex logarithm computation is excluded from the practical congestion control algorithm. Therefore the computational overhead is reduced. Meanwhile, the small sampling time in the router, 0.1 s, also reduces the router computational overhead compared with that of XCP, which needs to do computation for each packet.

5.1. Packet header

It is easier to implement the AECP algorithm with the AECP packet header. As shown in Fig. 4, the AECP header is inserted into the place between the IPv6 header and the TCP header. The "Next Header" field is used to identify the following protocol header (e.g., TCP header) and the length of "Next Header" field (8 bit) in AECP header equals to the "Next Header" field in the IPv6 header. The length of "Pre-allocation rate factor" field (16 bit) in the AECP header equals to the CWND field in the TCP header. Therefore the total length of the AECP header is 3 Bytes which is much shorter than IP header and the TCP header. The routers along the route modify the pre-allocation rate factor field to directly control the sending rate of the sender.

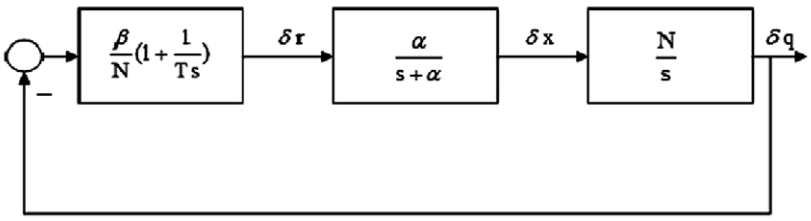


Fig. 3. Block of the close-loop control system.

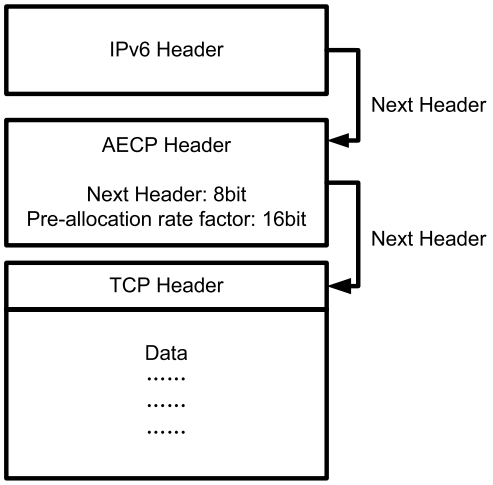


Fig. 4. AECP packet header.

5.2. Sender

On packet departures, the AECP sender initializes the pre-allocation rate factor to −1. Whenever a new ACK packet arrives, the sender reads the pre-allocation rate factor $r(k-1)$ from the ACK packet, and tracks the RTT τ. As a result the sender adjusts the sending rate as follows:

$$x(k+1) = x(k)^{1-2\tau} \cdot r(k-1)^{2\tau}. \tag{48}$$

5.3. Router

The router mainly generates its pre-allocation rate factor and inserts the factor into the headers of all passing packets. During the sampling interval, the router tracks the total amount of data that has arrived into the queue. At each sampling point, the router tracks the instantaneous queue length and computes the average incoming traffic rate $\sum x(k)$. Based on this information, the router estimates the pre-allocation rate factor as follows:

$$r(k+1) = r(k) \cdot \exp\left\{0.1\left(1 - \frac{\sum x(k) + (q(k) - 100)/2}{C}\right)\right\}. \tag{49}$$

In addition, the router checks if the locally pre-allocation rate factor is smaller than the one carried in the packet. If so, it replaces the corresponding field in the packet with the new value. In this manner, after traversing the whole path, each packet obtains the pre-allocation rate factor from the most congested link.

5.4. Receiver

An AECP receiver is similar to a TCP receiver except that when acknowledging a packet, the AECP receiver copies the extension AECP header from the data packet to its acknowledgment packet.

