Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Networks 50 (2006) 1816–1830
Computer Networks
www.elsevier.com/locate/comnet

A distributed utility max–min flow control algorithm ☆

Hyang-Won Lee *, Song Chong

Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), 373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, Republic of Korea

Received 30 August 2004; received in revised form 21 March 2005; accepted 31 July 2005
Available online 6 September 2005

Responsible Editor: N.B. Shroff

Abstract

A fair allocation of utility (application-layer performance) is essential in providing QoS (Quality of Service) guarantee. However, there have been few researches in the literature of utility-fair network resource allocation scheme. In this paper, we propose a distributed utility max–min flow control algorithm which accommodates application diversity in that it does not require the concavity of utility functions, and is scalable in that it does not require any per-flow operation in the network. The algorithm is proved to be convergent under the assumption that there exists a single bottleneck link and the communication delay between any two links in the network is bounded. Although the convergence of the algorithm is analyzed only for the case of a single bottleneck link, we show through simulations that the proposed algorithm works as designed for the case of multiple bottleneck links as well.

Keywords: Flow control; Utility max–min fairness; Non-linear/non-convex programming; Convergence analysis

1. Introduction

One of the key goals in the next generation network is to guarantee quality of service (QoS) which represents the level of application-layer performance (utility). To reach this goal, it is necessary to identify application specific demands and to provide network resource allocation schemes taking into account the demands. The application specific demand can be characterized by bandwidth, delay and delay jitter etc. which are called QoS parameters. For example, real-time multimedia applications require strict bandwidth, delay and delay jitter guarantee since to provide acceptable performance in playing back the encoded multimedia, at least the encoding rate should be guaranteed and the timely delivery of a packet is essential. On the other hand, data applications do not require strict delay jitter guarantee because their performances strongly depend on the time spent to complete data deliveries. A utility function is used to mathematically describe

☆ This work was supported by the center for Broadband OFDM Mobile Access (BrOMA) at POSTECH supported by the ITRC program of the Korean Ministry of Information and Communication (MIC) under the supervision of the Institute of Information Technology Assessment (IITA).
* Corresponding author. Tel.: +82 42 869 5473; fax: +82 42 869 8073.
E-mail addresses: mslhw@netsys.kaist.ac.kr (H.-W. Lee), song@ee.kaist.ac.kr (S. Chong).

1389-1286/$ - see front matter
doi:10.1016/j.comnet.2005.07.018

the QoS characteristics of an application. In this work, we deal with only a bandwidth utility function, which describes the application-layer performance with respect to the bandwidth allocated to an application.

By using the concept of utility functions, there have been numerous researches to provide utility-aware network bandwidth allocation schemes based on optimization theory [1–6]. The objective that they have in common is the maximization of aggregate utility. Under the objective, by decomposing the problem into network problem and user problem [2,5] or by appealing to the duality theory in optimization theory [3], it is easy to construct a distributed algorithm, which is provable to be convergent. However, this framework has two serious drawbacks. One is that the utility functions are restricted to be strictly concave. Because the utility functions of real-time or rate-adaptive applications are not concave [7], those applications cannot be handled in the framework. The other is that the framework cannot achieve a utility-fair bandwidth allocation. Achieved in the framework is only the *maximization of aggregate utility* and *bandwidth proportional fairness* for logarithmic utility functions, or *arbitrarily close approximation to bandwidth max–min fairness* for a generalized utility function which is concave [6]. Lee et al. [8] showed that if such algorithms developed for concave utility functions are applied to non-concave utility functions, the system can be unstable and can cause excessive congestion in the network. They proposed a distributed rate control algorithm which can handle both concave and sigmoidal-like utility functions, but the utility fairness was not considered either.

On the other hand, in [9], Cao and Zegura defined a utility max–min fairness and proposed a bandwidth allocation scheme that achieves the *utility max–min* fairness. Although their work is the first step to the utility max–min flow control and does not need the concavity of utility functions which is indispensable in the framework mentioned above, the proposed scheme has many problems. Firstly, it is not realistic because each link in the network should know the utility functions of all flows passing through the link and maintain the set of flows saturated at the link. Secondly, they prove that the algorithm finds a utility max–min fair rate allocation vector in finite time, but the delay, which is essential in the communication network, is not considered in the proof. To overcome the limitations mentioned so far, Cho and Chong [10] proposed a control-theoretic utility max–min flow control algorithm which resolves the problems of [9], and showed that the algorithm converges to a utility max–min fair rate vector by using Dewey and Jury's stability criterion [11].

In this paper, we address a *utility max–min* flow control algorithm based on optimization theory. We first propose a utility max–min flow control algorithm provided that there exists a single bottleneck link in the network, and then extend the algorithm to the case of multiple bottleneck links. The proposed algorithm requires neither the concavity of utility functions nor any per-flow operation in the network. Moreover, we mathematically prove, for the case of a single bottleneck link, that the algorithm converges to a utility max–min fair point with bounded communication delays. In this paper, we do not analyze the convergence of the algorithm accommodating multiple bottleneck links due to its complexity, but we show through simulations that the algorithm works as designed. To the best of our knowledge, our work is the first paper dealing with utility max–min flow control based on optimization theory.

Of course, the utility max–min fairness is not a better choice than the utility maximization in the aspect of utility efficiency. In other words, the utility max–min fair strategy can lead to utility inefficiency, as can do the bandwidth max–min fair strategy to bandwidth inefficiency [12]. If there exists a session whose utility function has a finite upper bound, then the utility allocated to all the other sessions sharing a link with the session cannot increase above the upper bound. We can easily see that this is not problematic if the value of utility at a utility max–min point is sufficiently smaller than the bound. However, as the utility max–min point approaches the bound (by the increased link bandwidth or by the departure of sessions), the problem will come up and get worse, i.e., most of the remaining bandwidth will be allocated to the session with bounded utility function because the utility max–min strategy finds the rate allocation achieving the equality of all sessions' utilities. Accordingly, if one is concerned with both fairness and efficiency, it is desirable to find and adopt other utility fairness criteria rather than utility max–min or utility maximization. This issue is out of the scope of this paper, and we leave it as a future study.

The rest of the paper is organized as follows. In Section 2, we formulate the utility max–min flow

control as an optimization problem for the case of a single bottleneck link. And then, a distributed algorithm for the case of a single bottleneck link is proposed and shown to be convergent in Section 3. In Section 4, we discuss and propose an extended algorithm to accommodate the case of multiple bottleneck links. In Section 5, we show through simulations that the proposed algorithm works as designed not only for the case of a single bottleneck link but also for the case of multiple bottleneck links. Finally, we conclude the paper in Section 6.

2. Problem formulation

Consider a network in which one or more sessions (source–destination pairs) exist, and let L and S be the set of all links and the set of all sessions in the network, respectively. C_l and S_l respectively denote the capacity of link l and the set of sessions sharing link l. Each session i is associated with a utility function U_i: $[0, M_i] \to R_+$ where M_i denotes the maximum possible transmission rate of session i. Below is the assumption on the utility functions we will use throughout the paper.

Assumption 1.

(a) Every $U_i(\cdot)$ is strictly increasing with $U_i(0) = 0$.
(b) Every $U_i(\cdot)$ is twice continuously differentiable.

Definition 1. Let x_i be the bandwidth allocated to session i. Then, a rate vector $x^* = [x_i^*, i \in S]^T$ is said to be utility max–min fair if it satisfies the following property for any other rate vector x: if $U_j(x_j^*) < U_j(x_j)$ for some j, then there exists i such that $U_i(x_i) < U_i(x_i^*) \leqslant U_j(x_j^*)$.

Definition 2. Suppose that a rate vector x^* is utility max–min fair, then session i is said to be bottlenecked at link l if $\sum_{k\in S_l} x_k^* = C_l$ and $U_i(x_i^*) \geqslant U_k(x_k^*), \forall k \in S_l$. In this case, link l is said to be a bottleneck link of session i.

Definition 1 and 2 are equivalent to the definition of bandwidth max–min fairness and bandwidth bottleneck except that in utility max–min fairness and bottleneck, utilities are compared instead of bandwidths.

Assume that every session is bottlenecked at a same link or equivalently there is only a single bottleneck link in the network. Note that this assumption also implies that no sessions are bottlenecked at their sources. We will discuss the case of multiple bottleneck links in Section 4 where we extend the algorithm proposed in this section. Let l and B_l be the single bottleneck link and the set of sessions bottlenecked at link l, respectively, then we have $B_l = S_l = S$. We formulate a utility max–min flow control problem as an optimization problem by

$$
\begin{aligned}
(\mathbf{P_I}): \quad \min \quad & \sum_{i\in B_l} \left(U_{\mathrm{av}}^l(x) - U_i(x_i)\right)^2 \\
\text{subject to} \quad & \sum_{i\in S_l} x_i \leqslant C_l \\
& x_i \geqslant 0, \ \forall i \in S,
\end{aligned}
$$

where $x = [x_i, i \in S]^T$, $U_{\mathrm{av}}^l(x) = \frac{1}{N_l}\sum_{i\in B_l} U_i(x_i)$ and $N_l = |B_l|$. Note that the absence of maximum possible transmission rates (M_i's) in the formulation does not cause a loss of generality, i.e., does not preclude the existence of maximum bandwidth limits. If such a maximum limit exists at source a, one can simply add a new source a' connected to a and set the bandwidth of the link between a and a' to the maximum limit. Then, letting the session start from a' exactly models the maximum bandwidth limit. The optimal solution of ($\mathbf{P_I}$) obviously achieves a utility max–min fairness among competing flows. However, it is not easy to obtain the optimal solution due to the capacity constraint, and moreover, there is no factor making the link fully utilized. So, instead of solving ($\mathbf{P_I}$) as it is, we relax the constraint while achieving utility max–min fairness and also making the link utilized at a target utilization level.

