1998 International Computer Symposium
Workshop on Computer Networks, Internet, and Multimedia
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

A SCHEME TO ACHIEVE WEIGHTED MAX-MIN FAIRNESS ALLOCATION FOR HIGH SPEED NETWORKS

Yen-Jen Chen and Suh-Yin Lee
Department of Computer Science and Information Engineering,
National Chiao Tung University, Hsin-Chu, Taiwan, R.O.C.
Email: {ctchen, sylee}@csie.nctu.edu.tw

Abstract

In this paper, a scheme is proposed to achieve the weighted max-min fairness on bandwidth allocation in networks. Since allocation is done according to the weights of all the user flows, the scheme supports various bandwidth requirements of users. The scheme can achieve the fairness allocation rapidly once the network condition changes. Before the fairness allocation is achieved, the switches an active flow passes through may allocate individual bandwidth shares to the flow. With the proposed scheme, for this active flow the smallest share can be quickly found and is allocated. For each active flow if its bandwidth shares at different switches converge to the smallest share, the weighted max-min fairness is achieved. The correctness has been proved in this paper. On the other hand, the scheme is efficient since each switch takes O(log N) to allocate bandwidth to N active flows.

Keywords: Flow control; Bandwidth allocation; Weighted max-min fairness; Constrained flow; Red black tree

1. Introduction

One of the most important issues of flow control for networks is to allocate the available bandwidth to user flows fairly. Each of the flows, assumed with a fixed route, would set its source rate to the allocated bandwidth to avoid network overflow. Two important criteria for the bandwidth allocation are the max-min fairness [1][2] and efficiency. The goal of the max-min fairness is to maximize the smallest flow bandwidth, and once this is achieved then next is to maximize the second smallest flow bandwidth and then next. The efficiency is in terms of the response time from the change of the network condition until a new max-min fairness allocation is achieved.

For example, in Fig. 1, a scheme A achieves the max-min fairness by the following way. Initially the switch s1 and s2 take τ seconds to allocate the bandwidth for the flows passing through link1 and link2. For fairness, s1 allocates the bandwidth of 75 Million bits per second (Mbps) for each of flow 1 and 2 and s2 allocates 50 Mbps for each of flows 1, 3, and 4. And then the switches take π seconds to propagate the allocation information to their neighbor nodes. After the first propagation, the source nodes of flow 3 and 4 get the allocation 50 Mbps while those of flow 1 and 2 get the allocation 75 Mbps. When s1 receives from s2 the information 50 Mbps of flow 1 at link2, it would allocate the new bandwidth 50 and 100 Mbps for flow 1 and 2, respectively. After the second propagation, the source nodes of flow 1 and 2 get the allocation 50 and 100 Mbps, respectively. Then the max-min fairness allocation 50, 100, 50, and 50 Mbps for flow 1, 2, 3, and 4 is achieved. Thus the response time for this example is $2\tau + 2\pi$. In fact, the scheme A is very efficient, since it takes only two times of propagation to achieve the fairness while many schemes need more.

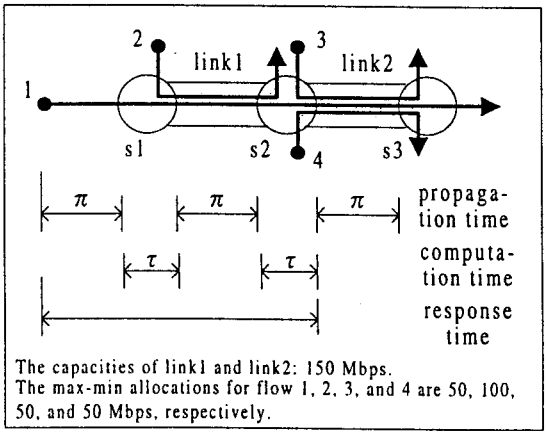


Fig. 1. Response time for max-min fairness

Recently, the max-min fairness has been adopted by the ATM Forum to evaluate the performance of different flow control schemes for the Available Bit Rate (ABR) service [2]. These well-known schemes for ABR either do not achieve the fairness or are not efficient. The Congestion Avoidance using Proportional Control (CAPC) scheme [3] takes a long response time to achieve the fairness. The Explicit Rate Indication for Congestion Avoidance (ERICA) scheme [4] overcomes the problem of CAPC, but not always achieve the fairness. The Max-Min scheme [5] solves the above problems while its algorithm takes O(N^2) time in the

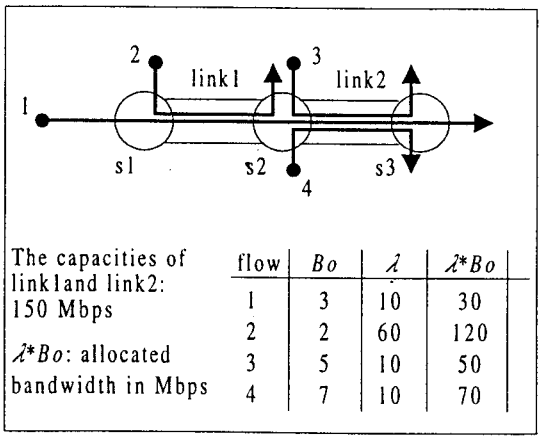
worst case to allocate bandwidth to N active flows passing through a link.

