Effairness: Dealing with Time in Congestion Control Evaluation

Sergey Gorinsky
Applied Research Laboratory
Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130, USA
gorinsky@arl.wustl.edu

Harrick Vin
Laboratory for Advanced Systems Research
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712, USA
vin@cs.utexas.edu

Abstract

Congestion control algorithms are traditionally evaluated in contrast to ideal capacity allocations that specify instantaneous efficient fair rates for application sessions but ignore time. While the latter is tenable for local networks and networks where all application sessions last long, instantaneous ideal allocations are inadequate standards for congestion control evaluation in dynamic wide-area networks. In this paper, we propose an alternative ideal of an effair allocation that explicitly accounts for unavoidable propagation delay. We develop an algorithm for computing the effair allocation and present a metric of effairness that quantifies how close the actual network services are to the effair allocation on the receiver, session, and network levels.

1. Evaluation of congestion control

While congestion control is a vibrant research area that has seen many hundred proposals, the topic of congestion control evaluation is recently attracting significant attention, e.g., within the Transport Modeling Research Group [3]. Any evaluation is inherently done in contrast to an ideal. In congestion control where the primary objective is to allocate network capacity to application sessions efficiently and fairly, standard ideal allocations ignore time, represent the network and each application session as a capacitated graph and capacitated flow through this graph respectively, and specify instantaneous efficient fair rates for all flows. Maxmin fairness [1], proportional fairness [10], and other instantaneous ideal allocations are easily generalized to include multicast flows [11].

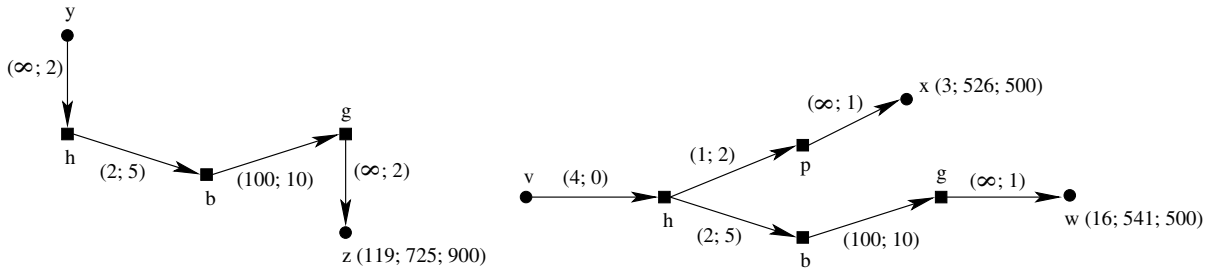
Ignoring time is tenable for local networks and networks where all application sessions last long. However, the Internet has emerged into a global high-capacity network where an application session might face large end-to-end propagation delay (e.g., few hundred milliseconds) which is comparable to or even higher than the time spent on transmitting the session data into the bottleneck link. Consequently, the Internet service provided to session s might depend on sessions that have finished transmitting before session s starts, and the current transmission rate of session s might affect the Internet service for sessions that will start after session s stops transmitting. Instantaneous ideal allocations are inadequate standards for congestion control evaluation in such dynamic wide-area network scenarios.

In this paper, we propose an effair allocation, an alternative ideal for capacity assignment in dynamic wide-area networks where propagation delay is non-negligible. First, Section 2 defines our model for the studied problem. In particular, we introduce streams to represent actual services provided by the network to all application sessions, including multicast and short-lived sessions. The stream representation captures the capacity and delay of network paths as well as the communication needs and capabilities of application end points. Section 3 presents the notion of a universal timescale. Using this notion, Section 4 develops an algorithm for computing the effair allocation. Section 5 proposes a hierarchical metric of effairness that quantifies how close the actual network services are to the ideal effair allocation on the receiver, session, and network levels. Section 6 concludes the paper with a summary.

2. Model

We model the **network** as a digraph. Square nodes depict hosts and routers. Directed edges denote communication links. Each edge is annotated with (c; d)

0-7695-3094-X/08 $25.00 © 2008 IEEE
DOI 10.1109/ICNS.2008.45



(a) unicast session υ (b) multicast session μ

Figure 1. Stream representation of network services provided to application sessions.

where c and d refer to the capacity and propagation delay of the link and have default measurement units of Mbps and ms respectively.

