Maxmin Fair Routing in Connection-Oriented Networks [*]

Shigang Chen, Klara Nahrstedt[†]
Department of Computer Science
University of Illinois at Urbana-Champaign
{s-chen5, klara}@cs.uiuc.edu

Abstract

The *maxmin fair bandwidth allocation* has been proposed as a flow/congestion control mechanism for managing the best-effort data traffic in connection-oriented networks. It achieves the fairness by assigning an equal share of bandwidth to each flow passing a link whenever possible; it improves the system throughput by fully utilizing the link capacity whenever possible. This type of fairness- and throughput-aware bandwidth allocation and corresponding routing schemes are important in multimedia connection-oriented networks for a class of services, such as ABR in ATM networks, which will carry heterogeneous types of best-effort traffic from non-real-time multimedia to ASCII text flows.

We present two contributions in this paper. The first contribution is to define a new set of *fairness-throughput* relational operators on the set of feasible bandwidth allocations and to show that the maxmin fair allocation maximizes the fairness-throughput performance measured by the new operators. The second contribution is to extend the concept of maxmin from the domain of flow/congestion control to the domain of routing. We propose a *maxmin fair routing algorithm* which selects the best route for a new flow such that the fairness-throughput performance of the maxmin fair allocation is maximized after the new flow joins in. The proposed algorithm achieves fairness by routing the new flow around congested links, which will improve the overall data throughput as well. Furthermore, an approximation algorithm is presented, which works better for networks with very dynamic states due to its simplicity and more relaxed information requirement on the network state.

[*]This work was supported by the ARPA grant under contract number F30602-97-2-0121 and the National Science Foundation Career grant under contract number NSF CCR 96-23867.

[†]Please address all correspondences to Klara Nahrstedt at Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, phone: (217) 244-6624, fax: (217) 244-6869.

1 Introduction

In traditional connectionless networks, data packets of a session (flow) may follow different routes to their destination node and network resources are effectively shared by packets from different sessions. However, this architecture does not meet the requirements of the future integrated-service networks. First, it does not support resource reservation which is vital for the provision of guaranteed quality of service (QoS). Second, data packets may experience unpredictable delay and arrive at the destination out of order which is undesirable for continuous real-time media such as audio and video. Hence, the next generation of high-speed wide-area networks will be connection-oriented. Extensive research and experiments have been done with the ATM technology [2, 5, 8, 10, 18].

The integrated services provided by connection-oriented high-speed networks such as ATM networks can be categorized into two broad classes. The first class provides *QoS services*. For example, in the ATM context, the CBR (constant bit rate) service can be used for transmitting real-time digitized voice and the VBR (variable bit rate) service can be used for transmitting real-time video which may be bursty at times. The second class provides *best-effort services*, e.g. the UBR (unspecified bit rate) and ABR (available bit rate) [1] services in ATM networks.

For the QoS services, resources (buffer space, processor time and link bandwidth) must be explicitly or implicitly reserved at each switch on the route of a flow in order to deliver data at the required throughput, or with a bounded end-to-end delay (delay variance), or both. Hence, the flow admission control and resource reservation are important for providing a sustainable quality to each flow accepted by the system [9, 21]. In this paper, we will not consider the QoS services and their implication on routing [19, 20].

On the other hand, for the best-effort services, resources are not reserved but shared among all flows in the network. The essential issues become fairness, effective resource utilization, overall throughput and average end-to-end delay, which are often conflicting measurements. The *maxmin fair* resource allocation was proposed as a flow/congestion control mechanism, which distributes the network bandwidth fairly among all competing flows [7]. It improves the throughput by a greedy strategy which assigns as much bandwidth to flows as the link capacity allows. It also provides a good delay-throughput tradeoff [7]. Charny et al. used the maxmin approach for the congestion control in ATM networks with explicit rate indication [6]. Ma et al. discussed the routing problem for high-bandwidth traffic in maxmin networks and aimed to improve the overall throughput [14].

[1] In the ABR service, a minimum bandwidth is specified, which provides the minimal QoS to the flow. However, above the minimum bandwidth, it is essentially best-effort.

Lu et al. studied the rate adaptation in mobile computing environments by using a weighted maxmin fair allocation [12].

This paper presents two contributions in the area of managing best-effort data flows by the maxmin fair bandwidth allocation. The first contribution is a further theoretical study. In contrast to Jaffe's objective function approach,[2] we define a new set of fairness-throughput relational operators on the set of feasible resource allocations and show that the maxmin allocation maximizes the fairness-throughput performance measured by the new operators. The second contribution is to extend the concept of maxmin from the domain of flow/congestion control to the domain of routing. We propose a *maxmin fair routing* algorithm which selects the best route for a new flow such that the fairness-throughput performance of the maxmin fair allocation is maximized after the new flow joins in. Since the proposed maxmin routing algorithm requires the knowledge of the routes of all existing flows, it is impractical for networks whose states are changing frequently as flows join and leave the networks. We thus present an approximation algorithm which is much simpler and does not require the information about individual flows. Furthermore, we augment the distributed bandwidth allocation algorithm proposed by Anna Charny et al [6] to provide the link state information needed by our approximation algorithm.

To the best of our knowledge, the only related work was done by Ma et al [14]. The routing algorithms in both [14] and this paper assume the network bandwidth is always assigned to the flows by the maxmin fair allocation. However, they try to optimize different performace metrics. Ma's algorithms use *throughput* as the only performacne metric and do not consider *fairness* at the routing level. In contrast, our routing algorithms optimize the *fairness-throughput* performance, the metric coming directly from the maxmin fair allocation itself, which are not optimized by Ma's algorithms.

