PAPER

An Asynchronous and Distributed Rate Control Mechanism for Elastic Services with Session Priorities

Tae-Jin LEE[†a)] ***and*** **Gustavo DE VECIANA**[††], ***Nonmembers***

SUMMARY We consider a rate control algorithm for elastic services to allocate bandwidth in a network subject to throughput and fairness constraints. Our algorithm achieves a max-min fair bandwidth allocation among contending elastic connections, and has desirable properties in that it can operate in a decentralized and asynchronous manner accounting in part for heterogeneity in round trip delays. The algorithm is simple and scalable in that, 1) the network links make local measurements of capacity and calculate local 'explicit rates' which are fed back to sources without requiring knowledge of the number of on-going connections, while 2) sources adjust their transmission rates so as not to exceed the received explicit rate indication. The algorithm is designed to track a "dynamic" network environment. We discuss its stability, convergence, and feasibility issues related to fair allocation and rate-based flow control. We also consider the role of sessions with priorities to differentiate among users with elastic services. Through rigorous analysis and simulations, we have shown that it has indeed desirable characteristics for networks with elastic services as well as other service types, which are expected to be common in future network environment.

key words: *rate control, max-min fairness, elastic services, asynchronous and distributed algorithm*

1. Introduction

Communication network resources, e.g., bandwidth, tend to be highly variable due to various service classes supported in high performance networks and due to dynamically varying network usage and congestion status. In such dynamic network environment, available bandwidth, remaining bandwidth in a network after allocating bandwidth to the stringent service classes (e.g., constant-rate services), is the useful resource to both users and service providers. It can be allocated among the users who are willing to receive less-stringent quality of service with less expensive cost. In this context, users can use the bandwidth efficiently with little sacrifice of service performance while network service providers are able to maximize the utilization of network resources or the revenue. ABR services in ATM networks and TCP in the Internet are among such *elastic services* [1].

The question of whether rate control mechanisms for elastic services should (or would) achieve a 'fair' allocation of resources among users sharing a network, has been the focus of both intensive research and debate [2]. There are two major views on the meaning of fairness, leading to alternative approaches to network control. The first, called *max-min fairness*, attempts to make the network transparent to users, i.e., resources are allocated so as to maximize the minimum throughput of users contending for network resources [3]. More general definitions of this type of fairness, might give priority, or weights to users, but have essentially the same structure [4]. The second approach, is economic in nature, and attempts to allocate resources so as to maximize the sum of the user's utilities—assuming such utility functions are available. Kelly [5] refers to the associated allocation as being *proportionally fair* and discusses cases where these two criteria coincide. Intuitively, in this case the throughput achieved by various users will in general depend on the number of bottleneck links the connections share. In a sense, max-min fairness attempts to maximize the worst case *individual* user performance, while the second approach maximizes the network's *overall* utility to users at the possible expense of some individuals. We will focus on the problem of achieving max-min fairness.

In order to support such elastic services in networks, typical mechanisms have relied on users' rate adjustment based on implicit or explicit feedback on network status, e.g., congestion and bandwidth availability. Many researchers have studied how to realize such mechanisms assuming fixed number of elastic connections. For research results and survey of ABR rate-based flow control, see [2], [6]–[9], and references therein. Some proposed mechanisms [10]–[14] show that they can achieve a notion of equilibrium point, i.e., fair bandwidth allocation [3] among elastic connections. Feedback control mechanisms for rate allocation of elastic connections to a single node have been considered in [15], [16]. A window-based congestion control achieving proportional fairness has been proposed in [17]. Rate control mechanisms should be designed to be simple and scalable in order to be viable in a large-scale networking environment where there are strong limitations on the complexity of the algorithms that can be implemented, see e.g., [9]. Recently, we have observed some results on this issue, e.g., a control theoretic framework to achieve max-min fair allocation of bandwidth [18] and a distributed rate allocation algorithm based on stochastic approximation of available capacity [19].

Several issues need to be addressed for rate control algorithms for elastic services:

- The *explicit rate control mechanism* should have fast

Manuscript received June 16, 2003.
Manuscript revised January 7, 2004.

[†]The author is with the School of Information and Communication Engineering, Sungkyunkwan University, 300 Cheoncheon-Dong, Jangan-Gu, Suwon City, Kyungki-Do 440-746, Korea.
[††]The author is with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78712, USA.
a) E-mail: tjlee@ece.skku.ac.kr

convergence characteristics.
- A *simple algorithm* is preferred to make the complexity of rate control reasonable.
- *Priorities* to connections are able to provide further *service differentiation*.
- In a large-scale network environment, a *distributed and asynchronous algorithm* is desirable.
- *Fairness* [2] needs to be provided to treat elastic connections fairly.
- It is desirable to *minimize the amount of information* required (e.g., number of on-going connections and status of links).

In this paper, we propose and investigate a simple, distributed and asynchronous flow control mechanism using explicit rate. We show rigorously that it achieves quickly the notion of max-min fair allocation of bandwidth. Our mechanism is *simple*, i.e., network switches only needs to compute total flow of elastic traffic and that of other traffic, and neither per-VC queueing nor per-VC counting of elastic connections is required. In addition, it is *distributed* and thus *scalable* to large scale networks. It is also *asynchronous*, i.e., network elements updates feedback information independently with one another, which commensurates with complex and independent network behaviors. Finally, it can be directly extended to encompass *prioritized services* via the notion of weighted max-min fairness. Our proposed mechanism extensively addresses the above requirements that a rate allocation algorithm for elastic services in multiservice networks should have. We demonstrate feasibility of our mechanism by rigorous analysis and simulations.

This paper is organized as follows. In Sect. 2 we review some notions related to max-min fairness that will be useful in the sequel and propose the synchronous algorithm. We show that the algorithm has a unique fixed point and it achieves max-min fair bandwidth allocation. Moreover we present a totally asynchronous version of the algorithm and its convergence to the same max-min fairness in Sect. 3. Then, we consider the role of round trip delays between sources and links. We extend the algorithm to the one with session priorities leading to the notion of *weighted max-min fairness* in Sect. 4. As a demonstration of our algorithm, we conduct simulations in Sect. 5, and conclude in Sect. 6.

2. Proposed Synchronous Algorithm

Our starting point is a simple mechanism for flow control proposed in [20]. The rationale for the mechanism is as follows (see Fig. 1) : suppose that n connections share a link (switch) with capacity c. If the capacity is to be shared evenly by the connections, then the fair rate $e(t)$ for each session, called "explicit rate," should be c/n. Assuming the sessions send traffic at this explicit rate, the link flow (typically measured) will be n times $e(t)$, i.e., $f(t) = ne(t)$. Now, since the number of active connections n may be unknown, we might estimate the number implicitly rather than monitoring it explicitly as other rate-based control schemes do [9], [16]. One can estimate the number of active connections using $\hat{n}(t) = f(t)/e(t)$. The explicit rate is then computed based on the estimated number, i.e., $e(t+1) = c/\hat{n}(t)$. Due to the capacity constraint, it is desirable to ensure that $e(t)$ can not exceed the link capacity c, that is, $e(t+1) = \min[\, c/\hat{n}(t), c\,]$. It may be preferable to limit the $e(t)$ to be even smaller than c, e.g., peak session rate for any connection in the network. We will see that this surprisingly simple mechanism can be extended to a network setting.

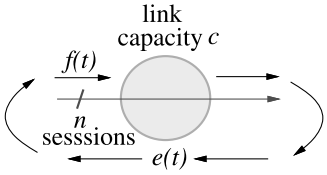

Fig. 1 A network with one link and n sessions (unconstrained sessions).