Each end point of an **application session** is hosted by a node and called either a **sender** or a **receiver** depending on whether the session transmits data from or to the end point. While unicast is a special case of multicast, we consider sessions with one sender and at least one receiver. The sender has immediate access to data transmitted during the session. Applications with multiple sending end points or intermittently available data are modeled as multiple sessions.

The **actual network service** provided to an application session is represented by a **stream**, our generalization of a flow [1]. The stream depicts the end points of the session as round nodes. A directed edge connects each end point with the hosting node. The edge directions are from the sender and to the receivers. To represent needs and capabilities of the sender and each receiver, we annotate the respective incident edge with $(c; d)$ where c denotes the rate at which the end point is willing and able to communicate with the hosting node, and d is the delay of such communication. We use $c = \infty$ to denote that the end point is able and willing to communicate with the hosting node at a higher rate than the network capacity. The stream also contains the multicast routing tree provided to the session by the network. Hence, the stream is a tree of directed paths from the sender to all the receivers of the session. Receiver k joins the session at time a_k, leaves the session at time f_k after receiving m_k of data, and is annotated with three-tuple $(a_k; f_k; m_k)$. A receiver may join late and leave early, and the value of m_k might be different for different k. Whereas our default measurement units for capacity and time are Mbps and ms respectively, we measure m_k in units of 10^3 bits (125 bytes) unless we explicitly specify otherwise.

The following is the first in a series of examples that illustrate our ideas throughout the paper.

Example 1 *Consider unicast session υ with sender* v *at host* h*, two-link network path through router* b*, and receiver* z *at host* g*. Link* hb *has capacity 2 Mbps and propagation delay 5 ms. Link* bg *has capacity 100 Mbps and propagation delay 10 ms. Both sender and receiver communicate with their respective hosts with delay 2 ms and at higher rates than the network capacity. The receiver starts receiving data at time $a_z = 119$ ms and leaves the session at time $f_z = 725$ ms after receiving $m_z = 9 \cdot 10^5$ bits of data. Figure 1a shows the stream representation for the network service provided to the unicast session.*

Now, consider multicast session μ with sender v *at host* h*, receiver* w *at host* g*, and receiver* x *at host* p*. The multicast routing tree for the session consists of two disjoint paths* hbg *and* hp*. Link* hp *has capacity 1 Mbps and propagation delay 2 ms. The sender transmits data to its hosting node at rate 4 Mbps and without delay. Both receivers receive data from their respective hosts with delay 1 ms and can sustain higher rates than the network capacity. Receiver* x *joins the session at time $a_x = 3$ ms and leaves at time $f_x = 526$ ms after receiving $m_x = 5 \cdot 10^5$ bits of data. Receiver* w *joins at time $a_w = 16$ ms and leaves at time $f_w = 541$ ms after receiving $m_w = 5 \cdot 10^5$ bits of data. Figure 1b depicts the stream representation for multicast session μ.*

If a link of the network does not serve any session, the superposition of all streams does not contain the edge corresponding to the link. In general, the stream superposition is a set of edge-disconnected components. Since network capacity allocation is independent for any two sessions represented by streams from different edge-disconnected components, our subsequent investigation considers each edge-disconnected component of the stream superposition independently. In particular, we consider only components that are trees. This constraint is not overly restrictive and accommodates most common settings for congestion control evaluation, e.g.,

networks with dumbbell [12], multiple-bottleneck [9], and multicast tree [2] topologies.

In the **ideal network service** for a session, application data propagate through the network without control overhead or extra delay at traversed nodes. Hence, our model abstracts away handshakes, packets, headers, queuing and transmission delays at routers, and other features particular to common networking technologies. Clocks at all nodes are synchronized. The clock synchronization is a tenable assumption for simulators, emulators, and other carefully managed environments for assessment of distributed protocols. Because evaluated congestion control designs have no leverage over applications or routing, receivers k join at the same times a_k and the session transmits along the same multicast tree as in the actual network service.

While the stream superposition provides a topological basis for the ideal network service, another foundation for the ideal capacity allocation in the temporal domain is the underlying notion of an instantaneous ideal allocation, such as maxmin fairness [1] or proportional fairness [10]. The temporal ideal allocation specifies a fair efficient rate $r_{\sigma l}(t)$ of each stream σ in its every node l at any instant t. We refer to this allocation as an **effair allocation** and to rates $r_{\sigma l}(t)$ as **effair rates**. The effair rates characterize the effair allocation completely. In particular, effair rate $r_{\sigma k}(t)$ of stream σ in receiver k determines the earliest time q_k when receiver k has received all m_k of data under the effair allocation:

$$\int_{a_k}^{q_k} r_{\sigma k}(t)\, dt = m_k. \tag{1}$$

We refer to q_k as the **effair finish time** of receiver k.

