ORMCC : A Simple And Effective Single-Rate Multicast Congestion Control Scheme

Jiang Li, Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180

Abstract— **There are two categories of multicast congestion control: multi-rate and single-rate. The former can adapt the transmission rate to heterogeneous receivers but requires layered transmission. The latter is simple to implement and deploy, and thus at least suitable for some situations. In this article, we propose a new single-rate scheme called *ORMCC*. It uses a new metric, *TRAC (Throughput Rate At Congestion)*, that makes the scheme simple as well as effective. We verified the performance by both theoretical analysis and simulations.**

Index Terms— **Multicast, congestion control, single-rate, drop-to-zero, TCP friendliness, feedback suppression, throughput rate at congestion (TRAC)**

I. Introduction

IP multicast is efficient for transmitting bulk data to multiple receivers. There are two categories of multicast congestion control. One of them is single-rate, in which the source controls the data transmission rate. The previous work include, for example, DeLucia et. al's work in [4], PGMCC[14], TFMCC[17], MDP-CC[9] and our prior work LE-SBCC [16]. The other is multi-rate (a.k.a layered multicast congestion control), in which receivers join just enough layers in the form of multicast groups to retrieve as much data as it can. The most noticeable among them are recently developed Fine-Grained Layered Multicast [2] and STAIR [3]. The single-rate category is easy to implement and deploy, because it does not require support from intermediate nodes. Although such schemes do not scale as well as multi-rate ones because they track the slowest receiver, they are at least suitable for some situations, e.g. the multicast in a not-so-heterogeneous environment, or bulk data transfer without concerns over delay. With some network support [8], we can also emulate multi-rate schemes by deploying single-rate schemes on selected intermediate nodes.

Our scheme is of the single-rate category. The contribution of our scheme is that it is simpler than previous work, while achieving the same or better performance. At the source and receivers, $O(1)$ states are maintained, the operations and computation are easy. We successfully address the problems of slowest receiver tracking, TCP-friendliness, and so-called drop-to-zero. Our feedback suppression works high effectively by suppressing over 95% feedback under normal situations. Also, we do not make any assumptions on network topology and intermediate nodes, and there is no need to measure RTTs from all receivers to the source, which can be a tedious problem especially without external instrumentation (e.g. GPS, NTP server). More important, the performance of our scheme can be verified by theoretical analysis also provided in this paper. The key idea is to base the scheme on a new metric, *TRAC (Throughput Rate At Congestion)*.

The general concept of our scheme is as follows: The source dynamically selects one of the slowest receivers as *Congestion Representative (CR)*, and only considers its feedback for rate adaption. The slowest receivers are those with the lowest average TRACs. When there is no CR, all receivers may send feedbacks to the source. Once a CR is selected, only the CR and those receivers with average TRAC lower than that of the CR can send feedbacks so that feedbacks are efficiently suppressed. Also notice that our scheme is not concerned with reliability issue and only consider congestion control. Therefore, it is applicable to both reliable and unreliable multicast.

An example operation can illustrate how our scheme works more clearly. In Figure 1 (a), the source has chosen a receiver behind the most congested path as CR by comparing average TRACs of receivers. Only the CR will send feedback while other receivers suppress their feedback. The feedback is CI(μ) in Figure 1 where CI means *congestion indication* and μ is TRAC measured by receiver. After a while, another path becomes the new most congested path. Those receivers behind that path will see average TRACs lower than that of the current CR, and will send feedbacks (Figure 1 (b)). As the result, one of them will be chosen as the new CR. After that, again, other receivers suppress their feedback (Figure 1 (c)).

In the following sections, we will briefly discuss some related work, followed by the ORMCC details. After that, we will analyze the performance of our scheme theoretically. Simulations results are presented at the end.

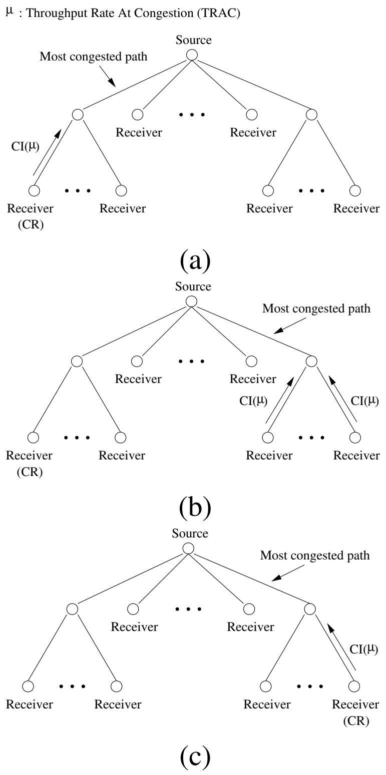

Fig. 1. EXAMPLE OF ORMCC OPERATION

II. Related Work

A. Single-Rate Schemes

DeLucia et. al's work in [4] is an early single-rate multicast congestion control scheme using representatives. It requires two types of feedback from receivers, *Congestion Clear (CC)* and *Congestion Indication (CI)*. Note that their CIs are single bit and thus different from ours. A fixed number of receiver representatives are maintained at the source. Whenever a CI is received by the source, if the sender of this CI is in the representative set, the representative is refreshed; if not, the sender will replace the representative that has not been refreshed for the longest time. Feedback from representatives is echoed by the source to suppress feedback scheduled at non-representative receivers. The source use only the feedback from representatives to do MIMD rate adaption.

The representative selection mechanism in that scheme is "simplistic"[4], but there is certain complexity involved in generating CC. The representative set is not guaranteed to include the slowest receiver, which means that the slowest receiver can be overloaded. Furthermore, it assumes that only a few bottlenecks cause the most congestion. Based on this assumption, receiver suppression is the only mechanism for filtering feedback from receivers. In a heterogeneous network, where there may be many different bottlenecks and asynchronous congestion, the assumption may not be true. Consequently, the transmission rate may be reduced more than necessarily and stay very low or close to zero. This is known as the *drop-to-zero* problem.

PGMCC [14], TFMCC [17] and MDP-CC [9] are recent work also using representatives. Although they use different policies for rate adaption, they all leverage the TCP throughput formula [13] [10] for allocating the slowest receiver, i.e the receiver with the lowest estimate TCP throughput according to the formula. Therefore, it is necessary for them to measure packet loss rate and RTT for all receivers.

PGMCC [14] keeps one representative as *acker*. The acker sends ACKs to the source which mimics the behavior of TCP. At the same time, NAKs with loss rate are sent from all other receivers. This is different from our scheme because we do not require separate ACK streams. The PGMCC source measures RTT between all receivers in terms of packet numbers, and compare the estimated throughput for updating acker. Due to the necessity of RTT measurement for all receivers, feedback suppression may have serious effect on PGMCC's performance. In fact, PGMCC does not provide a feedback suppression mechanism.

TFMCC [17] adjusts the rate according to the estimated rate calculated by the representative. RTTs are measured by receivers with a somewhat complex procedure. TFMCC comes with feedback suppression which is an enhanced version of [7] and is probabilistic timer-based. Therefore, the total number of feedbacks is the function of the total number of receivers, and additional delay is introduced into feedback.

MDP-CC [9] increases/decreases the transmission rate exponentially toward the target rate. Similar to TFMCC, The target rate is also calculated by the representative. In contrast to PGMCC and TFMCC, MDP-CC maintains a pool of representative candidates for representative update. As shown in that paper, maintaining multiple representative candidates requires much effort. MDP-CC can use probabilistic timer-based feedback suppression which has the same potential problem as that of TFMCC.

LE-SBCC [16] is our prior work. It only requires single bit NAKs from receivers, and the source has three cascaded filters to filter receiver feedback before using it for rate adaption. The computation complexity at the source is $O(1)$.However, it needs $O(n)$ states at the source, and network aggregation can also lead to performance degradation. ORMCC does not have these drawbacks.

