$\mathcal{H}^\infty$–BASED FLOW CONTROL FOR ATM NETWORKS WITH MULTIPLE BOTTLENECKS

İnci Munyas–Elmas and **Altuğ İftar**

Department of Electrical and Electronics Engineering
Anadolu University
26470 Eskişehir, Turkey

imunyas@anadolu.edu.tr aiftar@anadolu.edu.tr

Abstract: In this paper, the implementation of an $\mathcal{H}^\infty$ controller, designed in another work for the flow control in communication networks with a single bottleneck, is considered for asynchronous transfer mode (ATM) networks with multiple bottlenecks. The controller calculates the data sending rates by using the information about the queue lengths at the switches. To observe the performance of the system in real ATM networks, a number of simulations are realised for certain realistic cases and the results are presented.

Keywords: Communication networks, flow control, robust control, time-delay systems, multiple bottlenecks

1. INTRODUCTION

For any communication network, *resource management*, that is, to make the best use of available resources while guaranteeing the quality of service for users is the main problem. One aspect of this problem is traffic management which deals with developing algorithms to control the traffic in the network. The traffic management mechanisms are designed to solve the congestion problem which may lead to long queueing delays or cell losses.

The ATM Forum defined a flow control framework for Available Bit Rate (ABR) service which is a rate–based closed–loop scheme (Bonomi and Fendick, 1995). The main goal of this scheme is to fairly allocate the bandwidth leftover from Constant Bit Rate (CBR) and Variable Bit Rate (VBR) users to ABR users. The remaining bandwidth is distributed to Unspecified Bit Rate (UBR) users. In the ABR flow control scheme, the feedback information from the network to the users is carried by the Resource Management (RM) cells. However, the existence of the time delays in transmission and queueing and that the number of the active and controllable sources are unknown are the challenges faced in flow control.

There are many papers in the literature dealing with flow and congestion control, e.g., (Altman *et al.*, 1998), (Özbay *et al.*, 1998), (Quet *et al.*, 2002), (BenMohamed and Meerkov, 1993), (Mascolo, 1997), (Gomez-Stern *et al.*, 2002), (Mascolo and Cavendish, 1996) and the references there in. The design of a controller for the single bottleneck case using $\mathcal{H}_\infty$ control approach is given in (Quet *et al.*, 2002). The designed controller is robust to uncertain time-varying multiple time-delays and forces the queue length at the bottleneck node to the desired steady-state value asymptotically and also satisfies a weighted fairness condition. Depending on this work, the controller design for multi–bottleneck networks is given in (Biberoviç, 2001), (Biberoviç *et al.*, 2001). The implementation of the controller, the design of which is given in these works, is presented in

(Munyas *et al.*, 2003). Although the flow control approach presented and simulated in these works are for the multi–bottleneck case, it cannot be directly applied to ABR traffic in ATM networks. Because, ABR flow control framework has some special rules to be followed and a very general network structure is considered in those works. Also in (BenMohamed and Meerkov, 1997), controller design for the multi–bottleneck case is considered. However, according to the design method presented in that work, it is assumed that the system reaches steady state which may not be a realistic assumption.

In this paper, a congestion control algorithm for ATM networks with multiple bottlenecks is considered. Multiple bottlenecks means that there may exist more than one bottlenecked output port at the switches on the path of any connection. When it is attempted to design an $\mathcal{H}^\infty$ controller for this Multi-Input-Multi-Output (MIMO) system, it is seen that the resultant mathematical model of the network is so complex that it is not feasible (if not impossible) to define an $\mathcal{H}^\infty$ optimization problem. Thus, the multi–bottleneck case in the controller design problem is reduced to the single bottleneck case by making an assumption which does not violate the requirements of the ABR control framework. Since the $\mathcal{H}^\infty$ controller for the single bottleneck case, for a general network structure, has already been designed in (Quet *et al.*, 2002), to implement the suggested algorithm, that controller is used. The $\mathcal{H}^\infty$ controllers are implemented at each output port of each node and each controller uses the queue length information of the port at which it is implemented. Furthermore, to deal with the case when a bottleneck node becomes a non-bottleneck, a reset mechanism is also included. The closed–loop system is simulated for several cases and the results for two representative cases are presented.