In this paper, a scheme, named Weighted Max-Min Fairness (WMF), is proposed to rapidly achieve the max-min fairness allocation. Its efficiency is better than that of the Max-Min scheme since it requires the same number of propagation as Max-Min to achieve the fairness and takes only O(log N) time in the worst case for bandwidth allocation at each switch. In addition, the scheme proposes the method for weighted bandwidth allocation, which supports users with various bandwidth requirements.

For weighted fair allocation, we define the weighted max-min fairness as follows. Assume each flow f is associated with a weight $Bo.f$. The normalized bandwidth $\lambda.f$ of flow f is the bandwidth allocated to f divided by its weight $Bo.f$. Let $\vec{\lambda}$ be the normalized bandwidth vector of all the active flows in a network, numbered from 1 to K, i.e. $\vec{\lambda} = (..., \lambda.i, ...)$, $i = 1, 2, .., K$.

Definition 1. (Feasibility) A normalized bandwidth vector $\vec{\lambda}$ is said to be feasible if it satisfies the following constraints: the normalized bandwidth $\lambda.i \geq 0$ for any active flow i and $F_l \leq CA_l$ for any link l in the network, where F_l is the sum of the allocated bandwidths of the active flows passing through link l and CA_l is the capacity of link l.

Definition 2. (Weighted Max-Min Fairness) A normalized bandwidth vector $\vec{\lambda}$ is weighted max-min fair if it is feasible and for each active flow i, for maintaining feasibility, the normalized bandwidth $\lambda.i$ cannot be increased without decreasing $\lambda.j$, for some active flow j where $\lambda.j \leq \lambda.i$.



The capacities of link1 and link2: 150 Mbps

$\lambda*Bo$: allocated bandwidth in Mbps

flow	Bo	λ	$\lambda*Bo$
1	3	10	30
2	2	60	120
3	5	10	50
4	7	10	70

Fig. 2. Weighted max-min fairness allocation

For example, Fig. 2. shows four flows in the 3-switch tandem network and the vector $\vec{\lambda}$ (10, 60, 10, 10), is a weighted max-min fairness allocation. Obviously, whenever we increase one element of the vector, there must be another element decreased for maintaining feasibility.

The remaining of this paper is as follows. In Section 2, the proposed scheme is described. Then the correctness of achieving the weighted max-min fairness allocation is shown in Section 3. A logarithm-time implementation for the scheme to allocate bandwidth at each switch is shown in Section 4. In Section 5, we give the comparison with some well-known flow control schemes. Finally, in Section 6 we give the conclusion.

2. Proposed scheme

The basic idea of the proposed scheme WMF is as follows. Initially, each active flow obtains a weighted bandwidth share at each switch. Thus an active flow might have different shares at the switches it passes through. The share of the flow at a switch is propagated to the neighbor nodes along the path of the flow. When a switch receives a bandwidth share of the flow from its neighbor node, it compares the external share with the bandwidth share of the flow in the switch itself (i.e the internal share). If the external share is less than the internal, the internal is updated to the external and the flow is said constrained since the internal is limited to the external. If the external share is no less than the internal, the internal will not change and the flow is said unconstrained. The bandwidth is allocated to the constrained flows according to their internal shares, and the remaining is distributed to the unconstrained flows proportional to their weights. The new bandwidth share of each flow at the switch will be propagated to the neighbor nodes. Finally, the bandwidth shares of the flow at the switches it passes through converge to a common share. For each active flow if its bandwidth shares at different switches converge at a common share, the weighted max-min fairness allocation is achieved.

