A Distributed Utility Max-Min Flow Control Algorithm

Hyang-Won Lee and Song Chong
Department of Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology (KAIST)
mslhw@netsys.kaist.ac.kr, song@ee.kaist.ac.kr

***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 congested node and the communication delay between any two nodes in the network is bounded. Although the convergence of the algorithm is analyzed for the case of a single congested node, we show through simulations that the proposed algorithm works as designed for the case of multiple congested nodes.**

I. 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 is 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 describe 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]–[5]. 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 [6], 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. Lee et al. [7] 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 [8], 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, based on control theory, Cho et al. [9] proposed a utility max-min flow control algorithm which resolves the problems of [8], and showed that the algorithm converges to a utility max-min fair rate vector by using Dewey and Jury's stability criterion [10].

In this paper, we propose 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 congested node in the network, and then extend the algorithm to the case of multiple congested nodes. The proposed algorithm requires neither the concavity of utility functions nor any per-flow operation in the network. Moreover,

This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment).

we mathematically prove, for the case of a single congested node, that the algorithm converges to a utility max-min fair point with bounded communication delays. Here, we do not analyze the convergence of the algorithm accommodating multiple congested nodes 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.

II. 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, S_l and B_l respectively denote the capacity of link l, the set of sessions sharing link l and the set of sessions bottlenecked at link l. Each session i is associated with a utility function $U_i : [m_i, M_i] \rightarrow R_+$ where m_i and M_i respectively denote the minimum required transmission rate and the maximum possible transmission rate of session i.

Assumption 1: (a) Every $U_i(\cdot)$ is strictly increasing.
(b) Every $U_i(\cdot)$ is twice continuously differentiable.
(c) For each $i \in S$, there exist constants $a_{i0}, a_{i1}, a_{i2} > 0$ such that $|U_i(\cdot)| \leq a_{i0}$, $|U_i'(\cdot)| \leq a_{i1}$, and $|U_i''(\cdot)| \leq a_{i2}$ on $[m_i, M_i]$.

Definition 1: 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 $U_i(x_i) < U_i(x_i^*) \leq U_j(x_j^*)$.
Definition 1 is equivalent to the definition of bandwidth max-min fairness except that in utility max-min fairness, utilities are compared instead of bandwidths.

Assume that every session is bottlenecked at a same node or equivalently there is only a single congested node in the network. Suppose that l is a congested link at the node, then $B_l = S_l = S$. Let x_i be the transmission rate of session i. We formulate a utility max-min flow control problem as an optimization problem by

$$(\mathbf{P_E}): \quad \min. \sum_{i \in B_l} \left(U_{av}^l(x) - U_i(x_i)\right)^2$$
$$\text{subject to} \quad \sum_{i \in B_l} x_i = \lambda C_l$$
$$m_i \leq x_i \leq M_i, \ \forall\, i \in S,$$

where $0 \leq \lambda \leq 1$ is a target utilization and $x = [x_i, i \in S]^T$, $U_{av}^l(x) = \frac{1}{N_l}\sum_{i \in B_l} U_i(x_i)$ and $N_l = |B_l|$. The optimal solution of $(\mathbf{P_E})$ obviously achieves a utility max-min fairness among competing flows and also a target utilization of link l. However, it is not easy to obtain the optimal solution due to the capacity constraint. So, instead of solving $(\mathbf{P_E})$ as it is, we relax the constraint by adding a penalty function as follows:

$$(\mathbf{P_P}): \quad \min. \sum_{i \in B_l} \left(U_{av}^l(x) - U_i(x_i)\right)^2 + \mu\left(\lambda C_l - \sum_{i \in B_l} x_i\right)^2$$
$$\text{subject to} \quad m_i \leq x_i \leq M_i, \ \forall\, i \in S$$
$$\mu > 0$$

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. For the proof, see the extended version of this paper [11].

Theorem 1: Suppose that $\sum_{i \in B_l} m_i \leq C_l$. Then, under part (a) of Assumption 1, $(\mathbf{P_E})$ and $(\mathbf{P_P})$ have the same and unique optimal solution.
This theorem says that the optimality condition of $(\mathbf{P_E})$ and $(\mathbf{P_E})$ is equivalently given as

$$\begin{aligned} &U_{av}^l(\tilde{x}^*) = U_i(\tilde{x}_i^*), \ \forall i \in B_l \\ &\lambda C_l = \sum_{i \in B_l} \tilde{x}_i^* \\ &m_i \leq \tilde{x}_i^* \leq M_i, \ \forall\, i \in S \end{aligned} \tag{1}$$

Hence, we may solve $(\mathbf{P_P})$ to find the optimal solution of $(\mathbf{P_E})$.

