Using Average Attenuation Factor to Locate the Most Congested Path for Multicast Congestion Control

Jiang Li, Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute, 110 Eighth Street, Troy, NY 12180
Office: 518-276-8979
Email: {lij6@cs, shivkuma@ecse}.rpi.edu

Abstract

A typical method to perform single-rate multicast congestion control is for the source to adapt the sending rate to the most congested path. In previous research, TCP throughput formula [4] [6] is used to estimate the expected throughput of all paths for locating the most congested one. In order to measure RTT used in the formula, the source needs to *exchange* packets with every receiver. In sequence, there can be feedback implosion. In this article, we try to solve this problem by proposing a new method to locate the most congested path, where the source only needs to *send* packets. In this new method, the *average attenuation factor (AAF)* (of throughput) is measured for each path in the multicast tree, and the path with the highest AAF is considered as the most congested. AAF describes how much and how often the input of a path is attenuated on its route. Its formal definition is presented as a part of this paper. By simulations, we verify the properties of AAF and show the effectiveness of our proposed method as well. Using AAF to locate the most congested path can be used as a building block in multicast congestion control. Once the path is located, the source can adjust its sending rate using different policies based on the feedback from the receiver behind this path.

Keywords

Multicast, congestion control, attenuation factor

1 Introduction

IP multicast is efficient for transmitting bulk data to multiple receivers. Multicast congestion control is divided into two categories, single-rate and multi-rate. In single-rate schemes, the source controls the data transmission rate and all receivers receive data at the same rate. In multi-rate schemes (a.k.a layered multicast congestion control), receivers join just enough layers in the form of multicast groups to retrieve data as fast as they can. While single-rate schemes are simpler, the multi-rate schemes can satisfy the needs of heterogeneous receivers at the cost of more complexity. At the same time, single-rate congestion control mechanisms can be used to control the sending rate of each layer in a multi-rate framework to provide smooth rate adaptation [3].

In single-rate multicast congestion control schemes, the source must adapts the sending rate to the most congested path, otherwise that path will be consistently congested. Therefore, it is critical to locate the most congested path. In previous research, the typical way (e.g. in PGMCC [7] and TFMCC [8]) is to measure the packet loss rate and RTT of each path between the source and receivers, and then calculate the expected throughput for each path with the average TCP throughput formula [4] [6]. The path with the lowest expected throughput is considered as the most congested. Since the source needs to *exchange* packets with every receiver to measure the RTTs on different paths in this method, there is a risk of feedback implosion when the receiver set is large.

To solve this problem, we propose a different method where the source only needs to *send* packets to (instead of *exchange* packets with) receivers. This method uses a new metric, called *average attenuation factor (AAF)* (of throughput), which indicates how much and how often the input of a path is attenuated on its route. As we will show in later sections, AAF grows with the degree of congestion on a path, where the degree of congestion are decided by bottleneck bandwidths, total input of the bottlenecks, RTT of crossing flows (which decides how fast their sending rates increase), and bottleneck numbers. We will formally define AAF in Section 2.

To use AAF, the source sends common input to different paths. The AAF of each path is measured by the receivers at the output end of the path. The highest AAF indicates the most congested path. After locating the congested path, the source can adopt different policies to adjust the sending rate based on the feedback from the receiver behind this path. Therefore, using AAF to locate the most congested path can be a build block of multicast congestion control algorithms.

In the following two sections, we define AAF and qualitatively study its properties. After that, we verify its properties with *ns*-2[1] simulations. Finally, we incorporate our proposed method into a simple single-rate multicast congestion control scheme, and demonstrate the effectiveness of the method by simulations.

2 Definition and Usage of Average Attenuation Factor

In this section, we are going to define *average attenuation factor* (AAF) (of throughput). We propose to use AAF to collect information about the bottlenecks[1] on a path so as to measure the degree of congestion on that path. We will refer to such a path as the *relevant path.* AAF is always measured by checking the output pattern of a flow at the end of the relevant path. This flow will be referred to as the *reference flow.* The second part of this section discusses the possible usage of AAF as a building block in multicast congestion control schemes.

2.1 Definition of AAF

To define AAF, we need to define *attenuation factor* (AF). Consider the two aspects of congestion: one aspect is how severe the congestion is at each occurrence, the other is how often it happens. We define the former by *per-instance attenuation factor* θ, and the latter by *congestion occurrence rate* ρ. AF is the combination of θ and ρ.

