A New Fair Bandwidth Allocation Algorithm for Multimedia Multicasting in DiffServ

Lie Qian¹, Yiyan Tang¹, Yuke Wang¹, Bashar Bou-Diab², and Wladek Olesinski²

1. Department of Computer Science
   University of Texas at Dallas
   Richardson, TX 75080 USA
   {lqian, yiyian, yuke}@utdallas.edu

2. Research and Innovation, Alcatel
   Ottawa, Ontario, K2K 2E6, Canada
   {Bashar.Bou-Diab, Wladyslaw.Olesinski } @alcatel.com

Abstract— Multicast is a bandwidth efficient mechanism for delivering the same data to multiple receivers simultaneously. Layered multicast, which is suited for high-bandwidth multimedia traffic like video, may be used to differentiate between receivers of the same multimedia content based on their processing and bandwidth resources. To achieve fair bandwidth allocation in layered multicast, max-min fair allocation solutions have been proposed. However, existing solutions either use non-scalable centralized computation or require core routers to maintain per-session information. Maintaining per-session information in core routers violates the core stateless principle of DiffServ architecture, which is proposed as a scalable QoS solution in the Internet. In this paper, we propose a new scalable distributed max-min fair bandwidth allocation algorithm for DiffServ, which runs iteratively and does not maintain per-session information in core routers. The new algorithm is proved capable of achieving max-min fair bandwidth allocation. Through analysis, we show that the new algorithm complies with core stateless DiffServ architecture to have O(1) storage complexity in core routers.

I. INTRODUCTION

Multicast is an efficient mechanism for delivery of data from one or multiple sources to a set of destinations (receivers) identified by a multicast group [1]. In this context, we consider a single source multicast only and define a multicast flow as a collection of packets identified by the same (S, G) address pair [2][3], where S is the source’s network address and G is a multicast group network address. Multicast routing protocols establish a logical multicast tree that is rooted at the source of a multicast flow and has the flow destinations as leaves [2]. Packets of the multicast flow are delivered from the source to destinations following the multicast tree.

Transmitting multimedia traffic over a best-effort network, such as the Internet, yields an unreliable multimedia service that is vulnerable to transient network congestions. Quality of Service (QoS), which refers to the capability of a network to take into account traffic characteristics and requirements while transporting the traffic, is a mechanism to provide traffic isolation and guarantees on traffic delivery metrics during network congestion. Network QoS architectures have been proposed to provide network performance guarantees. The Integrated Services (IntServ) [4] architecture aims to provide absolute QoS guarantees through per-flow resource reservation in all routers in the network. IntServ is not scalable for large scale networks. Differentiated Services [5] (DiffServ) aggregates traffic into classes and does not require per-flow resource reservation in core routers.

Today’s Internet is a heterogeneous network, in which the paths to different destinations have different bandwidths and the destination hosts have different processing capacities. It is desirable in such a heterogeneous network to fairly allocate network bandwidth on links among different multicast sessions. Here, a session consists of the traffic from the same source to multiple destinations serving the same application or multimedia content, such as a clip of video or audio. Fair bandwidth allocation for multicast has two goals: (1) every destination of the same multicast session receives data at a rate commensurate with its processing capabilities and the bandwidth of the path leading to it (intra-session fairness), and (2) the bandwidth on links is allocated fairly between different multicast sessions (inter-session fairness) [6][7].

Layered multicast has been proposed to achieve intra-session fairness for multimedia content [8-10]. With layered multicast, multimedia signals of a session are encoded into multiple layers, each layer adding to the quality of the multimedia session. The destinations adapt to congestion and bandwidth capacity by requesting different number of layers. A session with layered traffic could be served by multiple multicast flows, one multicast flow for each layer.

To evaluate inter-session fairness, max-min fairness is a widely accepted fairness criterion [7,11-18]. Many solutions have been proposed to achieve max-min fairness [7,11-13]. The existing solutions either rely on unscalable centralized computation or utilize per-session state information in core routers. In this paper, we propose a new scalable fair bandwidth allocation algorithm to achieve max-min fairness in the DiffServ framework. The new algorithm has no per-session state information maintained in core routers and distributes the computation among the Bandwidth Broker (BB), edge routers and core routers to improve scalability. The correctness of the new algorithm is proved and its complexity is analyzed.