We consider a network consisting of a set of buffered links $\mathcal{L}$ each with a (typically measured) current bandwidth availability $\vec{c} = (c_\ell, \ell \in \mathcal{L})$. Suppose a set $\mathcal{S}$ of sessions share the network, where each session $s \in \mathcal{S}$ has a set of links $\mathcal{L}_s$ associated with it. The set $\mathcal{L}_s$ is intended to define an end-to-end connection through the network. More than one session might share each link, thus we let $\mathcal{S}_\ell$ be the set of sessions crossing link ℓ[†].

Suppose each link $\ell \in \mathcal{L}$ measures the aggregate flow $f_\ell(t)$ it is currently supporting, and computes a local 'explicit rate' $e_\ell(t)$ based on an estimated effective number of connections. In a scenario with greedy sources the session rates $a_s(t)$ are adjusted to be the smallest among the explicit rates of all links along the route of the session. We propose the following distributed and synchronous algorithm:

$$a_s(t) = \min_{\ell \in \mathcal{L}_s}[\, e_\ell(t)\,], \qquad s \in \mathcal{S}, \tag{1}$$

$$f_\ell(t) = \sum_{s \in \mathcal{S}_\ell} a_s(t), \qquad \ell \in \mathcal{L}, \tag{2}$$

$$e_\ell(t+1) = \min\left[\, \frac{c_\ell e_\ell(t)}{f_\ell(t)}, c_\ell \,\right], \qquad \ell \in \mathcal{L}. \tag{3}$$

In our mechanism, each intermediate link (switch) ℓ along the route from a source to a destination computes the aggregate flow $f_\ell(t)$, which conveys the total sum of session rates $a_s(t)$ from the sessions passing through link ℓ (see (2)). Note that the flow measurement at each switch does not require the information on the number of sources (connections) since the only thing each switch needs to do is to count the total number of cells (packets) passing through the switch and compute the aggregate flow. So switches need not know individual $a_s(t)$ in detail. Based on the aggregate flow at each link ℓ, the switch estimates the explicit rate $e_\ell(t+1)$ that it can sustain (see (3)). The minimum of these explicit rates along the route from a source to a destination is chosen to be the session rate to which the source needs to adjust (see (1)). Thus the rate adjustment for the sessions

[†]In general one might allow for a multi-point session, say s, by allowing the set $\mathcal{L}_s$ to be a rooted tree on the network.

(end hosts) $a_s(t)$ and the aggregate flows $f_\ell(t)$ are captured by the proposed iterative algorithm, which extends the idea of computing explicit rate in the single link network.

The goal of this type of rate adjustment is to ensure that capacities are fully exploited while achieving max-min fair rate allocation. Note that the explicit rate at each link ℓ is updated in a *decentralized* manner using local information c_ℓ, $e_\ell(t)$ and $f_\ell(t)$ and exchanges of information along each session's path (rate adjustments) rather than requiring exchanges of global states, e.g., whether each session is constrained or not at the link. The algorithm has clear advantages in terms of minimizing the information required to determine the max-min fair allocation in that 1) it needs not keep track of the number of active connections and 2) it needs not maintain information on which sessions are constrained at each link. We shall show that the fixed point equation associated with the iterative algorithm (1)–(3) has a unique solution which is the max-min fair allocation. And the iterative synchronous algorithm is shown to converge geometrically to the fixed point, i.e., max-min fair allocation.

In order to investigate that the proposed algorithm achieves max-min fair allocation of bandwidth, we review the notion of max-min fairness. The main idea underlying max-min fairness can be explained as follows: each connection crossing a link should get as much bandwidth as other such connections unless that session is constrained elsewhere. In other words, available resources are allocated equally among unconstrained sessions. Max-min fairness has the following characteristics:

- each session has a bottleneck link;
- and, unconstrained sessions at a given link are given their equal share of the available capacity.

To formally define max-min fairness, we will use **the bottleneck** property [3]. It will be useful to consider the max-min fair allocation in terms of a *hierarchy* of sets of bottleneck links and sessions [4] and *fair shares* [21].

2.1 Existence and Uniqueness

We shall assume the following:

Assumption 2.1: (Bottleneck Link Assumption) Each bottleneck link has at least one session for which it is the unique bottleneck link.

This implies that sessions might have more than one bottleneck link, but if this is the case, each of the bottleneck links should carry at least one session for which it is the unique bottleneck. Assumption 2.1 is a little weaker than that in [3], but more generalized than that in [4], wherein it is assumed "single" bottleneck link per session.

Define $\vec{e} = (e_\ell, \ell \in \mathcal{L})$ and $\vec{a} = (a_s, s \in \mathcal{S})$. Consider the following fixed point equation derived from the iterative algorithm (1)–(3)

$$\vec{e} = g(\vec{e}) = (g_\ell(\vec{e}), \ell \in \mathcal{L}) \tag{4}$$

where

$$e_\ell = \min\left[\frac{c_\ell e_\ell}{f_\ell}, c_\ell\right] = g_\ell(\vec{e}) \quad \text{for all} \quad \ell \in \mathcal{L}, \tag{5}$$

and where

$$f_\ell = \sum_{s\in\mathcal{S}_\ell} a_s,\ \ell \in \mathcal{L} \quad \text{and} \quad a_s = \min_{\ell\in\mathcal{L}_s}[\ e_\ell\ ],\ s \in \mathcal{S}. \tag{6}$$

We show the existence and uniqueness of a solution $\vec{e}^*$ to the fixed point equation (5), and further establish that the corresponding rate allocation $\vec{a}^*$ obtained by (6) is unique and satisfies the max-min fairness criterion.

Theorem 2.1: (Existence and Uniqueness) Suppose Assumption 2.1 holds, then the fixed point equation (5) has a unique solution $\vec{e}^* = (e^*_\ell, \ell \in \mathcal{L})$. The associated session rates $\vec{a}^* = (a^*_s, s \in \mathcal{S})$ satisfy the max-min fairness criterion, and are thus unique.

Proof: Let $\vec{0}$ denote zero vector with same dimension as $|\mathcal{L}|$. Since $\mathbb{E} = \{\vec{e} \in \mathbb{R}^{|\mathcal{L}|} |\ \vec{0} \le \vec{e} \le \vec{c}\}$ is compact and $g : \mathbb{E} \to \mathbb{E}$ is continuous, it follows by the Brouwer Fixed Point Theorem [22] that (5) has at least one solution. It follows from (5) that for any link $\ell \in \mathcal{L}$,

$$e^*_\ell = \min\left[\frac{c_\ell e^*_\ell}{f^*_\ell}, c_\ell\right] \Rightarrow \begin{cases} e^*_\ell = c_\ell & \text{if } f^*_\ell < c_\ell \\ e^*_\ell = c_\ell \frac{e^*_\ell}{f^*_\ell} & \text{if } f^*_\ell = c_\ell, \end{cases}$$

thus we have that

$$f^*_\ell = \sum_{s\in\mathcal{S}_\ell} a^*_s,\ \ell \in \mathcal{L} \qquad \text{and} \qquad a^*_s = \min_{\ell\in\mathcal{L}_s}[\ e^*_\ell\ ],\ s \in \mathcal{S}.$$

We will show that the session rate allocation $\vec{a}^*$ corresponds to a max-min fair allocation. Consider an arbitrary session $s \in \mathcal{S}$, we show that it has at least one bottleneck link. Consider $s \in \mathcal{S}$, then $a^*_s = \min_{\ell\in\mathcal{L}_s}[\ e^*_\ell\ ] = e^*_{\ell^*}$ for some $\ell^* \in \mathcal{L}_s$. Suppose that the link flow $f^*_{\ell^*} < c_{\ell^*}$, but then $a^*_s = e^*_{\ell^*} = c_{\ell^*}$, which contradicts $f^*_{\ell^*} < c_{\ell^*}$. Thus $f^*_{\ell^*} = c_{\ell^*}$. Now consider the sessions through ℓ^*. For each such session $r \in \mathcal{S}_{\ell^*}$, either $a^*_r = e^*_{\ell^*} = a^*_s$ ("constrained" at link ℓ^*) or $a^*_r < e^*_{\ell^*} = a^*_s$ (constrained elsewhere). Thus $a^*_s \ge a^*_r$ for all $r \in \mathcal{S}_{\ell^*}$, whence ℓ^* is a bottleneck link for session s. Therefore, $\vec{a}^*$ is a max-min fair allocation which is unique by the Theorem in [21].