The rest of the paper is organized as follows. The network and flow model is given in Section 2. The maxmin fair bandwidth allocation is briefly reviewed and an optimization property is given in Section 3. We define the problem of maxmin routing in Section 4 and propose an algorithm to solve the problem. An approximation algorithm and a dynamic bandwidth allocation protocol are discussed in Section 5 and Section 6, respectively. Finally, Section 7 draws the conclusion.

[2]Jaffe showed that the maxmin allocation minimized the defined performance objective function, which is given in Lemma 2 with some modification and different notations.

$\langle N, E\rangle$	the set N of nodes and the set E of links in the network
F	the set of flows in the network
$F(l)$	the set of flows through the link l, $F(l) \subseteq F$
$L(f)$	the set of links on the route of the flow f
l_b	the bottleneck link
$c(l)$	the capacity of the link l
B	a feasible bandwidth allocation
B_m	the maxmin bandwidth allocation

Table 1: Notations

2 Network and Flow Model

A network is modeled as a graph $\langle N, E\rangle$, where N is a set of nodes which are fully connected by a set E of full-duplex, directed communication links. Each link l has a bandwidth capacity $c(l)$. Let F be the set of flows in the network. We study the connected-oriented network where each flow has a fixed source (destination) and is assigned a fixed route through which all packets of that flow are transmitted in FIFO order [6, 7, 11]. We consider best-effort flows in this paper. By assuming the flow source has sufficient data and processing power and the network is always the bottleneck, a best-effort flow transmits data packets at the highest bandwidth the network supports [14]. Examples are connections for transmitting files, browsing web pages, or retrieving database information. For a flow $f \in F$, the set of links on its route is denoted as $L(f)$. The set of flows through a link l is denoted as $F(l)$. Other notations will be introduced when needed. Table 1 gives a quick reference to the notations used in this paper.

3 Maxmin Bandwidth Allocation

The maxmin fair bandwidth allocation [3] is first briefly reviewed and then an important optimization property is given by Theorem 1. The optimization property will be extended to the *maxmin fair routing* in Section 4. [4]

[3]Readers are referred to [6, 7] for more details.

[4]The maxmin fair routing is a new problem defined and solved in this paper, which is different from the maxmin fair allocation studied in the referred papers. Their difference will be detailed in Section 4.

3.1 A brief review

The maxmin allocation was first proposed by Jaffe [7] as a flow control technique which distributes the network bandwidth fairly among the best-effort flows. Much further research [1, 3, 4, 6, 15] has been done since then. It has been accepted by the ATM Forum as one of the traffic management approaches for the ABR service.

Its name comes from the fact that the maxmin allocation always *maximizes* the bandwidth allocated to those flows that receive the *minimum* bandwidth among all flows. Two basic properties are:

1. *Fairness property:* At each link l, every flow $f \in F(l)$ is allocated an equal share of the link capacity. However, if f receives a lower bandwidth at another link on its route, the bandwidth of f is allocated according to the bandwidth allocation at the bottleneck link on its route.

2. *Maximum throughput property:* The entire capacity of a link l must be allocated to the flows in $F(l)$ unless every flow f in $F(l)$ has a bottleneck link elsewhere on its route which limits the maximum bandwidth f can receive.

The maxmin bandwidth of each flow is determined by the bottleneck link on its route. A global bottleneck based algorithm which assigns the maxmin bandwidth to each flow was described in [6, 14] and is repeated below. A distributed algorithm was given in [7].

A global bottleneck link l_b is defined as the link which has the smallest bandwidth per flow, $\frac{c(l_b)}{|F(l_b)|}$. Since l_b is the most congested link in the network, it is the bottleneck link for each flow $f \in F(l_b)$. We allocate an equal share of the link capacity of l_b, i.e. $\frac{c(l_b)}{|F(l_b)|}$, to each f. Then all flows in $F(l_b)$ are removed from the network. The link capacities are reduced by the bandwidth consumed by the removed flows: For each $f \in F(l_b)$, the capacity of every link on the route of f is reduced by $\frac{c(l_b)}{|F(l_b)|}$. After that, the above procedure is repeated until every flow is assigned a bandwidth and removed from the network.

3.2 Fairness-throughput optimality

We formalize an important property for the maxmin allocation. A *feasible bandwidth allocation* $B : F \to R^+$ is a function which satisfies the following condition

$$\forall l \in E, \sum_{f \in F(l)} B(f) \leq c(l)$$

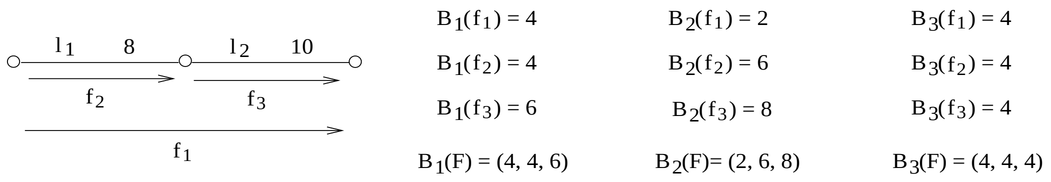


Figure 1: Consider three different allocations, $B_1(F) = (4,4,6)$, $B_2(F) = (2,6,8)$, and $B_3(F) = (4,4,4)$, among which B_1 is the maxmin fair allocation. $B_1(F) > B_2(F)$ because it is fairer. $B_1(F) > B_3(F)$ because it generates more overall throughput.