B. Multi-Rate Schemes

Ideally, the multi-rate multicast congestion control can satisfy heterogeneous receivers because each of them receives data at its own rate. The most noticeable among them are recently developed Fine-Grained Layered Multicast [2] and STAIR [3]. However, the multi-rate schemes are *closely coupled* with routing and IGMP, which implies some potential problems. For example, different groups for layers could follow different routes [11]. Aggregated multicast trees [5] do not necessarily prune trees dynamically and hence break the assumptions of the multi-rate schemes. The slackness of response to congestion due to long leave latency continues to be an issue. Besides, frequent group joins and leaves can introduce significant load at routers.

III. ORMCC DETAILS

In this section, we will present the details of the whole scheme, summarized by the features of ORMCC.

A. Feedback Required from Receivers – CI(μ)

When receivers detect congestion by packet losses, they need to inform the source so that the source can adjust the transmission rate accordingly. In ORMCC, the feedback is *Congestion Indications with TRAC (CI(μ)s)*. Like NAKs, CI(μ)s may be sent only when a receiver detects packet losses. Assume that at the arrival of packet A, a receiver detects that some packets have been lost. It will then send a CI(μ) to the source. Such a CI(μ) contains (1) The sequence number of A, for the sake of RTT measurement, and (2) The output rate measured at the arrival of A, for the sake of CR allocation. Another difference between CI(μ) and NAK is that, CI(μ)s are not sent for every packet loss, but for every detection of packet losses if without suppression. With suppression, CI(μ)s may be sent with a less frequency.

To avoid oscillation, we average the output rate over several consecutive packets. Note that the output rates used here are different from those in normal sense, because they are measured only when packet losses are detected due to congestion. To distinguish them, we give the notation *Throughput Rate At Congestion (TRAC)*.

B. Allocation of The Slowest Receiver

ORMCC compares average TRAC of all receivers to allocate the slowest ones, and choose one of them as the *Congestion Representative (CR)*. By using TRAC, it avoids computing TCP throughput formula [10] [13] which requires RTT and packet loss rate.

Since TRACs are measured during packet losses at receivers, they indicate how much bandwidth a flow can get out of the fully loaded bottleneck, assuming congestion is the only reason for packet losses. The less it can get, the more congested the bottleneck is. Therefore, we choose one receiver with the lowest average TRAC as the CR, and let the source only consider the CI(μ)s from it for rate adaption. Average TRAC is calculated by means of *EWMA (Exponentially Weighted Moving Average)*. Denote TRAC as U, average TRAC as $E(U)$, deviation as U_σ. With a sample U, $E(U)$ and U_σ are updated as $U_\sigma = (1-\alpha)U_\sigma + \alpha|E(U) - U|$, $E(U) = (1-\alpha)E(U) + \alpha U$.

The receivers help the source to select a receiver with the lowest average TRAC by sending in CI(μ)s only if their average TRACs are low enough to qualify them as CR. More details of how receivers help will be covered in the subsection III-D of feedback suppression.

C. Update of CR under Dynamic Conditions

Network conditions always keep changing, and we need to continuously keep our choice of CR up-to-date. There are mainly two situations under which CR needs to be updated: (1) The congestion seen by the current CR remains unchanged, but another receiver sees more severe congestion. (2) The previously most congested path is improved so that the current CR sees less congestion than other receivers, or the current CR leaves the multicast session.

Tracking the slowest receiver by examining average TRACs can deal with situation (1), but to cope with situation (2) needs more effort. Under this situation, there can be no CI(μ)s from the current CR. Recall that the source only consider the CI(μ)s from the CR for rate adaption and ignores all other CI(μ)s. If the source does not change CR in time, the transmission rate will be out of control. To detect that, we estimate an upper bound (denoted as T^{cr}_{max}) of the response time when the bottleneck is fully loaded (denoted as T^{cr}). Suppose we somehow detect that the bottleneck is fully loaded at time t. If there has been no CI(μ) from the current CR for T^{cr}_{max} since t, we can say that the current CR is now inactive and needs to be changed.

Let's look at Figure 2. When the CR is still active, we measure samples of T^{cr}. When the transmission rate reaches $E(U^{cr}) + 4U^{cr}_\sigma$, where $E(U^{cr})$ and U^{cr}_σ are the average and deviation of the current CR's TRAC respectively, we assume that bottleneck becomes fully loaded and start to count. Let the current time be t_0. At a later time t_1, the first CI(μ) since t_0 arrives at the source from the CR. $t_1 - t_0$ is then a sample of T^{cr} and we update the average and deviation of T^{cr} with EWMA. T^{cr}_{max} is the average value of T^{cr} plus eight times its deviation. When the CR is not active, for the duration of T^{cr}_{max} since we start to count, no CI(μ) will be received by the source. The source then requests feedback from other receivers for new CR selection, as described in next subsection III-D.

There is one small trick we use to bias the choice of CR toward those receivers with higher RTTs. After a new CR is chosen, we set a grace period of $2RTT_{max}$, where

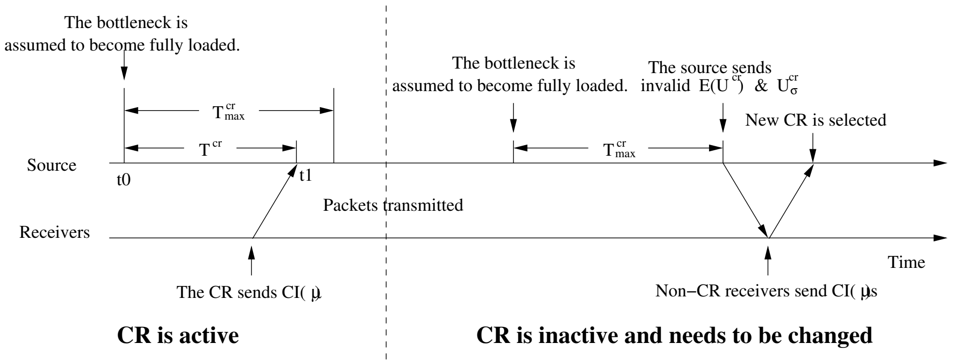

Fig. 2. UPDATE OF CONGESTION REPRESENTATIVE (CR)

RTT_{max} is the maximum RTT the source has ever seen. Within this period, if the source receives a CI from another receiver with similar average TRAC as that of the CR, it will update CR to this receiver, since this one tends to have longer RTT. Grace period is not reset after CR switches within grace period.

D. Feedback Suppression by Receivers

Effective feedback suppression can reduce the risk of feedback implosion, and allow a multicast congestion control scheme to be used for large groups. In ORMCC, the source conveys the average $E(U^{cr})$ and the deviation U^{cr}_{σ} of the CR's TRAC to receivers whenever the CR is updated or $E(U^{cr})$ and U^{cr}_{σ} are changed. Only if a receiver's own average TRAC $E(U)$ is less than $E(U^{cr}) - U^{cr}_{\sigma}$[1], will it send CI($\mu$)s. Note that a receiver does not maintain TRAC deviation U_{σ} because it is computed by the source with the help of TRAC in CI(μ)s.

$E(U^{cr})$ and U^{cr}_{σ} conveyed by the source can be of invalid values so that all receivers can send CI(μ)s. This is needed when the current CR is inactive and the source needs to trigger feedbacks from all receivers for new CR selection (Figure 2).

Clearly, no timer is involved in our feedback suppression, no knowledge of the whole group is needed. Unlike other probabilistic timer-based feedback suppression schemes, CI(μ)s are not scheduled at all before being suppressed. Yet, it is effective since the amount of CI(μ)s sent to the source is independent of the total receiver number. More insight will be given in a later analysis at IV-D.