2. PROBLEM STATEMENT AND CONTROLLER DESIGN

In this work, a network which consists of l source–destination pairs communicating over N nodes is considered. The network traffic consists of flows corresponding to each source–destination pair, (s,d). In this way, the traffic sent from multiple users connected to a node s to one or more users connected to a node d, forms a single connection between s and d. Let $\mathcal{C}$ denote the set of such connections and $\mathcal{C}_{i_k}$ be the set of connections traversing the kth output port of the ith node. Besides, let $s(j)$ and $d(j)$, $j \in \mathcal{C}$, $j = 1, \ldots, l$, denote the source and destination nodes of the jth connection, respectively. l_{i_k} represents the number of elements in the set $\mathcal{C}_{i_k}$.

Let $p_j = \{N_j^1, \ldots, N_j^{n_j}\}$, $j = 1, 2, \ldots, l$, denote the sequence of the nodes, the path, traversed by the jth connection where n_j is the number of nodes in p_j. For each connection j, $r^{j,0}(t)$ denotes the traffic demand of $s(j)$. Each node i has buffers at its output ports for storing packets waiting to be transmitted. The number of output ports of node i is denoted by k_i while i_k represents the kth output port of node i. Traffic at each output port is transmitted according to the First-In-First-Out (FIFO) discipline. $s(j)$ is allowed to send data through the network with the following rate;

$$r^{j,s}(t) = min\left\{r^{j,0}(t); r^j_{N_j^1}(t - \tau^{j,b}_{N_j^1}(t)); \ldots; r^j_{N_j^{n_j}}(t - \tau^{j,b}_{N_j^{n_j}}(t))\right\}, j \in \mathcal{C} \quad (1)$$

Here, $\tau^{j,b}_{N_j^m}(t)$, $m = 1, \ldots, n_j$, represents the backward time delay from the mth node in p_j to $s(j)$. $r^j_{N_j^m}(t)$ is the rate command determined by the mth node for the jth connection at time t. It is also assumed that the routing policy is static. Therefore, the sequence of the nodes and output ports of the nodes traversed by the connections are already known. Since each connection uses only one output port of a node, the command rates determined by the nodes are actually the rates calculated by the controller implemented at the corresponding output port of the node. The dynamics of the queue length at i_k can be described as

$$\dot{q}_{i_k}(t) = \sum_{j \in \mathcal{C}_{i_k}} \left[r^{j,b}_{i_k}(t) - \rho^{j,s}_{i_k,i_+}(t)\right] \quad (2)$$

where i_+ denotes the node following node i on the path of the jth connection. Here, $r^{j,b}_{i_k}(t)$ is the rate of data received at i_k from the jth source at time t and $\rho^{j,s}_{i_k,i_+}(t)$ is the rate of data, belonging to the jth connection, sent from i_k to the next node on the path of connection j at time t. The round trip delay for the jth connection observed at i_k at time t, $\tau^j_{i_k}(t) := h^j_{i_k} + \delta^j_{i_k}(t)$ where $h^j_{i_k}$ is the nominal part and $\delta^j_{i_k}(t)$ is the time varying uncertainty, is defined as $\tau^j_{i_k}(t) = \tau^{j,b}_{i_k}(t) + \tau^{j,f}_{i_k}(t)$. Here, $\tau^{j,b}_{i_k}(t) := h^{j,b}_{i_k} + \delta^{j,b}_{i_k}(t)$ is the backward time delay from the controller at i_k to s(j), where $h^{j,b}_{i_k}$ is the time invariant known nominal backward delay and $\delta^{j,b}_{i_k}(t)$ is the time varying backward time delay uncertainty, while $\tau^{j,f}_{i_k}(t) := h^{j,f}_{i_k} + \delta^{j,f}_{i_k}(t)$ is the forward time delay from s(j) to i_k, where $h^{j,f}_{i_k}$ is the time invariant known nominal forward delay and $\delta^{j,f}_{i_k}(t)$ is the time varying forward time delay uncertainty. Moreover, $\tau_{i_k,i_+}(t)$ is the time delay between the kth port of node i and node i_+ and have the same form as the time delays between a source and a node. Then, the queue length at i_k can be rewritten as follows;