Let $B(F)$ be the *list* of values $(B(f) \mid \forall f \in F)$ in the increasing order. Note that in mathematics $B(F)$ normally represents a *set* of values $\{B(f) | f \in F\}$. In this paper we make a different interpretation by introducing an increasing order to $B(F)$ and therefore using it as an ordered *list*. A link l is said to be *saturated* by B if

$$\sum_{f \in F(l)} B(f) = c(l)$$

Definition 1 *Given two feasible bandwidth allocations B and B' on F, we define the fairness-throughput relations: (1) $B(F) = B'(F)$ if the two lists are identical, and (2) $B(F) > B'(F)$ if there exists a prefix of $B(F)$, $(b_1, b_2, ..., b_i)$, and a prefix of $B'(F)$, $(b'_1, b'_2, ..., b'_i)$, such that $b_i > b'_i$ and $b_j = b'_j, 1 \le j \le i-1$.*

The above relations place a total order on the set of feasible allocations. The ordering is based on two performance measurements, *fairness* and *throughput*. In more descriptive and less precise words, an allocation is larger than the other if it is fairer and/or generates more throughput, which is illustrated by Figure 1, where three flows, f_1, f_2 and f_3, share two links with capacities 8 and 10, respectively. Consider three different allocations, B_1, B_2 and B_3. By Definition 1, $B_1(F) > B_2(F)$, The reason is that B_1 is fairer as it splits the capacity of link l_1 equally between f_1 and f_2 and thus maximizes the smallest element in the list of $B_1(F)$. B_3 is also fair. However, it does not fully utilize the capacity of link l_1. Hence, $B_1(F) > B_2(F)$.

Fairness and throughput are often conflicting measurements. For example, $B_2(F)$ has more overall throughput [5] but $B_1(F)$ is fairer between f_1 and f_2. The fairness-throughput relations defined in Definition 1 evaluate an allocation based on a measurement which provides a tradeoff between the fairness and the overall throughput. We shall establish a theorem showing that the maxmin allocation maximizes the fairness-throughput performance, i.e. $B_m(F) \ge B(F)$, where B is an arbitrary feasible allocation and B_m is the notation specially for the maxmin allocation here and in the rest of the paper. The proof of the following theorem is given in the Appendix.

[5] We define the *overall throughput* as the aggregate throughput of all flows.

Theorem 1 *$B_m(F) \geq B(F)$, for any feasible allocation B.*

4 Maxmin Routing

In this section, the optimization property of the maxmin bandwidth allocation illustrated in Theorem 1 is extended from the domain of flow control to the domain of routing. We first define the concept of *maxmin routing*.

4.1 Definition of maxmin routing

Let $G\langle N, E\rangle$ be a network where bandwidth is alway allocated to flows by maxmin. Let F be the set of existing flows, each of which has a fixed route. Consider a new flow f_0. The task of routing is to assign a route r to f_0. Each different route for f_0 results in a different maxmin bandwidth allocation $B_{m,r}$ on $F \bigcup \{f_0\}$. The purpose of *maxmin routing* is to find a route r such that $B_{m,r}(F \bigcup \{f_0\}) \geq B_{m,r'}(F \bigcup \{f_0\})$, for any feasible route r' of f_0.

The maxmin routing is a new problem which is different from the maxmin allocation studied by the previous publications. The problem solved by the latter is as follows: given a network and a set of flows with fixed routes, how to assign the network bandwidth to the flows such that the network performance is optimized. The maxmin routing, however, assumes the network bandwidth is assigned based on the maxmin allocation. It then introduces another dimension, new flows. The problem to be solved is how to assign routes to new flows such that the performance of the maxmin allocation can be maximized.

4.2 A maxmin routing algorithm

We propose a maxmin routing algorithm. The algorithm first adds f_0 to every link in the network [6] and then iteratively removes f_0 from the links with the minimum bandwidth per flow until the route for f_0 is found. By removing f_0 from the links with the minimum bandwidth per flow, the algorithm effectively routes f_0 around the most congested links and therefore maximizes the bandwidth allocated to the congested flows, which equals maximizing the low end of $B_{m,r}$ and thus equals maximizing $B_{m,r}$ because the low end of $B_{m,r}$ is of more significance by definition.

The algorithm below consists of two phases. In the first phase, the bottleneck link of f_0 is found, which determines the maxmin bandwidth for f_0; in the second phase, the algorithm finds

[6]Note that we are not adding the flow to the links of the *real* network but to the data structure representing the network at a node doing the source routing.

the rest of the route which maximizes $B_{m,r}$. We mark links either green or red. Green links are candidates to form a route for f_0; red links are either not on any paths from the source to the destination or considered to be congested and thus rejected by the algorithm. A path consisting of only green links is called a *green path*. When we say "remove a flow f from the network", we mean, $\forall l \in L(f), F(l) = F(l) - \{f\}$ and $c(l) = c(l) - B_{m,r}(f)$, where $B_{m,r}(f)$ is the bandwidth assigned to the flow by the algorithm. When we say the source (destination), we mean the source (destination) of f_0. The first phase of the algorithm is as follows.