Now, consider a solution $\vec{e}^*$. The explicit rate $e^*_{\ell^*}$ at each bottleneck link ℓ^* must be unique since by Assumption 2.1 the link is the only bottleneck for at least one session s of which the session rate is unique, i.e., $a^*_s = \min_{\ell\in\mathcal{L}_s}[\ e^*_\ell\ ] = e^*_{\ell^*}$, and the explicit rate of non-bottleneck link is its link capacity c_ℓ which is unique. So the uniqueness of the solution $\vec{e}^*$ follows. ■

2.2 Synchronous Iterative Algorithm without Delayed Information

In this subsection, we assume that explicit rate updates and flow adjustments occur synchronously on some discrete time step. In other words, the explicit rates at links are updated exactly at the same time.

Theorem 2.2: (Convergence of Synchronous Iterative Algorithm) Suppose Assumption 2.1 holds, then the explicit rates $\vec{e}(t) = (e_\ell(t), \ell \in \mathcal{L})$ in the iteration (1)–(3) converge geometrically to the fixed point $\vec{e}^*$ of (2.1) and the associated session rates $\vec{a}^*$ achieve the max-min fair rate allocation.

Proof: Recall the hierarchy of bottleneck links and sessions defined in Sect. 2. Note that $\mathcal{U}^{(i)}$ is the cumulative set of bottleneck links in levels 1 to i and $\mathcal{V}^{(i)}$ is the cumulative set of bottleneck sessions in levels 1 to i. Theorem 2.2 is a consequence of Lemma 2.1. ■

Lemma 2.1: (Convergence of Explicit Rates and Session Rates) Given an initial vector $\vec{e}(0)$, there exists a time t_N, where N is the number of hierarchy levels, such that for all $t \geq t_N$, the explicit rates of bottleneck links $\ell \in \mathcal{U}^{(N)}$ and the associated session rates $s \in \mathcal{V}^{(N)}$ converge geometrically to e^*_ℓ and a^*_s, respectively. Moreover, the explicit rates of non-bottleneck links $\ell \in \mathcal{L} \setminus \mathcal{U}^{(N)}$ also converge geometrically to the corresponding link capacities c_ℓ for $t \geq t_{N+1}$, where $t_{N+1} \geq t_N$, that is

$$\max_{\ell \in \mathcal{U}^{(N)}} |e_\ell(t) - e^*_\ell| \leq A_N \gamma_N^t,$$
$$\max_{s \in \mathcal{V}^{(N)}} |a_s(t) - a^*_s| \leq B_N \gamma_N^t,$$
$$\max_{\ell \in \mathcal{L} \setminus \mathcal{U}^{(N)}} |e_\ell(t) - c_\ell| \leq A_{N+1} \gamma_{N+1}^t,$$

and $0 < \gamma_N < 1, 0 < \gamma_{N+1} < 1, A_N > 0, B_N > 0, A_{N+1} > 0$.

Lemma 2.1 is the key lemma to show the convergence of the synchronous iterative algorithm (1)–(3) and is proved in [21]. For that purpose, we also need Lemmas [21], which present monotonicity of lower bound $\underline{e}_\ell(t)$ and upper bound $\overline{e}_\ell(t)$ of explicit rate $e_\ell(t)$. In addition, we use Lemmas [21] where both lower and upper bound are shown to converge geometrically to e^*_ℓ.

The proof of Theorem 2.2 uses the following ideas. Consider a link ℓ whose flow consists of constrained and unconstrained sessions, see Fig. 2. Neither the constrained flow α_ℓ nor the number of unconstrained connections n_ℓ are known explicitly. The explicit rate update is given by

$$\begin{aligned} e_\ell(t+1) &= \min\left[\frac{c_\ell e_\ell(t)}{f_\ell(t)}, c_\ell \right] \\ &= \min\left[\frac{c_\ell e_\ell(t)}{\alpha_\ell + n_\ell e_\ell(t)}, c_\ell \right] \triangleq g_\ell(e_\ell(t)). \end{aligned}$$

It can be shown that $g_\ell(.)$ is a pseudo-contraction [22] and $e_\ell(t+1) = g_\ell(e_\ell(t))$ is a pseudo-contracting iteration converging to e^*_ℓ, the fixed point of $g_\ell(.)$. Note that we do not have a fixed point at zero if we start from non-zero $e_\ell(0)$ since $e_\ell(t+1) \geq e_\ell(t)$ when $e_\ell(t) \leq e^*_\ell$ for all t. Figure 3 shows how the pseudo-contracting property arises. Thus

$$|e_\ell(t+1) - e^*_\ell| \leq \xi_\ell |e_\ell(t) - e^*_\ell|, \qquad 0 < \xi_\ell < 1,$$

where $e^*_\ell = (c_\ell - \alpha_\ell)/n_\ell$ is the fair share of the remaining capacity $(c_\ell - \alpha_\ell)$ to be allocated to the n_ℓ unconstrained sessions (see Sect. 2 for the definition of 'fair share'). We can

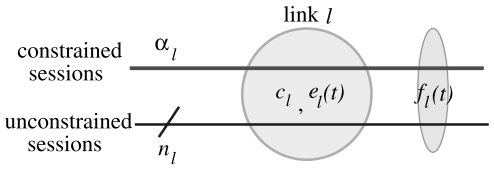

Fig. 2 Constrained and unconstrained sessions on a link ℓ.

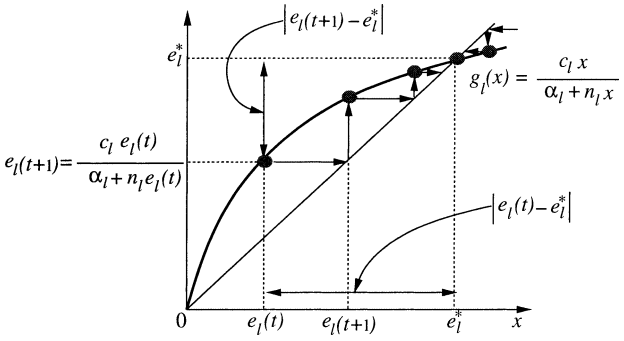

Fig. 3 A pseudo-contraction of $g_\ell(.)$.

show similar pseudo-contraction properties in the network setup using the bottleneck hierarchy.

At each level of the bottleneck hierarchy, the explicit rates of the associated bottleneck links can be shown to eventually have lower and upper bounds $\underline{e}_\ell(t)$ and $\overline{e}_\ell(t)$, i.e.,

$$\underline{e}_\ell(t) \leq e_\ell(t) \leq \overline{e}_\ell(t) \text{ for } \ell \in \mathcal{L}^{(i)},$$

and they converge geometrically to the fair share $e^*_\ell = x^i = (c_\ell - \alpha^{i*}_\ell)/n^i_\ell$ for $\ell \in \mathcal{L}^{(i)}$ at the i^{th} bottleneck level. So the explicit rates of i^{th} level bottleneck links converge to e^*_ℓ geometrically. We can show that the algorithm quickly achieves max-min fairness using these properties by induction on the bottleneck hierarchy. Furthermore, the explicit rates of non-bottleneck links $e_m(t)$ converge to the link capacities c_m geometrically.