E. Rate Adaption

ORMCC is a rate-based scheme. When increasing the rate, we mimic the behavior of TCP. That is, if there are no CI(μ)s from the CR, the transmission rate is increased by s/RTT per RTT, where s is the packet size, RTT is that between the source and the CR. If a CI is received from the CR, let the TRAC in this CI be μ, we adjust the transmission rate to the minimum of $\beta\mu$ and the current rate. β is 0.75 in our implementation. The analysis in a later section IV-B will discuss the choice of β's value. CI(μ)s from other non-CR receivers will be ignored, and at most one rate cut is allowed per RTT.

[1] We don't use $E(U) \geq E(U^{cr})$ because we want to be conservative and keep CR stable.

F. RTT Estimation

Unlike NAKs, which include the sequence number of a lost packet, CI(μ)s include the sequence number of a packet upon the arrival of which packet losses are detected. The source calculates the difference between the sending time of this packet and the arriving time of this CI(μ) to get a sample of RTT. By doing this, we avoid the unnecessary delay between the supposed arriving time of a lost packet and the time of its loss being detected. Nevertheless, RTT estimated by CI(μ)s is still the upper bound, because it is measured at packet losses and thus includes bottleneck queueing delay. On average, it is larger than RTT estimated by ACKs as in TCP.

As the summary of this section, the features of ORMCC are listed in the following:

- $O(1)$ **States** The numbers of states maintained by source and receivers are $O(1)$.
- **Simple Operations** Operations of source and receivers are all simple, without requiring intense computation. In particular, there is no need to do per-receiver RTT estimation.
- **Effective Feedback Suppression** With our non-probabilistic-timer-based feedback suppression mechanism in place, the amount of feedbacks is independent of the total number of receivers.

The pseudo code of ORMCC's algorithm is presented in the appendix for reference.

IV. THEORETICAL ANALYSIS

A. Capability of Tracking The Slowest Receiver

In this part, we are going to show that an ORMCC flow always track the slowest receiver, i.e. the receiver behind the most congested path. For convenience, we are going to refer the path between the source and the CR as *Representative Path*.

Let's consider a multicast session using ORMCC. Suppose there are N ($N > 1$) different paths on the multicast tree. Let R_i be the receiver behind path i. Without loss of generality, assume R_1 is the current CR. The source will choose another receiver R_j ($j \neq 1$) as the new CR only if R_j sees a lower throughput rate at congestion (TRAC) than that seen by R_1. To see when a R_j will see a lower TRAC on average, first we are going to calculate the TRACs on all paths from 1 to N, given the instantaneous ORMCC sending rate at which a burst of packet losses begins.

For the analysis, we have the following definitions (all i are from 1 to N):

$\lambda_{i,t}$: Instantaneous sending rate of the ORMCC flow at time t on path i.

$\mu_{i,t}$: Instantaneous throughput rate of the ORMCC flow corresponding to λ_i.

$\lambda^o_{i,t}$: The sum of instantaneous sending rates of all but the ORMCC flows sharing the bottleneck on path i at time t.

W_i : Bandwidth of the bottleneck on path i.

Q : Buffer size of a bottleneck. Here we assume that all bottlenecks have the same buffer size. If a queue is constantly non-zero, we will treat the part which is emptied and (partly) filled as the whole queue.

s : Packet size. We assume that all the packet sizes are equivalent.

RTT_i : RTT of path i.

Δ : The sending rate increment of the ORMCC flow per unit time. $\Delta = s/RTT_1 > 0$.

Δ_i : The sum of the sending rate increments per unit time of all but the ORMCC flows sharing the bottleneck on path i, without packet losses occuring at the bottleneck, assuming all do AIMD. In reality, most likely Δ_i changes randomly, therefore we consider its average value in an aggregate sense. $\Delta_i > 0$.

γ_i : $\gamma_i = \Delta_i/\Delta$.

Moreover, we assume that sending rates are increased continuously, as well as that all packet losses are due to congestion. Also, drop-tail buffer management is assumed for bottlenecks.[2]

Let's consider path 1 first (Figure 3). Suppose at time t_1, there is a burst of packet losses. The bottleneck queue must be full at this moment. The sum of sending rates of

[2]Although our analysis is based on drop-tail routers, ORMCC also works well with RED routers. It has been confirmed by simulations, though for space reason, the results are not included.

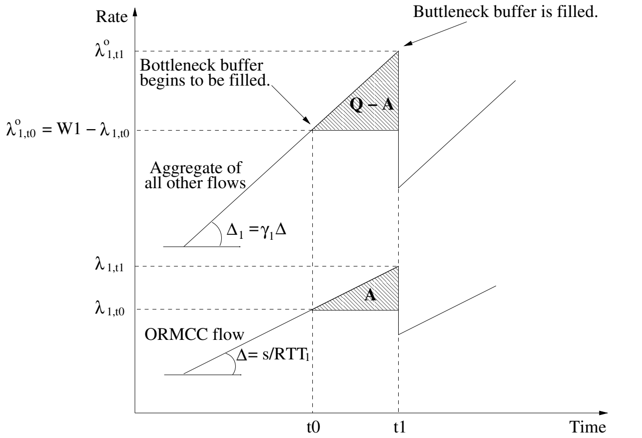

Fig. 3. EVOLUTION OF THE ORMCC FLOW'S SENDING RATE ON THE REPRESENTATIVE PATH

all the flows going through the bottleneck, $\lambda_{1,t_1} + \lambda^o_{1,t_1}$, must be larger than the bottleneck bandwidth W_1. Recall that $\Delta > 0$ and $\Delta_1 > 0$, meaning that without packet losses at the bottleneck, the sum of sending rates keep increasing. Consequently, at an earlier moment t_0, the sending rate sum must be equal to W_1, i.e.,

$$\lambda_{1,t_0} + \lambda^o_{1,t_0} = W_1 \tag{1}$$

Since the sending rate of the ORMCC flow grows by Δ per unit time,

$$\lambda_{1,t_1} = \lambda_{1,t_0} + (t_1 - t_0)\Delta \Rightarrow \lambda_{1,t_0} = \lambda_{1,t_1} - (t_1 - t_0)\Delta \tag{2}$$

$\lambda^o_{1,t}$ grows by $\Delta_1 = \gamma_1\Delta$ per unit time, therefore,

$$\lambda^o_{1,t_1} = \lambda^o_{1,t_0} + (t_1 - t_0)\Delta_1 = \lambda^o_{1,t_0} + (t_1 - t_0)\gamma_1\Delta \tag{3}$$

From (1), (2) and (3), we have,

$$\begin{aligned} \lambda^o_{1,t_1} &= W_1 - (\lambda_{1,t_1} - (t_1 - t_0)\Delta) + (t_1 - t_0)\gamma_1\Delta \\ &= W_1 - \lambda_{1,t_1} + (t_1 - t_0)(1 + \gamma_1)\Delta \end{aligned} \tag{4}$$

We also assume that at t_0, the bottleneck queue size is zero. Since at t_1, the queue is full, the queue is filled by sending rate increments during $[t_0, t_1]$. Recalling that the total sending rate grows by $\Delta + \Delta_1 = (1 + \gamma_1)\Delta$ per unit time, and the assumption of all flows' doing AIMD, we have,

$$\frac{1}{2}(t_1 - t_0)^2(1 + \gamma_1)\Delta = Q \Rightarrow t_1 - t_0 = \sqrt{\frac{2Q}{(1 + \gamma_1)\Delta}}$$

Together with (4),

$$\lambda^o_{1,t_1} = W_1 - \lambda_{1,t_1} + \sqrt{2\Delta Q(1 + \gamma_1)} \tag{5}$$