This paper is organized as follows. In Section II, we review the network model, DiffServ, max-min fairness, and existing bandwidth allocation algorithms. The new algorithm is presented in Section III, proved in Section IV and analyzed in Section V. The conclusion is given in Section VI.

II. BACKGROUND

In this section, we review the network model, DiffServ QoS architecture, the max-min fairness criterion and related works in max-min fair bandwidth allocation.

A. Network Model

Routers in a network are categorized into edge routers and core routers. Edge routers are the points connected with sources/destinations. Core routers connect edge routers in the same network. A multicast session has one Source Designated Router (SDR) connected to the source and multiple Destination Designated Routers (DDRs) that are connected to

* This work is sponsored by Alcatel.
the destinations. Traffic of one session is transported in one or multiple multicast groups (for layered multicast) from a SDR to its DDRs through one or multiple multicast trees built by a multicast routing protocol, such as Protocol Independent Multicast (PIM) [19].

Following the definition in [7], we call every pair of SDR and DDR in a multicast session a Virtual Session (VS). A VS $j$ has a SDR $\text{Src}(j)$, a DDR $\text{Dest}(j)$ and a minimum rate requirement $\lambda_j$. We assume that the traffic for VS $j$, which can be composed of traffic from different layers delivered by different multicast groups, follows the same path path$(j)$ from $\text{Src}(j)$ to $\text{Dest}(j)$.

### B. DiffServ QoS Architecture

Unlike IntServ [4] that has undesirable scalability limitation in core routers, DiffServ [5] offers a scalable QoS architecture by aggregating traffic from different flows at the network edge into classes identified by DiffServ Code Points (DSCP) encoded in IP packets’ headers. In DiffServ, edge routers may maintain per-flow state information to classify and mark incoming packets with proper DSCP values. Core routers do not maintain any per-flow state information but process packets on a per-class basis according to their DSCP classification. Bandwidth Broker (BB) [20] is a dedicated node that performs functions such as admission control, policy management, lightweight bandwidth reservation and so on in a DiffServ network.

### C. Max-Min Fair Bandwidth Allocation

Layered transmission schemes [8][10], which are proposed to multimedia traffic like video [21], are used to achieve intra-session fairness in heterogeneous networks. In layered transmission, a multimedia signal is encoded into multiple digital layers that can be incrementally combined to provide progressive refinement [16]. The destinations adapt to bandwidth by receiving different numbers of layers.

Given a network topology, existing sessions’ membership, path$(j)$ and minimum rate requirement $\lambda_j$ of each VS $j$ in the network, a bandwidth allocation process determines one rate $r_j$ for each VS $j$ in the network. After bandwidth allocation, $\text{Dest}(j)$ subscribes to proper number of layers in $j$’s session to receive traffic at rate $r_j$. Under a rate allocation, if session $i$ contains $n$ VSs on link $l$, say $\{VS1, VS2, \ldots, VSn\}$, the rate allocated to session $i$ on link $l$ is denoted as $\lambda_{\alpha}=\max\{r_{VS1}, r_{VS2}, \ldots, r_{VSn}\}$. A rate allocation is feasible, if the rate allocated for every VS $j$ is equal to or larger than $\lambda_j$ and $\sum_{i\in C_l}C_j$ for all links in the network, where $C_j$ is the bandwidth capacity of link $l$ or the bandwidth reserved to multicast sessions in the same DiffServ class.

The definition of max-min fair rate allocation is a widely accepted fairness criterion [7][11-18] to evaluate inter-session fairness. A rate allocation is max-min fair if 1) the allocation is feasible and 2) it is not possible to maintain its feasibility while increasing the rate of a VS without decreasing that of any other VS which is allocated equal or lower rate [7].

The example depicted in Figure 1 illustrates max-min fairness. In the example, two multicast sessions, session 1 and 2, have the same SDR $\text{Src}$ and VSs have their minimum rate requirements shown in Figure 1. There are 8 links, $L_1$ to $L_8$, with capacities $C_1$ to $C_8$. The max-min fair bandwidth allocation algorithm determines the max-min fair rate allocation $\{r_{VS1}, r_{VS2}, r_{VS3}, r_{VS4}\}$. There could be multiple feasible rate allocations, but it has been proved in [22] that the max-min fair rate allocation is unique, which is $(4\text{Mb}, 4\text{Mb}, 1\text{Mb}, 2\text{Mb})$ in this example. Under this max-min fair rate allocation, $r_{VS1}$, $r_{VS2}$, $r_{VS3}$, and $r_{VS4}$ could not increase due to the feasibility constraint. Increasing $r_{VS4}$ will decrease $r_{VS3}$, whose rate is smaller than $r_{VS4}$ (Definition of Max-min fairness). In this paper, we use $r_{j,max-min}$ to denote the rate assigned to VS $j$ under the unique max-min fair bandwidth allocation.