Based on the previous result, we can construct a box in a space of dimension $|\mathcal{L}|$ at each time t by taking the maximum of geometric converging sequences among all the links, see Lemma 2.1. The box shrinks as updates proceed, and it includes all the possible explicit rates at a specific time t, so that any sequence of explicit rates converge to the fair shares or link capacities. These boxes provide the foundation for proving that 'asynchronous' updates will converge as will be discussed in the following subsection.

3. Proposed Asynchronous Algorithm with Round Trip Delays

In practice, the explicit rate indications $e_\ell(t)$ of links will experience delays while they propagate back to the sources and until they are eventually reflected in the incident flows on the link. We assume in Sect. 2 that newly modified explicit rates at time t appear by the time the update is made in the link flow $f_\ell(t)$ without delay. That is the link flow reflects the explicit rates computed at time t. This condition is relaxed in

Sect. 3. The issue of *feasibility*, i.e., maintaining link flows not exceeding link capacities is also discussed, and the algorithm with session priorities is presented in Sect. 4.

3.1 Asynchronous Iterative Algorithm without Delayed Information

In the synchronous algorithm, updates of the explicit rates at links are assumed to be perfectly synchronized. In practice, this is unlikely to be the case, so next we consider how asynchronism would affect convergence. We use the asynchronous model in [22] to formulate a totally asynchronous version of the algorithm and prove its convergence.

Each link $\ell \in \mathcal{L}$ may not have access to the most recent values of components of $\vec{e}$. That is the flow on link ℓ may reflect old information about other links' states. Let T^ℓ denote a set of times at which e_ℓ is updated. We shall assume that there is a set of times $T = \{0, 1, 2, \cdots\}$ at which one or more components of $\vec{e}(t)$ are updated. An asynchronous iteration can be described by

$$e_\ell(t+1) = \begin{cases} \min\left[\frac{c_\ell e_\ell(t)}{f_\ell(t)}, c_\ell \right] \triangleq g_\ell(\vec{e}(t)), & t \in T^\ell \\ e_\ell(t), & \text{otherwise.} \end{cases} \tag{7}$$

Note that $f_\ell(t)$ depends on the possibly outdated explicit rates indication in the network, i.e.,

$$\begin{aligned} f_\ell(t) &= h_\ell\left(e_1(\tau_1^\ell(t)), e_2(\tau_2^\ell(t)), \cdots, e_\ell(\tau_{|\mathcal{L}|}^\ell(t))\right) \\ &= \sum_{s \in \mathcal{S}_\ell} \min_{m \in \mathcal{L}_s}[\, e_m(\tau_m^\ell(t))\,], \end{aligned}$$

where $\tau_m^\ell(t)$ is the most recent time for which e_m is known to link ℓ through incident flow $f_\ell(t)$ at the link (see (1)–(3)), $0 \le \tau_m^\ell(t) \le t$ for all $t \in T$ and $\tau_\ell^\ell(t) = t$ for all $t \in T^\ell$. In the asynchronous iterative algorithm, the explicit rate e_ℓ is updated using the link flow carrying explicit rates $e_m(\tau_m^\ell(t))$ known to ℓ when $t \in T^\ell$, otherwise it remains unchanged. It is assumed here that $\tau_m^\ell(t) \to \infty$ as $t \to \infty$. This assumption implies that every link updates its explicit rate infinitely often as $t \to \infty$. In this case following theorem applies.

Theorem 3.1: (Convergence of Asynchronous Iterative Algorithm) The explicit rates $\vec{e}(t)$ in the asynchronous implementation proposed in (7) converge to the fixed point $\vec{e}^*$ of (5) and the associated session rates $\vec{a}(t)$ converge to the max-min fair rates $\vec{a}^*$.

Proof: From Lemma 2.1, let $A = \max[A_N, A_{N+1}]$, $\gamma = \max[\gamma_N, \gamma_{N+1}]$, $C = \max_{\ell \in \mathcal{L}}[c_\ell]$, and

$$E(t) = \begin{cases} \{\vec{e} \mid \|\vec{e} - \vec{e}^*\|_\infty \le A\gamma^t\}, & t \ge t_{N+1}, \\ \{\vec{e} \mid \|\vec{e} - \vec{e}^*\|_\infty \le \max[A\gamma^{t_{N+1}}, C]\}, & t < t_{N+1}, \end{cases}$$

then following conditions hold.

1. We have $E(t+1) \subset E(t)$, and $g(\vec{e}) \in E(t+1)$ for all t and $\vec{e} \in E(t)$. The sequence $\{\vec{e}(t)\}$ converges to $\vec{e}^*$ by Theorem 2.2 (Convergence of Synchronous Iterative Algorithm).
2. The set $E(t)$ satisfies the Box Condition [22] for all t. That is there exist sets $E_i(t) \subset E_i(0)$ for all t, such that
$$E(t) = E_1(t) \times E_2(t) \times \cdots \times E_{|\mathcal{L}|}(t).$$
3. Initial explicit rate vector $\vec{e}(0)$ is in the set $E(0)$.

Thus by Asynchronous Convergence Theorem [22], the asynchronous iteration (7) converges to $\vec{e}^*$. ■

Asynchronous convergence ensures that although links update explicit rates independently, the allocation will converge as in the synchronous algorithm, though it may take longer to do so.

In Sect. 2.2 and Sect. 3.1, we considered the convergence of synchronous and asynchronous decentralized updates based on local information. In practice delays will be incurred in the communication between sources and links. We consider the role of the delays in the following subsection.

3.2 Iterative Algorithm with Round Trip Delays

In the preceding analysis, the link flow $f_\ell(t)$ was assumed to be the sum of session rates $a_s(t)$ traversing link ℓ. The session rates were in turn assumed to be $a_s(t) = \min_{\ell \in \mathcal{L}_s}[\, e_\ell(t)\,]$, i.e., the incident flows at time t reflect the computed explicit rates at the time t with no delay. In reality, the link flows would be immediately measured at each link, and would depend on *delayed* explicit rate indications computed at links and sent back to sources in order to control the source rates. We will present an example to show the oscillations that arise due to propagation delay.

Consider the network shown in Fig. 4 with one link shared by two sessions. Suppose the link capacity is $c_\ell = 1$, the initial explicit rate is $e_\ell(0) = 0.25$, and the *Round Trip Delay (RTD)* is assumed to be 1 time unit for both sessions. Thus the explicit rate takes at most 1 unit of time to propagate back to the sources and be reflected in the incident flow on the link, i.e., $f_\ell(t) = 2e_\ell(t-1)$. The explicit rate update would be

$$e_\ell(t+1) = \min\left[\frac{c_\ell e_\ell(t)}{f_\ell(t)}, c_\ell \right] = \min\left[\frac{e_\ell(t)}{2e_\ell(t-1)}, 1 \right],$$

which results in the oscillation shown in Fig. 5.