Assuming all flows going through the bottlenck have the same priority, since at time t_1 the bottleneck is working at its full load, we know

$$\begin{aligned} \mu_{1,t_1} &= \frac{\lambda_{1,t_1}}{\lambda_{1,t_1} + \lambda^o_{1,t_1}} W_1 \\ &\overset{(5)}{=} \frac{\lambda_{1,t_1}}{1 + \frac{1}{W_1}\sqrt{2\Delta Q(1 + \gamma_1)}} \end{aligned} \tag{6}$$

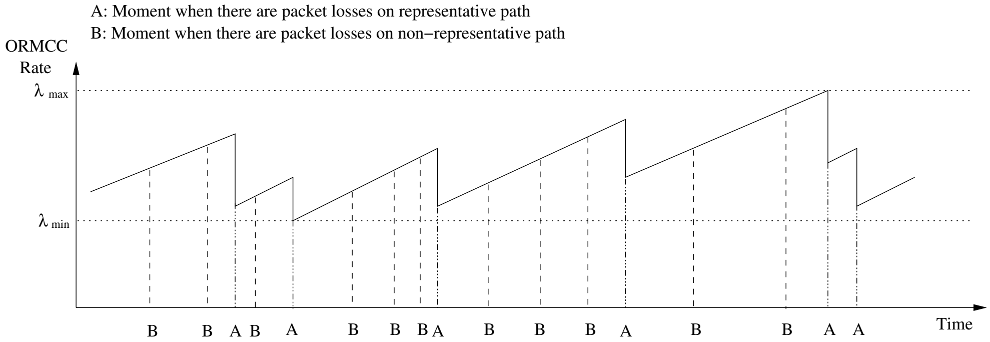

Fig. 4. THE ORMCC SOURCE ONLY CONSIDERS THE CONGESTIONS ON THE REPRESENTATIVE PATH FOR RATE ADAPTION

On any other path j $(j = 2 \dots N)$, since the ORMCC source ignores the congestion indications on this path (Figure 4), the sending rate of the ORMCC flow still grows by Δ per unit time. With t_1 of the same meansing as before, according to a derivation similar to that above, we have,

$$\mu_{j,t_1} = \frac{\lambda_{j,t_1}}{1 + \frac{1}{W_j}\sqrt{2\Delta Q(1+\gamma_j)}} \quad (7)$$

Consider λ_{i,t_1} $(i = 1 \dots N)$. Assume that the sending rate of the ORMCC flow varies between λ_{min} and λ_{max} (Figure 4), then λ_{i,t_1} is a sample value of a random variable Λ_i with sample space as $[\lambda_{min}, \lambda_{max}]$. Due to the randomness in the realistic networks, we can assume that Λ_i's are identically distributed, and their expected values are the same, i.e. $E(\Lambda_i) = E(\Lambda_j)$ $(i \neq j)$.

Let receiver i be the receiver behind path i. μ_{i,t_1} $(i = 1 \dots N)$ is the TRAC measured at receiver i. According to (7), μ_{i,t_1} is a function of λ_{i,t_1}, and thus is a random sample. Denote the corresponding random variable as U_i, and assume that W_i, Q, s, γ_i and RTT_1 are all constant, [3] we have,

$$U_i = \frac{\Lambda_i}{1 + \frac{1}{W_i}\sqrt{2\Delta Q(1+\gamma_i)}}$$
$$E(U_i) = \frac{E(\Lambda_i)}{1 + \frac{1}{W_i}\sqrt{2\Delta Q(1+\gamma_i)}}$$

As designed in ORMCC, for $j = 2 \dots N$, only upon detection of $E(U_j) < E(U_1)$ (the average TRAC of the current CR receiver 1) will receiver j send congestion indication (CI) packets back to the source, which then update the congestion representative (CR) to receiver j. From the expression of $E(U_i)$ above, we have,

$$\begin{aligned} & E(U_j) < E(U_1) \\ \Leftrightarrow \quad & \frac{E(\Lambda_j)}{1 + \frac{1}{W_j}\sqrt{2\Delta Q(1+\gamma_j)}} < \frac{E(\Lambda_1)}{1 + \frac{1}{W_1}\sqrt{2\Delta Q(1+\gamma_1)}} \\ \Leftrightarrow \quad & \frac{W_j}{\sqrt{1+\gamma_j}} < \frac{W_1}{\sqrt{1+\gamma_1}} \qquad \text{since } E(\Lambda_j) = E(\Lambda_1) \quad (8) \end{aligned}$$

[3]Even if some of them change randomly, we can still assume they are mutually independent.

We can see that $W_i/\sqrt{1+\gamma_i}$ $(i = 1 \dots N)$ indicates the degree of congestion on the bottleneck of path i. In fact, if the bottleneck has less bandwidth, i.e. W_i is smaller, $W_i/\sqrt{1+\gamma_i}$ has a lower value; if more flows are sharing a bottleneck, the sum of their per-unit-time rate increments Δ_i is higher, $\gamma_i = \Delta_i/\Delta$ is then larger, which in turn also makes $W_i/\sqrt{1+\gamma_i}$ lower. Therefore, (8) actually shows that as long as a non-representative path (path j) experiences a more serious congestion than the representative path (path 1) does, the receiver behind path j will see lower average TRAC $E(U_j)$, and will send CI(μ)s back to the source, making the source change CR. Namely, *an ORMCC flow always tracks the slowest receiver.*

B. TCP-Friendliness on Representative Path

By representative path, we mean the path which the congestion representative (CR) is behind. In the following, we are going to show that an ORMCC flow is friendly to a TCP flow on the representative path, by showing that they get approximately equal share of the bottleneck bandwidth, with the assumption that their RTT estimations and packet sizes are the same. More strictly speaking, we want to show that, with proper choice of rate reduction factor β for ORMCC, V_t^{MCC}/V_t^{TCP} oscillates around 1, where V_t^{MCC} and V_t^{TCP} denote the sending rates of the TCP flow and ORMCC flow at time t respectively. Those two flows are assumed to be the only flows on the representative path. A sample of the rate evolution is given in Figure 5.

Like other TCP throughput analysis papers [13] [10] have done, our analysis focuses only on TCP's congestion avoidance behavior. During congestion avoidance period, when without packet losses, a TCP source increases its congestion window by $1/N$ packet upon the receipt of per ACK, where N is the current congestion window size. A TCP source transmit all the packets in its congestion window in one RTT, therefore, the window grows by 1 packet

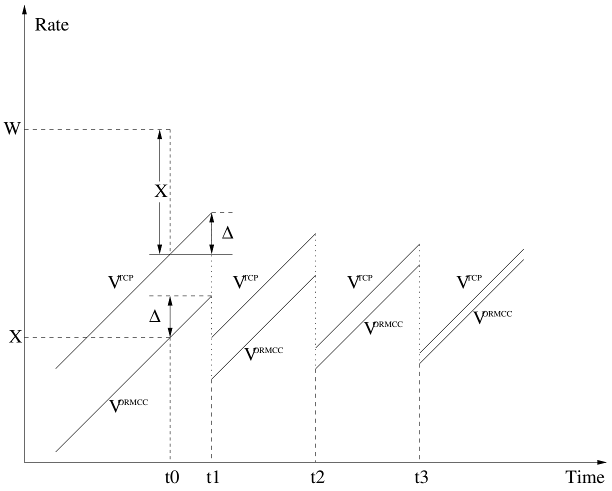

Fig. 5. EVOLUTION OF THE SENDING RATES OF TCP AND ORMCC FLOWS

per RTT, [4] which corresponds to the fact that its sending rate is increased by s/RTT per RTT, where s is the packet size. An ORMCC source increases its sending rate at the same pace, as covered in scheme description. At packet loss, a TCP source will reduce its congestion window by half, which is equivalent to cutting its sending rate by half.