$$q_{i_k}(t) = \int_0^t \sum_{j \in \mathcal{C}_{i_k}} \left(1 - \dot{\delta}^{j,f}_{i_-,i_k}(\nu)\right) \rho^{j,s}_{i_-,i_k}\left(\nu - \tau^{j,f}_{i_-,i_k}(\nu)\right) d\nu - \int_0^t \sum_{j \in \mathcal{C}_{i_k}} \rho^{j,s}_{i_k,i_+}(\nu) d\nu + q_{i_k}(0), \quad (3)$$

where i_- denotes the node preceding node i on the path of the jth connection (for details refer to (Munyas-Elmas, 2003)). Here, it is assumed that $\frac{d}{dt}(t - \tau^{j,f}_{i_-,i_k}(t)) > 0$ is satisfied for $i = 1, \ldots, n$, $k = 1, \ldots, k_i$ and $j = 1, \ldots, l_{i_k}$, implying $\dot{\tau}^{j,f}_{i_-,i_k}(t) < 1$ or $\dot{\delta}^{j,f}_{i_-,i_k}(t) < 1$.

The rate of data sent from i_k to the next node, $\rho^s_{i_k,i_+}(t)$, depends on the value of the queue length at i_k;

$$\rho^s_{i_k,i_+}(t) = \begin{cases} c_{i_k}(t), & q_{i_k}(t) > 0 \\ min\left\{c_{i_k}(t); r^b_{i_k}(t)\right\}, & q_{i_k}(t) = 0 \end{cases} \quad (4)$$

where $c_{i_k}(t)$ is the capacity of the kth output port of the ith node at time t. Besides, the rate of that flow belonging to the jth connection is

$$\rho^{j,s}_{i_k,i_+}(t) = \begin{cases} \dfrac{q^j_{i_k}(t)}{q_{i_k}(t)} \rho^s_{i_k,i_+}(t), & q_{i_k}(t) > 0 \\ \dfrac{r^{j,b}_{i_k}(t)}{r^b_{i_k}(t)} \rho^s_{i_k,i_+}(t), & q_{i_k}(t) = 0 \end{cases} \quad (5)$$

where $r^b_{i_k}(t) = \sum_{j \in \mathcal{C}_{i_k}} r^{j,b}_{i_k}(t)$ and $\rho^s_{i_k,i_+}(t) = \sum_{j \in \mathcal{C}_{i_k}} \rho^{j,s}_{i_k,i_+}(t)$ are satisfied. It can be seen from (3) that the queue length at i_k depends on the rate of data sent from the preceding node. However, this rate depends on the value of the queue length formed at the output port of the preceding node, that is, whether the port is a bottleneck or not. Therefore, when the queue length expression for an output port is written, it is seen that all the previous output ports should be checked to see whether those ports are bottlenecks or not. This MIMO structure, which originates from the basics of ABR control framework, is too complex to define an $\mathcal{H}^\infty$ optimization problem. Here, if the output ports which are bottlenecked have already been known, the source would be allowed to send data with the rate calculated by the controller implemented at the most bottlenecked port. Therefore, each output port on the path of a connection is let to calculate a rate command for the source *assuming* that the most bottlenecked port is itself. Then, the source is allowed to send data with the rate equal to the minimum of all calculated rate commands and the demand of the source. This strategy complies with the ABR control framework (see Section 4). In this strategy, the assumed queue length formed at port i_k can be written as follows,