One way of preventing oscillation is to update the explicit rate at each link only after the worst case RTD, D_ℓ has elapsed, where D_ℓ is the worst case RTD of the sessions sharing link ℓ. In other words, explicit rate $e_\ell(t)$ is updated only after the link receives newly modified source rates regulated by the last computed local explicit rate. This scheme

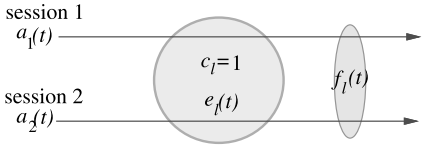

Fig. 4 Network example with two ABR sessions with round trip delay.

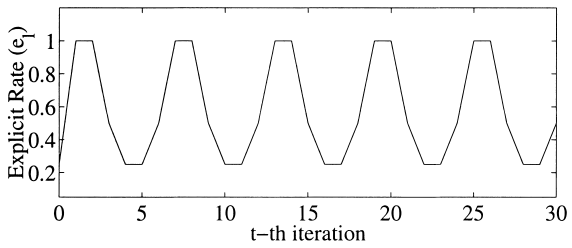

Fig. 5 Oscillation of explicit rate in the network example without considering RTD.

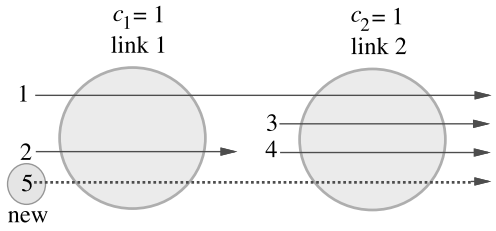

Fig. 6 A network with a new session 5.

can be shown to converge to the same max-min fair allocation. The explicit rate update of link ℓ is then

$$e_\ell(t+1) = \min\left[\frac{c_\ell e_\ell(t-D_\ell)}{f_\ell(t)}, c_\ell\right], \ell \in \mathcal{L}. \quad (8)$$

Thus stability can be achieved by delaying updates, or alternatively as suggested in [20] by damping the measurements and computation. The proof of convergence is the same as that of the synchronous convergence result stated in Theorem 2.2.

3.3 Feasibility Issue of Rate Control Mechanism

An allocation is said to be *feasible* if the link flows do not exceed the link capacities. In our algorithm, link flow may temporarily exceed capacity causing queue buildups. For example, Figs. 6 and 7 show a case where a new session 5 is setup after the other sessions in the network have reached the max-min fair allocation. Before the new session 5 is setup, two and three connections traverse link 1 and link 2, respectively, while link capacities c_1 and c_2 are all 1. So link 2 is a bottleneck link and the fair share $e_2 = \frac{1}{3}$, and the sessions 1, 3, and 4 adjust their rates to the fair share ($a_1 = a_3 = a_4 = \frac{1}{3}$). Then the available bandwidth at link 1 is $\frac{2}{3}$ ($= 1 - \frac{1}{3}$). So the fair share $e_1 = \frac{2}{3}$ and the session rate $a_2 = \frac{2}{3}$ since only session 2 is contending for the available bandwidth at link 1. Thus it achieves max-min fair allocation $\vec{a}^* = (\frac{1}{3}, \frac{2}{3}, \frac{1}{3}, \frac{1}{3})$, and then it quickly adjusts to its new max-min fairness $\vec{a}^* = (\frac{1}{4}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ after the new session.

The infeasibility can be mitigated by damping the computation of explicit rates. Damping of explicit rates by network adjustment will lessen the abrupt ramp-up or down of rates, and allow sufficient time for the network to adapt to the varying session rates and link flows and presumably prevent from excessive infeasibility. It can be shown that the

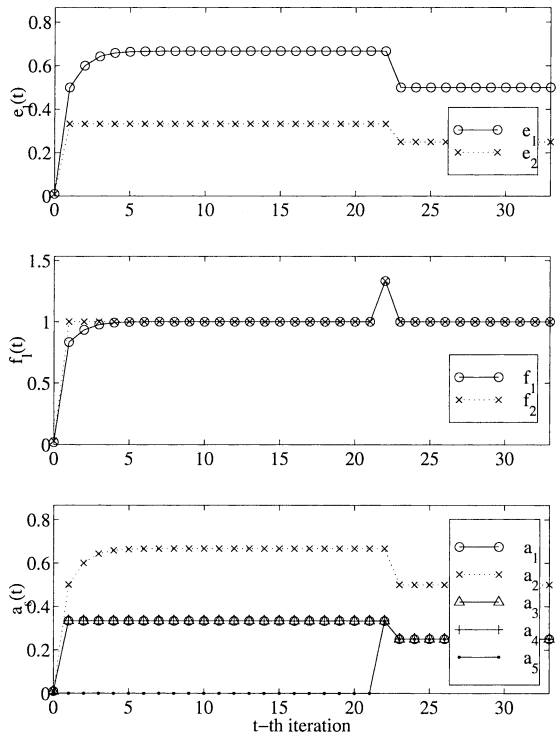

Fig. 7 Explicit rates, link flows and session rates when a new session 5 is setup.

damped version of the algorithm also converges to the solution of the algorithm without damping by similar steps followed in the proof of Theorem 2.2.

Another approach to manage the variability in a dynamic environment is to constrain sources to make slow rate adjustments particularly upon entering and increasing their rates: the session rates can not be increased rapidly when they are admitted to a network, rather they are permitted to increase their rates by only a little amount at a time so that the network will have sufficient time to recognize the number of connections by measuring the flow. This approach can be considered as damping source behavior.

We believe that single bit indication of queue status can be used in conjunction with our scheme to prevent the excessive queue buildup when link flows exceed the available resource transiently. In the scheme, sources slow down their change of rates if queue starts to build up, otherwise they speed up to achieve the desired max-min fairness quickly. In a sense, the single bit indication scheme can take care of the feasibility while the explicit rate control mechanism can ensure fast convergence to fair rates. The single bit queue indication scheme might be jointly combined with the damping at sources such as linear growth of source rate. While damping of explicit rates at network links and/or damping of session rates at sources manages to keep the queue from growing beforehand, the single bit indication scheme primarily reduces the queue already built-up.

An even more conservative approach would be to employ a safety margin on available capacity. Suppose we have

network utilization factor ρ, where $0 < \rho < 1$, and the link capacity to be shared is only $\bar{c}_\ell = \rho c_\ell$. We then have spare capacity $(1-\rho)c_\ell$ to absorb the transient overshoot above virtual capacity $\bar{c}_\ell$ leading to implicit control of queue buildup. The max-min fair allocation of resources would be defined with respect to the new capacities $\bar{c}_\ell, \ell \in \mathcal{L}$.

Combining these ideas, we can significantly improve the feasibility and thus performance in a dynamic network environment. There have been algorithms designed to guarantee feasibility. They, however, might also experience transient infeasibility if the instantaneous available bandwidth is highly variable, which might be typical in integrated services networks, and buffering should be provided to tackle the problem [9], [16].

4. Iterative Algorithm with Session Priority

It may be useful to allow sessions to have different priorities in an attempt to differentiate applications with different QoS. We can formulate an iterative algorithm wherein a priority w_s, where $w_s \geq 1$, of a session s plays a role as follows:

$$a_s(t) = w_s \min_{\ell \in \mathcal{L}_s}[\ e_\ell(t)\ ], \qquad s \in \mathcal{S}, \tag{9}$$

$$f_\ell(t) = \sum_{s \in \mathcal{S}_\ell} a_s(t), \qquad \ell \in \mathcal{L}, \tag{10}$$

$$e_\ell(t+1) = \min\left[\ \frac{c_\ell e_\ell(t)}{f_\ell(t)}, c_\ell\ \right], \qquad \ell \in \mathcal{L}. \tag{11}$$

Note that the computation of explicit rates is still conducted locally in a decentralized manner and the priority w_s is dealt with at the source leading to the same structure of distributed computation as in the preceding algorithms.