![Figure 1 Example of Max-min Fair Bandwidth Allocation](image)

**D. Related Works in Max-Min Fair Bandwidth Allocation**

Solutions have been proposed to achieve max-min fairness for multicast traffic [7,11-13]. Back pressure based multicast scheduling proposed in [12] achieves fairness through specially designed scheduling in routers and is not scalable because of its per-flow scheduling in core routers. The centralized algorithm in [11] computes long term max-min fair rates for destinations, and does not scale well with large number of sessions. The distributed algorithm proposed in [7] requires neither centralized rate computation nor per-flow scheduling. The defined notations and the algorithm proposed in [7] are presented in Table 1 and 2.

In the distributed algorithm proposed in [7], every router in the network maintains the information of passing multicast sessions including their saturation statuses, allocated bandwidth, and minimum rate requirements. The algorithm runs iteratively to find a rate allocation saturating all VSs. The definition of VS saturations is shown in Table 1. In $k$th iteration, routers compute the session Link Control Parameters (LCP) $\eta_{\alpha}(k)$ shown in Step S.3 for each link in the network. For each unsaturated VS $j$ in the network, a backward message from $\text{Dest}(j)$ to $\text{Src}(j)$ collects $\eta_{\alpha}(k)$ on path$(j)$ and computes $r_j(k)$ at $\text{Src}(j)$ in Step S.4. A forward rate message distributes the computed $r_j(k)$ to links on path$(j)$ for the computation of $\lambda_{\alpha}(k)$ in Step S.5 and notifies $\text{Dest}(j)$ of $r_j(k)$. When $r_j(k)$ is available at $\text{Dest}(j)$, a probe message is sent toward $\text{Src}(j)$ to collect the saturation status of $j$ on path$(j)$. This distributed algorithm runs iteratively until all VSs are saturated in Step S.7. It has been proved that the unique max-min fair bandwidth allocation saturates all VSs in the network. Under such bandwidth allocation, each VS $j$ in the network is allocated rate as $r_{j,max-min}$. The same authors also...
propose a discrete version algorithm in [13]. The existing distributed max-min fair bandwidth allocation algorithms in [7] and [13] require core routers to maintain per-session information for passing sessions, which prohibits them from being used in core-stateless QoS architectures like DiffServ.

\[ m(l) \text{ - Set of sessions passing through link } l \]

\[ m(k,l) \text{ - Set of virtual sessions of session } k \text{ passing through link } l \]

\( C_l \) - Capacity of link \( l \)

\( \Delta_l \) - Multicast session that the virtual session belongs to

\( M_l \) - Min. rate requirement for virtual session \( i \)

\( M_{\max} \) - Min. bandwidth requirement for session \( i \) on link \( l \)

\( l \) - Set of links traversed by virtual session \( i \)

\( r(k) \) - Rate allocated to session \( k \) on link at the end of the \( t \)th iteration.

\( r(k) \) - Rate allocated to session \( k \) on link at the end of the \( t \)th iteration.

\( \Delta_i(k) \) - Rate allocated to session \( k \) on link \( l \) at the end of the \( t \)th iteration.

\( \Delta_{\max} \) - Maximum rate amongst all VSNs of \( m(l) \) that is \( \sum_{i=1}^{m(l)} \Delta_i(k) = C_l \) and \( r(k) = \Delta_{\max}(k) \)

A session is saturated on link \( l \) if all the virtual sessions of this session traveling through the link are saturated.

\( S(k) \) - Set of unsaturated VSNs at the end of the \( t \)th iteration

\( \Xi(k) \) - Set of unsaturated sessions on link \( k \) at the end of the \( t \)th iteration.