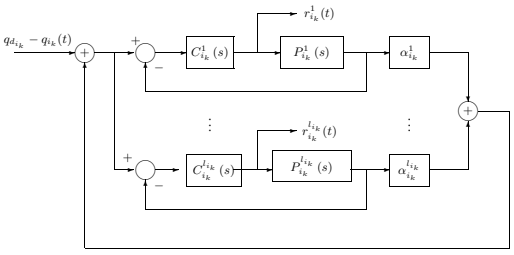

Fig. 1. The structure of the controller $K_{i_k}(s)$, (Quet *et al.*, 2002)

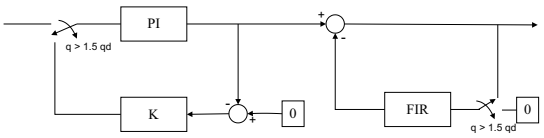

Fig. 2. Reset mechanism

$$q_{i_k}(t) = \int_0^t \sum_{j \in \mathcal{C}_{i_k}} \left(1 - \dot{\delta}^{j,f}_{i_k}(t)\right) r^j_{i_k}(\nu - \tau^j_{i_k}(\nu)) d\nu - \int_0^t \rho^s_{i_k,i_+}(\nu) d\nu + q_{i_k}(0) \quad (6)$$

From this expression, it is seen that the multi–bottleneck case in our problem is reduced to the single bottleneck case. Here, if the solution given in (Quet *et al.*, 2002) is applied for each port on the path of each connection, the $\mathcal{H}^\infty$ controllers that will be implemented at these ports can be found. The structure of the controller at each port i_k is given in Fig. 1. Here, $P^j_{i,k}(s) := \frac{1}{\alpha^j_{i,k} s} e^{-h^j_{i,k} s}$ and

$$C^j_{i,k}(s) = \frac{n \alpha^j_{i,k} \gamma^j_{i,k}}{2\sqrt{2\sum_{j=1}^n \left(\delta^{j,+}_{i,k}\right)^2}} \left(\frac{s h^j_{i,k} - k^j_{i,k}}{s h^j_{i,k}}\right) \frac{1}{1 + F^j_{i,k}(s h^j_{i,k})} \quad (7)$$

where $F^j_{i,k}(s h^j_{i,k})$ is a finite impulse response (FIR) filter of duration $h^j_{i,k}$ and $k^j_{i,k}$ and $\gamma^j_{i,k}$ are constants to be calculated, which depend on $\delta^{j,+}_{i_k}$, $\beta^j_{i_k}$ and $\beta^{j,f}_{i_k}$, the assumed bounds on $\left|\delta^j_{i_k}(t)\right|$, $\left|\dot{\delta}^j_{i_k}(t)\right|$ and $\left|\dot{\delta}^{j,f}_{i_k}(t)\right|$, respectively. Besides, $\alpha^j_{i_k}$, $j = 1, \ldots, l_{i_k}$, are the weighted fairness coefficients given for each port and satisfy $\sum_{j=1}^{l_{i_k}} \alpha^j_{i_k} = 1$. These coefficients allow the controllers implemented at the ports to allocate different bandwidths for each connection at steady-state (see (Quet *et al.*, 2002)).

The reset mechanism shown in Fig. 2 is used to handle the transition from the bottlenecked case to the non-bottleneck case. When high congestion occurs at a port, the queue length of this port grows up to values much larger than the desired