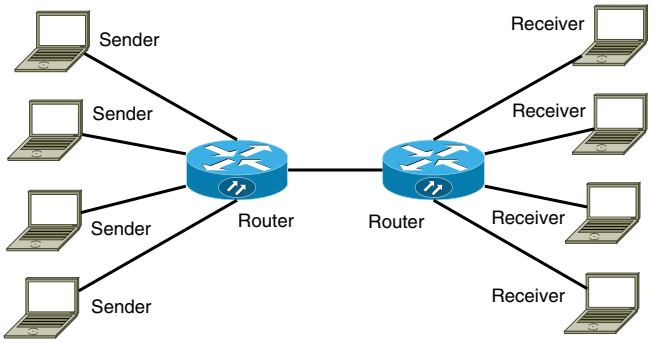


Fig. 5. Dumbbell topology.

6. Simulation results

In this section, we present some simulation results of the performance of AECP. We compare it with TCPSack, XCP, and VCP. We use ns2 for the simulation experiments. A tail-drop discipline at the router buffer is deployed and the buffer size is set to the product of bandwidth and delay. In all experiments the data packet size is 1000 bytes, while the ACK packet size is 40 bytes. For all the graphs, rate, utilization, packet loss rate and queue length are sampled over 1 s intervals.

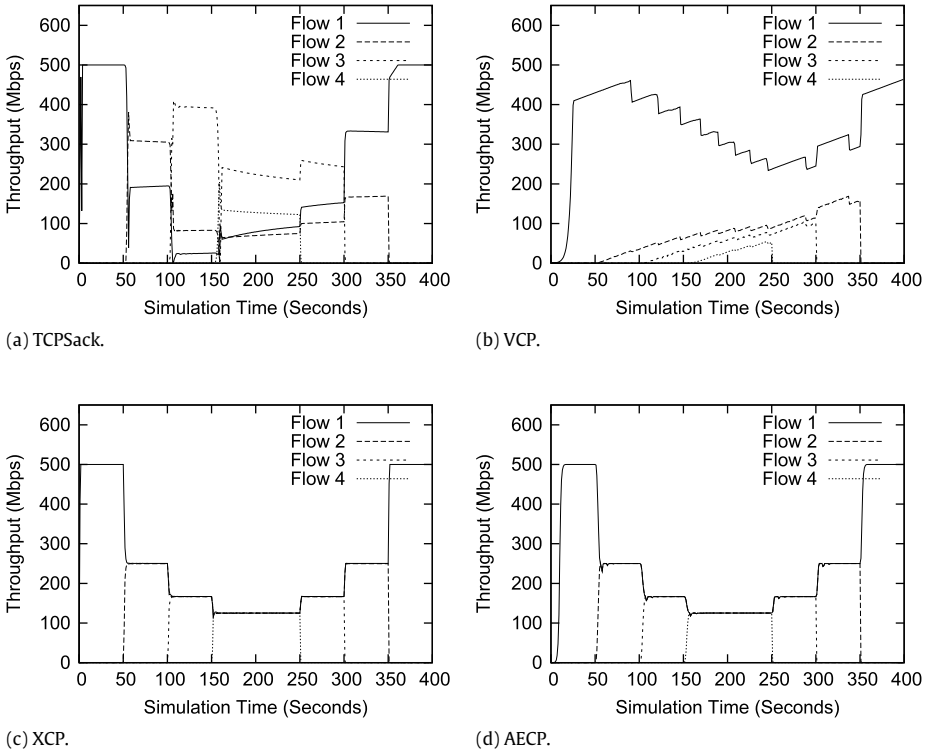
6.1. Convergence

In this experiment, we evaluate the performance of transport protocols in the simplest case where a single bottleneck link is shared by multiple flows. The dumbbell topology used here is depicted in Fig. 5. It consists of source/destination hosts, two routers, and links between the hosts and routers. We ran four flows with RTT of 50 ms, 100 ms, 200 ms, and 400 ms, while the bottleneck bandwidth is 500 Mbps. These flows start at 0 s, 50 s, 100 s and 150 s and stop at 400 s, 350 s, 300 s, 250 s, respectively. The rate curves are shown in Fig. 6.