3. Universal timescales

Before presenting how to compute the effair allocation for an edge-disconnected component of the stream superposition, we first address the fundamental challenge of time differences in distributed settings. Due to unavoidable propagation delays, the effair rate of a receiver might depend on future arrivals in other streams. To quantify the effect of propagation delay, we introduce the following concept of a time shift:

Definition 1 *The* **time shift** *from node i to node j is:*

$$s_{ij} = \sum_{e \in p_{ij}} (-1)^{u_e} d_e \tag{2}$$

where p_{ij} is the set of edges on path ij, d_e refers to the propagation delay of edge e, $u_e = 0$ for the edges directed from i to j, and $u_e = 1$ for the edges directed from j to i.

Due to our focus on stream superposition components with tree topologies, the time shift from node i to node j has a unique value. While the delay of an edge on path ij is added to or subtracted from the time shift depending on the edge direction, the reverse traversal of the path changes the sign but not the absolute value of the time shift, i.e., $s_{ji} = -s_{ij}$.

The key idea of our solution is to create a universal timescale:

Definition 2 *A* **universal timescale** *for a stream superposition component is a timescale such that a change of capacity allocation anywhere in the component affects the whole component instantaneously.*

The notion of universal timescales allow us to distinguish a special node called an anchor:

Definition 3 *The* **anchor** *of a stream superposition component is a receiver that joins the earliest on a universal timescale, with ties broken in favor of a receiver with the earliest arrival on the original timescale.*

If receiver n is the anchor of a stream superposition component, then $a_k \geq a_n + s_{nk}$ for each receiver k in the component. While there exist an infinite number of universal timescales, the concepts of time shift and anchor form a basis for defining a unique universal timescale:

Definition 4 *The* **canonical universal timescale** *for a stream superposition component is the universal timescale that denotes every instant in node k as*

$$\tau_k = t - s_{nk} \tag{3}$$

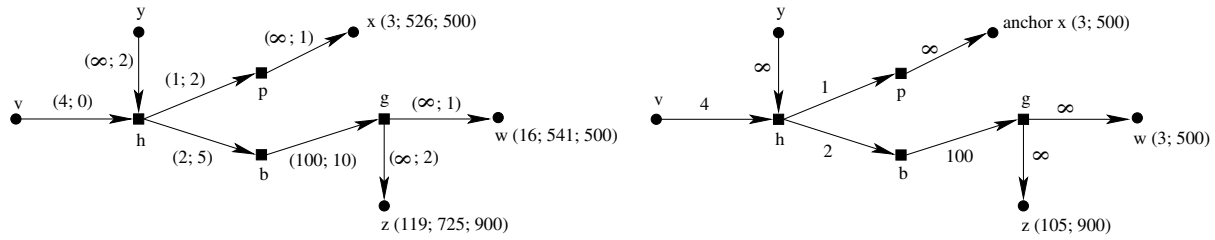
where t represents the same instant on the original timescale, and s_{nk} is the time shift from anchor n of the component to node k.

Note that time in the anchor is the same on both original and canonical universal timescales. With receiver n being the anchor of a stream superposition component, we use α_k to represent the arrival time of receiver k on the canonical universal timescale:

$$\alpha_k = a_k - s_{nk}. \tag{4}$$

Example 2 *Figure 2a shows the stream superposition for the network services provided to the unicast and multicast sessions in Example 1. The superposition consists of a single edge-disconnected component.*

By Definition 1, the time shift from receiver x *to receiver* z *is* $s_{xz} = -1 - 2 + 5 + 10 + 2 = 14$ *ms*



(a) stream superposition

(b) effair graph

Figure 2. From the stream superposition to the effair graph.

because path xz *goes against edges* px*,* hp *and along edges* hb*,* bg*,* gz*. The time shift from* z *to* x *equals* $s_{zx} = -14$ *ms. The time shift from* x *to receiver* w *is equal to* $s_{xw} = 13$ *ms. The time shift from sender* v *to sender* y *is* $s_{vy} = 0 - 2 = -2$ *ms.*