Per-Instance Attenuation Factor (PIAF) θ: PIAF provides the information about each congestion occurrence. One *congestion occurrence* is an event that one or more packets of the reference flow are lost [2] between two consecutive received packets of the flow. Assume that the packet loss is detected at time t, the θ of this congestion occurrence is defined as:

$$\theta = \frac{I(t, \Delta t) - O(t, \Delta t)}{I(t, \Delta t)} \tag{1}$$

[1]By bottlenecks, we refer only to those links where the input is larger than the output.
[2]We assume that all packet loss is due to congestion.

where $O(t, \Delta t)$ is the output of the reference flow between $t - \Delta t$ and t, and $I(t, \Delta t)$ is the corresponding input of the reference flow that generates the output of $O(t, \Delta t)$. In other words, θ is the packet loss rate of the reference flow detected between $t - \Delta t$ and t. When there is no congestion (i.e. no packet loss), $\frac{I(t,\Delta t) - O(t,\Delta t)}{I(t,\Delta t)}$ is always one and we cannot infer any information from it. Therefore, θ is only defined when there is congestion. It shows the proportion of the input that the relevant path can successfully forward when fully loaded.

Congestion Occurrence Rate (COR) ρ: COR observes the other aspect of congestion. It indicates the frequency of congestion. For a particular congestion occurrence, we define ρ as $1/T$, where T is the time between this congestion occurrence and the prior one.

Attenuation Factor (AF) Θ: Given the PIAF (θ) and COR (ρ) for a certain congestion occurrence, the AF (Θ) of this congestion occurrence is defined as the product of ρ and θ, i.e.

$$\Theta = \rho \cdot \theta \tag{2}$$

Average Attenuation Factor (AAF) $\overline{\Theta}$: Since network conditions usually change randomly, θ and ρ are random variables and so is Θ. It is inaccurate to measure the degree of congestion on a path with a single sample of Θ. Instead, we should use the average attenuation factor (AAF) $\overline{\Theta}$, i.e. the expected value of Θ. For simplicity, we assume θ and ρ are independent, and therefore,

$$\overline{\Theta} \doteq E(\Theta) = E(\theta) \cdot E(\rho) \tag{3}$$

2.2 Usage of AAF as a building block in multicast congestion control

The multicast congestion control procedure can be divided into two steps: First, the source locates the most congested path; second, the source adapts the sending rate according to the feedback from behind that path using various kinds of policies.

AAF can be leveraged for the first step. To use it, let the source sends the same input to all paths in the multicast tree as any multicast source would do. Each receiver measures the AAF by checking the output of the reference flow. The AAF values by different receivers then can be compared (either centrally by the source or distributedly by the receivers[3]). The source chooses the path with the highest AAF value as the most congested one.

After that, the source can request ACK or NAK from the receivers, and does AIMD (additive increase/multiplicative decrease) or MIMD (multiplicative increase/multiplicative decrease) or other types of rate adaptation based on the feedback.

3 The Properties of Average Attenuation Factor

In the following, we will show that the average attenuation factor grows with several factors. These factors are intuitively considered as the indicators of degree of congestion. They include

(1) The total input of a bottleneck.

[3]How to perform comparison of AAF values is out of the scope of this paper and therefore is omitted.

(2) The reciprocal of the bottleneck bandwidth.

(3) The number of bottlenecks.

(4) 1/RTT of the crossing flows (the flows going through the same bottleneck with the reference flow).

As a result, it is appropriate to use attenuation factor to measure the degree of congestion on a path.

3.1 AAF increases in proportion to the total input of a bottleneck

Assume that there is a congestion occurrence at time t at a bottleneck. Let the bottleneck bandwidth be W. During $(t - \Delta t, t)$, N packets of the reference flow and totally M packets of the crossing flows arrive at the bottleneck. Let $w = W \cdot \Delta t$. The number of the reference flow packets that pass the bottleneck is then a random variable K. Assuming that the flows going through the bottleneck are asynchronous, the probability mass function of K given N and M is,

$$P(K = k|N, M) = \frac{\binom{N}{k}\binom{M}{w-k}}{\binom{N+M}{w}} \tag{4}$$

This is a hypergeometric distribution $\mathbf{HYP}(w, N, N + M)$. Therefore, the expectation value of K given known N and M is

$$E(K|N, M) = w\frac{N}{N + M}$$

Since $\theta = 1 - \frac{K}{N}$ (as defined at Equation (1)),

$$E(\theta|N,M) = 1 - \frac{w}{N+M} = 1 - \frac{W}{N+M}\Delta t \quad (5)$$

Note that $N+M$ is actually the total input of the bottleneck during $(t-\Delta t, t)$. Therefore, $E(\theta)$ increases with the total bottleneck input. The total input in question here is the total input at a particular moment. Consequently, we can assume it has no effect on ρ. According to Equation (3), we know $\overline{\Theta}$ also increases with the total input of a bottleneck.