We can now define a *bottleneck hierarchy with priority* and a notion of *weighted fair share* following a similar procedure as in Sect. 2. Let $\overline{\mathcal{U}}^{(i)} = \cup_{j=1}^{i} \overline{\mathcal{L}}^{(j)}$ and $\overline{\mathcal{V}}^{(i)} = \cup_{j=1}^{i} \overline{\mathcal{S}}^{(j)}$ be the cumulative set of bottleneck links and sessions, respectively, in levels 1 to i of the hierarchy with priority. The *weighted fair share* $\overline{x}_\ell^i$ can be defined as a weighted fair partition of available capacity at link ℓ in the i^{th} level of the hierarchy:

$$\overline{x}_\ell^i = \frac{c_\ell - \overline{\alpha}_\ell^{i*}}{\overline{n}_\ell^i}, \tag{12}$$

where $\overline{\alpha}_\ell^{i*} = \sum_{s \in \mathcal{S}_\ell \cap \overline{\mathcal{V}}^{(i-1)}} \overline{a}_s^*$ is the total flow of sessions through ℓ constrained by bottleneck links in $\overline{\mathcal{U}}^{(i-1)}$, and $\overline{n}_\ell^i = \sum_{\mathcal{S}_\ell \setminus \overline{\mathcal{V}}^{(i-1)}} w_s$ is the effective number of sessions through ℓ unconstrained by the links in $\overline{\mathcal{U}}^{(i-1)}$. Note that $\overline{\alpha}_\ell^{1*} = 0$ in the 1^{st} level of the hierarchy.

Based on the weighted fair share, the set of i^{th} level bottleneck links and sessions with priority can be defined as:

$$\overline{\mathcal{L}}^{(i)} = \{\ell \in \mathcal{L} \setminus \overline{\mathcal{U}}^{(i-1)} \mid f_\ell^* = c_\ell$$
$$\text{and for all } s \in \mathcal{S}_\ell, \frac{\overline{a}_s^*}{w_s} = \overline{x}^i = \min_{m \in \mathcal{L} \setminus \overline{\mathcal{U}}^{(i-1)}} \overline{x}_m^i\},$$

$$\overline{\mathcal{S}}^{(i)} = \{s \in \mathcal{S} \setminus \overline{\mathcal{V}}^{(i-1)} \mid s \in \mathcal{S}_\ell \text{ and } \ell \in \overline{\mathcal{L}}^{(i)}\}. \tag{13}$$

Here $\overline{\mathcal{L}}^{(i)}$ is the set of i^{th} level bottleneck links such that the sessions in $\overline{\mathcal{S}}^{(i)}$ are allocated weighted minimum fair share in the network, i.e., for $s \in \overline{\mathcal{S}}^{(i)}$, $\frac{\overline{a}_s^*}{w_s} = \min_{r \in \mathcal{S} \setminus \overline{\mathcal{V}}^{(i-1)}} \frac{\overline{a}_r^*}{w_r} = \overline{x}^i$, thus each session sharing the link receives bandwidth in proportion to its priority, i.e., $\overline{a}_s^* = w_s \overline{x}^i$. Note that $\overline{x}^i = \overline{x}_\ell^i$ for $\ell \in \overline{\mathcal{L}}^{(i)}$ is the weighted fair share of the bottleneck links in the i^{th} level hierarchy. The construction of the bottleneck hierarchy with priority is analogous to that of Sect. 2 resulting in set of bottleneck links and sessions with priority $\overline{\mathcal{L}}^{(1)}, \cdots, \overline{\mathcal{L}}^{(N)}$ and $\overline{\mathcal{S}}^{(1)}, \cdots, \overline{\mathcal{S}}^{(N)}$.

One can show that (9)–(11) will converge and allocate bandwidth to the sessions proportional to the weights w_s, i.e., $\overline{a}_s^* = w_s \overline{x}^i$ for all $s \in \overline{\mathcal{S}}^{(i)}$, which is "weighted fair" leading to *weighted fair allocation* $\vec{\overline{a}}^* = (\overline{a}_s^*, s \in \mathcal{S})$.

5. Simulations

We have proposed a simple rate control framework for elastic services using explicit rate and analyzed its convergence to (weighted) max-min fair allocation of bandwidth both synchronously and asynchronously. In this section, we conduct simulations to verify the analysis results and to further investigate our proposed algorithm. We have used NIST ATM simulator [23], and modified switch, broadband terminal equipment, and traffic source modules to incorporate our distributed and asynchronous rate control algorithms.

We adopt several notations defined in ABR context, i.e., Minimum Cell Rate (MCR), m_s, Allowable Cell Rate (ACR), a_s, and Peak Cell Rate (PCR), p_s, which denote guaranteed minimum rate, allowable transmission rate, and upper bound of transmission rate, respectively. We will assume that each session s has a dedicated access link $\ell_s \in \mathcal{L}_s$ with a capacity p_s corresponding to the source's PCR. This will ensure that $a_s \leq p_s$ and make the description of algorithm simple since we consider $a_s(t) = \min_{\ell \in \mathcal{L}_s}[\ e_\ell(t)\ ]$ instead of using $a_s(t) = \min_{\ell \in \mathcal{L}_s}[\ p_s, e_\ell(t)\ ]$. Moreover we assume persistent greedy elastic connections, which transmit at their current ACR.

In our rate control mechanism, each switch approximates aggregate flow $f_\ell(t)$ of elastic connections and available capacity for elastic connections by measuring traffic during a measurement window. The duration of the measurement window has to be short enough to capture the variation of total flow or available capacity and long enough to lessen computing overheads at switches. Then it computes a local 'explicit rate' $e_\ell(t)$ at every update intervals. We set the update interval to be greater than the maximum RTD D_ℓ since the update interval is required to be greater than or equal to the maximum RTD to prevent oscillation as shown in Sect. 3.2. RTD can be measured by investigating Resource Management (RM) cells transmitted from the sources at regular intervals. We assume that RM cells are generated at every other 32 data cells.

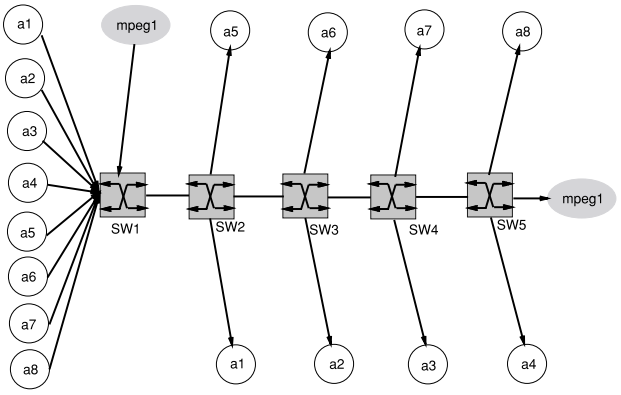

Fig. 8 Simulation scenario with five switches, eight elastic connections and one mpeg source.

Table 1 Arrival and departure time of connections.

Connection	Arrival time [sec]	Departure time [sec]	Switches traversed
a1	0	∞	sw1, sw2
a2	0	∞	sw1, sw2, sw3
a3	1	3	sw1, sw2, sw3, sw4
a4	2	4	sw1, sw2, sw3, sw4, sw5
a5	0	∞	sw1, sw2
a6	0	∞	sw1, sw2, sw3
a7	1	∞	sw1, sw2, sw3, sw4
a8	2	∞	sw1, sw2, sw3, sw4, sw5
mpeg1	0	∞	sw1, sw2, sw3, sw4, sw5

Switching devices send computed explicit rate to the sources, if the current transmission rate indication in the packet is higher than the computed explicit rate at the switch. Note the explicit rate is modified either on the forward or backward trip. Thus the source receives the minimum explicit rate for links along its route. The role of each source is to adjust the current transmission rate ACR $a_s(t)$ so that it does not exceed the explicit rate indication, or the session's PCR constraint.