It is necessary for a scheme to determine efficiently if flows are constrained or unconstrained, since the ways of allocating bandwidth to the two kinds of flows are different. It is found that for a link the normalized bandwidth allocated to the unconstrained flow is always larger than that of the constrained. Thus the active flows passing through a link can form an ordered list according to the increasing order of their normalized bandwidths. The list is named Buoy_list since the sublists of constrained and unconstrained flows are like the parts of a buoy under and over the water, respectively. WMF takes advantage of binary search on Buoy_list to find the first unconstrained flow, named critical flow, and thus all the constrained and unconstrained flows are found. Since flows would become active or idle, Buoy_list changes dynamically. To achieve the insertion or deletion of a flow in the Buoy_list with N active flows in O(log N) time, Buoy_list is implemented as a balanced binary search tree, named Buoy_tree.

By control packets the bandwidth shares of active flows at any switch can be propagated to its neighbor nodes. WMF assumes that the source node of each flow marks its control packets with "forward" and sends them into the network periodically. The control packets pass through the switches along the path of the flow. After

they are received by the destination node, their marks are replaced with "backward" and returned to the source node along the original path. The bandwidth share of the flow is stored in the expected rate (ER) field of control packets. Initially, the ER is set to the peak rate of the flow. After the control packet passes through a switch, the ER is set to the bandwidth share of the flow at the switch. The source of the flow sets its sending rate to the ER of the control packet, as soon as it receives the control packet with the mark "backward".

Each switch maintains the information (*ERf*, *ERb*) for each active flow passing through it, where *ERf* and *ERb* denote the value of ER of the latest received control packet marked with "forward" and "backward", respectively. Let *ERm* be the minimum of *ERf* and *ERb*. In other words, *ERm* is the currently smallest external share of a flow. Thus, for a flow at a switch, if its normalized *ERm* (i.e. $\frac{ERm}{Bo}$) is less than its normalized bandwidth share at the switch, it is constrained; otherwise, unconstrained. At each switch a process, named DIVIDING, of WMF is used to divide the active flows passing through a link into constrained and unconstrained. The following is the notations used in DIVIDING.

- CA the capacity of the link.
- N the number of the active flows passing through the link.
- i, j, k flow identifier. (i, j, k are integers from 1 to N.)
- L the set of the active flows which are constrained.
- H the set of the active flows which are unconstrained.
- ERm the minimum of ERf and ERb of a flow.
- λ the normalized bandwidth of a flow.
- Bo the weight of a flow.
- $ERf.i$, $ERb.i$, $ERm.i$, $Bo.i$, $\lambda.i$ ERf, ERb, ERm, Bo, and λ of a flow i.

Assume all the active flows passing through the link have been sorted according to their normalized *ERm*, i.e. $\frac{ERm}{Bo}$. If two flows have the same $\frac{ERm}{Bo}$, they are further sorted according to their flow identifiers. Let the sorted flow list be $i_1, i_2, \ldots, i_N$ and be defined as follows.

Definition 3. The **Buoy_list**, $i_1, i_2, \ldots, i_N$, of a link is an ordered list of the active flows through the link, where

$$\frac{ERm.i_p}{Bo.i_p} \le \frac{ERm.i_q}{Bo.i_q} \quad \text{for} \quad p < q, \quad \text{and if}$$

$$\frac{ERm.i_p}{Bo.i_p} = \frac{ERm.i_q}{Bo.i_q}, \; i_p < i_q.$$

To efficiently distinguish the constrained from unconstrained flows, the process DIVIDING scans the active flows in Buoy_list. Let f be a flow passing through the link. When a control packet of flow k arrives at the switch and $ERm.k$ changes or when flow k becomes active or idle, the process is executed as follows.

Process DIVIDING (Buoy_list, k);

<1> if ($ERm.k$ changes)

/* re-sort the Buoy_list since $ERm.k$ changes */

Delete (Buoy_list, k);

Insert (Buoy_list, k);

}

if (flow k becomes active) Insert (Buoy_list, k);

if (flow k becomes idle) delete (Buoy_list, k);

/* Let Buoy_list be $i_1, i_2, \ldots, i_N$ */

<2> $L = \{\}$; $H = \{i_1, i_2, \ldots, i_N\}$; $p = 1$;

<3> $$\lambda = \frac{(CA - \sum_{q=1}^{p-1} ERm.i_q)}{\sum_{q=p}^{N} Bo.i_q};$$

<4> if ($\frac{ERm.i_p}{Bo.i_p} < \lambda$) {

$L = L \cup \{i_p\}$; $H = H - \{i_p\}$; $p = p + 1$;

if ($p \le N$) goto <3>;

}

<5> $r = p$; $\lambda.i_r = \lambda$;

First, DIVIDING adjusts Buoy_list to reflect the change due to flow k. Then all the flows passing through

the link are assumed unconstrained and placed in the set H initially. The link bandwidth minus the bandwidth allocated to the constrained flows is allocated to the unconstrained flows proportional to their weights. Thus the normalized fair share of flow i_p is λ, computed in step <3>. If $\frac{ERm.i_p}{Bo.i_p} < \lambda$, the normalized bandwidth allocated to i_p is limited to $\frac{ERm.i_p}{Bo.i_p}$. Flow i_p is thus constrained and moved into the set L. On the other hand, if $\lambda \le \frac{ERm.i_p}{Bo.i_p}$, i_p is unconstrained. Finally, the active flows are partitioned into the sets L and H, by the flow i_r, which is defined as follows.