3.2 AAF increases in reverse proportion to the RTTs of crossing flows

For the discussion, we assume that all the crossing flows in the network adjust their sending rates according to AIMD (additive increase and multiplicative decrease) policy, and they increase their sending rates by s/RTT once per RTT. (where s is the packet size). We assume all packet sizes are the same.

Consider a bottleneck. Suppose there are two consecutive congestion occurrences at time t_1 and t_2 respectively. From t_1 to t_2, a crossing flow has increased its sending rate by Λ in order for the total input rate of the bottleneck to exceed the bottleneck capacity. Referring to the definition of ρ, we have,

$$\rho = \frac{1}{t_2 - t_1} = \frac{1}{\frac{\Lambda}{s/RTT}} = \frac{s}{\Lambda \cdot RTT}$$

Since Λ and RTT changes randomly, assuming s is constant,

$$E(\rho) = \frac{s}{E(\Lambda) \cdot E(RTT)} \tag{6}$$

At the same time, if the RTT of a flow is larger, its source will increase the rate at a slower pace, and therefore the average input to the bottleneck is less. According to analysis in the above section (Section 3.1), $E(\theta)$ decreases. Based on Equation (3), we know AAF increases as the RTTs of crossing flows decreases.

3.3 AAF increases in reverse proportion to the bottleneck bandwidth

Intuitively, if the bandwidth of a bottleneck is smaller, its total throughput and thus the throughput of each flow on this link are smaller. Indeed, Equation (5) shows that $E(\theta)$ increases when the bottleneck bandwidth W decreases. Also, Λ in Equation (6) decreases with W, which means that flows on the bottleneck link need to increase their sending rates less to congest the link. Consequently, $E(\rho)$ increases as well when W decreases. Since $\overline{\Theta} = E(\theta) \cdot E(\rho)$, $\overline{\Theta}$ increases as the bottleneck bandwidth decreases.

3.4 AAF increases in proportion to the number of bottlenecks

To show $\overline{\Theta}$ increases with the bottleneck amount, we will show that both $E(\theta)$ and $E(\rho)$ increases with the bottleneck amount.

Assume the momentary output of the ith bottleneck on the relevant path is O_i, and the input generating this output is I_i. Let n be the total number bottlenecks on the path. As defined in Equation (1),

$$\theta = \frac{I_n - O_n}{I_1} = 1 - \frac{O_n}{I_1} = 1 - \prod_{i=1}^{n} \frac{O_i}{I_i} \quad \text{since } I_i = O_{i+1}$$

Since $O_i/I_i < 1$[4], as n increases, $\prod_{i=1}^{n} \frac{O_i}{I_i}$ decreases, and θ increases, so does $E(\theta)$.

On the other hand, according to the definition of ρ, $E(\rho) = 1/E(T)$ is the reciprocal of the average time between consecutive congestion occurrences. Consider a period of τ, and let m_i be the number of congestion occurrences on the ith bottleneck. $E(\rho)$ within this period is then,

$$E(\rho) = \frac{1}{\tau / \sum_{i=1}^{n} m_i} = \frac{1}{\tau} \sum_{i=1}^{n} m_i$$

As the bottleneck number n increases, $\sum_{i=1}^{n} m_i$ as well as $E(\rho)$ increases.

Conclusively, since both $E(\theta)$ and $E(\rho)$ increases with the bottleneck number n, $\overline{\Theta}$ increases in proportion to the number of bottlenecks.

4 Simulations

In this section, we will use *ns*-2 simulations to verify the properties of average attenuation factor (AAF) $\overline{\Theta}$ discussed in the previous section. We will also show the effectiveness of using $\overline{\Theta}$ to choose the most congested path in a multicast tree. The router buffer size is set to 25 packets, and the packet sizes are all 1000 bytes. For reference, the changes of average per-instance attenuation factor (PIAF) θ and average congestion occurrence rate (COR) ρ are

[4] If $O_i/I_i = 1$, the link is not considered a bottleneck.

shown along with AAF.