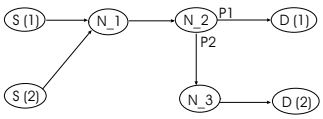

Fig. 3. Example network model

value of the queue length. Meanwhile, to force the queue length to its desired value, the command rate calculated by the controller decreases to very high negative values. Then, the port becomes non-bottlenecked. When the queue length begins to grow again, it is observed that the controller do not calculate the corresponding positive valued rates. This situation occurs because it takes a long time for the controller to recover from those high negative values. To avoid this situation, the controller is resetted to zero when the queue length becomes greater than 1.5 times the corresponding desired value. To solve the problems observed at the transition from the non-bottleneck case to the bottlenecked case, a reset mechanism was proposed in (Tiftikçi, 2003). However, with the reset mechanism presented in the present work, the problems considered in that work are also avoided. Considering (7), it is seen that the controller is formed by cascading a proportional plus integral (PI) controller with a feedback loop including an FIR filter on the feedback path. When the condition $q_{i_k} > 1.5q_{d_{i_k}}$ is satisfied, the input to the FIR filter is switched to zero, which nullifies the filter state in finite-time. In this way, the FIR part does not have any effect on the output of the controller. Hence, the controller can be resetted to zero by making the PI part of the controller track a zero signal. To achieve this aim, the feedback path including a gain K is connected to the input of the PI part of the controller when $q_{i_k} > 1.5q_{d_{i_k}}$ is satisfied. The gain K should be chosen as positive to provide stability of the feedback system. Choosing K large will make the response faster, however, will increase the sensitivity to disturbances and measurement noises.

3. SIMULATION RESULTS

To observe the performance of the controllers, a number of simulations are realised for network models with different number of source–destination pairs, switches, various parameter values and network conditions. However, due to space limitations only two cases are included here. Further cases may be found in (Munyas-Elmas, 2003). In both cases considered here, the network shown in Fig. 3 is considered. The desired queue lengths are 30 packets for all output ports. The paths followed by the connections are:

$p_1 = \{N_{1_1}, N_{2_1}, d_{1_1}\}$, $p_2 = \{N_{1_1}, N_{2_2}, N_{3_1}, d_{2_1}\}$

Here, d_{j_k} represents the kth port of the destination node of the jth source.

Table 1. Design parameters.

i_k, j	$h^j_{i_k}$	$\delta^{j,+}_{i_k}$	$\alpha^j_{i_k}$	$\beta^j_{i_k}$	$\beta^{j,f}_{i_k}$
$1_1, 1$	1	2	0.6	0.1	0.01
$1_1, 2$	2	3	0.4	0.2	0.02
$2_1, 1$	1	2	1	0.1	0.01
$2_2, 2$	2	3	1	0.2	0.02
$3_1, 2$	2	3	1	0.2	0.02
$d_{1_1}, 1$	1	2	1	0.1	0.01
$d_{2_1}, 2$	2	3	1	0.2	0.02

Table 2. Implementation parameters.

i_k, j	$h^{j,b}_{i_k}$	$\delta^{j,b}_{i_k}$	$h^{j,f}_{i_k}$	$\delta^{j,f}_{i_k}$
$1_1, 1$	0.9	$0.5\sin(\frac{2\pi}{50}t)$	0.1	$0.1\sin(\frac{\pi}{50}t)$
$1_1, 2$	1.85	$0.5\sin(\frac{2\pi}{50}t)$	0.15	$0.1\cos(\frac{\pi}{50}t)$
$2_1, 1$	0.8	$0.5\sin(\frac{2\pi}{50}t)$	0.2	$0.1\sin(\frac{\pi}{50}t)$
$2_2, 2$	1.75	$0.5\sin(\frac{2\pi}{50}t)$	0.25	$0.1\cos(\frac{\pi}{50}t)$
$3_1, 2$	1.65	$0.5\sin(\frac{2\pi}{50}t)$	0.35	$0.1\sin(\frac{\pi}{50}t)$
$d_{1_1}, 1$	0.65	$0.5\sin(\frac{2\pi}{50}t)$	0.35	$0.\sin(\frac{\pi}{50}t)$
$d_{2_1}, 2$	1.6	$0.5\sin(\frac{2\pi}{50}t)$	0.4	$0.1\sin(\frac{\pi}{50}t)$

Case 1: The values of the parameters used in the controller design are given in Table 1 while the implementation parameters are given in Table 2. The demands of the sources, $r^{0,1}(t)$ and $r^{0,2}(t)$ and the outgoing capacities $c_{1_1}(t)$, $c_{d_1}(t)$ and $c_{d_2}(t)$ are the same and equal to 100 p/s. Besides, $c_{2_1}(t) = 20$, $c_{2_2}(t) = 30$ and $c_{3_1}(t) = 60$ p/s. The results for this case are given in Fig. 4. As expected, congestion has occurred at both ports of the second node. The controllers at these ports forces the queue lengths to their desired values. Since all the capacities and demands remain the same for the rest of time, the queue lengths do not change.