As shown in Fig. 6, TCPSack cannot achieve max–min fairness and VCP is too slow to achieve max–min fairness because of the difference of RTT. XCP is the fastest no matter converging to efficiency or fairness. We have explained the reason in Section 1 that the convergence to efficiency and fairness of XCP is a constant convergence. AECP is a little bit slower than XCP, but it performs much more faster than TCPSack and VCP because the convergence to efficiency and fairness of AECP is an exponential convergence. Additionally, AECP and XCP both achieve max–min fairness independent of RTT.

6.2. Robustness

This experiment investigates the robustness of AECP in the presence of web, burst, and reverse traffics. The dumbbell topology is also used here, where the bottleneck bandwidth is 500 Mbps and the round trip propagation delay is 50 ms. For comparison purposes, two simulations are conducted.



(a) TCPSack. (b) VCP.

(c) XCP. (d) AECP.

Fig. 6. The rate dynamics of four flows with TCPSack, XCP, VCP and AECP.

The first simulation has only ten high speed flows on the forward path without disturbing traffic. The first flow starts at time zero, and the other flows enter into the network at five second intervals. In the second simulation, in addition to the ten high speed flows as the first simulation, there are another ten high speed flows on the backward path, and the average web traffic of 80 Mbps generated by 400 random on–off sources is always on. These web flows arrive according to the Poisson process. Moreover, burst CBR traffic of 160 Mbps generated by twenty UDP sources is injected into the network at 100 s, and then all UDP sources drop out at 200 s.

The rate dynamic curves in the first simulation and the second simulation are shown in Figs. 7 and 8 respectively. For the clarification of curves, we only show the fifth flow and the tenth flow in the figures. The average utilization, average packet loss rate and average queue length of the bottleneck link of the first simulation and the second simulation are listed in Table 1.

As shown in Fig. 7 and the S1 columns in Table 1, AECP and XCP converge to an equal rate in steady state, and the average utilizations of the bottleneck link are very high in the absence of disturbing traffic. In addition, the average queue length of AECP approaches the expected queue length q_0. However, Fig. 8 and the S2 columns in Table 1 show that TCPSack and XCP oscillate significantly in the presence of disturbing traffic. Moreover, the average utilizations of TCPSack, VCP and XCP degrade sharply, and the average loss rates increase obviously. Fig. 8 shows that the AECP flows converge to an equilibrium rate of 42 Mbps even with web traffic. When burst traffic appears at 100 s, the AECP flows give up the bandwidth rapidly. At 105 s, the AECP flows converge to a new equilibrium rate (26 Mbps). After the burst traffic leaves at 200 s, the AECP algorithm grapes the available bandwidth and converges to the previous equilibrium rate quickly. Table 1 also shows that AECP achieves better robust, higher link utilization and lower packet loss rate than the other protocols even in the presence of crossing traffic.

Table 1
The average utilization, average packet loss rate and average queue length.