4.1 AAF increases in proportion to the total input of a bottleneck

In this simulation, the sending rates of all the flows change randomly according to the uniform distribution. In the one-bottleneck network (Figure 1), the reference flow is sent from node S1 to R1, the crossing flows go from node S2 to R2. They share the bottleneck of 10Mbps between the two routers.

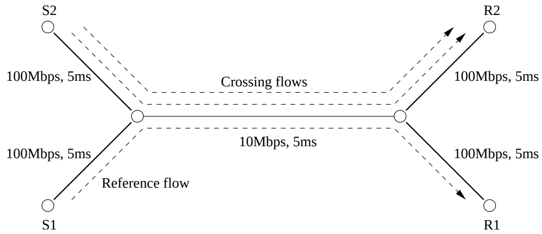
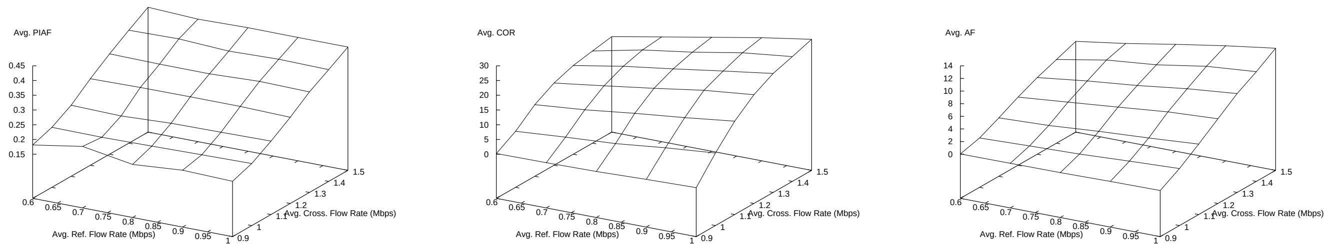


Figure 1: NETWORK WITH ONE BOTTLENECK

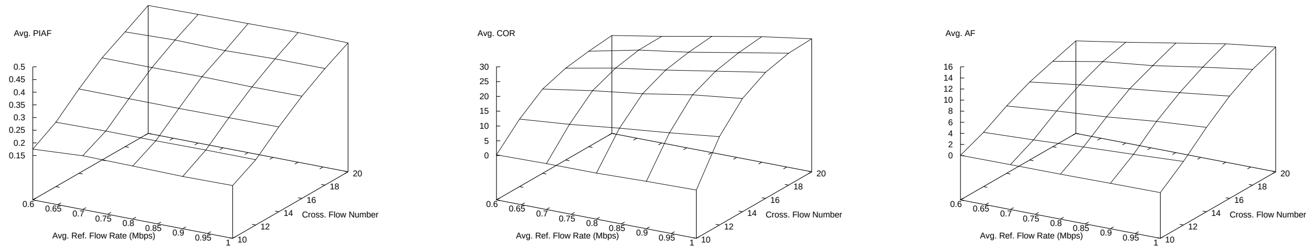
To increase the total input of the bottleneck, we use two different methods: a) to increase the average sending rates of the crossing flows without changing the number of the crossing flows, and b) to increase the number of the crossing flows while keeping their average sending rates constant. In both methods, we tested with five different values (0.6Mbps to 1.0Mbps with 0.1Mbps stepping) for the average sending rate of the reference flow.

In the first method, simulations were run with ten crossing flows. Their average sending rates are chosen from seven different values (0.9Mbps to 1.5Mbps with 0.1Mbps stepping). The results are shown in Figure 2. In the second method, simulations were run with six different numbers (10,12,14,16,18,20) of the crossing flows with their average sending rates as 0.9Mbps. The results are shown in Figure 3. We can see from both sets of results that the average AF grows with the total input of bottleneck.



Average PIAF (θ) Average COR (ρ) Average AF ($\overline{\Theta}$)

Figure 2: AAF GROWS WITH THE AVERAGE RATES OF CROSSING FLOWS.