First, we transform the capacity constraint (inequality constraint) into an equality form by introducing a slack variable v_l. Let $v_l = (1 - \lambda)C_l$ for $0 \leqslant \lambda \leqslant 1$, then ($\mathbf{P_I}$) can be rewritten as

$$
\begin{aligned}
(\mathbf{P_E}): \quad \min \quad & \sum_{i\in B_l} \left(U_{\mathrm{av}}^l(x) - U_i(x_i)\right)^2 \\
\text{subject to} \quad & \sum_{i\in S_l} x_i = \lambda C_l \\
& x_i \geqslant 0, \ \forall i \in S \\
& 0 \leqslant \lambda \leqslant 1.
\end{aligned}
$$

Note that λ is actually a target utilization and this formulation is equivalent to ($\mathbf{P_I}$) except that a target link utilization λ can be achieved here. From now on, we will omit $0 \leqslant \lambda \leqslant 1$ in the formulation because we can set the value of λ as we wish. We now relax the equality constraint by adding a penalty function as follows:

$$(\mathbf{P_P}):\ \min\ \sum_{i\in B_l}\left(U^l_{\mathrm{av}}(x)-U_i(x_i)\right)^2+\mu\left(\lambda C_l-\sum_{i\in S_l}x_i\right)^2$$
$$\text{subject to}\quad x_i\geqslant 0,\ \forall i\in S$$
$$\mu>0,$$

where μ is a penalty factor. This problem is much easier to solve than ($\mathbf{P_E}$). Moreover, the following theorem shows that ($\mathbf{P_E}$) and ($\mathbf{P_P}$) have the same and unique optimal solution.

Theorem 1. *Suppose that there exists a single bottleneck link and the link is l. Then, under part* (a) *of Assumption* 1, ($\mathbf{P_E}$) *and* ($\mathbf{P_P}$) *have the same and unique optimal solution.*

Proof. Let $\tilde{x}^*$ be an optimal solution of ($\mathbf{P_P}$), then the following condition holds.

$$\begin{aligned}&U^l_{\mathrm{av}}(\tilde{x}^*)=U_i(\tilde{x}^*_i),\ \forall i\in B_l\\&\lambda C_l=\sum_{i\in B_l}\tilde{x}^*_i\\&\tilde{x}^*_i>0,\ \forall i\in S.\end{aligned}\tag{1}$$

We first prove the existence of such a solution $\tilde{x}^*$. The above condition obviously results in the minimum of the objective function of ($\mathbf{P_P}$). Combining the first and second equation of (1) yields

$$\tilde{x}^*_i+\sum_{j\in B_l,j\neq i}U_j^{-1}\left(U_i(\tilde{x}^*_i)\right)=\lambda C_l,\quad \forall i\in B_l.\tag{2}$$

Due to part (a) of Assumption 1, the left hand side of (2) is zero when $\tilde{x}^*_i=0$ and a strictly increasing function of $\tilde{x}^*_i$ for all $i\in B_l$, and thus $\tilde{x}^*$ satisfying the first and second equation of (1) exists. It is easy to see, from part (a) of Assumption 1, that such an $\tilde{x}^*$ satisfies the third condition of (1).

The condition (1) is nothing but the optimality condition of ($\mathbf{P_E}$). Thus if we let x^* be an optimal solution of ($\mathbf{P_E}$), x^* satisfies (1). We now show that $\tilde{x}^*$ satisfying (1) is unique so that $x^*=\tilde{x}^*$. Suppose $x^*\neq\tilde{x}^*$, then there exist $i,j\in B_l$ such that $\tilde{x_i}^*>x^*_i$ and $\tilde{x_j}^*<x^*_j$ because $\lambda C_l=\sum_{i\in B_l}\tilde{x}^*_i=\sum_{i\in B_l}x^*_i$. As a consequence, we have $U_i(\tilde{x_i}^*)>U_i(x^*_i)$ and $U_j(\tilde{x_j}^*)<U_j(x^*_j)$, which implies that $U_i(x^*_i)\neq U_j(x^*_j)$ because $U_i(\tilde{x_i}^*)=U_j(\tilde{x_j}^*)$. This contradicts to the assumption that x^* is an optimal solution. Therefore, $\tilde{x}^*$ is unique so that $x^*=\tilde{x}^*$. □

Hence, we will solve ($\mathbf{P_P}$) instead of ($\mathbf{P_I}$).

Let $X_l=\sum_{i\in S_l}x_i$ and $f_l(x)$ be the objective function of ($\mathbf{P_P}$), i.e., $f_l(x)=\sum_{i\in B_l}(U^l_{\mathrm{av}}(x)-U_i(x_i))^2+\mu(\lambda C_l-X_l)^2$. If $f_l(x)$ is convex, the optimal solution is easy to find because any stationary point of a convex objective function $f(x)$, i.e., x such that $\nabla f(x)=0$, is a global minimizer of $f(x)$. However, $f_l(x)$ is not always convex. Fig. 1 shows an example of non-convex $f_l(x)$. Nevertheless, we have the following theorem which shows that a stationary point of $f_l(x)$ is the unique optimal solution of ($\mathbf{P_P}$).

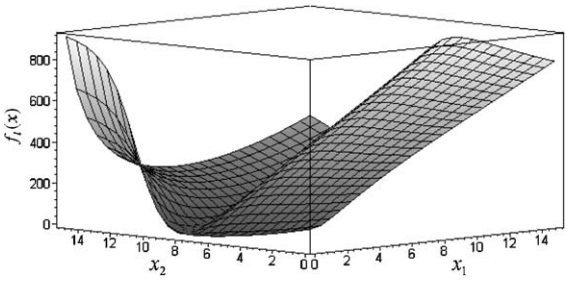


Fig. 1. Plot of $f_l(x)$ when $U_1(x_1)=10\log(x_1+1)$ and $U_2(x_2)=\frac{30}{1+e^{-(x_2-10)}}$.

Theorem 2. *Suppose that there exists a single bottleneck link and the link is l. Then, under part* (a) *of Assumption* 1, x^* *is the unique optimal solution of* ($\mathbf{P_P}$) *if and only if* $\nabla f_l(x^*)=0$.

Proof. Let $\nabla_i f_l(x)=\partial f_l(x)/\partial x_i$, then

$$\nabla_i f_l(x)=-2\left\{U_i'(x_i)\left(U^l_{\mathrm{av}}(x)-U_i(x_i)\right)+\mu(\lambda C_l-X_l)\right\},\tag{3}$$

and we have $\nabla_i f_l(x)=0,\ \forall i\in S$ at a stationary point of $f_l(x)$. The proof of the sufficiency is trivial because the unique optimal solution of ($\mathbf{P_P}$), say x^*, satisfies the condition (1) and consequently $\nabla f_l(x^*)=0$ holds. For the converse, we show that $U^l_{\mathrm{av}}(x)=U_i(x_i),\forall i\in B_l$ and $\sum_{i\in B_l}x_i=\lambda C_l$ must hold at the stationary point. Assume that $\sum_{i\in B_l}x_i<\lambda C_l$ at the stationary point, then it is true that $U^l_{\mathrm{av}}(x)<U_i(x_i),\forall i\in B_l$ because we have $U_i'(x_i)>0,\forall i\in B_l$ and $\mu>0$. However, $U^l_{\mathrm{av}}(x)<U_i(x_i),\forall i\in B_l$ cannot hold due to $\sum_{i\in B_l}\{U^l_{\mathrm{av}}(x)-U_i(x_i)\}=0$. Similarly, it can be shown that $\sum_{i\in B_l}x_i>\lambda C_l$ is impossible to happen at the stationary point. Thus, at the stationary point, there hold $\sum_{i\in B_l}x_i=\lambda C_l$ and $U^l_{\mathrm{av}}(x)=U_i(x_i),\forall i\in B_l$. This is equivalent to the optimality condition (1), so we can conclude that the stationary point of $f_l(x)$ is unique and is the unique optimal solution of ($\mathbf{P_P}$). □

Accordingly, our objective is now to find the unique stationary point of $f_l(x)$.

3. Asynchronous distributed algorithm and its convergence

In this section, we discuss a distributed implementation of our utility max–min flow control algorithm and its convergence. We find the stationary point by gradient projection method which is most commonly used to solve constrained optimization problems.