Definition 4. (Critical flow) For a link, with the process DIVIDING, the Buoy_list $i_1, i_2, \ldots, i_N$ is partitioned into constrained and unconstrained flow sets, L and H. If $H = \{i_r, i_{r+1}, \ldots, i_N\}$ for $1 \le r \le N$, i_r is the critical flow. If $H = \{\}$, there is no critical flow.

With the critical flow, for a flow i_p, the normalized bandwidth λi_p is $\frac{ERm.i_p}{Bo.i_p}$ if i_p is in L; otherwise λi_p is λi_r. There is an important property about the normalized bandwidth as follows.

Theorem 1. For a link, the normalized bandwidth of any constrained flow is less than that of any unconstrained flow.

Proof) All the unconstrained flows are of the same normalized bandwidth as that of the critical flow i_r. Since i_{r-1} is the constrained flow with the largest normalized bandwidth, to prove the theorem we compare the normalized bandwidth λi_r with λi_{r-1}. From the step <3> and <4> of DIVIDING, we have that $\lambda i_r = \frac{(CA - \sum_{q=1}^{r-1} ERm.i_q)}{\sum_{q=r}^{N} Bo.i_q}$ and $\lambda i_{r-1} = \frac{ERm.i_{r-1}}{Bo.i_{r-1}} < \frac{(CA - \sum_{q=1}^{r-2} ERm.i_q)}{\sum_{q=r-1}^{N} Bo.i_q}$. We write $\lambda i_r - \lambda i_{r-1}$ as follows.

$$\lambda i_r - \lambda i_{r-1} = \frac{(CA - \sum_{q=1}^{r-2} ERm.i_q) - ERm.i_{r-1}}{\sum_{q=r-1}^{N} Bo.i_q - Bo.i_{r-1}} - \frac{ERm.i_{r-1}}{Bo.i_{r-1}}$$

The result of the right side of the "=" sing in the above expression is positive, since for any real numbers a, b, c, and d, $\frac{c-a}{d-b} - \frac{a}{b} > 0$ if $\frac{c}{d} > \frac{a}{b}$ and $d > b > 0$. Thus $\lambda i_r > \lambda i_{r-1}$. Proven.

Theorem 1 can be used to determine if an active flow f is constrained or not. If $\frac{ERm.f}{Bo.f} < \lambda i_r$, f is constrained and λf is $\frac{ERm.f}{Bo.f}$; otherwise, f is unconstrained and $\lambda f = \lambda i_r$. Finally, WMF update ER of a control packet as follows. Immediately before a control packet of flow f departs from the switch, the ER of the packet is set to the bandwidth allocated to f, i.e. $\lambda f * Bo.f$.

Evidently, the process DIVIDING takes O(N) time to find the constrained and unconstrained flow sets, L and H. In fact, if the critical flow i_r is found, L and H are determined. With the following theorem, i_r can be found by the binary search on the Buoy_list, which takes only O(log N) time.

Theorem 2. Let i_p be a flow of Buoy_list. Let θi_p be the supposed share of i_p, where

$$\theta i_p = \frac{CA - \sum_{q=1}^{p-1} ERm.i_q}{\sum_{q=p}^{N} Bo.i_q}. \quad (1)$$

Flow i_p is constrained if and only if $\frac{ERmi_p}{Bo.i_p} < \theta i_p$.

Proof) If i_p is constrained, i_p must pass the check in the step <4> of DIVIDING and is moved into L. Thus

$\frac{ERmi_p}{Bo.i_p} < \theta.i_p$. Next, we will show if i_p is not constrained, $\frac{ERmi_p}{Bo.i_p} \geq \theta.i_p$. If i_p is unconstrained, the normalized bandwidth $\lambda.i_p$ is $\frac{CA - \sum_{q=1}^{r-1} ERm.i_q}{\sum_{q=r}^{N} Bo.i_q}$, where i_r is the critical flow. Thus, we have

$$\lambda.i_p - \theta.i_p = \frac{(CA - \sum_{q=1}^{r-1} ERm.i_q)}{\sum_{q=r}^{N} Bo.i_q} - \frac{(CA - \sum_{q=1}^{r-1} ERm.i_q) - \sum_{q=r}^{p-1} ERm.i_q}{\sum_{q=r}^{N} Bo.i_q - \sum_{q=r}^{p-1} Bo.i_q}. \quad (2)$$