Let $X_l = \sum_{i \in B_l} x_i$ and $f_l(x)$ be the objective function of $(\mathbf{P_P})$, i.e., $f_l(x) = \sum_{i \in B_l} \left(U_{av}^l(x) - U_i(x_i)\right)^2 + \mu\left(\lambda C_l - X_l\right)^2$. Then, $\nabla_i f_l(x) = \partial f_l(x)/\partial x_i$ is written by

$$\nabla_i f_l(x) = -2\left\{U_i'(x_i)\left(U_{av}^l(x) - U_i(x_i)\right) + \mu\left(\lambda C_l - X_l\right)\right\} \tag{2}$$

and $\nabla f_l(x) = [\nabla_i f_l(x), i \in S]^T$. If $f_l(x)$ is convex, the optimal solution is easy to find because any stationary point of a convex function $f(x)$, i.e., x such that $\nabla f(x) = 0$, is a global minimizer of $f(x)$. However, we can easily show that $f_l(x)$ is not always convex. Nevertheless, we have the following theorem which shows that a stationary point of $f_l(x)$ is the unique optimal solution of $(\mathbf{P_P})$. See [11] for the proof.

Theorem 2: Under part (a) of Assumption 1, x^* is the unique optimal solution of $(\mathbf{P_P})$ if and only if $\nabla f_l(x^*) = 0$.
Accordingly, our objective is now to find the unique stationary point of $f_l(x)$.

III. Asynchronous Distributed Algorithm and Its Convergence

In this section, we discuss the 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.

A. Distributed Implementation

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

$$x_i \leftarrow [x_i - \gamma \nabla_i f_l(x)]_i^+ \tag{3}$$

where $\gamma > 0$ is a step size and $[\cdot]_i^+$ denotes the projection on the interval $[m_i, M_i]$ and $\nabla_i f_l(x)$ is given by (2). 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. We assume that in the header of a packet, there exist two fields, U and C, used to carry the information for the computation (3) 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 $ACK.C = DAT.C$.
```

As seen in the above, different from the previous work [8], the proposed algorithm does not require any per-flow operation in the network, thereby being scalable. Furthermore, the algorithm in [9] 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 IV, these features also hold for the case of multiple congested nodes.

B. 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. For the analytical tractability, we assume that only the most recent values of x_i's are used to compute $AggRate^l$ and $AvgU^l$. Note, however, that $AggRate^l$ and $AvgU^l$ are the values averaged over time in the implementation.

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, $x^i(t) = \big(x_j(\tau_j^i(t)), j \in S\big)$. Then, if $t \in T^i$, the update of x_i is given by

$$x_i(t+1) = \big[x_i(t) - \gamma \nabla_i f_l(x^i(t))\big]_i^+ \tag{4}$$

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 $t \geq 0$, at least one of the elements of the set $\{t, t+1, \ldots, t+B-1\}$ belongs to T^i.
(b) There holds

$$\max\{0, t-B+1\} \leq \tau_j^i(t) \leq t, \tag{5}$$

for all $i, j \in S$ and all $t \geq 0$.
By (a), the interval of any consecutive update is bounded, and by (b), the communication delay between any two sources is bounded. Let $m = [m_i, i \in S]^T$ and $M = [M_i, i \in S]^T$, then we have the following theorem. See [11] for the proof.

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 $m \leq x(0) \leq M$, every limit point of $\{x(t)\}$ generated by the asynchronous algorithm (4) is the unique minimizer of $f_l(x)$.
From this theorem, we can notice that with a sufficiently small step size $\gamma > 0$, the algorithm converges to the unique optimal solution.

IV. Extension to The Case of Multiple Congested Nodes

So far, we have proposed and analyzed a utility max-min flow control algorithm for the case of a single congested node. In this section, we discuss the extension of the utility max-min flow control algorithm to the case of multiple congested nodes.

Suppose that there exist multiple congested nodes in the network and one of the bottleneck links is l, then B_l is not equal to S_l any more. Thus, if $U_{av}^l(x)$ is updated as stated in the above subsection, i.e., updated whenever a data packet arrives on link l, then $U_{av}^l(x)$ becomes the average value of the utilities of all sessions passing through l. However, what $U_{av}^l(x)$ 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 packet. BL denotes the current bottleneck link, thus if $DAT.BL$ is equal to link l, $AvgU^l$ is updated by $DAT.U$. $MinU$ is set to a big number $MaxU$ by the source and is compared with $AvgU^l$ of each link l on the path to find a new bottleneck link. That is, if $MinU > AvgU^l$, then $NewBL$ and $MinU$ is respectively set to link l and $AvgU^l$. Below is the pseudocode of our algorithm to accommodate multiple congested nodes.