3.1. Distributed implementation

Suppose that link l is a single bottleneck link, and that each session i can be expressed as a pair (source i, receiver i). Then, the transmission rate of source i is computed by

$$x_i \leftarrow [x_i - \gamma \nabla_i f_l(x)]_i^+, \tag{4}$$

where $\gamma > 0$ is a step size, $[\cdot]_i^+$ denotes $\max\{0, \cdot\}$, and $\nabla_i f_l(x)$ is given by (3). Link l maintains two main variables, $AggRate^l$ and $AvgU^l$, which respectively contain the averaged aggregate arrival rate and the averaged utility of sessions bottlenecked at the link. The initial values of $AggRate^l$ and $AvgU^l$ are all zero. We assume that in the header of a packet, there exist two fields, U and C, used to carry the information for the computation (4), and that a receiver generates and sends an acknowledgement(ACK) packet as soon as it receives a data-(DAT) packet. The following are the algorithms carried out by links, sources and receivers. Here, the averaging factors α and β fall in $(0, 1)$, and *CurRate* denotes the instantaneous aggregate arrival rate.

Link l

Upon every arrival of N^a packets:

Estimate $CurRate^l$.

$AggRate^l \leftarrow (1-\alpha)\cdot AggRate^l + \alpha \cdot CurRate^l$

Upon arrival of a data packet:

$AvgU^l \leftarrow (1-\beta)\cdot AvgU^l + \beta \cdot DAT.U$

$DAT.U \leftarrow AvgU^l - DAT.U$

$DAT.C \leftarrow \lambda C_l - AggRate^l$

Source i

According to its transmission rate:

Send a data packet with $DAT.U = U_i(x_i)$.

Upon arrival of an ACK packet:

$x_i \leftarrow [x_i + 2\gamma\{U_i'(x_i)\cdot ACK.U + \mu \cdot ACK.C\}]_i^+$

Receiver i

Upon receipt of a data packet:

Generate and transmit an ACK packet with $ACK.U = DAT.U$ and $ACK.C = DAT.C$.

We can see from the above that in contrast to the previous work [9], our algorithm does not require any per-flow operation in the network, thereby being scalable. Furthermore, the algorithm in [10] estimates the number of locally bottlenecked sessions at each link which is not simple, but our algorithm does not need that. As we will mention in Section 4, these features also hold for the algorithm accommodating multiple bottleneck links.

3.2. Convergence analysis

In the real network, the communication delays between any two nodes are inevitable and the global clock synchronization is impossible. Accordingly, to analyze the convergence, we need to express the distributed implementation as an asynchronous distributed algorithm, which involves the communication delay and the asynchronism.

Let $T = \{1, 2, \ldots\}$ be a set of indices which correspond to the sequence of physical times at which one or more components of vector x are updated. For each $i \in S$, define $T^i \subseteq T$ to be a set of times at which x_i is updated. For each i, $j \in S$ and each $t \in T^i$, a variable $\tau_j^i(t) \in T$ denotes the time at which x_j used in the update of x_i is generated. Note that $t - \tau_j^i(t)$ and $t - \tau_i^i(t)$ can be viewed as a communication delay from source i to source j via link l and the round trip delay of source i, respectively. Let $x^i(t)$ be a local rate vector at each source $i \in S$, that is, $x^i(t) = (x_j(\tau_j^i(t)), j \in S)$. Then, if $t \in T^i$, the update of x_i is given by

$$x_i(t+1) = [x_i(t) - \gamma \nabla_i f_l(x^i(t))]_i^+. \tag{5}$$

Otherwise, $x_i(t+1) = x_i(t)$.

Assumption 2 (*Partial asynchronism*). There exists an integer $B > 0$ such that: (a) For every $i \in S$ and for every $t \in T$, at least one of the elements of the set $\{t, t+1, \ldots, t+B-1\}$ belongs to T^i. (b) There holds

$$\max\{0, t-B+1\} \leqslant \tau_j^i(t) \leqslant t, \tag{6}$$

for all $i, j \in S$ and all $t \geqslant 0$.

By (a), the interval of any consecutive update is bounded, and by (b), the communication delay

between any two sources is bounded. We can prove the following theorem.

Theorem 3. *Under Assumption* 1 *and* 2, *there exists a constant* $\gamma_0 > 0$ *such that if* $0 < \gamma < \gamma_0$, *then starting from any initial rate vector* $x(0) \geqslant 0$, *every limit point of* $\{x(t)\}$ *generated by the asynchronous algorithm* (5) *is the unique minimizer of* $f_I(x)$.

Proof. See Appendix A. □

From this theorem, we can notice that with a sufficiently small step size $\gamma > 0$, the algorithm converges to the unique optimal solution.

4. Extension to the case of multiple bottleneck links

So far, we have proposed and analyzed a utility max–min flow control algorithm for the case of a single bottleneck link. For the generalization of the algorithm, we have to resolve two issues. One is the problem of M_i, which is the maximum possible transmission rate of session i. That is, some sessions can be bottlenecked at their sources by M_i's, but we could not deal with the case in the above section. The other is the case of multiple bottleneck links, by which we mean that there exist one or more bottleneck links in the network. In this section, we discuss the extension of the utility max–min flow control algorithm focusing on the two issues.

Suppose that there exist multiple bottleneck links in the network, then B_l is not necessarily equal to S_l for all $l \in L$. Moreover, some sessions could be bottlenecked at their sources. Thus, if $U^l_{\mathrm{av}}(x)$ is updated as stated in the above subsection, i.e., updated whenever a data packet arrives on link l, then $U^l_{\mathrm{av}}(x)$ becomes the average value of the utilities of all sessions passing through l. However, what $U^l_{\mathrm{av}}(x)$ is expected to be is the average utility of sessions bottlenecked at link l. It is therefore necessary to identify which sessions are bottlenecked at link l.

To do this, we need three more fields, *BL*, *NewBL* and *MinU*, in the header of a data(DAT) packet. *DAT.BL* denotes the current bottleneck link of the session to which *DAT* belongs, is set by the source, and is used to decide whether or not $AvgU^l$ is updated by *DAT.U*. In detail, if *DAT.BL* is link l, $AvgU^l$ is updated by *DAT.U*. Otherwise, $AvgU^l$ is not updated. Consequently, $AvgU^l$ becomes the average utility of sessions bottlenecked at link l. *DAT.NewBL* and *DAT.MinU* respectively denote the new bottleneck link and the minimum utility allocated by the links on the path. *DAT.MinU* is initially set to $U_i(M_i)$ by the source and is compared with $AvgU^l$ of each link l on the path to find a new bottleneck link. In detail, if $MinU > AvgU^l$, then *DAT.NewBL*, *DAT.MinU* and *DAT.C* are set to l, $AvgU^l$ and $\lambda C_l - AggRate^l$ respectively. As the algorithm in the above section, the initial values of $AggRate^l$ and $AvgU^l$ are all zero, and $DAT.NewBL = -1$ means the new bottleneck link of a session is its source.

While the sessions are finding their ultimate bottleneck links, the non-zero average utility can be induced even at non-bottleneck links. Once the ultimate bottleneck links of all sessions are identified or equivalently the system reaches the steady state in finding the bottleneck links, the average utility at the non-bottleneck links should be zero by the definition of $AvgU^l$. Suppose that we reset the average utility of a non-bottleneck link to zero to follow the definition of $AvgU^l$, right after finding all sessions' ultimate bottlenecks. Then, the link will be considered as a bottleneck link of all sessions passing through it because the sessions identify their bottlenecks by comparing $AvgU^l$'s on their paths. But, the link is not an ultimate bottleneck link and thus the sessions will ultimately find their bottleneck links which are the same as the ones found before. This phenomenon will be repeated and thus the algorithm will not converge. Suppose otherwise that we leave the non-zero average utility at a non-bottleneck link unchanged. If the link is always a non-bottleneck link, then leaving the value unchanged will not cause any problem. Otherwise, there is a possibility that the algorithm converges to a point where the link is over-utilized. For example, suppose that link l is not currently a bottleneck link but it has large $AvgU^l$ which has been induced during transient period. Note that it is impossible to know how large the value will be. By the change of the network condition, the link could be a bottleneck link of some sessions. However, the sessions are likely to consider the link as a non-bottleneck because it has large $AvgU^l$, although the link is actually a bottleneck link. Consequently, the link will be over-utilized.

This problem can be resolved by nulling the average utility only at a link which has no sessions bottlenecked at it, but is over-utilized. To prevent the instability mentioned above, we recommend that the value is gradually reduced towards zero instead of immediately resetting the value to zero. Note that a session's utility is averaged into $AvgU^l$ at its

bottleneck link, and thus a link can easily recognize if it is a bottleneck link or not. In the algorithm, we halve $AvgU^l$ if the link is being over-utilized but $AvgU^l$ has not been updated while receiving N^b consecutive packets. Of course, there can be many ways to decrease the value and the way may be selected according to the requirements including convergence speed, overshoot and so forth. Below is the pseudocode of the extended algorithm.

Link l

Upon every arrival of N^a packets:

Estimate $CurRate^l$.

$AggRate^l \leftarrow (1-\alpha)\cdot AggRate^l + \alpha \cdot CurRate^l$

Upon arrival of a data packet:

If $DAT.BL == l$

$AvgU^l \leftarrow (1-\beta)\cdot AvgU^l + \beta \cdot DAT.U$

If $DAT.MinU > AvgU^l$

$DAT.MinU \leftarrow AvgU^l$

$DAT.C \leftarrow \lambda C_l - AggRate^l$

$DAT.NewBL \leftarrow l$

Upon every arrival of N^b consecutive packets without update of $AvgU^l$:

If $AggRate^l > \lambda C_l$

$AvgU^l \leftarrow AvgU^l/2$

Source i

According to its transmission rate:

Send a data packet with $DAT.U = U_i(x_i)$, $DAT.MinU = U_i(M_i)$, $DAT.BL = BL$, $DAT.C = M_i$ and $DAT.NewBL = -1$.