1. For every link l that is on a path from the source to the destination, add f_0 to $F(l)$ and mark l as a green link. Mark other links red.
2. Find the global bottleneck link l_b which has the smallest bandwidth per flow, $\frac{c(l_b)}{|F(l_b)|}$.
   (a) If l_b is a red link, [7] then
      i. assign bandwidth $\frac{c(l_b)}{|F(l_b)|}$ to every flow in $F(l_b)$,
      ii. remove all flows in $F(l_b)$ as well as link l_b from the network, and
      iii. repeat Step 2.
   (b) If l_b is a green link and not all green paths from the source to the destination pass l_b, then
      i. remove f_0 from $F(l_b)$,
      ii. mark l_b as a red link, and
      iii. repeat Step 2.
   (c) If l_b is a green link and all green paths from the source to the destination pass l_b, then
      i. l_b is the bottleneck link for f_0, and will be denoted as l_0 in the second phase,
      ii. assign bandwidth $\frac{c(l_b)}{|F(l_b)|}$ to every flow in $F(l_b)$ including f_0,
      iii. remove all flows in $F(l_b)$ except f_0 from the network, [8] and
      iv. go to the second phase of the algorithm.

Let l_0 be the bottleneck link of f_0, which is found at Step 2(c) in the first phase. Let $B_{m,r}(f_0)$ be the bandwidth assigned to f_0. The second phase is to find the rest links which together with l_0 will form a route for f_0. It has the same control structure as the first phase except the subtle differences in the calculation of the global bottleneck and the treatment of f_0. The first phase focuses on finding

[7]By the construction of the algorithm, $f_0 \notin F(l_b)$ if l_b is a red link; $f_0 \in F(l_b)$ if l_b is a green link.
[8]We can not remove f_0 from the network because the route of f_0, i.e. $L(f_0)$, is unkown.

the bottleneck link l_0 and determining $B_{m,r}(f_0)$ whereas the second phase focuses on finding the rest of the route. Their conceptual differences make us decide to separate them, which also seems to make the algorithm more understandable. When the second phase terminates, $L(f_0)$ contains the links which form the route for f_0.

1. $L(f_0) = \{l_0\}$.

2. Among the links in $E - L(f_0)$, find the global bottleneck link l_b which has the smallest bandwidth per flow. The bandwidth per flow of a link $l \in E - L(f_0)$ is calculated by $\frac{c(l)-B_{m,r}(f_0)}{|F(l)|-1}$ if l is a green link or $\frac{c(l_b)}{|F(l_b)|}$ if l is a red link.

   (a) If l_b is a red link, then

      i. assign bandwidth $\frac{c(l_b)}{|F(l_b)|}$ to every flow in $F(l_b)$,

      ii. remove all flows in $F(l_b)$ as well as link l_b from the network,

      iii. repeat Step 2.

   (b) If l_b is a green link and not all green paths from the source to the destination pass l_b, then

      i. remove f_0 from $F(l_b)$,

      ii. mark l_b as a red link, and

      iii. repeat Step 2.

   (c) If l_b is a green link and all green paths from the source to the destination pass l_b, then

      i. $L(f_0) = L(f_0) \bigcup \{l_b\}$,

      ii. assign bandwidth $\frac{c(l_b)-B_{m,r}(f_0)}{|F(l_b)|-1}$ to every flow in $F(l_b)$ except f_0,

      iii. remove all flows in $F(l_b)$ except f_0 from the network, and

      iv. terminate if the links in $L(f_0)$ form a path from the source to the destination, or repeat Step 2 otherwise.

4.3 An example

The algorithm is stepped through by using an example in Figure 3. The initial network state is given by Figure 2 (a) as $c(l_1) = 15$, $c(l_2) = 8$, $c(l_3) = 30$, $c(l_4) = 5$, $c(l_5) = 15$, and $F = \{f_1, f_2, f_3\}$. We want to assign a route to a new flow, f_0, whose source and destination are s and t, respectively. In the execution of the algorithm, we only consider those links l with non-empty $F(l)$.