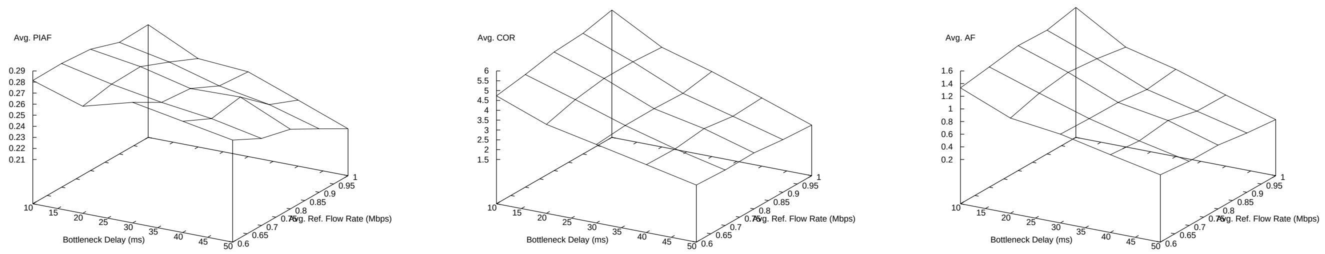
Average PIAF (θ) Average COR (ρ) Average AF ($\overline{\Theta}$)

Figure 3: AAF GROWS WITH THE NUMBER OF CROSSING FLOWS.

4.2 AAF increases in reverse proportion to the RTTs of crossing flows

In this simulation, we still use the topology in Figure 1, but we vary the bottleneck link delay between 10 ms and 50 ms with stepping as 10ms. For each choice of the bottleneck link delay, we again tested with five different values (0.6Mbps to 1.0Mbps with 0.1Mbps stepping) for the average sending rate of the reference flow. TCP (Reno) is used for the crossing flows.

Figure 4 shows that as the link delay increases, the AAF ($\overline{\Theta}$) decreases, which agrees with our analysis in Section 3.2.

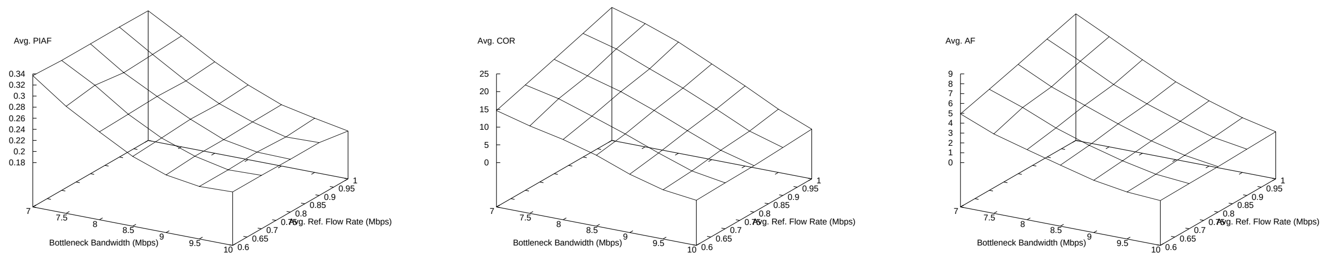


Average PIAF (θ) Average COR (ρ) Average AF ($\overline{\Theta}$)

Figure 4: AAF DECREASES AGAINST THE RTT OF CROSSING FLOWS.

4.3 AAF increases in reverse proportion to the bottleneck bandwidth

Similar to the configuration of first simulation, there are 10 crossing flows of random sending rates with the average of 0.9Mbps for each. The bandwidth of bottleneck in Figure 1 is chosen from seven different values (7.0Mbps to 10Mbps with 0.5Mbps stepping), and the average sending rate of the reference flow is configured as before. The results in Figure 5 shows that the AAF ($\overline{\Theta}$) decreases against the bottleneck bandwidth.



Average PIAF (θ) Average COR (ρ) Average AF ($\overline{\Theta}$)

Figure 5: AAF DECREASES AGAINST THE BOTTLENECK BANDWIDTH.