Upon receipt of an ACK packet:

$x_i \leftarrow [x_i + 2\gamma\{U_i'(x_i)\cdot ACK.U + \mu \cdot ACK.C\}]_i^+$

$BL = ACK.NewBL$

Receiver i

Upon receipt of a data packet:

Generate and transmit an ACK packet with $ACK.U = DAT.MinU - DAT.U$, $ACK.C = DAT.C$ and $ACK.NewBL = DAT.NewBL$.

Here, $[\cdot]_i^+$ denotes the projection on the interval $[0, M_i]$. Let L_i be the set of links on the path of session i. Then, we can see from the above source algorithm that when session i is bottlenecked at its source, i.e., $U_i(M_i) < AvgU^l$, $\forall l \in L_i$, x_i increases and is eventually throttled by M_i because the source always sees $ACK.U \geqslant 0$ and $ACK.C = M_i$ in this case. By this way, we handle the case when a session is bottlenecked at its source. We note that BL is initially set to the source, so before the first $ACK.BL$ is known to the source, $DAT.BL$ is always set to the source although we omit that in the pseudocode for simplicity. After the first ACK arrives at the source, the algorithm operates as shown in the above pseudocode.

The scalability still holds here because the algorithm does not require any per-flow operation, but more fields in the packet and more operations at the link are needed compared to the case of a single bottleneck link. Once the bottleneck link of a session is identified, it takes more than $\frac{1}{2}$RTT (round trip time) for the bottleneck link to be known to the source. Nevertheless, the algorithm converges as we show in the next section.

5. Simulation results

In this section, we show through simulations that the proposed utility max–min flow control algorithm works as designed. We used NS-2 network simulator [13] for the simulation which was carried out for the two cases, that is, the case of a single bottleneck link and the case of multiple bottleneck links. For the case of a single bottleneck link, we show that the algorithm converges to a utility max–min point for various utility functions. For the case of multiple bottleneck links, we verify the convergence of the algorithm under the scenario where a single link or several links can be bottleneck links according to the dynamic join and leave of sessions. The scenario also includes the case where some sessions are bottlenecked at their sources. Because the algorithm proposed in Section 4 is a generalized version of the one in Section 3, we used the former for all simulations.

Before showing simulation results, we introduce the utility functions used throughout the simulation. Table 1 summarizes the utility functions. As shown in the table, we use five types of utility functions

Table 1
The utility functions

Type	Short	Coefficient	Utility function
Logarithm	LOG	$a > 0$	$a \cdot \log(x+1)$
Linear	LINE	$a > 0$	$a \cdot x$
Sigmoid	SIG	$a, b, c > 0$	$a \cdot \left(\frac{1}{1+e^{-b(x-c)}} - \frac{1}{1+e^{bc}}\right)$
Arctan	ATAN	$a > 0$	$a \cdot \arctan(x)$
Quadratic	QUAD	$a > 0$	$a \cdot x^2$

which satisfy Assumption 1 and are convex, concave or neither convex nor concave. As mentioned in many papers, the logarithmic utility function represents an elastic application such as FTP whereas the sigmoidal function approximates the utility of a real-time application. Other utility functions are used to show that our algorithm can accommodate various types of utility functions although they may not have any practical correspondence. We specify a utility function by using the short forms in the second column and the coefficients in the third column. For example, LOG(a) and SIG(a,b,c) represent $a \cdot \log(x+1)$ and $a \cdot (\frac{1}{1+e^{-b(x-c)}} - \frac{1}{1+e^{bc}})$, respectively.

5.1. The case of a single bottleneck link

Fig. 2 shows the network topology in which we examine the proposed algorithm for the case of a single bottleneck link. In the figure, S_i, R_i and RT_i denote source i, receiver i and router i, respectively. The pair (S_i, R_i) constitutes session i and there are totally 10 sessions ($i = 1$–10). The utility function of each session is given as: $U_1(x_1) = \text{LOG}(1.5)$, $U_2(x_2) = \text{LOG}(2.0)$, $U_3(x_3) = \text{LINE}(0.15)$, $U_4(x_4) = \text{LINE}(0.2)$, $U_5(x_5) = \text{SIG}(10,0.5,10)$, $U_6(x_6) = \text{SIG}(10,0.3,20)$, $U_7(x_7) = \text{ATAN}(1.5)$, $U_8(x_8) = \text{ATAN}(2.5)$, $U_9(x_9) = \text{QUAD}(0.005)$ and $U_{10}(x_{10}) = \text{QUAD}(0.01)$. For better presentation, we show the utility functions in Fig. 3. The link capacity in Fig. 2 is set to be 100 Mbps, and to ensure that no sessions are throttled at their sources or access links, the maximum transmission rates (M_i) and the access link capacities of all sessions are set to be 500 Mbps. The delay of each link is randomly assigned, and the maximum of the round-trip delays of all sessions is 200 ms. The parameters used in the simulation are $\gamma = 0.001$, $\mu = 0.01$, $\lambda = 0.95$, $\alpha = 0.01$ and $\beta = 0.01$.

Fig. 4 shows the simulation results including the aggregate arrival rate, average utility, transmission rates and corresponding utilities. For all sessions, their utilities shown in Fig. 4(d) are all equal to

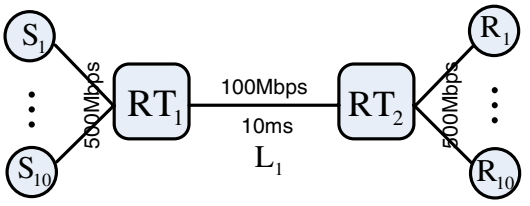


Fig. 2. The network topology for the case of a single bottleneck link.

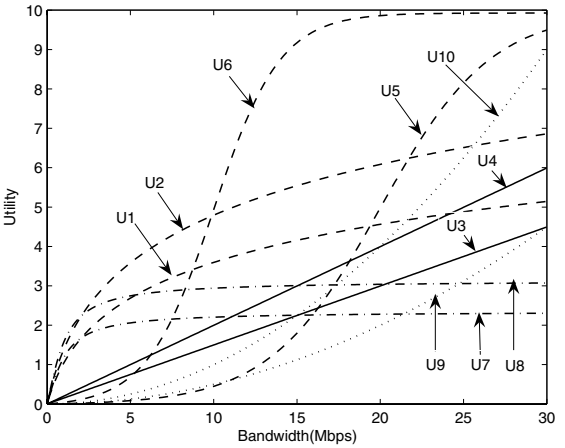


Fig. 3. Utility functions used in the simulation.

the average utility shown in Fig. 4(b). Moreover, we can see from Fig. 4(a) that the target link utilization 0.95 is achieved. So, we can conclude that for the case of a single bottleneck link, the proposed utility max–min flow control algorithm converges to the unique utility max–min point satisfying (1). Observe that the utilities of all the sessions cannot increase over $\frac{1.5\pi}{2}$ which is the upper bound of $U_7(x_7)$, and thus the transmission rates of them except session 7 cannot increase even if the remaining bandwidth is enough. Furthermore, the remaining bandwidth will be allocated to session 7. This observation implies that we need to set an upper bound (M_i) on the transmission rate of a session with bounded utility function, and the issue will be discussed in the following subsection.

5.2. The case of multiple bottleneck links

We further verify the convergence of our algorithm over the network topology shown in Fig. 5. There are 20 sessions which join and leave the network according to the arrival–departure timing diagram in Fig. 6. The utility functions of session 1–10 are equivalent to those in the above subsection, and the utility functions of other sessions are: $U_{11}(x_{11}) = \text{LOG}(1)$, $U_{12}(x_{12}) = \text{LOG}(2.0)$, $U_{13}(x_{13}) = \text{LINE}(0.3)$, $U_{14}(x_{14}) = \text{LINE}(0.2)$, $U_{15}(x_{15}) = \text{SIG}(8,0.5,15)$, $U_{16}(x_{16}) = \text{SIG}(9,0.3,17)$, $U_{17}(x_{17}) = \text{ATAN}(3)$, $U_{18}(x_{18}) = \text{ATAN}(4)$, $U_{19}(x_{19}) = \text{QUAD}(0.01)$ and $U_{20}(x_{20}) = \text{QUAD}(0.03)$. As mentioned at the end of the above subsection, we set the maximum transmission rates of the sessions with bounded utility functions to: $M_5 = 20$ Mbps, $M_6 = 30$ Mbps,