Case 2: The design and implementation parameters used in this case are the same as the ones used in Case 1. Different from that case, $c_{2_1}(t) = 25$ p/s. and $c_{3_1}(t)$, $c_{d_1}(t)$ and $c_{1_1}(t)$ are as given in Fig. 5. When the results, given in Fig. 6, are considered, it is seen that with the decrement in $c_{d_1}(t)$ after 60 s., d_1 becomes the bottleneck after that moment and hence, q_{2_1} decreases to zero. It is observed that there still exists some changes in q_{2_1} corresponding to the changes in $q_{d_{1_1}}$. However, q_{2_1} becomes zero after a while since the incoming data rate to the port 2_1 becomes smaller than its capacity. Similarly, with the decrement in $c_{3_1}(t)$ after 75 s., the 3rd node becomes the bottleneck after that moment and hence, q_{2_2} decreases. Although q_{2_2} increases following the decrements in q_{3_1} after some time, it decreases to zero again since the incoming data rate to the port 2_2 becomes smaller than its capacity. Furthermore, a decrement occurs in $c_{1_1}(t)$ at 100 s. Since $c_{1_1}(t)$ is the smallest capacity from that moment on, the queue lengths of $q_{d_{1_1}}$ and q_{2_1} decrease to zero and q_{1_1} converges to its desired steady–state. It is observed that expected command rates reach expected steady–state values and make the sources share and use the port capacities accordingly.

From the results of both cases, it is seen that the reset mechanism work as expected and avoid the problems occurring at the transitions from the bottlenecked case to non-bottlenecked case. Examining the actual sending rates of the sources, one can observe that the actual sending rate of each source becomes equal to the rate calculated by the most congested port, with a delay equal to the backward delay from that port. Though the corresponding plots cannot be presented due to space limitations, the data sending rates from one port to another are bounded with the outgoing capacities of the corresponding ports, as expected.

4. IMPLEMENTATION ISSUES

The flow control presented in this work can be implemented by exactly following the ABR flow control framework. Each controller, implemented at an output port of a switch on the path of a connection, calculates a command rate for the source and compares it with the rate written in the Explicit Rate, ER, field of the RM cell. Then, the controller writes the smaller rate in the ER field. Meanwhile, the congestion indication, CI, bit in the RM cell is also checked. If $CI = 1$ in an incoming RM cell, this means that congestion occurs in one of the previous switches. In this case, the value of the CI bit is not changed and the RM cell is sent to the next switch. If $CI = 0$ and the port is itself congested, than the CI bit is set to 1 and then, the RM cell is sent to the next switch. As the last step the source compares the rate written in the ER field with its demand and adjusts its actual sending rate to the smaller one. Besides, the resulting actual sending rates of the sources are bounded, $MCR \leq ACR \leq PCR$ where MCR and PCR are the quality of service parameters which denote the minimum and peak cell rates, respectively, while ACR is the allowed cell rate.

5. CONCLUDING REMARKS

In this work, the $\mathcal{H}^\infty$ controller designed for a general communication network with single bottleneck in (Quet *et al.*, 2002) is implemented in ATM networks with multiple bottlenecks. The controllers are implemented at each output port of each switch. The implementation is carried out in consistence with the ABR flow control framework. Besides, a reset mechanism is used to avoid the controllers to calculate high negative valued rates at the transitions from the bottlenecked to the non-bottleneck case.

Simulation results show that the controllers force the queue lengths at the bottleneck ports to their desired values asymptotically at steady–state. Besides, the data sending rates of the sources are adjusted to the rates determined by the controller of the most bottlenecked port and these rates also satisfy the weighted fairness condition.

REFERENCES

Altman, E., T. Başar and R. Srikant (1998). Robust rate control for ABR sources. *In Proc. of the INFOCOM'98, San Fransisco, California, U.S.A.* pp. 166–173.