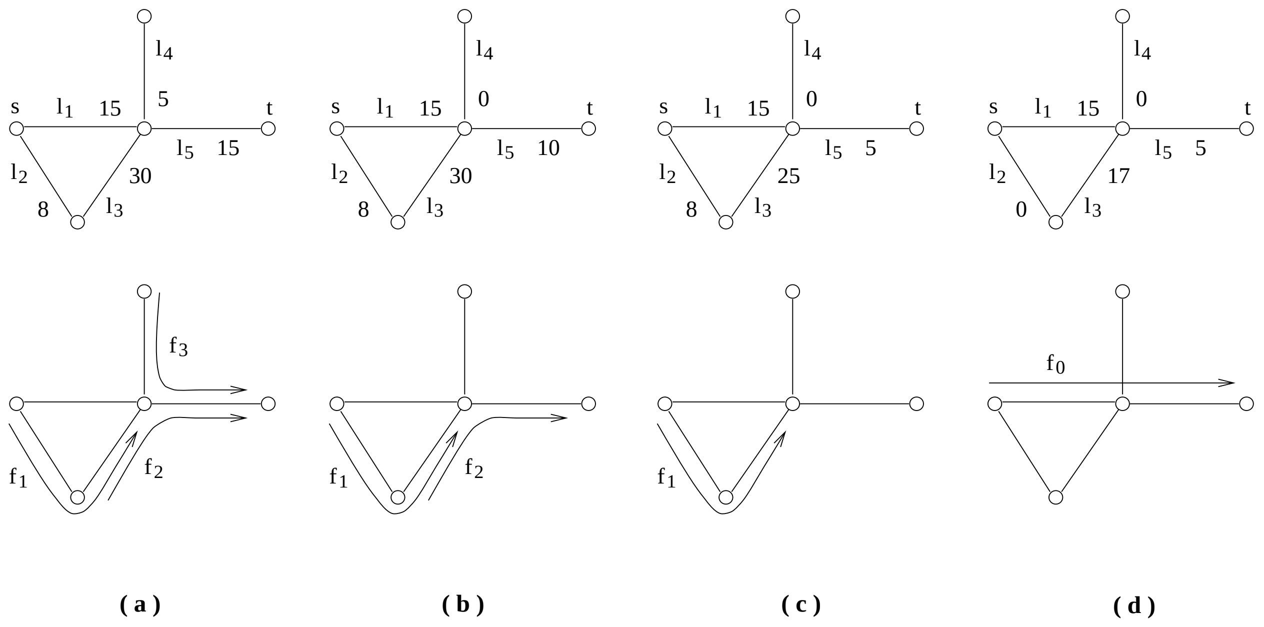
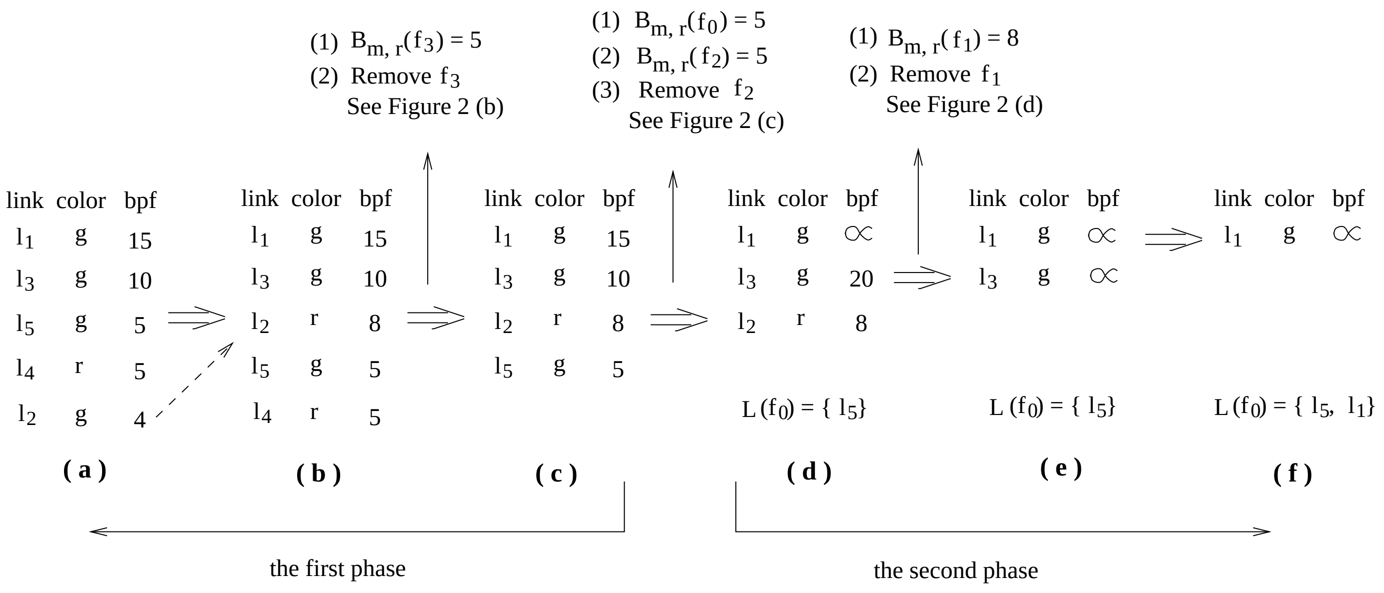


Figure 2: (a) The initial network state: $c(l_1) = 15$, $c(l_2) = 8$, $c(l_3) = 30$, $c(l_4) = 5$, $c(l_5) = 15$, and $F = \{f_1, f_2, f_3\}$. The route of each flow is shown in the figure. (b) The network state after f_3 is removed. (c) The network state after f_2 is removed. (d) The network state after f_1 is removed. The route for f_0 is found, which is $\{l_1, l_5\}$.

At the first iteration of the first phase (Figure 3 (a)), l_2 is identified as the global bottleneck and marked red. At the second iteration, l_4 becomes the global bottleneck and a bandwidth of 5 is assigned to f_3, which is then removed from the network. The new network state is shown in Figure 2 (b). At the third iteration, l_5 is the global bottleneck. Since l_5 is on every green path from s to t, it must be on the route to be assigned to f_0. Hence, l_5 is identified as the bottleneck link of f_0. According to the algorithm, f_0 and f_2 receives an equal share of the link capacity of l_5, which is 5. Then, we proceed into the second phase.

At the first iteration of the second phase (Figure 3 (d)), l_2 is the global bottleneck, and a bandwidth of 8 is assigned to f_1. Notice that the average bandwidth per flow (bpf) of l_1 is infinite. That is because $F(l_1) = \{f_0\}$. See the algorithm for how to calculate the bpf of a green link. At the second iteration, since there is a green path $\{l_1, l_5\}$ which does not pass l_3, l_3 is marked red. At the last iteration, l_1 is added into $L(f_0)$ to form a route for f_0.