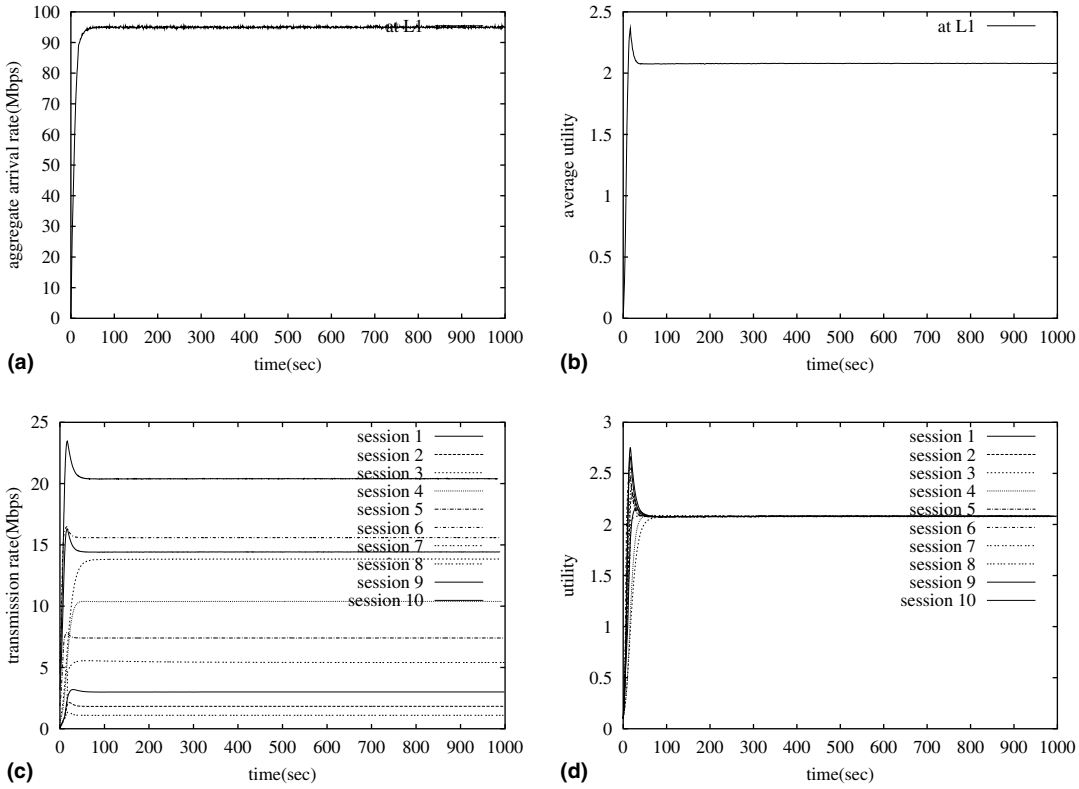


Fig. 4. Results with a single bottleneck link. (a) Aggregate arrival rate. (b) Average utility. (c) Transmission rates. (d) Utilities.

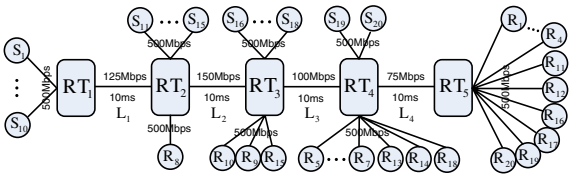


Fig. 5. The network topology for the case of multiple bottleneck links.

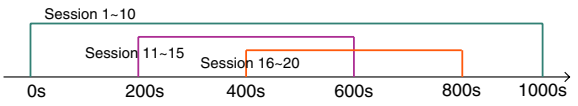


Fig. 6. The arrival and departure of sessions.

$M_7 = 5$ Mbps, $M_8 = 10$ Mbps, $M_{15} = 20$ Mbps, $M_{16} = 30$ Mbps, $M_{17} = 15$ Mbps and $M_{18} = 20$ Mbps. For all the other sessions, $M_i = 500$ Mbps is used, and all the access link capacities are set to be 500 Mbps. The link capacities in the core network are set to $C_1 = 125$ Mbps, $C_1 = 150$ Mbps, $C_3 = 100$ Mbps and $C_4 = 75$ Mbps. The delay of each link is randomly assigned, and the maximum of the round-trip delays of all sessions is 200 ms. The parameters are all equal to those in the above subsection except that the step size γ is set to 0.0025.

The simulation results are shown in Fig. 7. Let us explain the results by each period. In $[0, 200]$, there are two different utility values, 2.97 and 2.06 as shown in Fig. 7(d), and we can see from Fig. 7(b) that 2.97 is the average utility at L_1. Moreover, the aggregate arrival rate at L_1 is 118.75 Mbps as shown in Fig. 7(a), which implies that the target utilization 0.95 is achieved at L_1. This obviously shows that L_1 is a bottleneck link in the period. The other value 2.06 is the utility of session 7 bottlenecked at its source. As shown in Fig. 7(c), its transmission rate is 5 Mbps which is equal to M_7. The target link utilization is not achieved at all the other links, so we can conclude that L_1 is a single bottleneck link in the period of $[0, 200)$ and the sessions bottlenecked at the link achieve the same utility. Note that the non-zero average utilities at the non-bottleneck links are induced while the sessions are finding their bottleneck links and the values are not set to zero as we stated in Section 4.

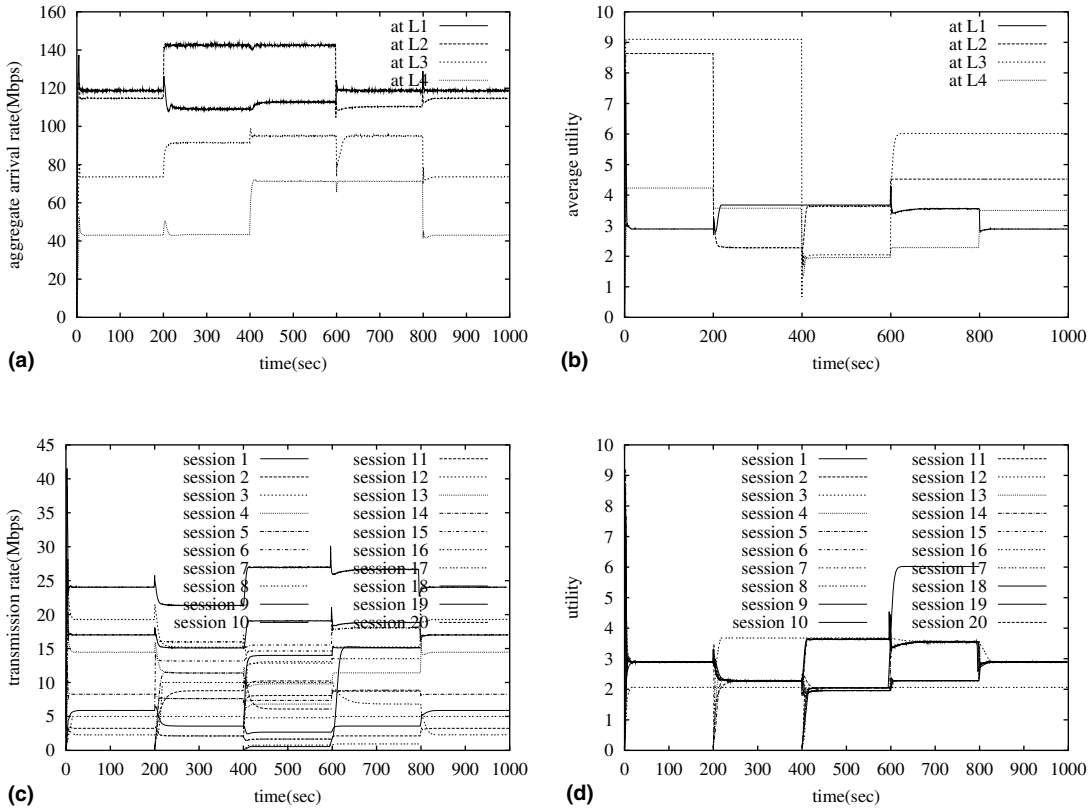


Fig. 7. Results with multiple bottleneck links. (a) Aggregate arrival rate. (b) Average utility. (c) Transmission rates. (d) Utilities.

In [200,400), session 11–15 join the network, and we can see from Fig. 7(a) that L_2 is a single bottleneck link because the target utilization is achieved only at L_2. But, Fig. 7(d) shows three different utility values, 2.06, 2.28 and 3.68, which implies that not all the sessions are bottlenecked at L_2. As shown in Fig. 7(b), 2.28 is the average utility at L_2, and as shown in Fig. 7(d), 2.06 and 3.68 are respectively the utilities of session 7 and 8 which might be bottlenecked at its source. Fig. 7(c) verifies this guess because the transmission rates of session 7 and 8 are equal to their maximum possible transmission rates 5 Mbps and 10 Mbps, respectively. Consequently, our algorithm achieves utility max–min fairness in this period. In [400, 600), session 16–20 join the network, and we can notice that L_2, L_3 and L_4 are bottleneck links from Fig. 7(a) which shows that $AggRate^2 = 142.5$ Mbps, $AggRate^3 = 95$ Mbps and $AggRate^4 = 71.25$ Mbps. As shown in Fig. 7(b), we have $AvgU^2 = 3.63$, $AvgU^3 = 2.04$ and $AvgU^4 = 1.96$, and these values also appear in Fig. 7(d). One more utility value 3.68 in Fig. 7(d) is the utility of session 8 bottlenecked at its source. Observe that in contrast to the previous period, session 7 is not bottlenecked at its source, and is bottlenecked at L_3.