Assume that congestion is the only reason for packet losses. Let W be the bottleneck bandwidth. It is obvious that packet losses can occur only if $V_t^{TCP}+V_t^{MCC} \geq W$. Suppose some packets are lost and both flows reduce their transmission rates at t_1 (Figure 5). Before the losses, since both V_t^{TCP} and V_t^{MCC} keep increasing, there must be a moment t_0 when $V_{t_0}^{TCP} + V_{t_0}^{MCC} = W$. For short, let $V_{t_0}^{MCC} = X$, then $V_{t_0}^{TCP} = W - X$. For the first step of analysis, we will show that with appropriate β,

$$\begin{cases} X < W - X \Rightarrow X/(W-X) < V_{t_1}^{MCC}/V_{t_1}^{TCP} \\ X > W - X \Rightarrow X/(W-X) > V_{t_1}^{MCC}/V_{t_1}^{TCP} \end{cases} \tag{9}$$

Let the moment just before the rate reduction at t_1 be t_1'. Because the TCP and ORMCC flows share the same path, we can safely assume that they detect packet losses and reduce transmission rates at the same time. For the TCP flow, suppose that at t_1', its transmission rate has been increased by Δ since t_0, i.e.

$$V_{t_1'}^{TCP} = W - X + \Delta$$

After a reduction by half,

$$V_{t_1}^{TCP} = \frac{V_{t_1'}^{TCP}}{2} = \frac{W - X + \Delta}{2}$$

Since the ORMCC flow increases its rate at the same pace, we have,

$$V_{t_1'}^{MCC} = X + \Delta$$

[4]We assume that a TCP receiver sends an ACK per received packet.

Assume that both flows have the same priority, i.e. their packets are forwarded by the bottleneck with the same probability. In consequence, at t_1', the ORMCC CR sees a receiving rate of

$$\frac{V_{t_1'}^{MCC}}{V_{t_1'}^{MCC} + V_{t_1'}^{TCP}} W = \frac{X + \Delta}{W + 2\Delta} W$$

According to the rate adaption policy of ORMCC,

$$V_{t_1}^{MCC} = \beta \frac{X + \Delta}{W + 2\Delta} W$$

Therefore,

$$\frac{V_{t_1}^{MCC}}{V_{t_1}^{TCP}} = \beta \frac{X + \Delta}{W + 2\Delta} W \Big/ \frac{W - X + \Delta}{2}$$

Now let's compare $X/(W-X)$ and $V_{t_1}^{MCC}/V_{t_1}^{TCP}$.

$$\begin{aligned} & \frac{X}{W-X} - \frac{V_{t_1}^{MCC}}{V_{t_1}^{TCP}} \\ = \; & \frac{2}{W - X + \Delta} \left[\left(\frac{1}{2} - \frac{\beta W}{W + 2\Delta} \right) X + \left(\frac{X}{2(W-X)} - \frac{\beta W}{W + 2\Delta} \right) \Delta \right] \end{aligned} \tag{10}$$

Since $W > X$ and $\Delta \geq 0$, $2/(W - X + \Delta) > 0$, and the positivity of (10) is decided by its second factor between square brackets. If we choose a value for β carefully so that,

$$\beta = \frac{1}{2} \frac{W + 2\Delta}{W}$$

The second factor of (10) becomes

$$0 \cdot X + \frac{\Delta}{2} \left(\frac{X}{W-X} - 1 \right) = \frac{\Delta}{2} \left(\frac{X}{W-X} - 1 \right) \tag{11}$$

It is easily seen that, if $X > W - X$, (11) > 0 so that (10) > 0; while if $X < W - X$, (11) < 0 so that (10) < 0. That is exactly what we want for (9).

With (9) eastablished, we can go further. Assume that at t_i, $i = 1 \ldots \infty$, both the TCP flow and the ORMCC flow reduce their rates (Figure 5), and their rates after reduction are $V_{t_i}^{TCP}$ and $V_{t_i}^{MCC}$ respectively. Also assume that $t_{i,0}$ is the closest moment before t_i so that $V_{t_{i,0}}^{TCP} + V_{t_{i,0}}^{MCC} = W$. Recall the meanings of X and $W - X$, (9) actually indicates that,

$$\begin{cases} V_{t_{i,0}}^{MCC} < V_{t_{i,0}}^{TCP} \Rightarrow V_{t_{i,0}}^{MCC}/V_{t_{i,0}}^{TCP} < V_{t_i}^{MCC}/V_{t_i}^{TCP} \\ V_{t_{i,0}}^{MCC} > V_{t_{i,0}}^{TCP} \Rightarrow V_{t_{i,0}}^{MCC}/V_{t_{i,0}}^{TCP} > V_{t_i}^{MCC}/V_{t_i}^{TCP} \end{cases} \tag{12}$$

Let's consider the situation that $V_{t_{i,0}}^{MCC} < V_{t_{i,0}}^{TCP}$ first, which is shown in Figure 5. Note that for any i,

$$V_{t_{i+1,0}}^{MCC} = V_{t_i}^{MCC} + A, \;\; V_{t_{i+1,0}}^{TCP} = V_{t_i}^{TCP} + A \quad A > 0$$

So,

$$\frac{V_{t_i}^{MCC}}{V_{t_i}^{TCP}} < \frac{V_{t_i}^{MCC} + A}{V_{t_i}^{TCP} + A} = \frac{V_{t_{i+1,0}}^{MCC}}{V_{t_{i+1,0}}^{TCP}} \overset{(12)}{<} \frac{V_{t_{i+1}}^{MCC}}{V_{t_{i+1}}^{TCP}}$$

Similarly, if $V_{t_{i,0}}^{MCC} > V_{t_{i,0}}^{TCP}$,

$$\frac{V_{t_i}^{MCC}}{V_{t_i}^{TCP}} > \frac{V_{t_{i+1}}^{MCC}}{V_{t_{i+1}}^{TCP}}$$

As the result, if the ORMCC flow rate is less than that of the TCP flow, it will grow until it exceeds the latter; likewise, if the ORMCC flow rate is more, it will get less and less until it is below the TCP flow rate. Hence, V_t^{MCC}/V_t^{TCP} oscillates around 1. Conclusively, we say that ORMCC is approximately TCP friendly, given that the rate reduction factor β is properly chosen.

With regard to the value of rate reduction factor β, recall that above in the analysis, we need to have that

$$\beta = \frac{1}{2}\frac{W + 2\Delta}{W}$$

Since $\Delta \geq 0$, $\beta \geq 0.5$. Consider the fact that TCP uses ACKs to measure RTTs. It can have lower RTT estimation than that of ORMCC which uses NAKs for this purpose. Thus, TCP can increase sending rate faster than ORMCC. For compensation, ORMCC at packet losses can reduce its transmission rate by less. In our implementation, we use a value of 0.75 and it works fine in simulations.

C. Immunity To Drop-to-zero Problem

The cause of drop-to-zero problem is the asynchronous packet losses on multiple paths. If a multicast source reduces the transmission rate too much on the losses, the rate will stay very low or even converge to zero. However, in nature, the source in ORMCC adapts the transmission rate according to the congestion on one single path while ignoring that on all others, there will be no drop-to-zero problem for ORMCC. In more details, if a receiver other than the current congestion representative (CR) sees a packet loss rate lower or equal to that by CR, it won't send CI(μ)s back to the source. The source won't see any CI(μ)s from it, thus of course won't reduce the transmission rate. Even if the source gets CI(μ)s from different receivers because there is a change of the most congested bottleneck, once it chooses a receiver as the new CR after a very short period of time (several RTTs), it will ignore CI(μ)s from all other receivers. Consequently, we can say that ORMCC is immune to drop-to-zero.