\( F_l(k) \) - Total bandwidth consumed by saturated sessions on \( k \) at the end of the \( t \)th iteration

\( \eta(k) \) - Link control parameter of session \( k \) on link \( l \) at the end of the \( t \)th iteration.

\( \eta_l(k) \) - Session link control parameter of session \( k \) passing link \( l \)

\( \theta_l \) - A variable used in computing link control parameter \( \eta(k) \)

Table 1 Definition of Variables

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S.1</td>
<td>Initializing ( k = 0 ), ( \eta_l(0) = 0 ), ( F_l(0) = 0 ), ( S(0) = {1,2,\ldots,M} ), ( \Xi(0) = m(l) ) for all links ( l ), ( r(0) = \mu_l ) for all virtual sessions ( i ).</td>
</tr>
<tr>
<td>S.2</td>
<td>If ( k = k + 1 ) then goto step S.3.</td>
</tr>
<tr>
<td>S.3</td>
<td>For each link ( l ) in the network compute the link control parameter ( \eta_l(k) ) if ( \Xi(k-1) \neq \emptyset ) then ( \eta_l(k) ) is the maximum possible ( \eta_l ) which satisfies ( F_l(k-1) + \sum_{i=1}^{m(l)} \Delta_i(k_l) = C_l ), else if ( \Xi(k-1) = \emptyset ), ( \eta_l(k) = \eta_l(k-1) ).</td>
</tr>
<tr>
<td>S.4</td>
<td>Compute ( r(1) ) for all virtual session ( i ) where ( r(1) = \min { \eta_l(k), \mu_l } ).</td>
</tr>
<tr>
<td>S.5</td>
<td>For each link ( l ) in the network compute the rate ( r(1) ) on link ( l ) for every session on this link in ( m(l) ), ( \Delta_i(k_l) = \max { \eta_l(k), \mu_l } ).</td>
</tr>
<tr>
<td>S.6</td>
<td>Compute the set of unsaturated virtual sessions after ( k )th iteration, ( S(k) = S(k-1) - { x : \exists l \in L \text{ s.t. } \sum_{i=1}^{m(l)} \Delta_i(k_l) = C_l \text{ and } r(k) = \Delta_{\max}(k) } ).</td>
</tr>
<tr>
<td>S.7</td>
<td>If ( S(k) ) is empty, i.e., all virtual sessions are saturated, terminate the algorithm and ( r(k) ) is the final rate allocation for all virtual sessions.</td>
</tr>
<tr>
<td>S.8</td>
<td>For each link ( l ), compute the set of unsaturated sessions on link ( l ) at the end of the ( k )th iteration, ( \Xi(k) = { i : n \in \Xi(k) \text{ s.t. } m(n) \cap S(k) \neq \emptyset } ).</td>
</tr>
<tr>
<td>S.9</td>
<td>For each link ( l ), where ( \Xi(k) \neq \emptyset ), compute the bandwidth consumed by the saturated sessions passing through link ( l ), ( F_l(k) = \sum_{i=1}^{m(l)} \Delta_i(k_l) ).</td>
</tr>
<tr>
<td>S.10</td>
<td>Go to step S.2.</td>
</tr>
</tbody>
</table>

Table 2 A Distributed Max-min Fair Bandwidth Allocation Algorithm

![Figure 2 Data Structure in the BB](image)

A. Description of the New Algorithm

Our new algorithm runs iteratively and achieves max-min fair bandwidth allocation when it terminates. The new algorithm can be described in 8 steps as following.

Step 1. Algorithm initialization

The BB invokes the algorithm by sending algorithm starting signal to SDRs and DRRs. The BB resets \( REC_{BB,i} \) for all links in the BB by setting \( r_i = \mu_i \) and setting saturation statuses of all sessions and VSNs to “unsaturated”.