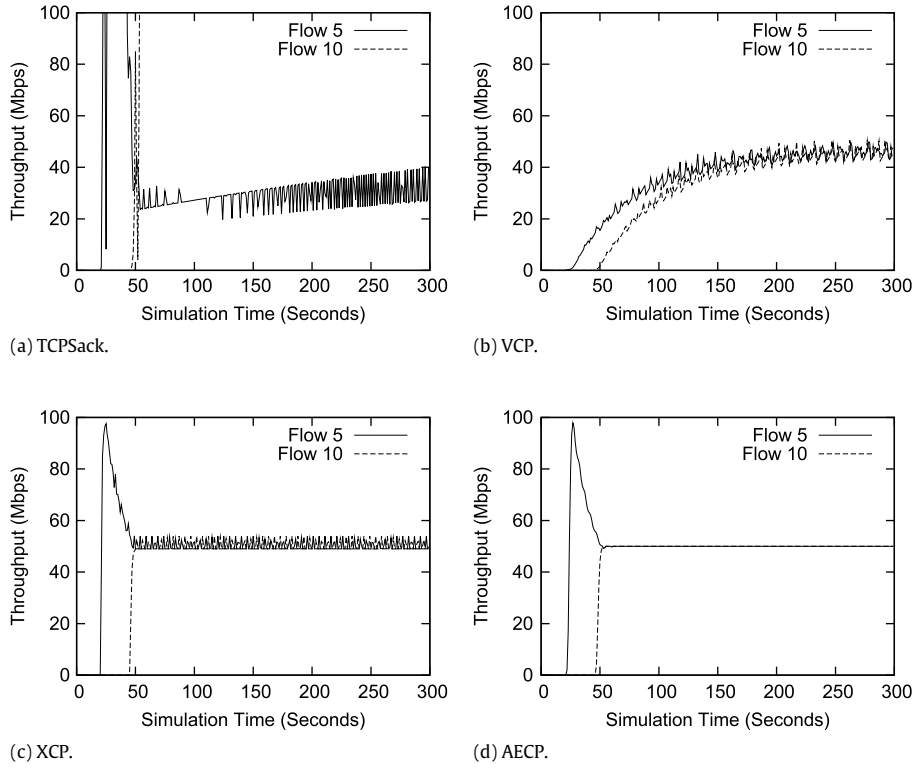
Protocol	Utilization (%)		Packet loss ratio		Queue length (Package)	
	S1	S2	S1	S2	S1	S2
TCPSack	100	87.14	2.47×10^{-4}	3.81×10^{-2}	19 443	3888
VCP	93.06	89.95	0	0	0.5	238
XCP	99.97	94.28	0	2.64×10^{-3}	2.0	3315
AECP	100	98.64	0	6.75×10^{-4}	100	1358

S1: The first simulation in the absence of crossing traffic.
S2: The second simulation in the presence of crossing traffic.

6.3. Multiple bottlenecks

Next, we study the performance of AECP with a more complex topology of multiple bottlenecks. For this purpose, we use a typical multiple bottlenecks topology with three links depicted in Fig. 9. All the links have a 20 ms one-way propagation delay. One high speed flow (i.e., flow 1) traverses all the links in the forward direction. In addition, each individual link has one crossing high speed flow (i.e., flow 2, flow 3, and flow 4) traversing in the forward direction. The middle link has the smallest bandwidth of only 250 Mbps, and the other links have the same bandwidth of 500 Mbps. All flows start at time zero.

