University of Alberta

Dynamic Network Resource Allocation

by

Yu Sheng

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Fall 2010
Edmonton, Alberta

Examining Committee

Mike MacGregor, Computing Science

Chinthananda Tellambura, Electrical & Computer Engineering

Ioanis Nikolaidis, Computing Science

Abstract

A fair and optimal mechanism is required for allocating bandwidth to virtual machine migration in a WAN environment. In this thesis, we propose a dynamic resource allocation algorithm running in either centralized or distributed environments. The centralized version of our algorithm collects information from individual users and dynamically allocates bandwidth according to their demands. The distributed version of our algorithm is running on the internal nodes (e.g. routers) in the network. In the distributed case, we show that even when the routers and the users do not exchange allocation information, the allocation is still stable and optimal if the users are elastic users. Another interesting problem we solved is emergency handling, which is also critical in virtual machine live migration.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, we discuss the problem of allocating network bandwidth to several sites that are concurrently migrating virtual machines. The network is a private network with leased links. Each site in the network is under control from a central manager. In other words, there is no malicious node in the network, and the data traffic is controllable. We define live migration as a fast and reliable virtual machine migration process, which provides the lowest system downtime. During live migration, the performance of the virtual machine is maintained at the maximum. If the system downtime is less than a few milliseconds, the user of the virtual machine will not even notice that the virtual machine has been migrated from one physical host to another.

The system we consider consists of a central physical machine and several distributed clients. We call the central physical machine the Central Manager Server (CMS.) The distributed clients each host one or more virtual machines; we call these clients VMHost. Each VMHost is connected to the CMS via a wide area network. The CMS is responsible for managing the VMHosts, and provides several useful services.

The first service that the CMS provides is virtual machine backup. Because physical hard disk is vulnerable, a periodic backup of the disk image of the virtual machine is necessary. Each virtual machine periodically sends its disk image to the CMS. Normally the virtual machine keeps running on the VMHost during the backup process. The user of the virtual machine will not notice the transmission of the backup data between the VMHost and the CMS.

The second service is disaster recovery. When the VMHost suffers a power failure or other emergency, the CMS can start up the backup virtual machines for that VMHost on some other VMHost. When the VMHost is recovered, the CMS will start migrating the virtual machines back to the original VMHost.

There are several problems that must be solved to properly and reliably supply these services. The first problem is to provide a virtual machine migration mechanism. The second problem is on the VMHost level. Each individual VMHost can host several virtual machines, and these virtual machines share the same network adapter of that host. As a result, internal bandwidth competition is inevitable. The goal is to provide a local bandwidth sharing scheme. The third problem is on the VMHosts-CMS level. The VMHosts are connected to each other and to the CMS over low bandwidth, high delay WAN links. The services envisioned can require the transfer of large data sets, and we face the competing goals of quick transfers for individual VMHosts whilst not starving any one site. Fair and efficient bandwidth allocation in an environment of dynamically changing demands is required. We assume that we are dealing with leased links so that we are not faced with competition from other anonymous, uncontrollable sources, as would be the case in the Internet. The work reported in this thesis focuses on this third problem.

However, we will show that our work is not limited to this specific problem. The algorithm designed for this specific application is general, and can be applied to many other areas. Moreover, we will see that a centralized algorithm has many limitations. Hence, we strive to extend our work and to find a better solution for generic resource allocation problems.

In this chapter, we give a brief discussion of our network resource allocation problem, describe the network model we use in our work, discuss the new features of our algorithm, and finally give an outline of the remaining chapters.

1.1 Background

Network resources, such as link bandwidth, are limited. Research in the field of resource allocation has received extensive attention in recent years. The goal

of this research has been to provide a scheme which allocates the resources in a fair and efficient manner. As a result, a number of centralized and distributed resource allocation schemes have been suggested.

The typical centralized resource allocation scheme has a client-server structure. The clients are the multiple users competing for finite network resources, *i.e.* link bandwidth. The server is the central manager which is responsible for performing the task of resource allocation. In order to allocate the resources, the central manager needs to know the status of all the users and the links in the network. Two methods have been developed to collect the information. The first one is the active method.The central manager periodically collects information from the users. Once the manager has enough information, it starts a new round of allocation. The second one is the passive method. If the user needs to change its current allocation, it sends the request to the central manager. Then the manager reacts accordingly. As we noted above, our first proposal is based on the centralized resource allocation scheme. The CMS is the central server, which is responsible for collecting the bandwidth requirements of each VMHost. The VMHost receives a bandwidth allocation from the CMS, and limits its bandwidth usage. We used active method to collect information in our work.

Although we solved the problems as defined above, there are several disadvantages of the centralized allocation scheme. First, the central manager is always required to perform the resource allocation task. If the central manager fails, then the allocation process will fail. The second disadvantage is that the communication between the users and the central manager will introduce overhead and therefore reduces efficiency. Because network resources are limited, any overhead can degrade system performance. The third disadvantage is that the method does not scale well. For example, it is hard to maintain a central manager in a network as large as the Internet.

To solve these problems, we extended the centralized resource allocation scheme to a distributed version. The structure of the distributed resource allocation scheme is similar to that of an ad-hoc network. There is no central manager in this scheme. Each user has a predefined evolving function which

updates its bandwidth demand. The user collects feedback (*e.g.* propagation delay or packet loss) from the network and reacts accordingly. The feedback might not be a precise reflection of the actual change in the bandwidth change in the network. For example, TCP is a typical window-based protocol which controls the user's transmission rate by collecting its packet loss rate. Once TCP detects a packet loss, it reduces the sending window size, and therefore reduces the transmission rate of the user. Because many factors (*e.g.* queue overflow on the router or bit corruption of the packet) can cause packet loss, TCP cannot tell the difference between these factors. As a result, a decrease of the user's transmission rate may be unnecessary, and therefore the performance is degraded unnecessarily. However, we will show that if all the users in the network are elastic users, then the result of the bandwidth allocation will eventually be stable and optimal.