In [600, 800), session 11–15 leave the network, and L_1 becomes a new bottleneck link while as L_2 becomes a non-bottleneck link as shown in Fig. 7(a). We can see from Fig. 7(b) that the average utilities of the bottleneck links are $AvgU^1 = 3.52$, $AvgU^3 = 6.02$ and $AvgU^4 = 2.28$. In this period, session 7 is bottlenecked at its source again so that there are totally four different utility values in Fig. 7(d). Hence, the sessions sharing a same bottleneck link achieve equal utility, which validates the convergence of the algorithm to a utility max–min point. In the last period [800, 1000), all the transmission rates and corresponding utilities go back to the values equal to those in [0, 200) as session 16–20 leave the network. In conclusion, the proposed algorithm works as designed even when there exist multiple bottleneck links in the network and even when some sessions are bottlenecked at their sources.

6. Concluding remarks

In this paper, we developed a utility max–min flow control algorithm based on optimization theory. The algorithm operates in asynchronous and distributed manner and does not require any per-flow operation in the network. In contrast to most of the researches dealing with utility functions, we do not restrict utility functions to be concave. Nevertheless, for the case of a single bottleneck link, we mathematically proved that the proposed algorithm converges to a unique global optimal point under some conditions. Although the analysis was performed for the case of a single bottleneck link, we showed through simulations that the algorithm works as designed for the case of multiple bottleneck links as well.

Max–min fairness is one of the most commonly used fairness criteria, but it is not an easy task to formulate the objective as an optimization problem because a bottleneck link should be identified for each flow. We guess that the formulation will be a mixed integer programming or a kind of minimax problem, which is not easy to solve. In fact, it seems to be very difficult even to find an objective function that achieves max–min fairness. Although we presented a formulation in this paper, it works only for a special case, i.e., the case of a single bottleneck link. We actually found that it is a daunting task even to involve M_i in the formulation. Due to the difficulty, we could not present the problem formulation for the general case, so finding the general formulation and proving the convergence would be an interesting issue of future study.

Acknowledgment

The authors would like to thank the editor and anonymous reviewers for their helpful comments and suggestions.

Appendix A. Proof of Theorem 3

A.1. The outline of the proof

For better understanding of the proof, we first sketch the proof. The goal is to show that (a) the amount of update goes to zero, i.e., $\lim_{t\to\infty} s_i(t) = 0$, $\forall i$ where $s_i(t) = \frac{1}{\gamma}\{[x_i(t) - \gamma\nabla_i f_l(x^i(t))]^+ - x_i(t)\}$. Once (a) is established, we can easily show that (b) the sequence $\{x(t)\}$ has a limit point x^* and (c) the error term $|\nabla_i f_l(x(t)) - \nabla_i f_l(x^i(t))|$ goes to zero for all i. By using (a)–(c), we can obtain the convergence of the algorithm to the unique optimal point. To show (a), we use non-negativity of the objective function, and descent property of the iteration, i.e., each update reduces the objective function. In more detail, we employ Lemma 3[1] to express the upper bound of the updated objective function value $f_l(x(t+1))$ in terms of $s_i(t)$, $\nabla_i f_l(x(t))$ and $\nabla_i f_l(x^i(t))$. After that, Lemmas 4 and 6 are used to rewrite the bound in terms of $\|s(t)\|_2$, and non-negativity of the function is used to show that $\|s(t)\|_2$ goes to zero. Finally, we show (b) and (c) by using Lemma 5.

A.2. The proof

We start the proof by introducing the following lemma. See [14] for the proof.

Lemma 1. *Let a function f: $D \to R^m$ be continuous for some domain $D \subset R^n$. Suppose that $\nabla f(x)$ exists and is continuous on D. If, for a convex subset $X \subset D$, there is a constant $K \geqslant 0$ such that*

$$\|\nabla f(x)\| \leqslant K$$

on X, then

$$\|f(x) - f(y)\| \leqslant K\|x - y\|, \quad \forall x, y \in X, \tag{A.1}$$

where $\|\cdot\|$ denotes an arbitrary norm.

By using this lemma, we can prove the following lemma which shows that $\nabla f_l(x)$ is Lipschitz continuous.[2]

Lemma 2. *Let $C = [\, C_l, i \in S]^T$, then there exists a constant $K > 0$ such that*

$$\|\nabla f_l(x) - \nabla f_l(y)\|_2 \leqslant K\|x - y\|_2, \quad \text{for } 0 \leqslant x, y \leqslant C.$$

Proof. By part (b) of Assumption 1, $|U_i(\cdot)|$, $|U_i'(\cdot)|$ and $|U_i''(\cdot)|$ are bounded over a compact set, and we define their bounds as: $|U_i(\cdot)| \leqslant a_{i0}$, $|U_i'(\cdot)| \leqslant a_{i1}$, and $|U_i''(\cdot)| \leqslant a_{i2}$ on $[0, C_l]$. First, observe that

[1] Lemmas 1 and 2 are needed for the proof of Lemma 3.

[2] A function satisfying (A.1) is said to be Lipschitz continuous or simply lipschitzian.

$$[\nabla^2 f_l(x)]_{ij} = \begin{cases} -2\left[U_i''(x_i)\{U_{\text{av}}^l(x) - U_i(x_i)\} + \frac{1-N_l}{N_l}(U_i'(x_i))^2 - \mu\right], & i = j \\ -2\left[\frac{1}{N_l}U_i'(x_i)U_j'(x_j) - \mu\right], & i \neq j, \end{cases}$$

and we have $\|\nabla^2 f_l(x)\|_2 \leqslant \|\nabla^2 f_l(x)\|_1$ because $\nabla^2 f_l(x)$ is symmetric and $\|\nabla^2 f_l(x)\|_2^2 \leqslant \|\nabla^2 f_l(x)\|_1^2 \cdot \|\nabla^2 \times f_l(x)\|_\infty^2$ (see p. 635 of [15]). Moreover, it follows from the definition of $\|\nabla^2 f_l(x)\|_1$ that

$$\begin{aligned} \|\nabla^2 f_l(x)\|_1 &\leqslant 2\max_{i\in B_l}\Bigg\{ |U_i''(x_i)| \cdot |U_{\text{av}}^l(x) - U_i(x_i)| \\ &\quad + |U_i'(x_i)|^2 + \frac{1}{N_l}|U_i'(x_i)| \sum_{j\in B_l} |U_j'(x_j)| + \mu N_l \Bigg\} \\ &\leqslant 2(a_0 a_2 + 2a_1^2 + \mu N_l) \triangleq K, \end{aligned}$$

where $a_0 = \max_{i\in S} a_{i0}$, $a_1 = \max_{i\in S} a_{i1}$ and $a_2 = \max_{i\in S} a_{i2}$. Consequently, it holds that $\|\nabla^2 f_l(x)\|_2 \leqslant K$ and thus $\nabla f_l(x)$ is Lipschitz continuous by Lemma 1. □

Since $\nabla f_l(x)$ is Lipschitz continuous, the next lemma holds for $f_l(x)$. See [15] for the proof.

Lemma 3 (Descent lemma). *If f: $R^n \to R$ is continuously differentiable and has the property $\|\nabla f(x) - \nabla f(y)\|_2 \leqslant K\|x - y\|$ for every x, $y \in R^n$, then*

$$f(x + y) \leqslant f(x) + y^T \nabla f(x) + \frac{K}{2}\|y\|_2^2.$$

Let $s_i(t) = \frac{1}{\gamma}\{x_i(t+1) - x_i(t)\}$, then $s_i(t) = \frac{1}{\gamma}\{[x_i(t) - \gamma\nabla_i f_l(x^i(t))]^+ - x_i(t)\}$.

Lemma 4. *For any i and t, we have*

$$s_i(t)\nabla_i f_l(x^i(t)) \leqslant -|s_i(t)|^2,$$

and thus

$$\sum_i s_i(t)\nabla_i f_l(x^i(t)) \leqslant -\|s(t)\|_2^2.$$

Proof. The projection theorem says that for some $x \in R$, $z \in X$ is equal to $[x]^+$ if and only if $(y - z)^T(x - z) \leqslant 0$ for all $y \in X$ (see p. 211 of [15]). From this theorem, we see that

$$\begin{aligned} &\left\{x_i(t) - [x_i(t) - \gamma\nabla_i f_l(x^i(t))]_i^+\right\} \\ &\quad \times \left\{x_i(t) - \gamma\nabla_i f_l(x^i(t)) - [x_i(t) - \gamma\nabla_i f_l(x^i(t))]_i^+\right\} \leqslant 0, \end{aligned}$$

which can be rewritten as

$$\begin{aligned} &-\gamma s_i(t)\{-\gamma s_i(t) - \gamma\nabla_i f_l(x^i(t))\} \leqslant 0 \\ &\Rightarrow s_i(t)\nabla_i f_l(x^i(t)) \leqslant -|s_i(t)|^2. \end{aligned}$$

Summing the above inequality over i yields the desired result. □

Below are the last two lemmas we need for the proof of the convergence.