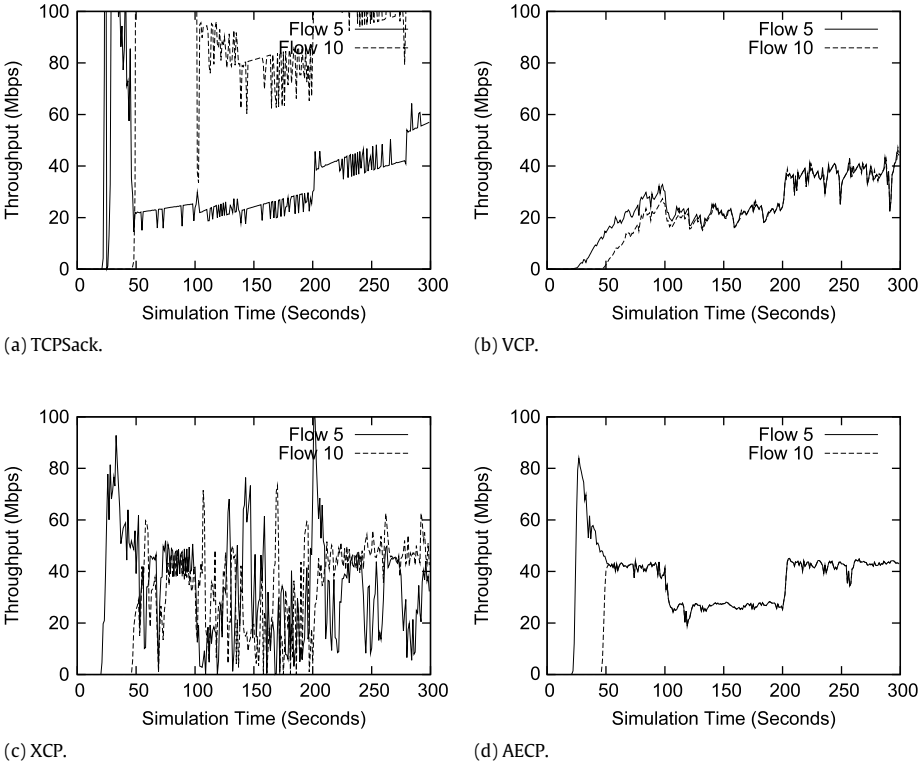
As shown in Fig. 10, the long flow in TCPSack cannot catch any bandwidth resource from the short flow. In VCP, the throughputs of flow 1, flow 2, flow 3, and flow 4 are 67 Mbps, 369 Mbps, 174 Mbps, and 369 Mbps, respectively. These results show that the fairness of TCPSack and VCP in multiple bottlenecks networks is really poor. In XCP, the throughput of flow 1 and flow 3 are 125 Mbps, but the throughput of flow 2 is 338 Mbps and the throughput of flow 4 is 365 Mbps in steady state. In AECP, the throughputs of flow 1 and flow 3 both achieve 125 Mbps and the throughput of flow 2 and flow 4 are 375 Mbps in steady state. In other words, AECP completely achieves max–min fairness while XCP approximately achieves max–min fairness.



(a) TCPSack. (b) VCP.

(c) XCP. (d) AECP.

Fig. 7. The rate dynamics of ten flows using TCPSack, VCP, XCP and AECP in the absence of web traffic, burst traffic and reverse traffic.



(a) TCPSack. (b) VCP.

(c) XCP. (d) AECP.

Fig. 8. The rate dynamics of ten flows using AECP and XCP in the presence of web traffic, burst traffic and reverse traffic.

7. Conclusion

In this paper we propose a new alternating evolution algorithm, which consists of the link and source algorithms for congestion control in high speed networks. The algorithm is also based on the explicit bandwidth pre-allocation factor, which is carried by the packet and is computed in routers according to the information of the router capacity, the aggregate load, and the instantaneous queue length. Therefore the routers do not need to keep the per-flow state and the overhead is low. The senders asymptotically