On a universal timescale, receivers x *and* w *join at the same time and earlier than receiver* z*. Since* x *joins earlier than* w *on the original timescale, receiver* x *is the anchor of the stream superposition. Original time* t *becomes time* $\tau_x = t$*,* $\tau_w = t - 13$ *ms, and* $\tau_z = t - 14$ *ms respectively in receivers* x*,* w*, and* z *on the canonical universal timescale. Hence, receivers* x*,* w*, and* z *join respectively at time* $\alpha_x = 3$ *ms,* $\alpha_w = 3$ *ms, and* $\alpha_z = 105$ *ms on the canonical universal timescale.*

4. Effair allocation

The canonical universal timescale is a theoretical construct that enables us to reduce the problem of computing the effair allocation to the classical problem of finding the instantaneous ideal allocation. Since a change in capacity allocation anywhere in the stream superposition component affects the whole component instantaneously on the universal timescale, we move the superposition component from the original to canonical universal timescale, determine the instantaneous ideal allocation for each moment on the canonical universal timescale, and then change clocks in all nodes back to the original timescale to obtain effair rates $r_{\sigma l}(t)$ of each stream σ in all its nodes l at any original time t.

The rest of the section elaborates this algorithm for computing the effair allocation. The move from the original to canonical universal timescale transforms the superposition component into an **effair graph** that annotates each edge with capacity c instead of $(c; d)$, each receiver with $(\alpha_k; m_k)$ instead of $(a_k; f_k; m_k)$, and explicitly identifies the anchor of the component. The absence of propagation delay d in the edge annotations reflects the instantaneity of any impact made by an allocation change on the other nodes throughout the effair graph.

Let ϕ_k denote the effair finish time of receiver k on the canonical universal timescale:

$$\phi_k = q_k - s_{nk} \tag{5}$$

where s_{nk} is the time shift from anchor n of the component to receiver k of stream σ. If canonical universal time τ is earlier than arrival time α_k or is at least effair finish time ϕ_k, we describe stream σ as being **inactive** in receiver k at time τ. If stream σ is inactive in receiver k at time τ, effair rate $\rho_{\sigma k}(\tau)$ of stream σ in receiver k at this canonical universal time τ equals zero:

$$\rho_{\sigma k}(\tau) = 0 \;\Leftrightarrow\; \tau < \alpha_k \text{ or } \tau \geq \phi_k. \tag{6}$$

For all non-leaf nodes of a stream, we expand the definition of inactivity recursively. If stream σ is inactive in all children of non-leaf node l at time τ, then stream σ is inactive in node l as well, and effair rate $\rho_{\sigma l}(\tau)$ of stream σ in node l at canonical universal time τ is also zero:

$$\rho_{\sigma l}(\tau) = 0 \;\Leftrightarrow\; \forall\, k \in C_{\sigma l}\;\; \rho_{\sigma k}(\tau) = 0 \tag{7}$$

where $C_{\sigma l}$ is the set of all children that node l has in stream σ.

Before the arrival of anchor n, all streams are inactive throughout the component, i.e., $\rho_{\sigma l}(\tau) = 0$ for each stream σ in all its nodes l for $\tau < \alpha_n$. Hence, we start computing non-zero effair rates at $\tau = \alpha_n$ and then proceed forward in canonical universal time.

Let $S(\tau)$ be the set of all receivers where the stream of the receiver is active at the currently considered time τ. We refer to $S(\tau)$ as an **active set** and construct a graph $G(\tau)$ by removing from the effair graph the edges incident to every node where all streams are inactive at time τ. Note that all directed paths in graph $G(\tau)$ terminate only in receivers belonging to $S(\tau)$. We provide $G(\tau)$ as an input to the algorithm for computing the instantaneous ideal allocation. The algorithm outputs ef-

1. Represent the actual network services provided to the application sessions as streams; construct the stream superposition; do the following steps for each edge-disconnected component of the stream superposition.
2. Find anchor n; construct the effair graph; switch to the canonical universal timescale; set $\rho_{\sigma l}(\tau) \leftarrow 0$ for each stream σ in its every node l for $\tau < \alpha_n$; set $\tau \leftarrow \alpha_n$.
3. Repeat the following until effair finish times ϕ_k are known for each receiver k: if stream σ is inactive in node l at time τ, set $\rho_{\sigma l}(\tau) \leftarrow 0$; construct graph $G(\tau)$; execute the algorithm for the instantaneous ideal allocation on input $G(\tau)$ to compute $\rho_{\sigma l}(\tau)$ for each stream σ in its every node l where stream σ is active at time τ; find next time δ when the active set changes; set $\rho_{\sigma l}(\omega) \leftarrow \rho_{\sigma l}(\tau)$ for each stream σ in its every node l for $\omega \in [\tau; \delta)$; set $\tau \leftarrow \delta$.
4. Set $\rho_{\sigma l}(\tau) \leftarrow 0$ for each stream σ in its every node l for all subsequent τ.
5. Return to the original timescale; set $r_{\sigma l}(t) \leftarrow \rho_{\sigma l}(t - s_{nl})$.