D. Effectiveness of Feedback Suppression

Without support from internal nodes, which is the situation that we assume for reality, most multicast feedback suppression schemes (e.g. [6], [15], [17], [7], [12]) use random timers for delaying receivers' feedback before sending them. However small it is, there is some feedback latency which may bring performance penalty. Since our feedback suppression is not based on timers, it does not suffer from this problem. Also, there is no need to know or estimate the total number of receivers like [7]. Moreover, we are going to show below that in ORMCC, the total number of feedbacks (i.e. CI(μ)s) sent to the source by all receivers in a multicast session, is independent of the total number of receivers. Instead, it depends on the switching frequency of the most congested bottleneck, as well as the number of receivers behind the new most congested bottleneck, plus the minimum RTT between them and the source. For convenience, we use the acronym *MCB* for *most congested bottleneck* in the following discussion.

We assume that there is only one MCB at any moment[5]. To begin the calculation, the following notations are needed[6].

N : Total number of receivers behind the new most congested bottleneck (MCB).
R_i : Receiver i behind the new MCB. ($i = 1 \ldots N$)
RTT_i : RTT between the source and R_i.
RTT_i^f : Forwarding (downstream) part of RTT_i.
RTT_{min} : The minimum of all RTT_i's.
p : Packet loss rate seen by receivers behind the new MCB.
v : Average transmission rate of the ORMCC flow.

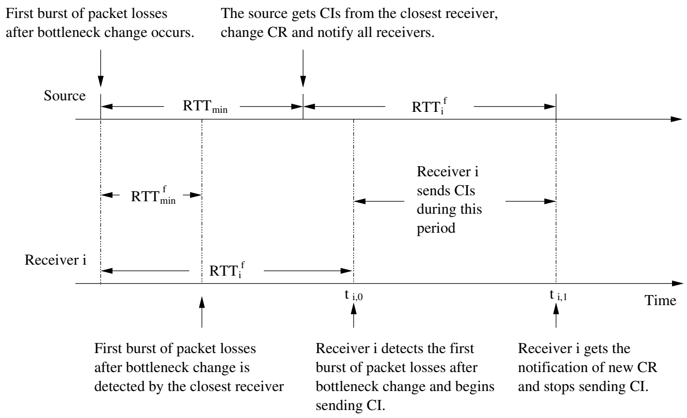

Fig. 6. FEEDBACK SUPPRESSION

Whenever there is a new MCB, according to the previous discussion of ORMCC's following the most congested path, only those receivers behind the new MCB will send CI(μ)s back to the source, and all of them except one will stop sending CI(μ)s once one of them is chosen as the new CR. More specifically, the source will

[5]There can certainly be multiple bottlenecks which have similar degree of congestion and are all most congested. However, the discussion still holds.

[6]Since the receivers involved here are all behind MCB, we can assume that they see the same degree of congestion and thus the same packet loss rates.

first see the CI(μ)s from the receiver with RTT_{min}, then change CR and tell all receivers of the change. For any R_i except the new CR, the duration of sending CI(μ)s is between the moment $t_{i,0}$ when they first detect packet loss after bottleneck change and the moment t_{i_1} when they know the new CR. According to Figure 6, $t_{i_1} - t_{i_0} = RTT_i^f + RTT_{min} - RTT_i^f = RTT_{min}$. Therefore, before a new CR is decided, the number of CI(μ)s sent from this receiver i is thus $pvRTT_{min}$, and the total number of CI(μ)s sent by all receivers behind the new MCB is,

$$\sum_{i=1}^{N} pvRTT_{min} = vpN \cdot RTT_{min}$$

Once a new CR is decided, only one receiver, namely the new CR, will send CI(μ)s. Let's call the period between two successive MCB switchings *MCBSP (MCB switching period)*. During a MCBSP of length t, the total number of CI(μ)s sent to the source is, assuming $t \geq RTT_{min}$,

$$vpN \cdot RTT_{min} + vp(t - RTT_{min}) = vp(t + (N-1)RTT_{min})$$

If a Poisson process with parameter λ[7] is assumed for MCB switching, for a multicast session of duration T, on average, the total number of CI(μ)s transmitted is approximately,

$$\lambda T \cdot vp\left(\frac{1}{\lambda} + (N-1)RTT_{min}\right) \quad (13)$$

We can see that, for a ceratin T,

$$\lim_{\lambda \to 0} \frac{\lambda T \cdot vp\left(\frac{1}{\lambda} + (N-1)RTT_{min}\right)}{vpT} = 1 + \lim_{\lambda \to 0} \lambda(N-1)RTT_{min} = 1$$

namely,

$$\lim_{\lambda \to 0} \lambda T \cdot vp\left(\frac{1}{\lambda} + (N-1)RTT_{min}\right) = vpT$$

That means, if during a multicast session, there is no MCB switching, the total number of CI(μ)s transmitted is approximately equal to the number of CI(μ)s sent from a single receiver behind the MCB. To make it clearer: if the MCB does not change during a multicast session, the volume of feedbacks is on the same level of unicast feedback!

Also, from (13), we find that the total number of transmitted CI(μ)s is independent of the total number of receivers in a multicast session. It depends on how fast MCB switches and the amount of receivers behind the new MCB, as well as the smallest RTT between those receivers and the source. Usually, MCB switches only once in many multiple RTT_{min}'s, and the amount of receivers behind the new MCB is much less than the overall number. Moreover, RTT_{min} is almost a negligible duration. Consequently, our feedback suppression mechanism is effective.

Finally, we must say that due to measurement errors in reality, the total number of CI(μ)s sent can be a little higher than what we have derived here. However, the difference won't be significant.

[7]Considering the reality, we can assume that MCB switching does not occur too frequently and $1/\lambda \geq RTT_{min}$.

V. SIMULATIONS

We have also run NS-2 [1] simulations to test different aspects of ORMCC, which include: (1) TCP-Friendliness, (2) Drop-to-zero avoidance, (3) Multiple bottleneck fairness, (4) Slowest receiver tracking, and (5) Feedback suppression. We have also compared the performance of our scheme with TFMCC in a dynamic heterogeneous network. In those simulations, the data packet size is 1000 bytes, the bottleneck buffer size is 50K bytes, the initial RTT is 200 milliseconds.

A. TCP-Friendliness and Drop-To-Zero Avoidance

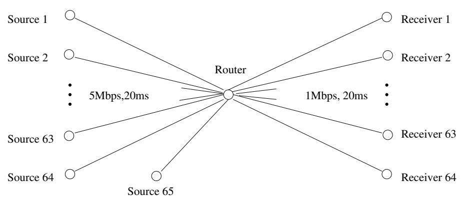

Fig. 7. 64-RECEIVER STAR TOPOLOGY WITH TCP BACKGROUND TRAFFIC

In the star topology with 129 end nodes (Figure 7), between each pair of source i and receiver i ($i = 1 \ldots 64$), there are one TCP Reno flow and one single-receiver ORMCC flow. Furthermore, there is a multi-receiver ORMCC flow from source 65 to all 64 receivers. Therefore, on the paths between the router and the receivers, there are asynchronous packet losses.

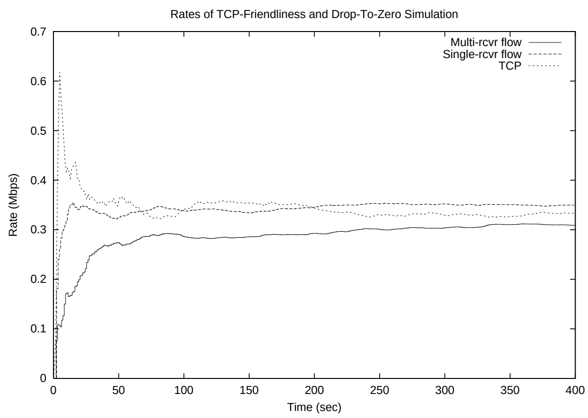

Fig. 8. ORMCC IS TCP-FRIENDLY AND IMMUNE TO DROP-TO-ZERO

In Figure 8, the single-receiver ORMCC flow and TCP flow are randomly chosen. we can see that the throughputs of ORMCC flows are close enough to that of TCP so that we can say that ORMCC is TCP-friendly. Furthermore, the throughput rate of multi-receiver ORMCC flow and that of single-receiver one are similar, both over 0.3Mbps, which indicates that ORMCC does not suffer from drop-to-zero problem.