```
Link $l$
```

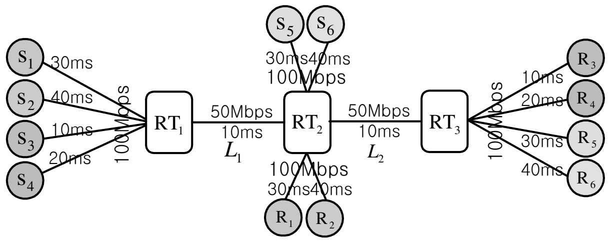


Fig. 1. The network topology

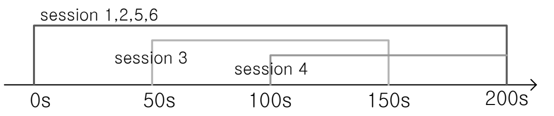


Fig. 2. The arrival and departure of sessions

```
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 $DAT.MinU > AvgU^l$
        $DAT.MinU \leftarrow AvgU^l$
        $DAT.C \leftarrow \lambda C_l - AggRate^l$
        $DAT.NewBL \leftarrow l$
Source $i$
    According to its transmission rate:
        Send a data packet with $DAT.U = U_i(x_i)$,
        $DAT.MinU = MaxU$ and $DAT.BL = BL$.
    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 $ACK.NewBL = DAT.NewBL$.
```

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 congested node. 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.

V. 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 [12]. The step size, the penalty factor and the target utilization used for simulations are $\gamma = 0.0035$, $\mu = 0.075$ and $\lambda = 0.95$.

We examine the proposed algorithm in the network topology shown in Fig. 1. 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 6 sessions. Each utility function is given as: $U_1(x_1) = 1.5\log(x_1+1)$, $U_2(x_2) = 2.0\log(x_2+1)$, $U_3(x_3) = 0.15x_3$, $U_4(x_4) = 0.2x_4$, $U_5(x_5) = 10\left(\frac{1}{1+e^{-0.5(x_5-10)}} - \frac{1}{1+e^5}\right)$ and $U_6(x_6) = 10\left(\frac{1}{1+e^{-0.3(x_6-20)}} - \frac{1}{1+e^6}\right)$. 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. The linear utility function represents an application whose satisfaction increases linearly to the bandwidth allocated to it. For better presentation, we show the utility function in Fig. 3. The link capacity between any two routers is set to 50Mbps. On the other hand, the link capacities of the access links are all set to 100Mbps so that no sessions are throttled there. The propagation delay of each link is displayed in the figure. Fig. 2 shows the arrival-departure scenario of sessions over time, and the theoretical fair rates under the scenario are summarized in Table I.

The simulation results are shown in Fig. 4. Observe from Fig. 4(a) that the utilizations of L_1 and L_2 are always a target value 0.95. In the period [0,50), session 1 and 2 share link L_1, and session 5 and 6 share link L_2, so the simulation scenario can be considered as two independent single-bottleneck scenarios. As seen in Fig. 4(c), in that period, the transmission rates of session 1 and 2 are equal to the theoretical ones and, we can see from Fig. 4(d) that the utilities of session 1 and 2 are equal to the average utility at L_1 shown in Fig. 4(b). For session 5 and 6, the same result is achieved at link L_2. This result shows that the utility max-min fairness is achieved for the case of single congested node while the link utilization is kept at a target value. From the time when session

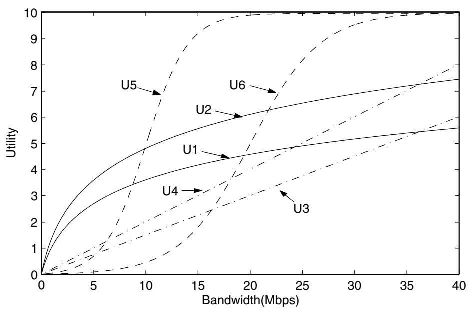
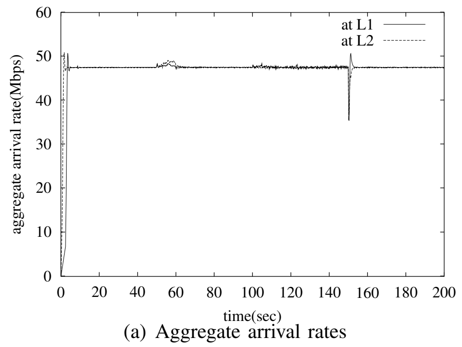


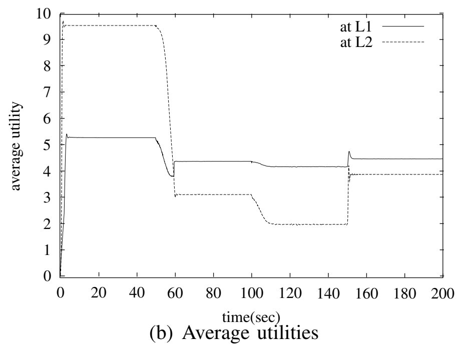
Fig. 3. Utility functions used in the simulation

TABLE I
THE THEORETICAL FAIR RATES(MBPS) OVER TIME

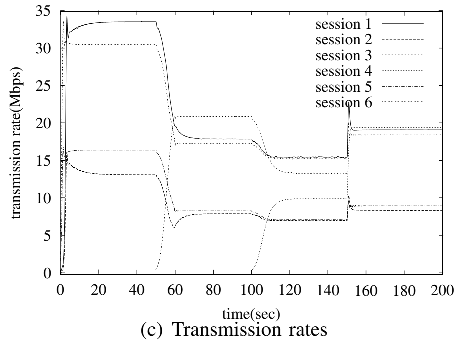
session	0∼50	50∼100	100∼150	150∼200
1	34.1	18.0	15.7	19.3
2	13.4	8.1	7.3	8.5
3	-	21.4	14.0	-
4	-	-	10.5	19.7
5	16.7	8.6	7.4	9.2
6	30.8	17.5	15.6	18.6



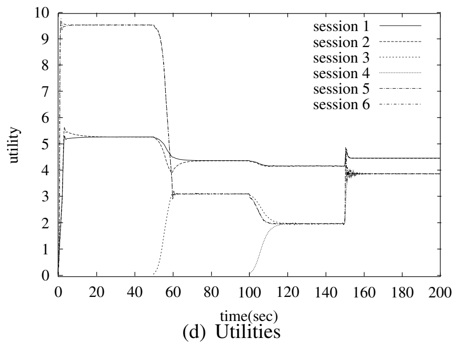