The resulting route is $r = L(f_0) = \{l_1, l_5\}$, which leads to a maxmin allocation $B_{m,r}(F \bigcup \{f_0\}) = (5, 5, 5, 8)$. Consider the other feasible route $r' = \{l_2, l_3, l_5\}$, whose maxmin allocation is $B_{m,r'}(f_0) = 4$, $B_{m,r'}(f_1) = 4$, $B_{m,r'}(f_2) = 6$, $B_{m,r'}(f_3) = 5$, and thus $B_{m,r'}(F \bigcup \{f_0\}) = (4, 4, 5, 6)$. We have $B_{m,r}(F \bigcup \{f_0\}) > B_{m,r'}(F \bigcup \{f_0\})$.



bpf: bandwidth per flow g: green r: red

Figure 3: The algorithm consists of two phases. Each phase consists of a number of iterations. At each iteration, (a)-(f), only those links l with non-empty $F(l)$ are considered. They are ordered in the decreasing order of bpf (bandwidth per flow); the last one in the list is the global bottleneck.

4.4 Discussion

The maxmin routing avoids the congested links by marking them red, and routes f_0 along those links whose per-flow bandwidth is as large as possible. Hence, it helps to improve the overall throughput of the network. However, in a long term, fairness may contradict throughput as the proposed algorithm may select an excessively long path which reduces the overall bandwidth available for flows arriving successively. Research showed that short routing paths tend to yield high overall throughput [13]. The proposed algorithm can be modified to take the path length into consideration. A maximum allowable length is specified for a new flow based on the distance from the source to the destination. After marking a green link l red, the algorithm tests whether there still exists a green path from the source to the destination whose length is bounded by the maximum allowable length. If the answer is false, the algorithm marks l back green, selects the shortest green path from the source to the destination and then terminates.

5 An Approximation Algorithm

The proposed maxmin routing algorithm requires the knowledge of the routes of all existing flows. When flows join and leave the network frequently, the communication overhead for collecting the routes of all flows will be excessively high. Many routing algorithms [14, 17, 20] rely on the state information of the links in the network, instead of that of the flows. The state of the links can be collected and maintained at each node by the link-state algorithm [16], which has been implemented on many internetworks. In the following, we approximate the proposed maxmin routing algorithm by using a new state defined on each link.

Each link l monitors the actual data rate $r(f)$ of every flow $f \in F(l)$.[9] By using the dynamic bandwidth allocation discussed in the next section, the actual data rate of a flow will approximate the expected maxmin bandwidth. The link l keeps track of the set $F_b(l)$ of flows whose bottleneck links are l. How to maintain $F_b(l)$ is also discussed in the next section. Note that, according to the maxmin allocation, the rates of flows in $F_b(l)$ are higher than those of flows in $F(l) - F_b(l)$ whose bottleneck links are elsewhere on their routes other than l [3]. A new state $r(l)$ is defined for l.

$$r(l) = max\{\frac{c(l) - \sum_{f \in F - F_b(l)} r(f)}{|F_b(l)| + 1}, \frac{c(l)}{|F(b)| + 1}\}$$

If a new flow f_0 is routed through l and l is the bottleneck link of f_0, $r(l)$ is an approximation of the bandwidth allocated to every flow in $F_b(l) \bigcup \{f_0\}$.

At every node in the network, the link state algorithm collects all $r(l), l \in E$. When a new flow f_0 arrives, a source routing is done by using the Dijkstra's algorithm to find a path p which maximizes the smallest $r(l)$ on the path, i.e. to maximize $\min_{l \in p}\{r(l)\}$. If there are multiple such paths, choose one which maximizes the second smallest $r(l)$, and so on. This procedure continues until a single path is found or a pre-determined maximum allowable length has been reached.

6 Distributed Dynamic Bandwidth Allocation

The bandwidth allocated to a flow changes as other flows join and leave the network dynamically. The source of a flow must adjust its data rate according to the network dynamics. The *dynamic bandwidth allocation* is used to adjust on the fly the bandwidths of all flows and inform the sources to change their data rates accordingly. The design requirement of dynamic bandwidth allocation is that, given any initial state, it must be able to converge to the maxmin allocation in finite time if there is no further arrival of new flows and no further leave of existing flows. Anna Charny et al [6] proposed a distributed algorithm which fulfills such a requirement. We modify the algorithm in order to maintain $F_b(l)$ and $r(l)$ at each link l and therefore provide the information needed by the

[9]Note that the monitoring as well as other link operations is in fact done by the node in charge of the link.

approximation algorithm in Section 5. The value $r(l)$ will be sent to each router in the network by the link-state algorithm for the purpose of maxmin routing.

1. The source of each flow f sends out forward control messages, RM cells in the ATM context, along its route periodically to determine the expected maxmin bandwidth. The rate of control messages should be bounded by certain low percentage of the average data rate.

2. When a link receives a forward control message, it assigns bandwidth $\frac{c(l) - \sum_{f \in F - F_b(l)} r(f)}{|F_b(l)|}$ to the message [10] and forwards the message to the next hop on its route. By traversing every link on its route, the forward control message keeps the smallest assigned bandwidth, and the link which assigns such a bandwidth is also kept as the bottleneck link.

3. When the destination receives a forward control message, it turns the message around as a backward control message which traverses back along the route to the source.

4. When a link receives a backward control messages, it checks whether it is the bottleneck link for this flow. If it is, $F_b(l) = F_b(l) + \{f\}$; otherwise, $F_b(l) = F_b(l) - \{f\}$. $r(l)$ is recomputed when necessary.

5. When the source receives a backward control message, it adjusts the data rate according to the smallest assigned bandwidth carried back by the message. [11]

7 Conclusion

In connection-oriented networks, one of the most challenging problems is how to share the resources fairly among the competing flows and at the mean time maximize the throughput. Fairness and throughput, however, are often conflicting measurements. By making a tradeoff between them, we defined the fairness-throughput relation on the set of feasible bandwidth allocations and showed that among them the maxmin fair allocation has the optimal fairness-throughput performance.