B. Multiple Bottlneck Fairness

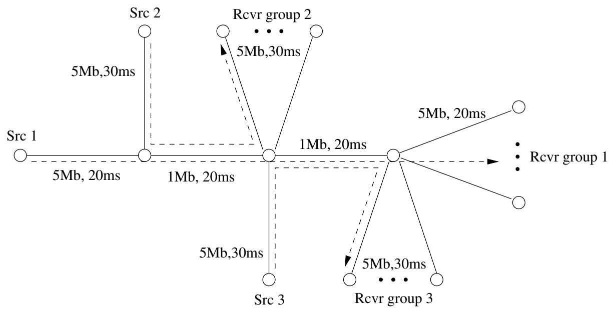

Fig. 9. LINEAR NETWORK WITH MULTIPLE BOTTLENECKS (TOTALLY 48 RECEIVERS)

Different notions of fairness define how the bottleneck bandwidth is shared by through traffic. To check the fairness our scheme can achieve, we ran a simulation on Figure 9. RED queues are used on the routers to eliminate the effect of RTT estimation. There is one multi-receiver multicast flow from Src 1 to receivers in Group 1, two single-receiver multicast flows from Src 2 to Group 2 and from Src 3 to Group 3 respectively. Each group has 16 receivers. According to proportional fairness, the long ORMCC flow should get one-third of the bottleneck bandwidth, 0.33Mbps. The result in Figure 10 shows that ORMCC achieve proportional fairness.

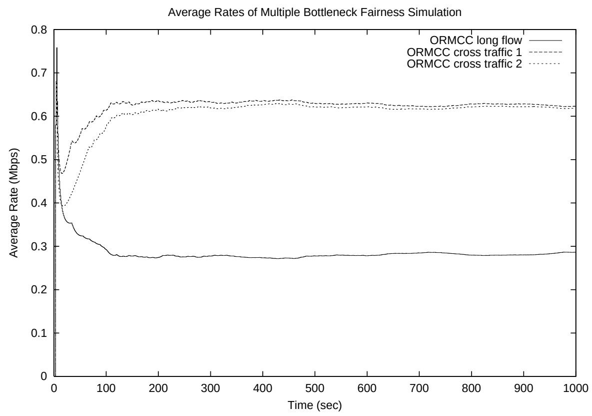

Fig. 10. ORMCC ACHIEVES PROPORTIONAL FAIRNESS. 0.33Mbps IS THE OPTIMAL RATE OF THE LONG FLOW.

C. Slowest Receiver Tracking

This simulation is used to test ORMCC's capability to quickly track the slowest receiver and select it as CR. In the tree topology of Figure 11, there is an ORMCC flow between the source and all the 32 receivers. Bottlenecks are dynamically generated, the most congested path and thus the slowest receiver also change on fly, as shown in Table I. The changes include both conditions causing CR switch, i.e. (1) A slower receiver appears, (2) The current slowest receiver is absent. Simulation result is shown in Figure 12. Vertical lines in it show when the ORMCC source switched CR. We can see that ORMCC update CR and adapt its transmission rate in a timely manner.

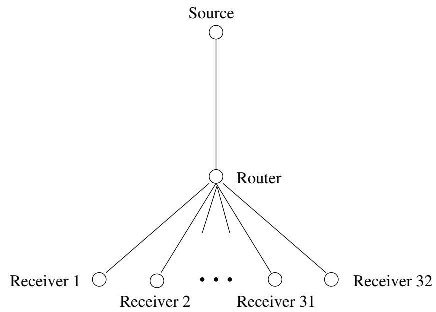

Fig. 11. ONE-LEVEL TREE WITH 32 RECEIVER NODES

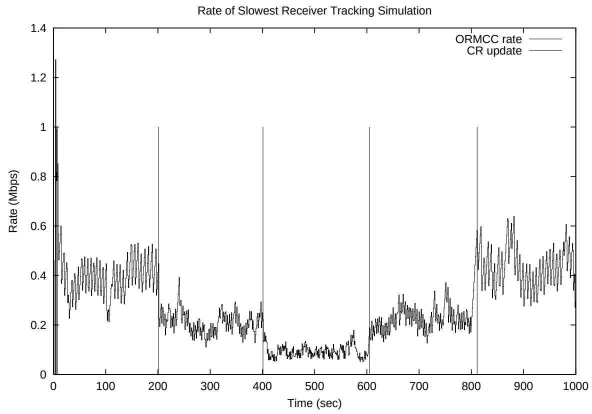

Fig. 12. ORMCC QUICKLY TRACKS THE SLOWEST RECEIVER

D. Feedback Suppression

To check the effectiveness of the feedback suppression mechanism in ORMCC, we refer back to the simulation of TCP-friendliness and drop-to-zero avoidance. In totally ten simulations, the average total number of CI(μ)s sent is 816 (standard deviation is 14.8), the average total number of suppressed CI(μ)s is 34601 (standard deviation is 422.0). The average number of CI(μ)s would have been sent by each receiver if without suppression, is $(34601 + 816)/64 \approx 553$. As we discussed in the analysis, realistic measurement error can lead to a little bit more CI feedbacks. Since $816 < 2 \times 553$, the deviation is ignorable. The high ratio of CI(μ)s suppressed, $34601/(34601 + 816) \times 100\% \approx \mathbf{97.7}\%$, shows that *our feedback suppression is very effective.*

TABLE I

DYNAMICS IN SLOWEST RECEIVER TRACKING SIMULATION

<table>
<tr><td></td><td>0 - 200 sec</td><td>200 - 400 sec</td><td>400 - 600 sec</td><td>600 - 800 sec</td><td>800 - 1000 sec</td></tr>
<tr><td rowspan="3">TCP Reno flows generating bottlenecks</td><td colspan="5">1 TCP Reno Flow shares Path 1 with ORMCC flow</td></tr>
<tr><td></td><td colspan="3">3 TCP Reno Flows shares Path 2 with ORMCC flow</td><td></td></tr>
<tr><td colspan="2"></td><td>7 TCP Reno Flows shares Path 2 with ORMCC flow</td><td colspan="2"></td></tr>
<tr><td>Slowest Receiver</td><td>Receiver 1</td><td>Receiver 2</td><td>Receiver 3</td><td>Receiver 2</td><td>Receiver 1</td></tr>
<tr><td>Most congested path</td><td>Path 1</td><td>Path 2</td><td>Path 3</td><td>Path 2</td><td>Path 1</td></tr>
<tr><td>Average bandwidth share for each flow on the most congested path</td><td>0.5Mbps</td><td>0.25 Mbps</td><td>0.125Mbps</td><td>0.25Mbps</td><td>0.5Mbps</td></tr>
</table>

(Path i is the path between the router and receiver i, $i = 1 \ldots 3$.)