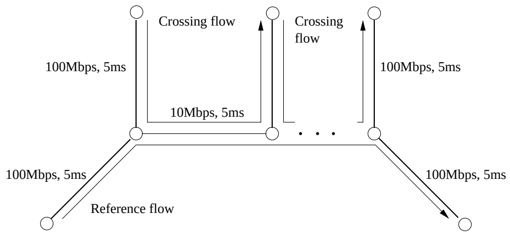
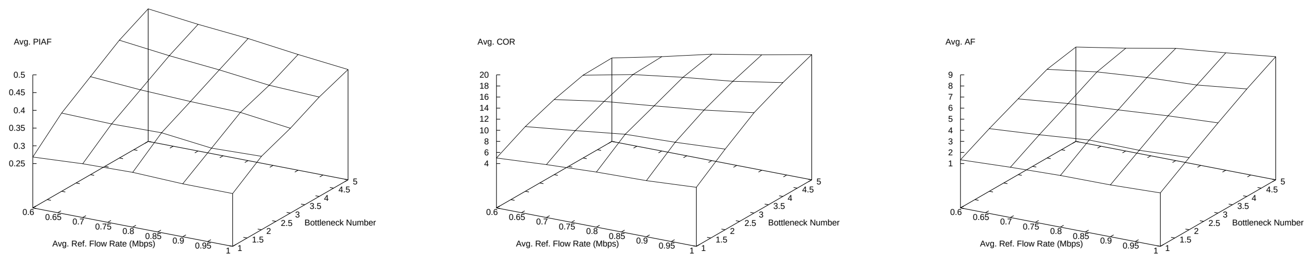


Figure 6: Network With Various Number of Bottlenecks

4.4 AAF increases in proportion to the number of bottlenecks

In Figure 6, there can be one to five bottlenecks of 10Mbps. On each bottleneck, there are 10 one-hop TCP (Reno) flows. The reference flow goes through all the bottlenecks. After running the simulations for different number of bottlenecks and different average sending rates of the reference flow, we got the results in Figure 7, which shows that as the reference flow goes through more bottlenecks, the AAF increases.



Average PIAF (θ) Average COR (ρ) Average AF ($\overline{\Theta}$)

Figure 7: AAF INCREASES WITH THE BOTTLENECK NUMBER.

4.5 The Effectiveness of Using AAF to Locate the Most Congested Path

To further verify that it is effective to use AAF for locating the most congested path, we constructed a simple single-rate multicast congestion control scheme. In the scheme, the source sends packets to all receivers at a single rate. It chooses the receiver which has the largest AAF as the representative, and adapts the sending rate only based on the feedback from the representative. NAK (negative acknowledgment, which is sent by receivers upon packet loss) is the only type of feedback packets we use. At the time of no loss, the sender increases the sending rate by s/RTT (s is the packet size) per RTT; upon the receipt of NAKs, the sender reduces the sending rate by half, but only does at most one reduction per RTT.

We tested two scenarios where the most congested path switches over time. It is expected that the receiver which detects the largest AAF also switches along with the most congested path, and the source always chooses the receiver behind the most congested path as the representative. The results agree with the expectations.

4.5.1 Scenario 1: Change the most congested path by varying the number of crossing flows

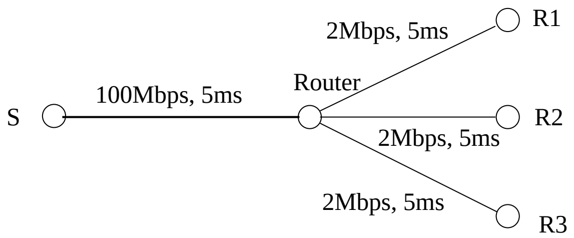


Figure 8: TREE TOPOLOGY USED FOR SCENARIOS 1 TEST

In the tree of Figure 8, packets are multicast from node S to receiver nodes R1, R2, and R3. Besides, there are TCP (new Reno) flows on the paths from the router to each receiver node. The simulation time is 500 seconds. Denote the link from the router to node Ri as link i. During the whole simulation, there are two TCP flows on link 1; between [100, 400], there are four on link 2; and between [200, 300], there are six on link 3. The most congested link at different times are shown in the following table:

Simulation Time	[0, 100]	[100, 200]	[200, 300]	[300, 400]	[400, 500]
Most Congested Link	link 1	link 2	link 3	link 2	link 1

The result is shown in Figure 9, where the sending rate has been averaged on 2-second scale. The vertical lines indicate that the representative is quickly switched after the most congested link has changed. In consequence, the sending rate is also adjusted lower.