We then used the same fairness-throughput relation as a performance measurement for routing and proposed a maxmin fair routing algorithm which assigns the best route to a new flow such that the performance of the maxmin fair allocation can be maximized after the new flow joins in. The route assignment consists of two phases. During the first phase, the algorithm finds the

[10] In the above formula, we ignore the bandwidth consumed by the control messages for the purpose of simplicity. Readers are referred to [6], where the traffic volume of control messages are considered.

[11] If the assigned bandwidth is much greater than the current data rate, the source may choose to increase the data rate gradually in order to avoid oscillation or temporary overloading of the bottleneck link.

bottleneck link for the new flow and determines the maxmin bandwidth of the new flow; during the second phase, the algorithm finds the best route by avoiding choosing the links which are relatively more congested. The proposed algorithm achieves fairness by routing the new flow around the congested links, which will improves the overall data throughput as well. We further developed an approximation algorithm which works better for networks with very dynamic states due to its simplicity and more relaxed information requirement on the network state.

Our maxmin routing algorithms look at a single new flow at a time. It would be interesting to study the maxmin routing problem in the context of multiple new flows, which will be more difficult because the route change of each flow may make the optimal routes of the others different. Heuristics will be needed for an efficient implementation with a good statistical performance in a long run. Another part of our future work is to compare with other routing algorithms by simulations and show how well our maxmin routing algorithms perform at every individual performance metrics such as average throughput per flow, throughput derivation and average packet delay.

References

[1] Baruch Awerbuch and Yuval Shavitt. Converging to approximated max-min flow fairness in logarithmic time. *accepted for Infocom'98, San-Francisco, CA*, April 1998.

[2] S. Berson and L. Berger. Ip integrated services with rsvp over atm. *IETF Internet Draft: draft-ieft-issll-atm-support-03.txt*, July 1997.

[3] D. Bertsekas and R. Galager. Data networks. *Prentice Hall, Englwood Cliffs, N.J.*, 1992.

[4] F. Bonomi and K. Fendick. The rate-based flow control framework for the available bit rate atm service. *IEEE Network*, pages 9(2):25–39, March/April 1995.

[5] Flavio Bonomi, Kerry Fendick, and Nanying Yin. Abr point-to-multipoint connections. *ATM Forum/95-0974R1*, August 1995.

[6] Anna Charny, David D. Clark, and Raj Jain. Congestion control with explicit rate indication. *IEEE International Conference on Communications*, 1995.

[7] Jeffrey M. Faffe. Bottleneck flow control. *IEEE Transactions on Communications*, COM-29(7):954–962, July 1981.

[8] Sonia Fahmy, Raj Jain, Shivkumar Kalyanaraman, Rohit Goyal, Bobby Vandalore, Xiangrong Cai, and Seong-Cheol Kim. Performance analysis of abr point-to-multipoint connections for

bursty and non-bursty traffic with and without vbr background. *ATM Forum/97-0422*, April 1997.

[9] Domenico Ferrari and Dinesh C. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE JSAC*, 8(3):368–379, April 1990.

[10] ATM Forum. Private network network interface (pnni) v1.0 specifications. June 1996.

[11] H. T. Kung, Trevor Blackwell, and Alan Chapman. Credit-based flow control for atm networks: Credit update protocol, adaptive credit allocation, and statistical multiplexing. *ACM Sigcom, London England UK*, pages 101–114, August 1994.

[12] Songwu Lu, Kang-Won Lee, and Vaduvur Bharghavan. Revenue-based rate adaptation in mobile computing environment. *Unpublished paper, University of Illinois at Urbana-Champaign, Electrical and Computer Engineering*, 1997.

[13] Q. Ma and P. Steenkiste. Quality-of-service routing with performance guarantees. *Proceedings of the 4th International IFIP Workshop on Quality of Service*, May 1997.

[14] Q. Ma, P. Steenkiste, and H. Zhang. Routing high-bandwidth traffic in max-min fair share networks. *Proceedings of SIGCOMM'96*, August 1996.

[15] J. Mosley. Asynchronous distributed flow control algorithms. *Ph.D. thesis, MIT, Dept. of Electrical Engineering and Computer Science, Cambridge, MA*, 1984.

[16] J. Moy. Ospf version 2, internet rfc 1583. March 1994.

[17] C. Parris, H. Zhang, and D. Ferrari. Dynamic management of guaranteed performance multimedia connections. *Multimedia Systems Journal*, 1:267–283, 1994.

[18] Wenge Ren, Kai-Yeung, and Hiroshi Suzuki. Performance evaluation of multipoing-pint abr and ubr. *ATM Forum/96-1402*, October 1996.

[19] H. F. Salama, D. S. Reeves, and Y. Viniotis. A distributed algorithm for delay-constrained unicast routing. *INFOCOM'97, Japan*, March 1996.

[20] Z. Wang and Jon Crowcroft. Qos routing for supporting resource reservation. *Journal of Selected Areas in Communications*, September 1996.

[21] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. Rsvp: A new resource reservation protocol. *IEEE Network*, September 1993.

Appendix: The proof of Theorem 1

See Table 1 for the explanation of the notations.