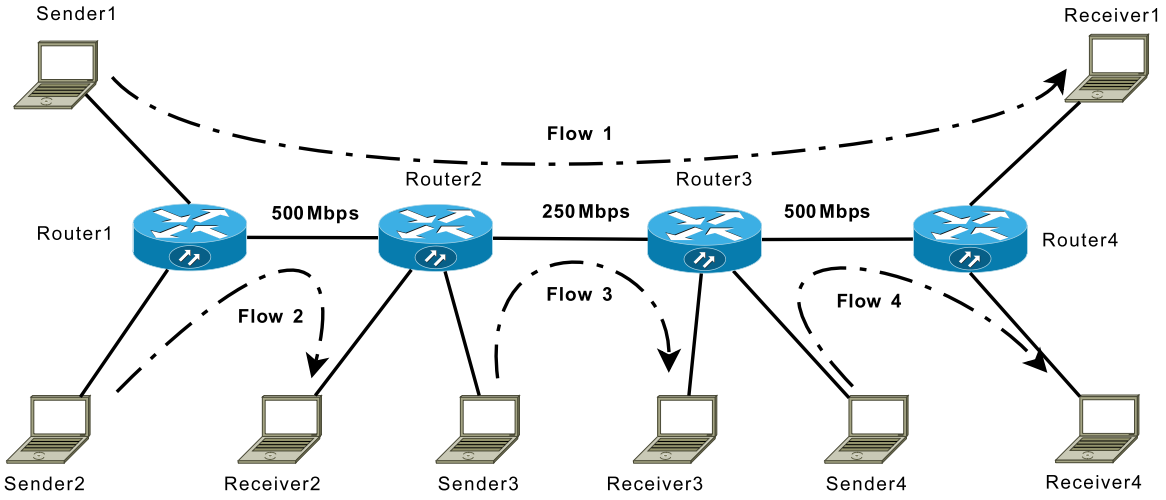
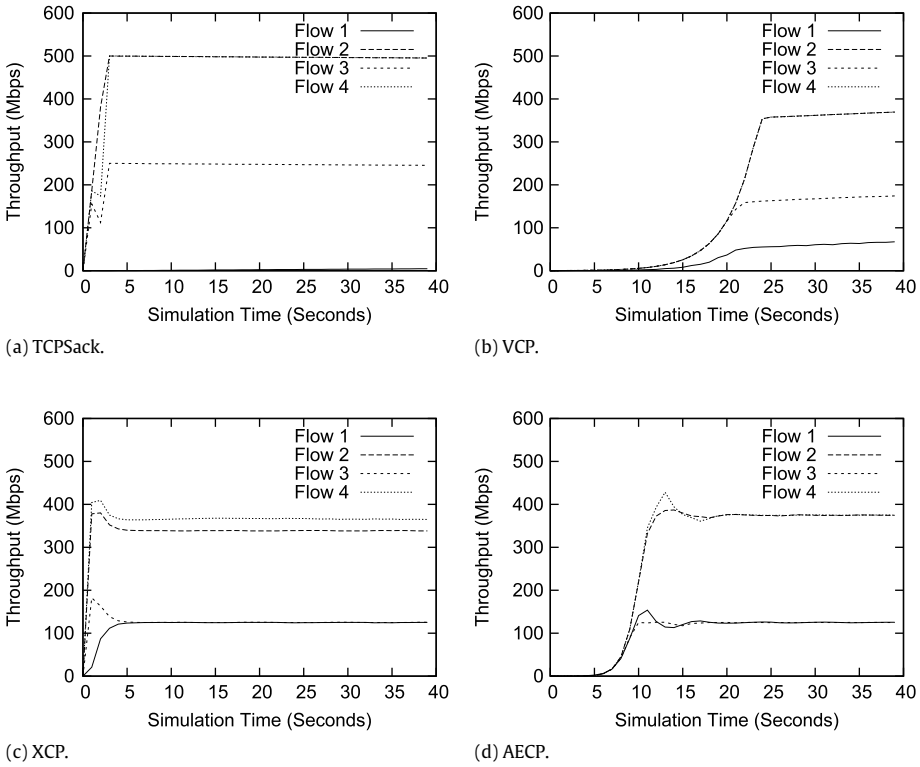


Fig. 9. Multiple bottlenecks topology.



(a) TCPSack. (b) VCP.

(c) XCP. (d) AECP.

Fig. 10. The rate dynamics of four flows using different protocol in a multiple bottleneck topology.

adjust the sending rate according to the pre-allocation bandwidth factor. After that the pre-allocation bandwidth factor changes asymptotically along with the varying aggregate load and the queue length in routers. Thereby the sending rate and the pre-allocation bandwidth factor form the alternating evolution and both achieve a steady state eventually.

Theoretic analysis and simulation experiments show that the algorithm is able to effectively and efficiently allocate bandwidth. The performance of our algorithm is evaluated through simulation in terms of convergence, robustness, fairness, queue length, link utilization, and packet loss ratio. The simulation results show that AECP not only provides fast convergence to efficiency and fairness, but is also strongly robust to crossing traffic, which are desirable for high speed networks.

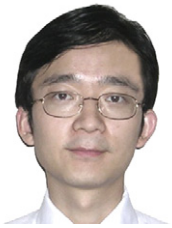
Acknowledgments

This work is co-sponsored by Natural Science Foundation of China (60803121, 60673152, 60773145), National High-Tech R&D (863) Program of China (2006AA01A101, 2006AA01A106, 2006AA01A108), Tsinghua National Laboratory for Information Science and Technology (TNLIST) Cross-discipline Foundation and NEC (China) Co. Ltd.