Since $\frac{(CA - \sum_{q=1}^{r-1} ERm.i_q)}{\sum_{q=r}^{N} Bo.i_q} = \lambda.i_p = \lambda.i_r \leq \frac{ERmi_q}{Bo.i_q}$ for $q \geq r$, we have $\lambda.i_p - \theta.i_p \geq 0$. Since i_p is unconstrained, $\frac{ERmi_p}{Bo.i_p} \geq \lambda.i_p$. Therefore $\frac{ERmi_p}{Bo.i_p} \geq \theta.i_p$. Proven.

Corollary 1. A flow i_p in Buoy_list is unconstrained if and only if $\frac{ERmi_p}{Bo.i_p} \geq \theta.i_p$.

Although the critical flow i_r can be found in O(log N) time, to insert or delete a flow in Buoy_list still takes O(N). The solution is to construct Buoy_list as a balanced binary search tree. The detail will be shown in Section 4.

3. Correctness

In this section we prove that the scheme WMF can achieve the weighted max-min fairness allocation.

Theorem 3. The scheme WMF can achieve the weighted max-min fairness allocation.

Proof) Assume there are K active flows in the network, numbered from 1 to K. With WMF, the normalized bandwidth allocated to each flow i is $\lambda.i$. Let $\vec{\lambda}$ be the normalized bandwidth vector of all active flows, i.e. $\vec{\lambda}$ = (..., $\lambda.i$, ...), i = 1, 2, ..., K. From Definition 2, $\vec{\lambda}$ is weighted max-min fair if $\vec{\lambda}$ is feasible and for each active flow i, for maintaining feasibility, $\lambda.i$ cannot be increased without decreasing $\lambda.j$, for which $\lambda.j \leq \lambda.i$ for some active flow j.

With the WMF scheme, consider the situation after a stable allocation (i.e. after the bandwidth shares of each active flow at different switches converge to a common share). If an active flow i is constrained at a link, the bandwidth allocated to i at the link is limited to the bandwidth share of flow i at another link at which flow i is unconstrained. Thus for any active flow i there is at least one link l at which the flow is unconstrained. Since flow i is unconstrained at link l, the normalized bandwidth of flow i is equal to that of the critical flow of link l. Therefore, the normalized bandwidth of any flow passing through link l is no greater than that of flow i. If we try to increase the normalized bandwidth $\lambda.i$ of flow i, we must decrease the $\lambda.j$ of another flow j passing through link l for maintaining feasibility. Clearly, $\lambda.i$ cannot be increased without decreasing $\lambda.j$ for some j, where $\lambda.j \leq \lambda.i$. Therefore, the vector $\vec{\lambda}$ is weighted max-min fair.

4. Implementation

The process DIVIDING can be implemented as an algorithm with log(N) complexity if we construct Buoy_list as a balanced binary search tree, named Buoy_tree, and employ Theorem 2 to search the tree for the critical flow i_r. Each node in Buoy_tree represents a flow in Buoy_list and contains the fields shown in Table 1.

SERm	*key*	*SBo*
lt	*id*	*rt*

Table 1. Node of Buoy_tree

For a node x the field *id* is the identifier of the flow represented by x. *lt* and *rt* are the node identifier of the left and right sons of x, respectively. The value $\frac{ERm}{Bo}$ of the represented flow is stored in the field *key*. The sum of the *ERm* and *Bo* of all the nodes in the tree rooted in x are stored in *SERm* and *SBo*, respectively. By specifying a field of node x with x^<*field*>, we show the properties of Buoy_tree as follows:

P1) if node y is in the left subtree of x, y^*key* ≤ x^*key* (if y^*key* = x^*key*, y^*id* < x^*id*).

P2) if node y is in the right subtree of x, $y\text{^}key \geq x\text{^}key$ (if $y\text{^}key = x\text{^}key$, $y\text{^}id > x\text{^}id$).

P3) $x\text{^}SERm = x\text{^}lt\text{^}SERm + ERm.(x\text{^}id) + x\text{^}rt\text{^}SERm$;

P4) $x\text{^}SBo = x\text{^}lt\text{^}SBo + Bo.(x\text{^}id) + x\text{^}rt\text{^}SBo$.