(a) Aggregate arrival rates



(b) Average utilities



(c) Transmission rates



(d) Utilities

Fig. 4. Results with multiple bottleneck link configuration

3 joins the network, the simulation scenario becomes to reflect the case of multiple congested nodes. In the period [50,100), the result agrees with what we expected theoretically. Notice that the utility of session 3 in that period is equal to the average utility at L_2, which implies that session 3 is bottlenecked at L_2. In the period [100,150), session 4 joins the network, and we can see from Fig. 4(d) that the bottleneck link of session 3 and 4 is L_2 because the utilities of them are equal to the average utility at L_2. The expected result is achieved for the period [150,200). In brief, the proposed utility max-min flow control algorithm works as designed when there exist multiple congested nodes in the network.

VI. Concluding Remarks

In this paper, we developed a utility max-min flow control algorithm which is based on the optimization theory. The algorithm operates in asynchronous and distributed manner and does not require any per-flow operation in the network. Different from most of the researches dealing with utility functions, we did not assume that every utility function is strictly concave. Nevertheless, for the case of a single congested node, we mathematically proved that the algorithm converges to a unique global optimal point under some conditions. Although the analysis was performed for the case of a single congested node, we showed through simulations that the algorithm works as designed for the case of multiple congested nodes as well.

References

[1] F. Kelly, "Charging and rate control for elastic traffic," *Euro. Trans. Telecommun.*, vol. 8, pp. 33–37, Dec. 1997.

[2] F. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *J. Oper. Res. Soc.*, vol. 49, no. 3, pp. 237–252, Mar. 1998.

[3] S. H. Low and D. E. Lapsley, "Optimization flow control-II: Implementation," *Internal Report, Melbourne University*, May 2000.

[4] S. Athuraliya and S. H. Low, "Optimization flow control-I: Basic algorithm and convergence," *IEEE/ACM Trans. Networking*, vol. 7, no. 6, pp. 861–874, Dec. 1999.

[5] R. J. La and V. Anantharam, "Utility-based rate control in the internet for elastic traffic," *IEEE/ACM Trans. Networking*, vol. 10, no. 2, pp. 272–286, Apr. 2002.

[6] S. Shenker, "Fundamental design issues for the future internet," *IEEE J. Selected Areas Comm.*, vol. 13, no. 7, pp. 1176–1188, Sep. 1995.

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

[8] Z. Cao and E. W. Zegura, "Utility max-min: An application-oriented bandwidth allocation scheme," in *Proc. IEEE INFOCOM 1999*, New York, Mar. 1999, pp. 793–801.

[9] J. w. Cho and S. Chong, "Utility max-min flow control using slope-restricted utility functions," *Submitted for Publication*.

[10] A. G. Dewey and E. I. Jury, "A stability inequality for a class of nonlinear feedback systems," *IEEE Trans. Automatic Control*, vol. 11, no. 1, pp. 54–62, Jan. 1966.

[11] H.-W. Lee and S. Chong, "A distributed utility max-min flow control algorithm," *Extended version of this paper*, 2004. [Online]. Available: http://netsys.kaist.ac.kr/∼mslhw/icc2005extd.pdf

[12] Ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/