References

[1] M. Allman, W. Stevens, TCP congestion control, RFC 2581.
[2] R.B. Banks, Growth and Diffusion Phenomena: Mathematical Frameworks and Applications, Springer, Berlin, 1994.

[3] D. Bertsekas, R. Gallager, Data Networks, 2nd ed., Prentice-Hall, 1991.
[4] D.M. Chiu, R. Jain, Analysis of the increase and decrease algorithms for congestion avoidance in computer networks, Computer Networks and ISDN Systems 17 (1989) 1–14.
[5] S. Deb, R. Srikant, Global stability of congestion controllers for the internet, in: 2002 IEEE Conference on Decision and Control, 2002.
[6] W. Feng, K. Shin, D. Kandlur, D. Saha, The blue active queue management algorithms, IEEE/ACM Transactions on Networking 10 (4) (2002) 513–528.
[7] S. Floyd, High-speed TCP for large congestion windows, RFC 3649.
[8] Sally Floyd, Van Jacobson, Random early detection gateways for congestion avoidance, IEEE Transactions on Networking 1 (4) (1993) 397–413.
[9] S. Floyd, S. Ratnasamy, S. Shenker, Modifying TCP's congestion control for high speeds network[EB/OL]. (2002-03-01) [2007-03-01]. http://www.icir.org/floyd/notes.html.
[10] A. Gruebler, N. Nakicenovic, Diffusion of Technologies and Social Behavior, Springer, Berlin, 1991.
[11] C.V. Hollot, Vishal Misra, Donald F. Towsley, Weibo Gong, A control theoretic analysis of RED, in: INFOCOM'01, 2001, pp. 1510–1519.
[12] Shousong Hu, Principle of Automatic Control, 4th ed., Science Publisher, 2001, (in Chinese).
[13] Xiaomeng Huang, Chuang Lin, Fengyuan Ren, A novel high speed transport protocol based on explicit virtual load feedback, Computer Networks 51 (7) (2007) 1800–1814.
[14] V. Jacobson, Congestion avoidance and control, Computer Communication Review 18 (4) (1988) 314–329.
[15] S. Jin, L. Guo, I. Matta, A. Bestavros, TCP-friendly SIMD congestion control and its convergence behavior, in: ICNP'01, 2001.
[16] D. Katabi, M. Handley, C. Rohrs, Congestion control for high bandwidth delay product networks, in: ACM SIGCOMM'02, August 2002.
[17] T. Kelly, Scalable TCP: Improving performance in high-speed wide area networks, in: The 1st Workshop on Protocols for Fast Long-Distance Networks, 2003.
[18] F. Kelly, Mathematical modeling of the Internet, Mathematics Unlimited, 2001.
[19] S. Kunniyur, R. Srikant, An adaptive virtual queue (avq) algorithm for active queue management, IEEE/ACM Transactions on Networking 12 (2) (2004) 286–299.
[20] Benjamin C. Kuo, Farid Golnaraghi, Automatic Control Systems, 8th ed., Wiley, New York, 2003.
[21] T. Lakshman, U. Madhow, The performance of tcp/ip for networks with high bandwidth-delay products and random loss, IEEE/ACM Transactions on Networking 5 (3) (1997) 336–350.
[22] D. Loguinov, H. Radha, End-to-end rate-based congestion control: Convergence properties and scalability analysis, IEEE/ACM Transactions on Networking 11 (4) (2003) 564–577.
[23] Dmitri Loguinov, Yueping Zhang, Derek Leonard, Jetmax: Scalable max–min congestion control for high-speed heterogeneous networks, Computer Networks 52 (6) (1988) 1193–1219.
[24] A.J. Lotka, Elements of Physical Biology, Williams and Wilkins Co., Baltimore, 1925.
[25] Steven H. Low, David E. Lapsley, Optimization flow control I: Basic algorithm and convergence, IEEE/ACM Transactions on Networking 7 (6) (1999) 861–874.
[26] Vishal Misra, Wei-Bo Gong, Donald F. Towsley, Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED, in: ACM SIGCOMM'00, 2000, pp. 151–160.
[27] Jeonghoon Mo, Jean Walrand, Fair end-to-end window-based congestion control, IEEE/ACM Transactions on Networking 8 (5) (2000) 556–567.
[28] J. Postel, Transmission control protocol, RFC 793.
[29] Scott Shenker, Fundamental design issues for the future internet, IEEE Journal on Selected Areas in Communication 13 (7) (1995).
[30] P.F. Verhulst, Recherches math matiques sur la loi d'accroissement de la population, Nouveaux meacute de l'Academie Royale des Sci. et Belles-Lettres de Bruxelles, 1845.
[31] G. Vinnicombe, On the stability of end-to-end congestion control for the internet. http://www-control.eng.cam.ac.uk/gv/internet/TR398.pdf.
[32] Michael Welzl, Scalable Performance Signalling and Congestion Avoidance, Kluwer Academic Publishers, 2003.
[33] J. Widmer, R. Denda, M. Mauve, A survey on TCP-friendly congestion control, IEEE Network 15 (3) (2001) 28–37.
[34] B. Wydrowski, Lachlan L.H. Andrew, Moshe Zukerman, MaxNet: A congestion control architecture for scalable networks, IEEE Communications Letters 7 (10) (2003) 511–513.
[35] Y. Xia, L. Subramanian, I. Stoica, S. Kalyanaraman, One more bit is enough, in: ACM SIGCOMM'05, 2005.
[36] L. Xu, K. Harfoush, I. Rhee, Binary increase congestion control for fast long distance networks, in: INFOCOM'04, 2004.
[37] Yueping Zhang, SeongRyong Kang, Dmitri Loguinov, Delayed stability and performance of distributed congestion control, in: ACM SIGCOMM'04, 2004.
[38] Yueping Zhang, Derek Leonard, Dmitri Loguinov, JetMax: Scalable maxmin congestion control for high-speed heterogeneous networks, in: INFOCOM'06, 2006.
[39] Rui Zhang-Shen, Nandita Dukkipati, Masayoshi Kobayashi, Nick McKeown, Processor sharing flows in the internet, in: IWQos'05, 2005.
[40] Jiang Zhu, Chia-Hui Tai, Nandita Dukkipati, Making large scale deployment of rcp practical for real networks, in: INFOCOM'08, 2008.