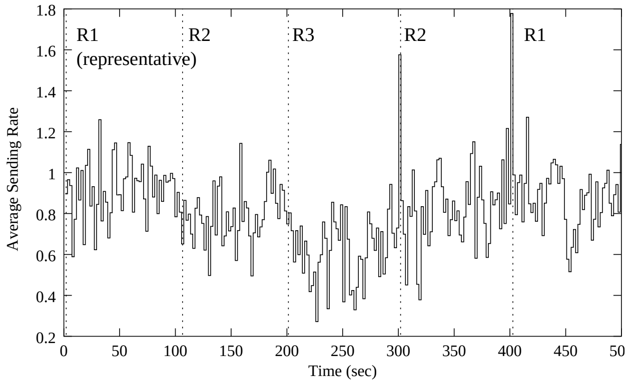


Figure 9: SCENARIO 1 TEST RESULT (The receiver behind the most congested path is always chosen as the representative)

4.5.2 Scenario 2: Change the most congested path by varying the number of bottlenecks

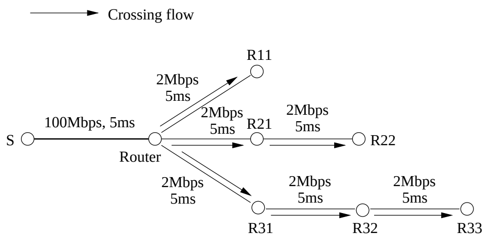


Figure 10: TREE TOPOLOGY USED FOR SCENARIOS 2 TEST

In this scenario, there are three different branches in the multicast tree (Figure 10). Multicast traffic is from node S to receiver nodes R11, R22, and R33. In addition, there is an one-hop TCP (new Reno) flow on each of the link between the router and the receiver nodes. The receiver nodes join and leave the multicast group at different moments: R11 is in the group for the whole simulation, R22 joins at 100th second and leaves at 400th second, R33 joins at 200th second and leaves at 300th second. The simulation time is 500 seconds. We observed that at moments around 101th, 201th, 305th, and 406th second, the representative was switched to R11, R22, R33, R22, and R11 respectively.

In summary, under different situations where a path becomes most congested either because more flows are sharing it or because it contains more bottlenecks, using AAF can locate it effectively.

5 Conclusion

We have defined *average attenuation factor (AAF)* (of throughput) which indicates how much and how often the input of a path is attenuated along its route. We proposed using AAF to locate the most congested path as a building block in multicast congestion control. Using AAF can eliminate the need for the source to exchange packets with every receiver, and thus can solve the feedback implosion problem which is present in existent solutions. We have analytically shown its relation to some commonly known factors that indicate the degree of congestion. We have also conducted simulations to verify the properties of AAF. The effectiveness of using AAF to locate the most congested path has been demonstrated by simulations for several scenarios.

As we have mentioned in the text, AAF's measured by receivers have to be compared in order to find the highest value. Careless design of the comparison mechanism (for example, all comparison is performed by the source) can still result in feedback implosion. Although at the first thought we can leverage some delay-based feedback suppression mechanisms (e.g. [2], [5]), it is possible to use AAF itself to develop a non-delay-based and more efficient mechanism. We schedule it for our future work.

References

[1] S. Bajaj, et al, “Improving Simulation for Network Research”, *Technical Report 99-702b, University of Southern California, March 1999, revised September 1999*

[2] Thomas T. Fuhrmann, Jorg Widmer, “On the Scaling of Feedback Algorithms for Very Large Multicast Groups”, *Computer Communications*, 24(5-6): 539-547, Mar. 2001.

[3] Gu-In Kwon, John Byers, “Smooth Multirate Multicast Congestion Control”, *To appear in IEEE INFOCOM '03*, April 2003.

[4] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott, “The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm”, *Computer Communications Review*, volume 27, number 3, Jul. 1997.

[5] Jorg Nonnenmacher, Ernst W. Biersack, “Scalable Feedback for Large Groups”, *IEEE/ACM Transactions on Networking*, 7(3): 375-386, Jun. 1999.

[6] Jitendra Padhye, Victor Firoiu, Don Towsley, Jim Kurose, “Modeling TCP Throughput: A Simple Model and its Empirical Validation,” *SIGCOMM*, Aug. 1998.

[7] L. Rizzo, “PGMCC: A TCP-friendly Single-Rate Multicast Congestion Control Scheme”, *SIGCOMM '00*, Aug '00.

[8] Jorg Widmer, Mark Handley, “Extending Equation-based Congestion Control to Multicast Applications”, *SIGCOMM 2001*, Aug. 2001.