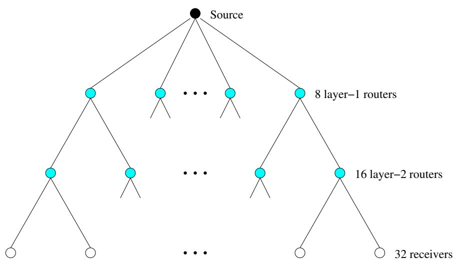

Fig. 13. HETEROGENEOUS DYNAMIC NETWORK

E. Comparision with TFMCC on Heterogeneous Dynamic Network

As the last simulation, we construct a dynamic network to test the stability and adaptability of ORMCC, and compare the performance of our scheme with TFMCC[17]. In Figure 13, each link has 2Mbps bandwidth. 6 links at the level, 12 links at the second level, and 24 links at the third level has 200ms delay. All other links have 20ms delay. However, on any path between the source and a receiver, there is at most one link of 200ms delay. On each link, 3 TCP Reno flow are randomly turned on and off according to exponential distribution with expected value as 60 seconds. These TCP flows dynamically generate bottlenecks and make the network heterogeneous. At last, there is a multicast flow from the source to all the receivers. The multicast flow can use either ORMCC or TFMCC.

We ran 10 simulations for each of the two schemes. In the following table we can see that with similar amount of feedbacks, ORMCC can achieve much higher throughput. That means, in dynamic environment, which indicates that ORMCC has better stability and adaptability in heterogeneous and dynamically changing networks.

Nonetheless, we notice that the throughput rates of both schemes are quite low. That is because TCP has slow start phase that can consume bandwidth quickly and generate bottlenecks of small bandwidth. Furthermore, both ORMCC and TFMCC are single rate multicast congestion control schemes. They track the slowest receivers, thus get low throughput.

	ORMCC	TFMCC
Average Throughput	63 Kbps	9.2Kbps
Througput Std. Dev.	16 Kbps	0.36 Kbps
Average Feedback Number	782	685.7
Feedback Number Std. Dev.	180.9	43.2

VI. CONCLUSION

We have proposed a single-rate multicast congestion control scheme in this paper. It uses an conventional concept of representative (we call it *Congestion Representative (CR)*. However, by leveraging a novel metric *TRAC*, it is simple while still being able to effectively address the problems of TCP-friendliness, drop-to-zero, slowest receiver tracking and feedback suppression. The numbers of states maintained by source and receivers are $O(1)$; operations of source and receivers are all simple without requiring intense computation, in particular there is no need to measure RTTs between all receivers and the source; non-probabilistic-timer-based feedback suppression is highly effective. Theoretical analysis and simulations both confirm the performance of ORMCC.

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] J. Byers, M. Luby, M. Mitzenmacher, "Fine-Grained Layered Multicast", *Infocom 2001*

[3] J. Byers, G. Kwon, "STAIR: Practical AIMD Multirate Multicast Congestion Control", *NGC 2001*

[4] Dante DeLucia, Katia Obraczka, "A Multicast Congestion Control Mechanism Using Representatives", *Proceedings of the IEEE ISCC 1998*

[5] A. Fei, J. Cui, M. Gerla, M. Faloutsos, "Aggregated Multicast: an Approach to Reduce Multicast State", *Globecom 2001*

[6] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, Lixia Zhang, “A Reliable Multicast Framework for Light-wight Sessions and Application Level Framing”, *IEEE/ACM Transactions on Networking*, Vol. 5(6): 784-803, Dec. 1997.

[7] 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.

[8] J. C. Lin, S. Paul, “RMTP: A reliable multicast transport protocol”, *Infocom 1996*, March 1996

[9] J. Macker, R. Adamson, “A TCP Friendly, Rate-Based Mechanism for Nack-Oriented Reliable Multicast Congestion Control”, *Globecom 2001*

[10] 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.

[11] T. Nguyen, K. Nakauchi, M. Kawada, H. Morikawa, T. Aoyama, “Rendezvous Points Based Layered Multicast”, *IEICE Trans. Commun.*, Vol. E84-B, No. 12, Dec. 2001.

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

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

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

[15] Puneet Sharma, Deborah Estrin, Sally Floyd, Van Jacobson, “Scalable Timers for Soft State Protocols”, *Proceedings of IEEE INFOCOM*, Apr. 1997.

[16] Puneet Thapliyal, Sidhartha, Jiang Li, Shivkumar Kalyanaraman, “LE-SBCC: Loss-Event Oriented Source-Based Multicast Congestion Control”, *Multimedia Tools and Applications*, Vol. 17, No. 2-3, pp. 257-294, Jul. - Aug. 2002.

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

APPENDIX
ALGORITHM

A. Source Operations

Variables :

λ : Transmission rate
μ : Throughput rate at congestion (TRAC) in the received CI(μ)
$E(\mu^{cr})$: Average TRAC of the CR
μ_σ^{cr} : Deviation of the CR TRAC
s : Packet size
RTT_{max} : Maximum RTT
RTT_{cr} : RTT between the source and the CR
T^{cr} : CR response time when the bottleneck is fully loaded
$E(T^{cr})$: Average of T^{cr}
T_σ^{cr} : Deviation of T^{cr}
cr_valid : Indicates whether the CR is valid
R : The receiver sending the received CI(μ)

Ininitialization:
cr_valid = **false**
$RTT_{max} = 0$

Event every RTT_{cr}**:**
if *There is no rate reduction within the recent* RTT_{cr} **then**
$\lambda \leftarrow \lambda + s/RTT^{cr}$
if $\lambda \geq E(\mu^{cr}) + 4\mu_\sigma^{cr}$ **and** *CR checking timer is not running* **then**
Start CR checking timer with time length $E(T^{cr}) + 8T_\sigma^{cr}$
$t \leftarrow$ the current time
endif
endif

Event when the CR checking timer expires
cr_valid $\leftarrow$ **false**

Subroutine : CutRate ()
if λ *has not been cut within the most recent* RTT_{cr} **then**
$\lambda \leftarrow \min(\lambda, 0.75\mu)$
Stop CR checking timer
endif

Event upon receipt of CI(μ**):**
if R *is CR* **then**
cr_valid $\leftarrow$ **true**
if *CR checking time is running* **then**
$\Delta \leftarrow$ current time $- t$
Update $E(T^{cr})$ and T_σ^{cr} with Δ
endif
Update $E(\mu^{cr})$ and μ_σ^{cr} with μ
Update RTT_{cr}
if $RTT_{max} < RTT_{cr}$ **then**
$RTT_{max} \leftarrow RTT_{cr}$
endif
Stop CR checking timer
do CutRate ()
return
endif

/* The CI(μ) is NOT from CR if reach here */
if *cr_valid is* **false then**
Choose R as the CR
Start CR grace period as $2RTT_{max}$
else if *In CR grace period* **then**
if *The RTT sample measured by this CI(*μ*) is larger than* RTT_{cr} **then**
Choose R as the CR
endif
/* NOT in CR grace period */
else if $\mu < E(\mu^{cr}) - \mu_\sigma^{cr}$ **then**
Choose R as the CR
endif

if *CR has been changed at the receipt of this CI(*μ*)* **then**
cr_valid $\leftarrow$ **true**
Update $E(\mu^{cr})$ and μ_σ^{cr} with μ
Update RTT_{cr}
if $RTT_{max} < RTT_{cr}$ **then**
$RTT_{max} = RTT_{cr}$
endif
do CutRate ()
endif

B. Receiver Operations

Variables :

μ : A throughput rate at congestion (TRAC) sample
$E(\mu)$: Average TRAC of this receiver
$E(\mu^{cr})$: Average TRAC of the CR
μ_σ^{cr} : Deviation of the CR TRAC

Event upon receipt of a packet:
if $E(\mu^{cr})$ *and* μ_σ^{cr} *has been changed* **then**
Update the local copy of $E(\mu^{cr})$ and μ_σ^{cr}
endif
if *This packet indicates packet losses* **then**
Meaure μ and update $E(\mu)$
if $E(\mu^{cr})$ *and* μ_σ^{cr} *are invalid* **or** $E(\mu) < E(\mu^{cr}) - \mu_\sigma^{cr}$ **then**
Send CI(μ)
endif
endif