Lemma 5. *For every i and every $t \geqslant 0$, we have*

$$\begin{aligned} |\nabla_i f_l(x(t)) - \nabla_i f_l(x^i(t))| &\leqslant 2\gamma\left(\frac{a_1^2}{N_l} + \mu\right) \\ &\quad \times \sum_{k\in B_l}\sum_{\tau=t-2B}^{t-1} |s_k(\tau)| + 2a_1^2\gamma \sum_{\tau=t-B}^{t-1} |s_i(\tau)|. \end{aligned}$$

Proof. First, observe that for all $k \in S$,

$$\begin{aligned} |x_k(t) - x_k^i(t)| &= \left|x_k(t) - x_k(\tau_k^i(t))\right| \\ &= \left|\sum_{\tau=\tau_k^i(t)}^{t-1} \gamma s_k(\tau)\right| \leqslant \gamma \sum_{\tau=t-B}^{t-1} |s_k(\tau)|. \quad \text{(A.2)} \end{aligned}$$

The last inequality is an immediate consequence of Assumption 2. As we stated in Section 3, we use averaged $U_{\text{av}}^l(x)$ and X_l in the algorithm. To involve that in $\nabla_i f_l(x^i(t))$, we define $U_{\text{av}}^{l,\text{est}}(x^i(t))$ and $X_l^{\text{est}}(x^i(t))$ to be the estimation of them. Then, they can be expressed as

$$\begin{aligned} U_{\text{av}}^{l,\text{est}}(x^i(t)) &= \sum_{\tau=t-B}^{t}\sum_{k\in B_l} \eta_k^i(\tau) U_k(x_k^i(\tau)), \\ &\sum_{\tau=t-B}^{t} \eta_k^i(\tau) = \frac{1}{N_l}, \qquad \forall k \in B_l, \\ X_l^{\text{est}}(x^i(t)) &= \sum_{\tau=t-B}^{t} \epsilon_l^i(\tau)\sum_{k\in S_l} x_k^i(\tau), \quad \sum_{\tau=t-B}^{t} \epsilon_l^i(\tau) = 1, \end{aligned}$$

where the weights $\eta_k^i(\tau)$ and $\epsilon_l^i(\tau)$ are all positive. $\nabla_i f_l(x(t))$ and $\nabla_i f_l(x^i(t))$ are given by

$$\begin{aligned} \nabla_i f_l(x(t)) &= -2\Bigg[U_i'(x_i(t))(U_{\text{av}}^l(x(t)) - U_i(x_i(t)) \\ &\qquad + \mu\left(C_l - \sum_{k\in S_l} x_k(t)\right)\Bigg], \\ \nabla_i f_l(x^i(t)) &= -2\big[U_i'(x_i(t))(U_{\text{av}}^{l,\text{est}}(x^i(t)) - U_i(x_i^i(t)) \\ &\qquad + \mu\big(C_l - X_l^{\text{est}}(x^i(t))\big)\big]. \end{aligned}$$

Notice that in $\nabla_i f_l(x^i(t))$, $U_i'(x_i(t))$ is used instead of $U_i'(x_i^i(t))$ because $U_i'(x_i(t))$ is calculated from the

current transmission rate $x_i(t)$ as shown in Section 3. It follows that

$$
\begin{aligned}
&|\nabla_i f_l(x(t)) - \nabla_i f_l(x^i(t))| \\
&\leqslant 2|U_i'(x_i(t))| \left\{ \sum_{k\in B_l} \left| \frac{1}{N_l} U_k(x_k(t)) - \sum_{\tau=t-B}^{t} \eta_k^i(\tau) U_k(x_k^i(\tau)) \right| \right. \\
&\quad \left. + |U_i(x_i(t)) - U_i(x_i^i(t))| \right\} + 2\mu \sum_{k\in S_l} \left| x_k(t) - \sum_{\tau=t-B}^{t} \epsilon_l^i(\tau) x_k^i(\tau) \right| \\
&\leqslant 2|U_i'(x_i(t))| \left\{ \sum_{k\in B_l} \frac{1}{N_l} \max_{t-B\leqslant\tau\leqslant t} \left| U_k(x_k(t)) - U_k(x_k^i(\tau)) \right| \right. \\
&\quad \left. + |U_i(x_i(t)) - U_i(x_i^i(t))| \right\} + 2\mu \sum_{k\in S_l} \max_{t-B\leqslant\tau\leqslant t} |x_k(t) - x_k^i(\tau)|.
\end{aligned}
$$

By using the mean value theorem and the boundedness of $U_i'(\cdot)$, we can write

$$
\begin{aligned}
&\leqslant 2a_{i1} \left\{ \frac{1}{N_l} \sum_{k\in B_l} \max_{t-B\leqslant\tau\leqslant t} |U_k'(y_k(t))| \cdot |x_k(t) - x_k^i(\tau)| \right. \\
&\quad \left. + \left| U_i'(z_i(t)) \right| \cdot |x_i(t) - x_i^i(t)| \right\} \\
&\quad + 2\mu \sum_{k\in S_l} \max_{t-B\leqslant\tau\leqslant t} \left| x_k(t) - x_k^i(\tau) \right| \\
&\leqslant 2a_1^2 \left\{ \frac{1}{N_l} \sum_{k\in B_l} \max_{t-B\leqslant\tau\leqslant t} \cdot |x_k(t) - x_k^i(\tau)| + \cdot |x_i(t) - x_i^i(t)| \right\} \\
&\quad + 2\mu \sum_{k\in S_l} \max_{t-B\leqslant\tau\leqslant t} \left| x_k(t) - x_k^i(\tau) \right|
\end{aligned}
$$

for some $y_k(t)$ between $[x_k(t)$ and $x_k^i(\tau)]$, and $z_i(t)$ between $[x_i(t)$ and $x_i^i(t)]$. Applying (A.2) to the above inequality yields

$$
\begin{aligned}
&\leqslant 2a_1^2 \left\{ \frac{1}{N_l} \sum_{k\in B_l} \max_{t-B\leqslant\tau\leqslant t} \gamma \sum_{\tau'=\tau-B}^{t-1} |s_k(\tau')| + \gamma \sum_{\tau=t-B}^{t-1} |s_i(\tau)| \right\} \\
&\quad + 2\mu \sum_{k\in S_l} \max_{t-B\leqslant\tau\leqslant t} \gamma \sum_{\tau'=\tau-B}^{t-1} |s_k(\tau')| \\
&= 2\gamma \left(\frac{a_1^2}{N_l} + \mu \right) \sum_{k\in B_l} \sum_{\tau'=t-2B}^{t-1} |s_k(\tau')| \\
&\quad + 2a_1^2 \gamma \sum_{\tau=t-B}^{t-1} |s_i(\tau)|,
\end{aligned}
$$

which is the desired result. □

Lemma 6. *We have*

$$
\begin{aligned}
&\sum_i |s_i(t)| \cdot |\nabla_i f_l(x(t)) - \nabla_i f_l(x^i(t))| \\
&\leqslant \gamma \left(\frac{a_1^2}{N_l} + \mu \right) \left\{ 2N_l B \|s(t)\|_2^2 + N_l \sum_{\tau=t-2B}^{t-1} \|s(\tau)\|_2^2 \right\} \\
&\quad + a_1^2 \gamma \left\{ B\|s(t)\|_2^2 + \sum_{\tau=t-B}^{t-1} \|s(\tau)\|_2^2 \right\}.
\end{aligned}
$$

Proof. By Lemma 5, we can write

$$
\begin{aligned}
&\sum_i |s_i(t)| \cdot |\nabla_i f_l(x(t)) - \nabla_i f_l(x^i(t))| \\
&\leqslant 2\gamma \left(\frac{a_1^2}{N_l} + \mu \right) \sum_i \sum_{k\in B_l} \sum_{\tau=t-2B}^{t-1} |s_i(t)| \cdot |s_k(\tau)| \\
&\quad + 2a_1^2 \gamma \sum_i \sum_{\tau=t-B}^{t-1} |s_i(t)| \cdot |s_i(\tau)| \\
&\leqslant \gamma \left(\frac{a_1^2}{N_l} + \mu \right) \sum_i \sum_{k\in B_l} \sum_{\tau=t-2B}^{t-1} (|s_i(t)|^2 + |s_k(\tau)|^2) \\
&\quad + a_1^2 \gamma \sum_i \sum_{\tau=t-B}^{t-1} (|s_i(t)|^2 + |s_i(\tau)|^2) \\
&= \gamma \left(\frac{a_1^2}{N_l} + \mu \right) \left\{ 2N_l B \|s(t)\|_2^2 + N_l \sum_{\tau=t-2B}^{t-1} \|s(\tau)\|_2^2 \right\} \\
&\quad + a_1^2 \gamma \left\{ B\|s(t)\|_2^2 + \sum_{\tau=t-B}^{t-1} \|s(\tau)\|_2^2 \right\},
\end{aligned}
$$

where the second inequality follows from the fact that $2ab \leqslant a^2 + b^2$. □