BenMohamed, L. and S. Meerkov (1993). Feedback control of congestion in store–and–forward datagram networks: The case of a single congested node. *IEEE / ACM Trans. on Networking* **1**, 693–708.

BenMohamed, L. and S. Meerkov (1997). Feedback control of congestion in packet switching networks: The case of multiple congested nodes. *International Journal on Communication Systems* **10**, 227–246.

Biberoviç, E. (2001). *Flow control in high–speed data communication networks.* M.S. Thesis. Anadolu University, Eskişehir, Turkey. (In Turkish).

Biberoviç, E., A. İftar and H. Özbay (2001). A solution to the robust flow control problem for networks with multiple bottlenecks. *In Proc. of the 40th IEEE Conference on Decision and Control, Orlando, FL, U.S.A.* pp. 2303–2308.

Bonomi, F. and K. W. Fendick (1995). The rate–based flow control framework for the available bit rate ATM service. *IEEE Network Magazine* **25**, 24–39.

Gomez-Stern, F., J. M. Fornes and F. R. Rubio (2002). Dead–time compensation for ABR traffic control over ATM networks. *Control Engineering Practice* **10**, 481–491.

Mascolo, S. (1997). Smith's principle for congestion control in high–speed ATM networks. *In Proc. of the IEEE Conference on Decision and Control, San Diego, California* pp. 4595–4600.

Mascolo, S. and D. Cavendish (1996). ATM rate based congestion control using a smith predictor: An EPRCA implementation. *In Proc. of GLOBECOM'96, England* pp. 569–576.

Munyas-Elmas, İ. (2003). *Developing flow control algorithms for ATM networks with multiple bottlenecks.* M.S. Thesis. Anadolu University, Eskişehir, Turkey. (In Turkish).

Munyas, İ., Ö. Yelbaşı and A. İftar (2003). Decentralized robust flow controller design for networks with multiple bottlenecks. *In Proc. of the European Control Conference, Cambridge, U.K.*

Özbay, H., S. Kalyanaraman and A. İftar (1998). On rate–based congestion control in high–speed networks: Design of an $\mathcal{H}_\infty$ based flow controller for single bottleneck. *In Proc. of the American Control Conference, Philadelphia, PA, U.S.A.* pp. 2376–2380.

Quet, P.-F., B. Ataşlar, A. İftar, H. Özbay, S. Kalyanaraman and T. Kang (2002). Rate–based flow controllers for communication networks in the presence of uncertain time–varying multiple time delays. *Automatica* **38**, 917–928.

Tiftikçi, A. (2003). *Improvement of the controllers used in multiple bottleneck ATM networks*. Graduation Project, Department of Electrical and Electronics Engineering, Anadolu University, Eskişehir, Turkey.

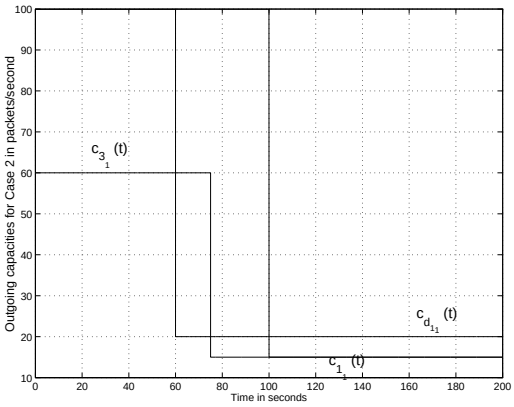

Fig. 5. Outgoing capacities for Case 2.