Figure 3. Summary of the algorithm for computing the effair allocation.

fair rates $\rho_{\sigma l}(\tau)$ for each stream σ in its every node l where stream σ is active at time τ.

The effair rates computed for time τ remain valid until the active set changes at time δ due to a new receiver arrival or because some receiver k from $S(\tau)$ has received all m_k of data. In the latter scenario, time δ becomes effair finish time ϕ_k of receiver k on the canonical universal timescale:

$$\int_{\alpha_k}^{\phi_k} \rho_{\sigma k}(\tau)\, d\tau = m_k. \tag{8}$$

Irrespective of which event changes the active set at time δ, the interval between times τ and δ does not see changes in the effair rate of any stream in any node:

$$\forall\, \omega \in [\tau; \delta)\ \ \rho_{\sigma l}(\omega) = \rho_{\sigma l}(\tau). \tag{9}$$

Then, we advance the currently considered time τ to δ, reconstruct graph $G(\tau)$, and repeat the rest of the above procedure until effair finish times are determined for all receivers in the stream superposition. At subsequent times τ, the active set is empty, and effair rate $\rho_{\sigma l}(\tau)$ for each stream σ in all its nodes l remains zero.

Finally, we move the completely computed effair allocation from the canonical universal to original timescale. At any original time t, the resulting effair rates $r_{\sigma l}(t)$ of streams σ in their nodes l are:

$$r_{\sigma l}(t) = \rho_{\sigma l}(t - s_{nl}) \tag{10}$$

and effair finish times q_k of receivers k are:

$$q_k = \phi_k + s_{nk} \tag{11}$$

where s_{nl} and s_{nk} denote the time shift from anchor n to node l and receiver k respectively. Figure 3 summarizes our algorithm for computing the effair allocation.

Example 3 *Let us return to unicast session υ and multicast session μ from Examples 1 and 2. Figure 2b presents the effair graph derived from the stream superposition shown in Figure 2a. With maxmin fairness chosen as the underlying instantaneous ideal allocation, we execute the algorithm summarized in Figure 3 to find the following effair rates $\rho(\tau) = (\rho_{\upsilon z}(\tau), \rho_{\mu x}(\tau), \rho_{\mu w}(\tau))$ on the canonical universal timescale:*

$$\rho(\tau) = \begin{cases} (0,0,0) & \textit{for } \tau < 3, \\ (0,1,2) & \textit{for } \tau \in [3; 105), \\ (1,1,1) & \textit{for } \tau \in [105; 401), \\ (2,1,0) & \textit{for } \tau \in [401; 503), \\ (2,0,0) & \textit{for } \tau \in [503; 703), \\ (0,0,0) & \textit{for } \tau \geq 703. \end{cases}$$

Returning to the original timescale, we determine the following effair rates of streams υ and μ in their receivers:

$$r_{\upsilon z}(t) = \begin{cases} 0 & \textit{for } t < 119, \\ 1 & \textit{for } t \in [119; 415), \\ 2 & \textit{for } t \in [415; 717), \\ 0 & \textit{for } t \geq 717, \end{cases}$$

$$r_{\mu x}(t) = \begin{cases} 0 & \textit{for } t < 3, \\ 1 & \textit{for } t \in [3; 503), \\ 0 & \textit{for } t \geq 503, \end{cases}$$

$$r_{\mu w}(t) = \begin{cases} 0 & \textit{for } t < 16, \\ 2 & \textit{for } t \in [16; 118), \\ 1 & \textit{for } t \in [118; 414), \\ 0 & \textit{for } t \geq 414. \end{cases}$$

Respective effair finish times of receivers z, x, *and* w *are $q_z = 717$ ms, $q_x = 503$ ms, and $q_w = 414$ ms.*

5. Effairness index

To quantify how close the actual network services are to the ideal effair allocation, we propose the following metric of effairness:

Definition 5 *Effairness for receiver k, session s, and the whole network is respectively:*

$$e_k = \frac{q_k - a_k}{f_k - a_k}, \quad e^s = \frac{\sum_{k \in S} e_k}{|S|}, \quad E = \frac{\sum_{s \in N} e^s}{|N|} \qquad (12)$$

where a_k, f_k, and q_k denote respectively the arrival time, finish time, and effair finish time of receiver k, S is the set of all receivers in session s, and N refers to the set of all sessions in the network.