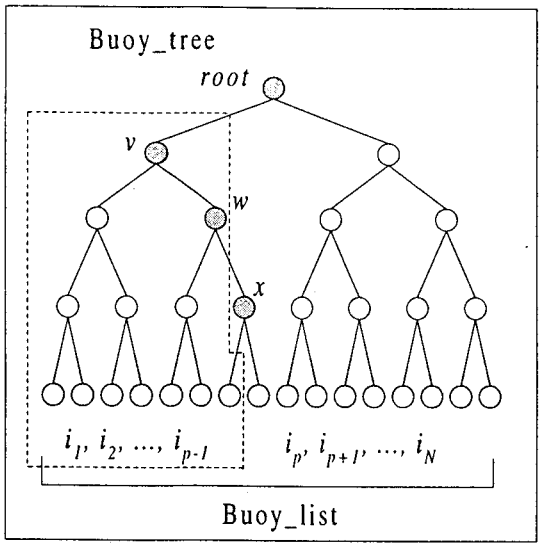


Fig. 3. Progress of searching in Buoy_tree

Let the node x in the Buoy_tree, as shown in Fig. 3, represent the flow i_p in the Buoy_list $i_1, i_2, \ldots, i_N$. With the property P1 and P2, the nodes framed with the dash line represent the sublist $i_1, i_2, \ldots, i_{p-1}$ in the Buoy_list. On the other hand, the nodes not framed represent $i_p, i_{p+1}, \ldots, i_N$.

Consider the process DIVIDING. In the step <1>, to delete or insert a flow k can be implemented by deleting or inserting the flow from or into Buoy_tree. These operations take O(log N) time if we use the red black tree [6][7] as Buoy_tree. A red black tree is a binary search tree of which every node is colored either red or black. The root of the tree is black, and, if a node is red, its children must be black. Since the tree is binary, each node in the tree has two pointers to its children. A pointer is null if the child it points to does not exist. Every path from a node down to any null pointer must contain the same number of black nodes. Thus, the height of a red black tree is at most 2 log(N + 1). The advantage of the red black tree is that the operations on the tree, such as searching, insertion, and deletion, take O(log N) time in the worst case.

The searching for the node, named critical node, representing the critical flow i_r starts from the root of Buoy_tree and advances down until a leaf. When the searching advances to a node x as shown in Fig. 3, x is checked if it is constrained or unconstrained by the inequality $x\text{^}key < \theta.i_p$, where i_p is the flow represented by x and $\theta.i_p$ is $\dfrac{CA - \sum_{q=1}^{p-1} ERm.i_q}{\sum_{q=p}^{N} Bo.i_q}$. Since $x\text{^}key$ is $\dfrac{ERmi_p}{Bo.i_p}$, from Theorem 2 if $x\text{^}key < \theta.i_p$, x is said constrained; otherwise, unconstrained. If x is constrained, the critical node must be in the right subtree of node x and thus the searching advances to the right son of x. If x is unconstrained, the critical node may be x or in the left subtree of x. Thus, x is kept as the possible critical node, and the searching advances to the left son of x. In the later advances, if a new possible critical node is found, the old one will be replaced by the new one. The searching ends at a leaf node, and the current possible critical node, if it exists, represents the critical flow i_r.

In the above, it is necessary to compute $\theta.i_p$ to check if node x is constrained or unconstrained. From the definition of $\theta.i_p$, the terms $\sum_{q=1}^{p-1} ERm.i_q$ and $\sum_{q=p}^{N} Bo.i_q$ is computed first. This can be achieved in O(1) time by using the *SERm* and *SBo* of the nodes the searching has advanced to. Let variables *lt_SERm* and *lt_SBo* be maintained in the progress of the searching. For example, in Fig. 3, when the searching advances to x, *lt_SERm* has contained the sum of $ERm.i_q$ of the flows i_q represented by the node v and w and the nodes in the left subtrees of v and w. Let T be the node identifier. Initially, T is set to *root*, and *lt_SERm* and *lt_SBo* are set to zero. Immediately before the searching advances to the next node, *lt_SERm* and *lt_SBo* are updated as follows.

If (the searching will advance to the right subtree of T) {

$lt\_SERm = lt\_SERm + T\text{^}SERm - T\text{^}rt\text{^}SERm$;

$lt\_SBo = lt\_SBo + T\text{^}SBo - T\text{^}rt\text{^}SBo$;

$T = T\text{^}rt$;

}

else $T = T\text{^}lt$;

Therefore, when the searching advances to node x (i.e. when T is set to x), the terms $\sum_{q=1}^{p-1} ERm.i_q$ and $\sum_{q=p}^{N} Bo.i_q$ are as follows.