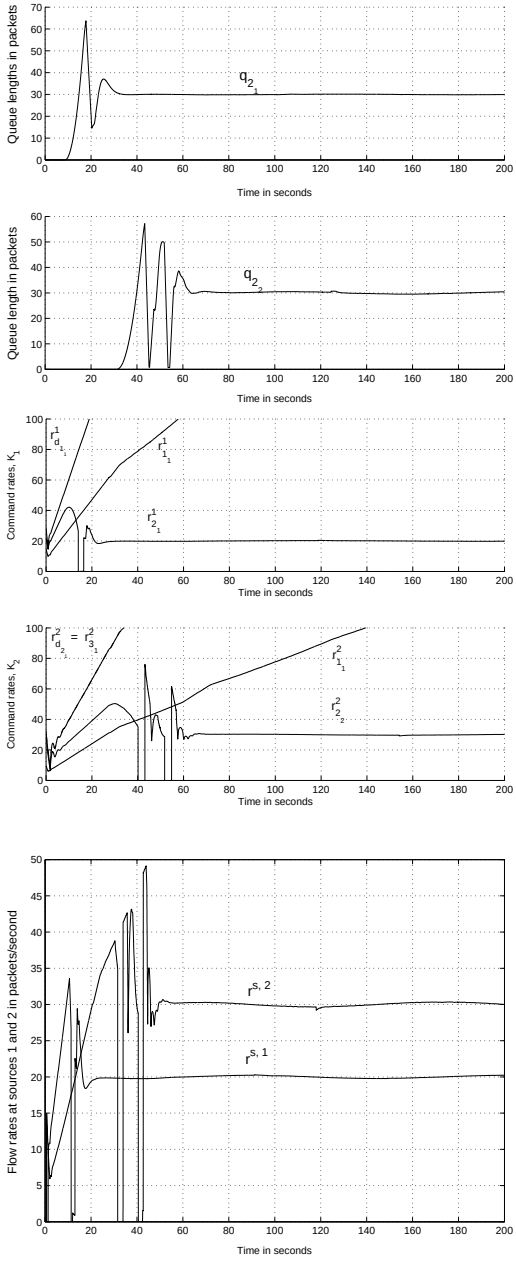

Fig. 4. Results for Case 1.

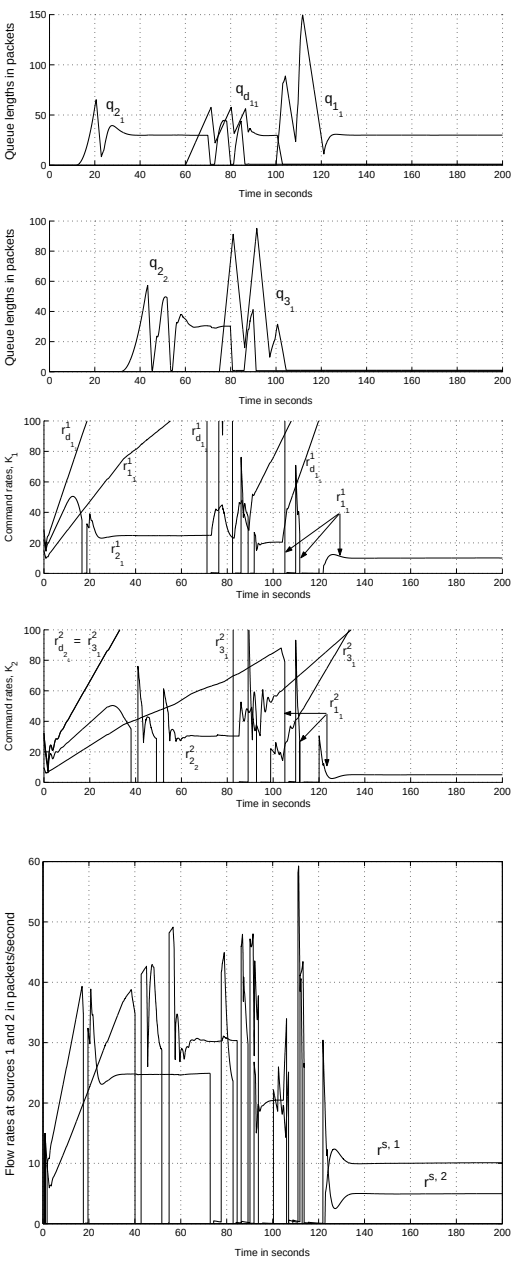

Fig. 6. Results for Case 2.