Xiaomeng Huang is an Assistant Professor of Department of Computer Science and Technology, Tsinghua University, Beijing, China. He received the Ph.D. degree of the Department of Computer Science and Technology, Tsinghua University, China in 2007. He is especially interested in large-scale distributed system, computer networks and etc.

Yongwei Wu is an Associate Professor of Computer science and Technology, Tsinghua University, Beijing, China. He received the Ph.D. degree in applied mathematics from the Chinese Academy of Sciences in 2002. He is especially interested in grid and cloud computing, distributed processing, and parallel computing.

Guangwen Yang is Professor of Computer Science and Technology, Tsinghua University, Beijing, China. He received the Ph.D. degree of the Department of Computer Science, Harbin Institute of Technology University, China in 1996. He is mainly engaged in the research of grid computing and parallel and distributed computing.

Weiming Zheng is a Professor of computer science and technology, Tsinghua University, China, where he received the B.S. and M.S. degrees in 1970 and 1982 respectively. His research interests include computer architecture, operating system, storage networks, and distributed computing.

Jinlei Jiang is an Assistant Professor with Department of Computer Science and Technology, Tsinghua University, China. He received a Ph.D. degree in computer science and technology from Tsinghua University in January 2004 with an honor of excellent dissertation. His research interests mainly focus on grid and cloud computing, computer-supported cooperative work, workflow management and services computing.