Proof of Theorem 3. By Lemma 3, we can write

$$
\begin{aligned}
f_l(x(t+1)) &= f_l(x(t) + \gamma s(t)) \\
&\leqslant f_l(x(t)) + \gamma \sum_i s_i(t) \nabla_i f_l(x(t)) \\
&\quad + \frac{1}{2} K\gamma^2 \|s(t)\|_2^2 \\
&= f_l(x(t)) + \gamma \sum_i s_i(t) \nabla_i f_l(x^i(t)) \\
&\quad + \frac{1}{2} K\gamma^2 \|s(t)\|_2^2 + \gamma \sum_i s_i(t) (\nabla_i f_l(x(t)) \\
&\quad - \nabla_i f_l(x^i(t))).
\end{aligned}
$$

By applying Lemmas 4 and 6, we can continue the proof as

$$f_l(x(t'+1)) \leqslant f_l(x(t')) - \gamma\|s(t')\|_2^2 + \frac{1}{2}K\gamma^2\|s(t')\|_2^2 + \gamma^2\left(\frac{a_1^2}{N_l}+\mu\right) \times \left\{2N_lB\|s(t')\|_2^2 + N_l\sum_{\tau=t'-2B}^{t'-1}\|s(\tau)\|_2^2\right\} + a_1^2\gamma^2\left\{B\|s(t')\|_2^2 + \sum_{\tau=t'-B}^{t'-1}\|s(\tau)\|_2^2\right\}.$$

Summing the above result from $t'=0$ to $t'=t$ yields

$$f_l(x(t+1)) \leqslant f_l(x(0)) - \gamma\left(1-\frac{1}{2}K\gamma\right)\sum_{\tau=0}^{t}\|s(\tau)\|_2^2 + \gamma^2\left(\frac{a_1^2}{N_l}+\mu\right)\left\{2N_lB\sum_{\tau=0}^{t}\|s(\tau)\|_2^2 + 2N_lB\sum_{\tau=0}^{t}\|s(\tau)\|_2^2\right\} + a_1^2\gamma^2\left\{B\sum_{\tau=0}^{t}\|s(\tau)\|_2^2 + B\sum_{\tau=0}^{t}\|s(\tau)\|_2^2\right\} = f_l(x(0)) - \gamma\left(1-\gamma\left(\frac{1}{2}K + 2(3a_1^2+2\mu N_l)B\right)\right)\sum_{\tau=0}^{t}\|s(\tau)\|_2^2 = f_l(x(0)) - \gamma\left(1-\frac{\gamma}{\gamma_0}\right)\sum_{\tau=0}^{t}\|s(\tau)\|_2^2,$$

where $\gamma_0 = \frac{1}{\frac{1}{2}K+2(3a_1^2+2\mu N_l)B}$. Taking $\lim_{t\to\infty}$ on both sides yields

$$\lim_{t\to\infty} f_l(x(t+1)) \leqslant f_l(x(0)) - \gamma\left(1-\frac{\gamma}{\gamma_0}\right)\sum_{\tau=0}^{\infty}\|s(\tau)\|_2^2.$$

Assume that $0<\gamma<\gamma_0$. Then there holds $\lim_{t\to\infty}s(t)=0$ because $f_l(\cdot)\geqslant 0$. Otherwise, $\sum_{\tau=0}^{t}\|s(\tau)\|_2^2$ grows to infinity as $t\to\infty$, which contradicts to the fact that $f_l(\cdot)\geqslant 0$. Thus, we have $\lim_{t\to\infty}s_i(t)=0$, $\forall i$. As a consequence $\lim_{t\to\infty}\|x(t+1)-x(t)\|=0$ and $\lim_{t\to\infty}|\nabla_i f_l(x(t)) - \nabla_i f_l(x^i(t))|=0$, $\forall i$ by Lemma 5. Let x^* be a limit point of $\{x(t)\}$, $\{t_k\}$ be a sequence such that $\lim_{k\to\infty}x(t_k)=x^*$, and τ_k be such that $|t_k-\tau_k|\leqslant B$ and $\tau_k\in T^i$. Note that such τ_k exists by part (a) of Assumption 2. Then, it follows that

$$\lim_{k\to\infty}\nabla_i f_l(x^i(\tau_k)) = \lim_{k\to\infty}\nabla_i f_l(x(\tau_k)) = \nabla_i f_l(x^*), \quad \forall i$$

since τ_k tends to infinity as $k\to\infty$. Finally, we can write

$$\left[x_i^* - \gamma\nabla_i f_l(x^*)\right]_i^+ - x_i^* = \lim_{k\to\infty}\left\{\left[x_i(\tau_k) - \gamma\nabla_i f_l(x^i(\tau_k))\right]_i^+ - x_i(\tau_k)\right\} = \lim_{k\to\infty}\gamma s_i(\tau_k) = 0, \quad \forall i.$$

We show that the limit point x^* is the unique stationary point $\tilde{x}^*$ of $f_l(x)$. First, $\tilde{x}^*$ obviously satisfies $[\tilde{x}^* - \gamma\nabla f_l(\tilde{x}^*)]^+ - \tilde{x}^* = 0$ because $\nabla f_l(\tilde{x}^*)=0$. Suppose $x^*\neq\tilde{x}^*$. Because $[x^* - \gamma\nabla f_l(x^*)]^+ - x^* = 0$, $(x-x^*)^T\nabla f_l(x^*)\geqslant 0$ for all $x\geqslant 0$ (see p. 213 of [15]). However, we have $(\tilde{x}^*-x^*)^T\nabla f_l(x^*)<0$ since $\tilde{x}^*$ is the unique minimizer of $f_l(x)$ and thus $\tilde{x}^*-x^*$ is the direction along which $f_l(x)$ decreases. This contradiction shows that x^* is $\tilde{x}^*$. □

References

[1] F. Kelly, Charging and rate control for elastic traffic, Euro. Trans. Telecommun. 8 (1997) 33–37.

[2] F. Kelly, A.K. Maulloo, D.K.H. Tan, Rate control for communication networks: shadow prices, proportional fairness and stability, J. Oper. Res. Soc. 49 (3) (1998) 237–252.

[3] S.H. Low, D.E. Lapsley, Optimization flow control—I: Basic algorithm and convergence, IEEE/ACM Trans. Network. 7 (6) (1999) 861–874.

[4] S. Athuraliya, V.H. Li, S.H. Low, Q. Yin, Rem: active queue management, IEEE Network 15 (3) (2001) 48–53.

[5] R.J. La, V. Anantharam, Utility-based rate control in the internet for elastic traffic, IEEE/ACM Trans. Network. 10 (2) (2002) 272–286.

[6] J. Mo, J. Walrand, Fair end-to-end window-based congestion control, IEEE/ACM Trans. Network. 8 (5) (2000) 556–567.

[7] S. Shenker, Fundamental design issues for the future internet, IEEE J. Select. Areas Comm. 13 (7) (1995) 1176–1188.

[8] J.-W. Lee, R.R. Mazumdar, N.B. Shroff, Non-convexity issues for internet rate control with multi-class services: stability and optimality, in: Proc. IEEE INFOCOM 2004, Hong Kong, 2004.

[9] Z. Cao, E.W. Zegura, Utility max–min: an application-oriented bandwidth allocation scheme, in: Proc. IEEE INFOCOM 1999, New York, 1999, pp. 793–801.

[10] J.w. Cho, S. Chong, Utility max–min flow control using slope-restricted utility functions, in: Proc. IEEE GLOBECOM 2005, St. Louis, 2005.

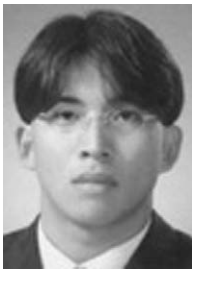
[11] A.G. Dewey, E.I. Jury, A stability inequality for a class of nonlinear feedback systems, IEEE Trans. Automat. Control 11 (1) (1966) 54–62.

[12] A. Tang, J. Wang, S.H. Low, Is fair allocation always inefficient, in: Proc. IEEE INFOCOM 2004, Hong Kong, 2004.

[13] Ns-2, Available from: <http://www.isi.edu/nsnam/ns/>.

[14] H.K. Khalil, Nonlinear Systems, third ed., Prentice Hall, NJ, 2002.

[15] D.P. Bertsekas, J.N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods, Athena Scientific, Belmont, MA, 1997.



Hyang-Won Lee received his B.S. and M.S. degrees in Electrical Engineering and Computer Science from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2001 and 2003, respectively. He is currently a Ph.D. student in the Department of Electrical Engineering and Computer Science, KAIST. His research interests include congestion/flow control, radio resource management and QoS in wireless networks.

Song Chong received his B.S. and M.S. degrees in Control and Instrumentation Engineering from Seoul National University, Seoul, Korea, in 1988 and 1990, respectively, and his Ph.D. degree in Electrical and Computer Engineering from the University of Texas at Austin in 1995. From 1994 to 1996, he was a Member of Technical Staff in the Performance Analysis Department at AT&T Bell Laboratories, Holmdel, New Jersey, USA. He is currently an Associate Professor with the Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Taejon, Korea. His research interests are in high-speed communication networks, high-performance switching/routing systems, multimedia networking and performance evaluation. He has published more than 25 papers in international journals and conferences and holds three U.S. patents with several others pending. He is an Editor and an Associate Publication Editor for the Journal of Communications and Networks. He has served as a Technical Program Committee member of IEEE INFOCOM ('97, '99, '03), an Organizing/Program Committee member of PV ('01–'04) and Technical Program Co-chair of ICBN '04.