$$\sum_{q=1}^{p-1} ERm.i_q = lt\_SERm + T\text{^}lt\text{^}SERm.$$

$$\sum_{q=p}^{N} Bo.i_q = root\text{^}SBo - (lt\_SBo + T\text{^}lt\text{^}SBo).$$

Since the Buoy_tree is not fixed, the values of the *SERm* and *SBo* of nodes change with the insertion and deletion of a node. In general, a new node y is inserted as a leaf according to property P1 and P2. Let f be the flow represented by y. Then *ERm.f* and *Bo.f* is added into the *SERm* and *SBo* of each node along the path from the root to y, respectively. To delete a node z, it is necessary to make z become a leaf first by the following way. Replace z with the leftmost (i.e. the smallest) node in the right subtree of z if the subtree is not null; otherwise, with the rightmost (i.e. the largest) node in the left subtree. Do this until z becomes a leaf. Let g be the flow represented by z. *ERm.g* and *Bo.g* is subtracted from the *SERm* and *SBo* of each node along the path from the root to z, respectively, after z is deleted. Since only the nodes along a path from the root to leaf are concerned, the insertion and deletion take O(log N) time.

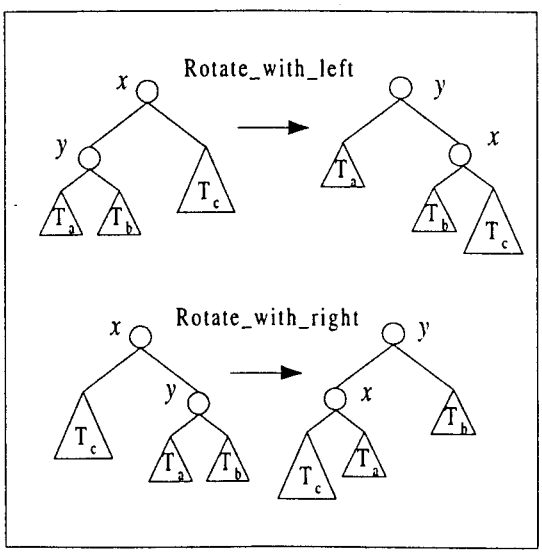


Fig. 4. Operation of tree rotations

When the insertion or deletion occur in a red black tree, the tree rotation is necessary to maintain the tree still being a red black tree. Two classical rotation operations are Rotate_with_left and Rotate_with_right shown in Fig. 4. For example, for Rotate_with_left, the left subtree of node x is rotated toward x, and then node y becomes the root of the tree. The *SERm* and *SBo* of x and y are recomputed, this takes O(1) time. The similar fact is for Rotate_with_right.

Since we employ the red black tree as Buoy_tree, the searching, insertion and deletion with rotation, and the computation of *SERm* and *SBo* on Buoy_tree take O(log N) times in the worst case. Therefore, we implement the process DIVIDING as a logarithm-time algorithm.

5. Comparison

It is shown in [5] that the response time of the Max-Min scheme to achieve the max-min fairness is less than that of the ERICA scheme. It is found in [8] that the response time of ERICA is much less than that of the CAPC scheme. Since the Max-Min scheme is better than the others in terms of the response time, we compare it with our scheme WMF. Because the fairness for the Max-Min scheme is not weighted, we set the weight of each flow to one in WMF for comparison.

For the Max-Min scheme, the bandwidth shares of active flows are also carried in ER of control packets and conveyed between network nodes. Each switch also maintains an *ERm* for each active flow passing through it. The active flows passing through a link are marked with either "constrained" or "unconstrained". The bandwidth allocated to each "constrained" flow is equal to its *ERm*. The remaining of the link bandwidth *CA* is evenly distributed to the "unconstrained" flows. At a switch, when the value of *ERm* of an active flow changes or a flow becomes active or idle, the fair share Λ of the "unconstrained" flows is computed as:

$$\Lambda = \frac{CA - \sum_{i:\text{"constrained" flow}} ERm.i}{\text{number of "unconstrained" flows}}. \quad (3)$$

All the "unconstrained" flows with the *ERm* no greater than Λ are marked with "constrained" and all the "constrained" ones with the *ERm* greater than Λ are marked with "unconstrained". Then Eq. (3) is computed and the flows are marked again, until the mark of any flow is not changed any more. Let Λ_{end} be the final value of Λ, which is the bandwidth allocated to the "unconstrained" flow. Clearly, it takes O(N^2) time in the worst case to divide the N active flows passing through a link into "constrained" and "unconstrained".