Step 2. LCP computation and distribution

At the beginning of \( k \)th iteration, the BB computes the LCP \( \eta(k) \) for each link \( l \) in the network as the maximum \( \theta_l \) that satisfies (1) based on information in \( REC_{BB,i} \). The BB sends the LCP \( \eta(k) \) to link \( l \)’s upstream router.

\[ F_l(k) = F_l(k-1) + \sum_{i=1}^{m(l)} \max \{ \theta_i(k), \mu_i \} = C_l \] (1)

Step 3. Backward rate message processing at routers

At the beginning of \( k \)th iteration, for each unsaturated \( VS_j \) in the network, \( Dest_j \) sends a backward rate message, denoted as \( BM_j(k) \), toward \( Src_j \) along the reverse path of \( path_j \). Initially, two rates are stored in \( BM_j(k) \) as (2) and (3).

\[ r_j^{\text{cur}}(k) = \text{min}(r_j^{\text{cur}}(k), \text{max}(r_j^{\text{cur}}(k), r_j^{\text{pre}}(k))) \] (2)

\[ r_j^{\text{cur}}(k) = \text{min}(r_j^{\text{cur}}(k), \text{max}(r_j^{\text{cur}}(k), r_j^{\text{pre}}(k))) \] (3)

where \( r_j \) is the LCP received from the BB. \( R \) adds link \( l \) into \( BM_j(k) \) with a temporary session link control parameter value \( r_j^{\text{back},LCP}(k) = \max \{ \eta_j(k), r_j^{\text{pre}}(k) \} \). The updated \( BM_j(k) \) is then forwarded toward \( Src_j \).

Step 4. Backward rate message at SDR

When router \( Src_j \) receives \( BM_j(k) \), \( Src_j \) updates rate records for session \( j \) as (3) and (4).

\[ r_j^{\text{alloc},LCP}(k) = r_j^{\text{alloc},LCP}(k) \vee \text{listed in } BM_j(k) \] (3)

\[ r_j^{\text{SLCP}}(k) = \max(r_j^{\text{SLCP}}(k), r_j^{\text{back},LCP}(k)) \vee \text{listed in } BM_j(k) \] (4)

Step 5. Forward rate message generation

In each iteration \( k \), router \( v \), the SDR of session \( i \), generates one forward rate message \( FM_k(k) \). For each received \( BM_j(k) \) in session \( i \), i.e., \( i = j \), \( v \) adds one forward record, called
FRecord and denoted as FR\(_{i}(k)\), into FM\(_{i}(k)\). FR\(_{i}(k)\) contains two rate values, \(r_{i,l}^{\text{backward}}(k)=r_{i,l}^{\text{cur}}(k)\) and \(r_{i,l}^{\text{forward}}(k)=\infty\) when SDR \(v\) receives BM\(_{l}(k)\) from all unsaturated VSs of session \(l\). \(v\) adds updated \(r_{i,l}^{\text{SLCP}}(k)\) list in the rate records of session \(i\) into FM\(_{i}(k)\). Meanwhile, \(v\) sends the updated \(r_{i,l}^{\text{alloc}}(k)\) list in the rate records of session \(i\) to the BB. The BB uses the received \(r_{i,l}^{\text{alloc}}(k)\) list to update the session link rates for session \(i\) as \(r_{i,l}=\max(r_{i,l}, r_{i,l}^{\text{alloc}}(k))\) on link \(l\). Then router \(v\) processes and forwards FM\(_{i}(k)\) as described in Step 6.

After the BB updates session link rates \(r_{i,l}\) for all unsaturated sessions, the “fully utilization status” for link \(l\), which indicates whether link \(l\) is fully utilized or not (if \(\sum_{i} r_{i,l}=C_{l}\)), will be sent to \(l\)’s upstream router by the BB.

**Step 6. Forward rate message processing at routers**

When the forward rate message FM\(_{i}(k)\) reaches router \(R\) for processing, the original FM\(_{i}(k)\) will be duplicated for each outgoing link that has VS of session \(i\) at downstream. For the duplicated message passing through link \(l\), denoted as FM\(_{i}(k)\), router \(R\) first updates each FR\(_{i}(k)\) in FM\(_{i}(k)\) as (5).

\[
 r_{i,l}^{\text{forward}}(k) = \min(r_{i,l}^{\text{forward}}(k), r_{i,l}^{\text{SLCP}}(k))
\]

Before \(R\) forwards FM\(_{i}(k)\) on \(l\), if \(l\) is fully utilized (notified by the BB), \(R\) performs saturation test for every unsaturated VS \(j\) in FM\(_{i}(k)\). In the saturation test, \(R\) flags the FRecords FR\(_{i}(k)\) in FM\(_{i}(k)\) as saturated if \(r_{i,l}^{\text{backward}}(k) = r_{j,l}^{\text{SLCP}}(k)\).