Lemma 1 *For any $f \in F$, there must exist a saturated link in $L(f)$, i.e.*

$$\exists l \in L(f), \sum_{f \in F(l)} B_m(f) = c(l)$$

Proof: This is due to the maximum throughput property (Page 5). The maxmin allocation always tries to increase the bandwidth of a flow until at least one link (bottleneck) on its route is saturated. $\square$

Lemma 2 *Consider the objective function $Y(B)$ defined on the set of feasible allocations B.*

$$Y(B) = \sum_{f \in F} \left| B(f) - \min_{l \in L(f)} \{ \max_{f' \in F(l)} B(f') \} \right|$$

Then, (1) the maxmin allocation B_m minimizes the above objective function and (2) $Y(B_m) = 0$.

Proof: A proof was given by Jaffe in [7]. [12] $\square$

Lemma 3 *For any $f \in F$, there exists a saturated link $l \in L(f)$ such that $B_m(f) = \max_{f' \in F(l)} B_m(f')$.*

Proof: Since $Y(B_m) = 0$ by Lemma 2, we have

$$B_m(f) = \min_{l \in L(f)} \{ \max_{f' \in F(l)} B_m(f') \} \tag{1}$$

and therefore there exists $l \in L(f)$ such that

$$B_m(f) = \max_{f' \in F(l)} B_m(f') \tag{2}$$

Let Ω be the set of links in $L(f)$ which satisfy (2). By (1) and the definition of Ω,

$$\forall l \in L(f) - \Omega, B_m(f) < \max_{f' \in F(l)} B(f')$$

However, the fairness property (Page 5) requires that f should receive an equal share of bandwidth which is as large as any other flow on the link $l \in L(f) - \Omega$ can receive and thus must be $\max_{f' \in F(l)} B(f')$,

[12] Note that the notations as well as the presentation here are different from those in [7]

unless f has a bottleneck link elsewhere on its route. That means Ω must contain a bottleneck link which prevents B_m from assigning a higher bandwidth to f. The immediate result is that there must be a saturated link in Ω because otherwise the maximum throughput property (Page 5) requires a higher bandwidth to be allocated to f on the links in Ω. $\square$

Theorem 1 $B_m(F) \geq B(F)$, for any feasible allocation B.

Proof: We prove it by induction on the number of flows in the network. A base case is first proved. For any network topology $\langle N, E\rangle$ and an arbitrary capacity function $c : E \rightarrow R^+$, consider a single flow f, i.e. $F = \{f\}$. By Lemma 1, there must be a saturated link in $L(f)$. The saturated link must be the one with the smallest capacity and hence $B_m(f) = \min_{l \in L(f)} c(l)$. For any feasible allocation B, $B(f) \leq \min_{l \in L(f)} c(l)$ by definition. Hence, $B_m(f) \geq B(f)$.

Assume that $B_m(F) \geq B(F)$ is true for any $\langle N, E\rangle$ and $c : E \rightarrow R^+$ with $|F| < k$. We prove the case where $|F| = k$. Let l_b be the global bottleneck link such that

$$\frac{c(l_b)}{|F(l_b)|} = \min_{l \in E} \frac{c(l)}{|F(l)|}$$

The maxmin allocation assigns an equal share of $c(l_b)$ to every flow in $F(l_b)$ so that,

$$\forall f \in F(l_b), B_m(f) = \frac{c(l_b)}{|F(l_b)|}$$

We now show that,

$$\forall f \in F - F(l_b), B_m(f) \geq \frac{c(l_b)}{|F(l_b)|}$$

By Lemma 3, there exists a saturated link $l \in L(f)$ such that $B_m(f) = \max_{f' \in F(l)} B(f')$. Since l is saturated, $\sum_{f' \in F(l)} B_m(f') = c(l)$. Hence,

$$B_m(f) = \max_{f' \in F(l)} B(f') \geq \frac{\sum_{f' \in F(l)} B_m(f')}{|F(l)|} = \frac{c(l)}{|F(l)|} \geq \frac{c(l_b)}{|F(l_b)|}$$

Therefore, $B_m(F) = B_m(F(l_b)) + B_m(F - F(l_b))$, where $+$ is the *concatenation* of two lists.

Let B be an arbitrary feasible bandwidth allocation. Consider two cases.

1. Case 1: $\exists f \in F, B(f) < \frac{c(l_b)}{|F(l_b)|}$. We have $B_m(F) > B(F)$ because the smallest element in $B_m(F)$ is $\frac{c(l_b)}{|F(l_b)|}$, which is greater than the smallest one in $B(F)$.

2. Case 2: $\forall f \in F, B(f) \geq \frac{c(l_b)}{|F(l_b)|}$. Since $\sum_{f \in F(l_b)} B(f) \leq c(l_b)$, we have $\forall f \in F(l_b), B(f) = \frac{c(l_b)}{|F(l_b)|}$. Hence, $B(F) = B(F(l_b)) + B(F - F(l_b))$. $B_m(F(l_b)) = B(F(l_b))$ since they are identical. By

the induction hypothesis, $B_m(F - F(l_b)) \geq B(F - F(l_b))$, for a network where the flows in $F(l_b)$ are removed and the capacities of the links are reduced as follows: for every $f \in F(l_b)$, the capacity on every link $l \in L(f)$ is reduced by $\frac{c(l_b)}{|F(l_b)|}$.

Hence, we have $B_m(F) \geq B(F)$ by synthesizing the above results

(a) $B_m(F) = B_m(F(l_b)) + B_m(F - F(l_b))$,

(b) $B(F) = B(F(l_b)) + B(F - F(l_b))$,

(c) $B_m(F(l_b)) = B(F(l_b))$, and

(d) $B_m(F - F(l_b)) \geq B(F - F(l_b))$.

Therefore, the theorem holds. $\square$