We have simulated a network of five switches and eight elastic connections (see Fig. 8). One MPEG source traverses all the switches in the network. The MPEG source consists of multiplexed 10 movies and news clips with the duration of 10 seconds [24]. In order to observe the behavior of the mechanism, elastic connections enter and leave the network dynamically as indicated in Table 1.

Capacity of links/switches is assumed to be 155 Mbps. The distances between the sources/destinations and the switches are set to 0.2 km, and the distance between the switches is set to 300 km, which corresponds to WAN environment. Assuming the signal propagation speed is 2×10^5 km/sec, the end-to-end distance of 100 km corresponds to 1 msec. So the maximum RTD is about 12 msec, and the update interval of explicit rates is assumed to be 12 msec. The oscillation phenomenon can be prevented by updating explicit rates every 12 msec. And the measurement interval for available capacity and flow was set to 0.6 msec. Thus available capacity and flow are measured 20 times during each update interval of explicit rates and the measurements are moving-averaged. The computation and measurement of explicit rates, flow, and available capacity are damped with damping coefficient of 0.8 to smoothen the abrupt variation.

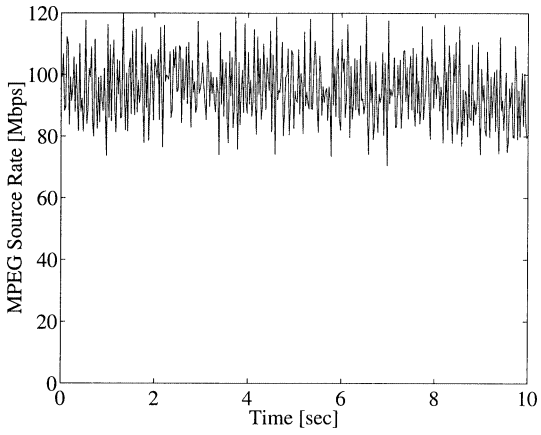

Fig. 9 An MPEG source collected from 10 news and video clips.

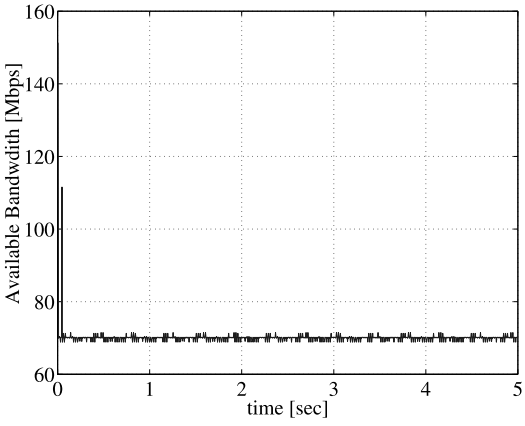

Fig. 10 Available capacity for elastic connections at switch 1.

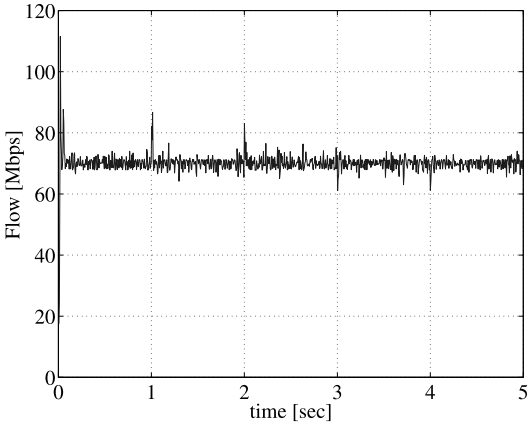

Fig. 11 Flow of elastic connections at switch 1.

The simulation results are shown in Fig. 10–Fig. 14. Since an MPEG source (Fig. 9) traverses all the switches during simulations, the measured available bandwidth for

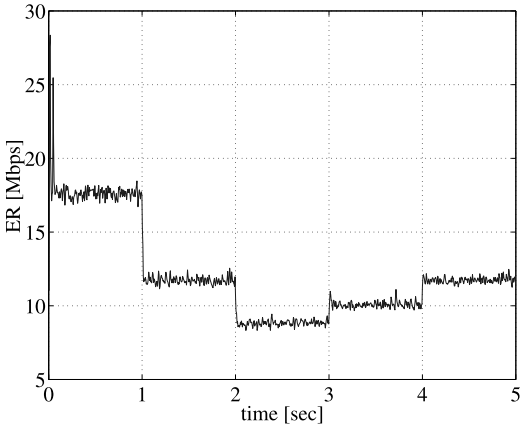

Fig. 12 Explicit rate computed at switch 1.

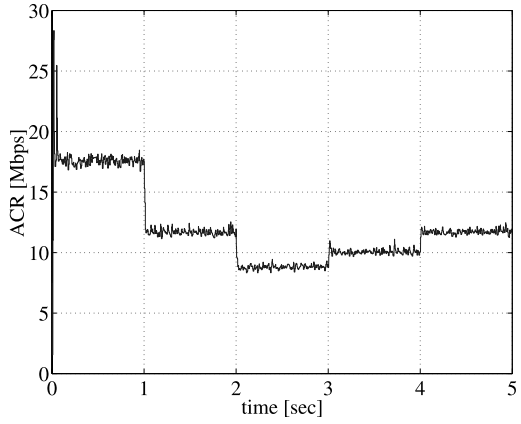

Fig. 13 Data rate at source 1.

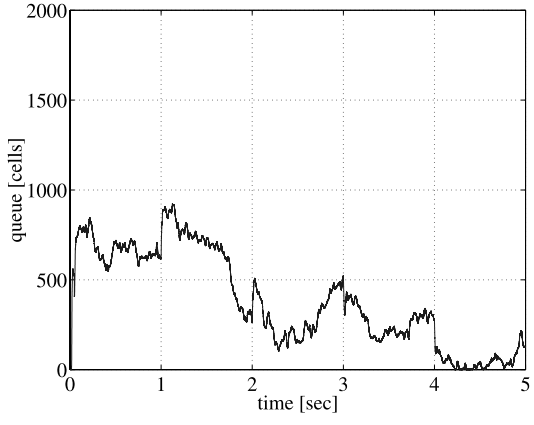

Fig. 14 Queue status at switch 1.

elastic connections varies around 70 Mbps (Fig. 10), which correctly tracks the available capacity for elastic connections after allocating sufficient bandwidth to the MPEG source.

We observe that the explicit rates are quickly updated as the number of elastic connections dynamically varies at switch 1 (see Fig. 12). Note that there are 4 connections during [0,1] sec, 6 during [1,2] sec, 8 during [2,3] sec, 7 during [3,4] sec, and 6 during [4,5] sec, respectively. The computed explicit rate indications were 17.5 Mbps, 11.7 Mbps, 8.8 Mbps, 10 Mbps and 11.7 Mbps during each second, which is matched with the correct fair share. Thus we verify that the proposed mechanism correctly tracks the varying traffic flow and updates explicit rates accordingly.

The actual flow measurement (70 Mbps) of elastic connections at switch 1 is shown to match with the remaining available bandwidth at switch 1 after the MPEG source is allocated its bandwidth (85 Mbps) among the switch capacity of 155 Mbps as shown in Fig. 11. And the switch capacity is fully utilized by the MPEG source and eight elastic connections as expected. Thus the available capacity and elastic flow measurement quickly reflects the network status. The sources are shown to follow the rate regulation fed back from the switches to the sources (Fig. 13).