**Step 7. Forward rate message at DDRs**

When Dest\(_{j}\) receives FM\(_{i}(k)\), it uses \(r_{j,l}^{\text{forward}}(k)\) in the message as the rate allocated for current iteration, \(r_{j,l}=r_{j,l}^{\text{forward}}(k)\). Dest\(_{j}\) checks the saturation status of VS \(j\) in FM\(_{i}(k)\) and sends the status to the BB. If VS \(j\) is not saturated in FM\(_{i}(k)\), Dest\(_{j}\) sends out the BM\(_{j}(k+1)\) with \(r_{j,l}^{\text{prev}}(k+1)=r_{j,l}(k)\) as in Step 3. Otherwise Dest\(_{j}\) stops sending BM\(_{j}\) for VS \(j\) and \(r_{j,l}(k)\) is VS \(j\)’s final allocated rate, \(r_{j,l}^{\text{max-min}}\).

**Step 8. Start the new round of iteration**

After updating saturation status for all VSs in \(k\)th iteration, the BB updates the session saturation statuses in REC\(_{i}\) for all links. If all VSs in the network are saturated, the algorithm terminates. Otherwise, the BB computes LCPs \(\eta(k+1)\) and the algorithm goes back to Step 2.

**B. Example**

We use the example in Figure 1 to illustrate how the new algorithm works. In the 1st round, BB computes LCPs as:

\[
[\eta_{l1}(1),...\eta_{l8}(1)] = \{2Mb,2.5Mb,1Mb,4Mb,4Mb,4Mb,4Mb,4Mb\}
\]

Following Step 2 and 3, the \(r_{j,l}^{\text{outer}}(1)\) received by \(v\) are:

\[
[r_{j,l}^{\text{outer}}(1),...r_{j,l}^{\text{outer}}(1)] = \{4Mb,2Mb,1Mb,2Mb,2Mb\}
\]

The BB detects the fully utilized links \([L_{1},L_{3},L_{7}]\) in the first round. Saturation tests are performed on these links in Step 6. After the forward rate message processing in Step 5, 6, 7, the allocated rates received by the destinations of VSs are:

\[
[r_{j,l}^{\text{prev}}(1),...r_{j,l}^{\text{prev}}(1)] = \{4Mb,4Mb,4Mb,2.5Mb,2Mb\}.
\]

VS1, VS3, and VS5 are detected as saturated on links \([L_{1},L_{3},L_{7}]\), respectively. One thing worth mentioning here is that \(r_{j,l}^{\text{outer}}(1)\) is 4Mb, and it is saturated on link \(L_{1}\). However, it is not detected because the saturation test performed in Step 6 is based on \(r_{j,l}^{\text{backward}}(1)=2Mb\). This will be corrected in the 2nd iteration.

In the 2nd iteration, only \(d_{3}\), \(d_{4}\) send backward rate messages to \(v\). Computed LCPs are:

\[
[\eta_{l1}(2),...\eta_{l8}(2)] = \{4Mb,3Mb,4Mb,4Mb,4Mb,4Mb,4Mb,4Mb\}
\]

The \(r_{j,l}^{\text{prev}}(2)\) received by \(v\) are \(r_{j,l}^{\text{outer}}(2)=4Mb\), \(r_{j,l}^{\text{outer}}(2)=3Mb\). The allocated rates received by the destinations are \(r_{j,l}^{\text{prev}}(2)=4Mb\), \(r_{j,l}^{\text{prev}}(2)=3Mb\) and both VSs are detected as saturated. The algorithm terminates with max-min fair allocation \([r_{j,l}^{\text{prev}}] = \{4Mb,4Mb,1Mb,3Mb,2Mb\}\).

**IV. CORRECTNESS PROOF**

In this section, we prove that the new algorithm can terminate at the unique max-min fair bandwidth allocation. Under the max-min fair bandwidth allocation, all VSs are saturated and each VS \(j\) is allocated with \(r_{j,l}^{\text{max-min}}\). In order to prove the correctness, we first have four lemmas as following.

**Lemma 1:** The link control parameters on the same link and the rates allocated to the same VS are non-decreasing in the new algorithm:

\(r_{j,l}(k+1)\geq r_{j,l}(k)\) and \(\eta(k+1)\geq\eta(k)\).