The hierarchical structure of the effairness index satisfies the common need to evaluate a congestion control algorithm from both application and network perspectives. While effairness of 1 represents the service identical to the effair ideal, values between 0 and 1 correspond to suboptimal service. According to recent studies, awareness of session sizes enables fair efficient algorithms that consistently provide lower delay than under an instantaneous ideal allocation [4, 5, 6, 7]. Such fair efficient algorithms attain effairness of greater than 1 on all the levels: for the receivers, sessions, and entire network.

While the following example is the last one in the paper, the longer preliminary version of this work uses effairness to evaluate file transfers over Transmission Control Protocol [8].

Example 4 *Effairness for receiver* z*,* x*, and* w *in our Examples 1 through 3 is $e_z = 0.987$, $e_x = 0.956$, and $e_w = 0.758$ respectively. Effairness for session υ and μ is $e^\upsilon = 0.987$ and $e^\mu = 0.857$ respectively. Effairness for the whole network is $E = 0.922$. Since multicast session μ transmits data at a single rate constrained by the 1-Mbps capacity of link* hp*, receiver* w *receives its data later than feasible with multiple-rate transmission. This suboptimal performance is reflected in the low value of e_w, which also reduces effairness for session μ and the whole network.*

6. Conclusion

In this paper, we studied the problem of congestion control evaluation in dynamic wide-area networks. After arguing that traditional instantaneous ideal allocations are inadequate standards for this task, we proposed an alternative ideal of an effair allocation that explicitly accounts for unavoidable propagation delay. Then, we designed an algorithm for computing the effair allocation and presented a hierarchical metric of effairness that quantifies how close the actual network services are to the effair allocation on the receiver, session, and network levels.

References

[1] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, 1987.

[2] J. Byers, G. Horn, M. Luby, M. Mitzenmacher, and W. Shaver. FLID-DL: Congestion Control for Layered Multicast. *IEEE Journal on Selected Areas in Communications*, 20(8):1558 – 1570, October 2002.

[3] S. Floyd. Metrics for the Evaluation of Congestion Control Mechanisms. Internet Draft draft-irtf-tmrg-metrics-11, work in progress, *http://www.icir.org/tmrg/*, October 2007.

[4] E. J. Friedman and S. G. Henderson. Fairness and Efficiency in Web Server Protocols. In *Proceedings ACM SIGMETRICS*, June 2003.

[5] S. Gorinsky, E. J. Friedman, S. Henderson, and C. Jechlitschek. Efficient Fair Algorithms for Message Communication. Technical Report WUCSE-2007-54, *www.arl.wustl.edu/~gorinsky/pdf/WUCSE-TR-2007-54.pdf*, Department of Computer Science and Engineering, Washington University in St. Louis, December 2007.

[6] S. Gorinsky and C. Jechlitschek. Fair Efficiency, or Low Average Delay without Starvation. In *Proceedings International Conference on Computer Communications and Networks (ICCCN)*, August 2007.

[7] S. Gorinsky and N. S. V. Rao. Dedicated Channels as an Optimal Network Support for Effective Transfer of Massive Data. In *Proceedings IEEE INFOCOM*, April 2006.

[8] S. Gorinsky and H. Vin. Effairness: A Metric for Congestion Control Evaluation in Dynamic Networks. Technical Report TR2001-31, *www.arl.wustl.edu/~gorinsky/pdf/TR2001-31.pdf*, Department of Computer Sciences, University of Texas at Austin, August 2001.

[9] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, and B. Vandalore. The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks. *IEEE/ACM Transactions on Networking*, 8(1):87–98, February 2000.

[10] F. Kelly. Charging and Rate Control for Elastic Traffic. *European Transactions on Telecommunications*, 8(1):33–37, January 1997.

[11] S. Sarkar and L. Tassiulas. Back Pressure Based Multicast Scheduling for Fair Bandwidth Allocation. In *Proceedings IEEE INFOCOM*, April 2001.

[12] J. Widmer and M. Handley. Extending Equation-Based Congestion Control to Multicast Applications. In *Proceedings ACM SIGCOMM*, August 2001.