In order to investigate the feasibility, we have also observed queue dynamics. We have checked if data build up in the queues during the operation. The number of packets (cells) in the queue at switch 1 is shown to be maintained below 1000, which is a sign of stable operation of our algorithm. Thus we have verified the correctness, network dynamics, and feasibility of our algorithms via simulations.

6. Conclusion

In this paper, we have investigated a simple flow control mechanism using explicit rate. We have formulated a decentralized iterative algorithm and shown that the solution of fixed point equation associated with the algorithm is unique, and the algorithm converges geometrically to the (weighted) max-min fair allocation of resources. We have proposed asynchronous version of the algorithm leading to the same (weighted) max-min fairness. These algorithms operate in a distributed manner accounting for the heterogeneity of a large-scale high speed network.

It has been shown that they quickly achieve notion of global max-min fair rate allocation for contending users sharing resources through decentralized adjustment of explicit rates. The algorithms are simple in that they do not require that the links keep track of the number of constrained and unconstrained connections as some rate based flow algorithms did. Hence it has clear scalability advantage in terms of both complexity and state information. In addition, priority can easily be incorporated by a weight given to each session, and thus differentiated quality of service per connection is possible. We have considered the feasibility issue and extended the algorithms so as to deal with priorities of sessions. We then have verified the algorithm by simulations showing that it indeed achieves the desirable characteristics for rate control of elastic connections.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments.

References

[1] G. de Veciana, T.J. Lee, and T. Konstantopoulos, "Stability and performance analysis of networks supporting elastic services," IEEE/ACM Trans. Netw., vol.9, no.1, pp.2–14, Feb. 2001.
[2] F. Bonomi and K.W. Fendick, "The rate-based flow control framework for the available bit rate ATM service," IEEE Netw. Mag., vol.9, no.2, pp.25–39, March/April 1995.
[3] D. Bertsekas and R. Gallager, Data Networks, Prentice Hall, 1992.
[4] E.M. Gafni and D. Bertsekas, "Dynamic control of session input rates in communication networks," IEEE Trans. Autom. Control, vol.29, no.11, pp.1009–1016, 1984.
[5] F. Kelly, "Charging and rate control for elastic traffic," Eur. Trans. Telecommun., vol.8, pp.33–37, 1997.
[6] R. Jain, S. Kalyanaraman, and R. Viswanathan, "The OSU scheme for congestion avoidance using explicit rate indication," ATM Forum, 94-0883, Sept. 1994.
[7] R. Jain, S. Kalyanaraman, and R. Viswanathan, "Simulation results: The EPRCA+ scheme," ATM Forum, 94-0988, Oct. 1994.
[8] K.Y. Siu and H.Y. Tzeng, "Adaptive proportional rate control for ABR service in ATM networks," Tech. Rep. 94-07-01, ECE Dept. U.C. Irvine, July 1994.
[9] A. Charny, K.K. Ramakrishnan, and A. Lauck, "Time scale analysis and scalability issues for explicit rate allocation in ATM networks," IEEE/ACM Trans. Netw., vol.4, pp.569–581, 1996.
[10] F. Bonomi, D. Mitra, and J.B. Seery, "Adaptive algorithms for feedback-based flow control in high-speed wide-area ATM networks," IEEE J. Sel. Areas Commun., vol.13, no.7, pp.1267–1283, Sept. 1995.
[11] S.P. Abraham and A. Kumar, "A stochastic approximation approach for max-min fair adaptive rate control of ABR sessions with MCRs," Proc. IEEE INFOCOM, pp.1358–1365, 1998.
[12] F.P. Kelly, A.K. Mauloo, and D.K.H. Tan, "Rate control in communication networks shadow prices, proportional fairness and stability," J. Oper. Res. Soc., vol.15, no.49, pp.237–255, 1998.
[13] R.J. La and V. Anantharam, "Charge-sensitive TCP and rate control in the Internet," Proc. IEEE INFOCOM, pp.1116–1175, 2000.
[14] S. Low and D. Lapsley, "Optimization flow control, i: Basic algorithm and convergence," IEEE/ACM Trans. Netw., vol.7, no.6, pp.861–874, Dec. 1999.
[15] L. Benmohamed and S.M. Meerkov, "Feedback control of congestion in packet switching networks the case of a single congested node," IEEE/ACM Trans. Netw., vol.1, no.6, pp.693–709, Dec. 1993.
[16] C.F. Su, G. de Veciana, and J. Walrand, "Explicit rate flow control for ABR services in ATM networks," IEEE/ACM Trans. Netw., vol.8, no.3, pp.350–361, June 2000.
[17] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," IEEE/ACM Trans. Netw., vol.8, no.5, pp.556–567, Oct. 2000.
[18] S. Chong, S. Lee, and S. Kang, "A simple, scalable, and stable explicit rate allocation algorithm for max-min flow control with minimum rate guarantee," IEEE/ACM Trans. Netw., vol.9, no.3, pp.322–335, June 2001.
[19] S.P. Abraham and A. Kumar, "A new approach for asynchronous distributed rate control of elastic sessions in integrated packet networks," IEEE/ACM Trans. Netw., vol.9, no.1, pp.15–30, Feb. 2001.
[20] C. Fulton, S.Q. Li, and C.S. Lim, "An ABR feedback control scheme with tracking," Proc. IEEE INFOCOM, pp.806–815, 1997.
[21] T.J. Lee, Traffic Management and Design of Multiservice Networks: The Internet and ATM Networks, Ph.D. Thesis, Dept. of ECE, University of Texas, Austin, 1999.
[22] D. Bertsekas and J.N. Tsitsiklis, Parallel and Distributed Computation: Numerical methods, Prentice Hall, 1989.
[23] The NIST ATM Simulator, NIST, 1995.
[24] MPEG sources, http://www-info3.informatik.uniwuerzburg.de/rose

Tae-Jin Lee received his B.S. and M.S. in electronics engineering from Yonsei University, Korea in 1989 and 1991, respectively, and his M.S.E. in eletrical engineering and computer science from University of Michigan, Ann Arbor in 1995. He received his Ph.D. in electrical and computer engineering from the University of Texas at Austin in 1999. From 1999 to 2001, he has been a senior engineer at Corporate R & D Center, Samsung Electronics Co. Ltd. Since 2001, he has been an Assistant Professor at the School of Information and Communication Engineering in Sungkyunkwan University, Korea. His research interests include performance evaluation, traffic management and design of communication networks and systems, wireless LAN/PAN, ad-hoc networks, and optical networks.

Gustavo de Veciana received his B.S., M.S, and Ph.D. in electrical engineering from the University of California at Berkeley in 1987, 1990, and 1993 respectively. He is currently a Professor at the Department of Electrical and Computer Engineering at the University of Texas at Austin. His research focuses on the design, analysis and control of telecommunication networks. Current interests include: measurement, modeling and performance evaluation; wireless and sensor networks; architectures and algorithms to design reliable computing and network systems. Dr. de Veciana has been an editor for the IEEE/ACM Transactions on Networking. He is the recipient of General Motors Foundation Centennial Fellowship in Electrical Engineering and a 1996 National Science Foundation CAREER Award, co-recipient of the IEEE William McCalla Best ICCAD Paper Award for 2000, and co-recipient of Best Paper in ACM Transactions on Design Automation of Electronic Systems, Jan 2002–2004.
E-mail: gustavo@ece.utexas.edu