**Lemma 2:** If \(r_{j,l}^{\text{prev}}(k)\) is true for all VS \(j\), then \(r_{j,l}^{\text{prev}}(k+1)\).

**Lemma 3:** In the new algorithm, the undetected saturated VSs and undetected fully utilized links at \(k\)th iteration will be detected in the \((k+1)\)th iteration.

**Lemma 4:** In the new algorithm, the saturated VS will not become unsaturated again and its allocated rate will not increase in subsequent iterations even if its DDR sends a backward rate message.

From all four Lemmas above, we can prove Theorem 1.

**Theorem 1:** The new algorithm saturates every VS \(j\) at rate \(r_{j,l}^{\text{max-min}}\) and achieves the unique max-min fair bandwidth allocation after termination.

**Proof of Theorem 1:** From Lemma 1 and Lemma 2, we conclude the new algorithm gets the allocated rates of each unsaturated VS \(j\) closer to their max-min fair rates \(r_{j,l}^{\text{max-min}}\) iteration by iteration, but \(r_{j,l}^{\text{max-min}}\) could not be exceeded. Due to the uniqueness of the max-min fair bandwidth allocation, which is the allocation that saturates all VSs, The latency in the detection of saturated VSs will not affect other saturated VSs.

**3VSs only saturate at the rate of \(r_{j,l}^{\text{max-min}}\).** A VS \(t\) saturated at a rate smaller than \(r_{j,l}^{\text{max-min}}\) requires another VS \(j\) saturated at a rate greater than its max-min rate \(r_{j,l}^{\text{max-min}}\). This, however, is not possible because of Lemma 4 and Lemma 2.

**4.Because the undetected saturated VS will be always detected at most one round of iteration later (Lemma 3), the rate of the VSs that are delayed in saturation will increase**
and finally saturate after all undetected saturated VSs are detected (Lemma 2).

Thus, the delay in saturation detection will not prevent any VS from becoming saturated at the rate $r_{\text{max-min}}$ and reach the unique max-min fair bandwidth allocation. □ End of proof

V. COMPLEXITY DISCUSSION

Our new max-min fair bandwidth allocation algorithm does not require core routers to store per-session information and avoids centralized computation. Complexities of the whole algorithm and complexities in one iteration are shown in Table 3, where $M$ is the total number of VSs and $N$ is the total number of sessions in a given network.

<table>
<thead>
<tr>
<th>Complexity of the Whole Algorithm</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Max # of Iterations</td>
<td>$M$</td>
<td></td>
</tr>
<tr>
<td>Storage</td>
<td>$O(M)$</td>
<td>$O(N)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Complexity of One Iteration</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># of msg round trips</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td># of backward rate msgs processed</td>
<td>$O(M)$</td>
<td>$O(M)$</td>
</tr>
<tr>
<td># of forward rate msgs processed</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td># of msgs from routers to BB</td>
<td>$O(M)$</td>
<td></td>
</tr>
<tr>
<td># of link control parameters computed</td>
<td>$O(1)$</td>
<td></td>
</tr>
</tbody>
</table>

Table 3 Complexity of the New Algorithm

Following Lemma 2, it is easy to show that at least one VS is saturated and detected in each round of iteration. Thus, the new algorithm terminates after at most $M$ rounds of iteration. In each round of iteration, the distributed algorithm requires only one round trip message exchanges between DDRs and SDRs and requires two round trip message exchanges from the BB to routers.

Core routers in our algorithm do not maintain per session or per VS information, but only keep one link control parameter for every link and have the storage complexity $O(L)$. As the principle of DiffServ QoS architecture, the complexity of our algorithm is pushed into the BB and the edge routers.

One communication bottleneck exists in the saturation notification from DDRs to the BB, which can be $O(M)$ in one iteration. One solution is to let SDRs collect VS saturation status from DDRs and send them in one message to BB. This way, the number of message round trips in one iteration becomes 1.5 and the number of messages from routers to BB is reduced to $O(N)$ in one iteration.

VI. CONCLUSIONS

In this paper, we propose a novel scalable max-min fair bandwidth allocation algorithm that can be deployed in DiffServ QoS architecture for multicast traffic. Our new algorithm relieves core routers from per-session information storage to comply with DiffServ’s core stateless principle. The new algorithm also avoids centralized computation through distributing the computation among all routers to improve the scalability. The new algorithm is proved to be able to reach max-min fair bandwidth.

REFERENCES