By Λ_{end}, the N active flows can be separated into i_1, i_2,..., i_{m-1} and i_m, i_{m+1},..., i_N, where $1 \le m \le N + 1$ and m satisfies the following conditions: 1) $ERm.i_1 \le ERm.i_2 \le \ldots \le ERm.i_{m-1} < \Lambda_{end}$ and 2) $\Lambda_{end} \le ERm.i_m \le ERm.i_{m+1} \le \ldots \le ERm.i_N$. From Max-Min, the bandwidth allocated

to flow i_p is either $ERm.i_p$ for $p < m$ or A_{end} for $p \geq m$. Consider the process DIVIDING of WMF, with which the active flows are moved into constrained flow set one by one until the critical i_r is found. Since Eq. (3) is the special case of Step <3> of DIVIDING while all weights are one, the critical flow i_r found by DIVIDING is the flow i_m done by the Max-Min scheme. Thus the results of allocating bandwidth to the N flows by Max-Min and WMF are the same. On the other hand, Max-Min and WMF use the same way to propagate the bandwidth share of the active flows. For both the reasons, Max-Min and WMF take the same number of times of control packet propagation to achieve the max-min fairness allocation. However, the response time of WMF is less than that of Max-Min since WMF takes only O(log N) time to allocate bandwidth to N active flows while Max-Min takes O(N^2).

6. Conclusion

Bandwidth allocation is one of the most important issues for flow control. Two important performance criteria concerned are the max-min fairness and response time to achieve the fairness. A lot of flow control schemes have been proposed, but they either cannot always achieve the fairness or take long time to achieve. The Max-Min scheme is one of the schemes with better performance in terms of response time. However, it takes O(N^2) in the worst case to allocate bandwidth to N active flows passing through a link. This is inefficient, especially for high speed networks, in which there are a large number of flows. We propose the scheme WMF, which can allocate weighted bandwidth and achieve the weighted max-min fairness allocation rapidly. The correctness of achieving the weighted max-min fairness is shown in Theorem 3. The response time to achieve the fairness is less than that of the Max-Min scheme, since WMF takes only O(log N) time to allocate bandwidth while Max-Min takes O(N^2). In fact, from the simulation Max-Min takes O (N) time in average for bandwidth allocation. Nevertheless, WMF is much more efficient than Max-Min when N is large.

Since WMF possesses the property of the weighted max-min fairness, it can support various bandwidth requirement of users. With the reservation strategy, WMF can guarantee the minimum bandwidth for each flow. Since it achieves weighted fairness allocation rapidly, it is suitable for variable-rate flows. For these reasons, it can support the Variable Bit Rate (VBR) service [9]. In the future, we will introduce the video traffics [10] into the network employing the WMF scheme. WMF will be compared with the schemes allocating bandwidth dynamically [11][12] according to the criteria of utilization, QoS, and efficiency.

References

[1] D. Bartsekas and R. Gallager, ed., Data Networks, Prentice Hall, 1992.

[2] F. Bonomi and K. W. Fendick, "The rate-based flow control framework for the available bit rate ATM service," IEEE Network Mag., vol.9, no.2, pp.25-39, March/April 1995.

[3] A. W. Barnhart, "Explicit rate performance evaluation," ATM Forum 94-0983R1, 1994.

[4] R. Jain, S. Kalyanaraman, R Viswanathan and R. Goyal, "A sample switch algorithm," ATM Forum 95-0178R1, 1995.

[5] D. H. K. Tsang and W. K. F. Wong, "A new rate-based switch algorithm for ABR traffic to achieve max-min fairness with analytical approximation and delay adjustment," Proc. IEEE INFOCOM'96, pp.1174-1181, 1996.

[6] M. A. Weiss, ed., Data Structures and Algorithm Analysis in C, Addison-Wesley, 1997.

[7] R. Sedgewick, ed., Algorithm in C, Addison-Wesley, 1990.

[8] D. H. K. Tsang, W. K. F. Wong, S. M. Jiang, and E. Y. S. Liu, "A fast switch algorithm for ABR traffic to achieve max-min fairness," Proc. IEEE International Zurich Seminar on Digital Communications, pp.19-23, 1996.

[9] Martin de Prycker, ed., Asynchronous Transfer Mode: Solution for Broadband ISDN, Ellis Horwood, 1993.

[10] M. Schwartz, ed., Broadband Integrated Networks, Prentice Hall, 1996.

[11] E. W. Fulp and D. S. Reeves, "On-line dynamic bandwidth allocation," Proc. IEEE International Conference on Network Protocols (ICNP'97), pp.134-141, 1997.

[12] E. W. Fulp, M. Ott, D. Reininger, and D. S. Reeves, "Paying for QoS: an optimal distributed algorithm for pricing network resources," Proc. International Workshop on Quality of Service (IWQOS '98), pp.75-84, May 1998.