Another debate concerns whether we choose static or dynamic allocation. A static allocation scheme distributes network resources in a prior reservation manner. Each user is allocated a fixed amount of bandwidth. In real networks, however, data flows do not last forever but arrive and leave at random times. Consider a case where two users are competing for a single link in the network. If we use a static allocation to distribute bandwidth to these users, then each of them could be assigned half of the bandwidth of the link. However, assume that one user is browsing the Web while the other is downloading files. The first user might not always fully utilize its allocated bandwidth, and the second might always be short of bandwidth. In this case, a significant amount of bandwidth is wasted due to the variation in the dynamic demands of the users. Therefore, we focus on using a dynamic allocation scheme, which adjusts the allocation result according to the users' demands for bandwidth.

We show that our method is very general and is applicable to a broad range of services. For example, although our algorithm runs in a centralized manner, we still have the flexibility to convert it to the distributed version. We will show this later in this paper. Another flexible aspect of our algorithm is that it is a general resource allocation algorithm that solves not only the network bandwidth allocation problem, but also problems like process scheduling in

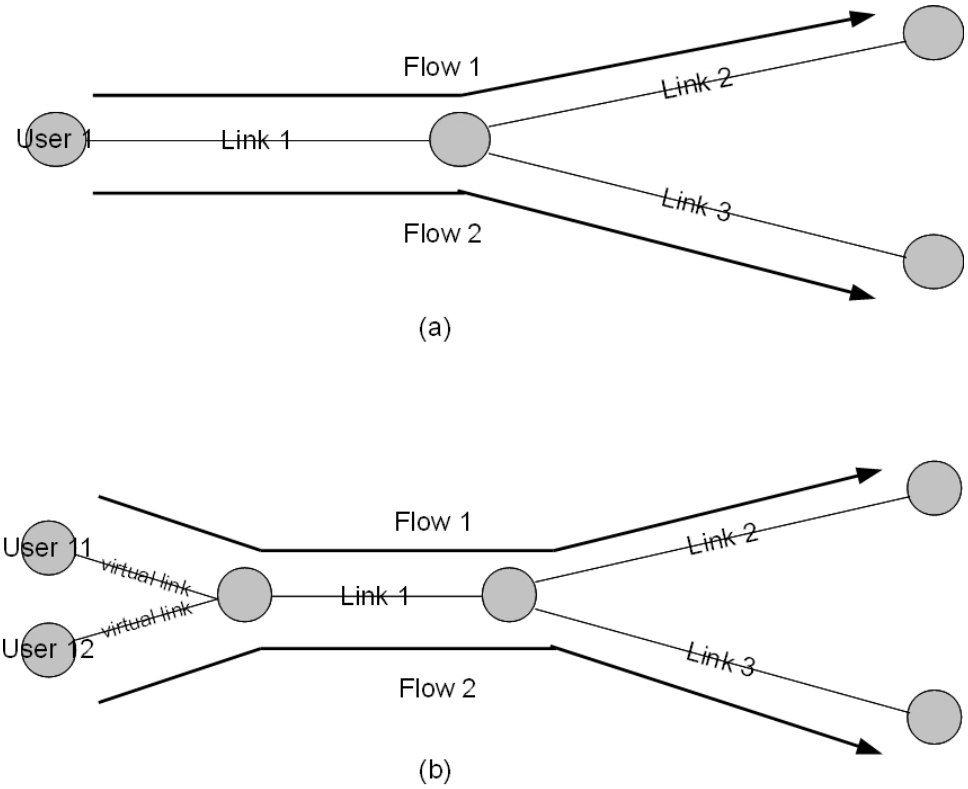


Figure 1.1: (a) Network model before conversion (b) Network model after conversion

the area of operating systems, *etc.*

1.2 Network Model

Consider a network NET which contains a set of users U and a set of links L. U contains n users while L contains m links. We define user U_i as a user in set U, where $i \in [1, n]$. We define link L_j as a link in set L, where $j \in [1, m]$. Link L_j has capacity C_j. We define the route of user U_i to be the concatenation of links through which the user's data flows, and we denote this route R_i. Accordingly, we use $L_j \in R_i$ to denote that link L_j is on the route R_i of user U_i. Each user U_i is allocated bandwidth x_i.

In the real world, there can be many users located at one network node. Moreover, each of these users can have multiple data flows. For example, one user can have both a web browser and file downloads running simultaneously. For our purposes, we assume that each independent flow constitutes a separate

user. For example, in the network topology shown in Fig. 1.1(a) there are three links L_1, L_2, and L_3. User U_1 has two flows. L_1 and L_2 are on route one of U_1, and L_2 and L_3 are on route two of U_1. In the formulation that follows, we would separate U_1 into two separate users U_{11} and U_{12}, and allocate bandwidth to the two flows separately.

To account for the divided users located in the same physical node, we introduce the concept of a 'virtual link' to distinguish this special link from real links. A 'virtual link' is a link with infinite capacity. Because we divide one physical user into multiple sub-users, each sub-user needs a 'virtual link' to connect to the network. Because a 'virtual link' has infinite capacity, it has no effect on the allocation result. The converted network is shown in Fig. 1.1(b).

We assume that the direction of the data flow of each user is unique. That is, the source and destination of each flow are fixed. This is because network links are typically full duplex, and therefore flows in opposite directions on the same link do not interfere with each other. If one user has flows in both directions on its route, then we divide this user into two sub-users. Each sub-user owns one of the two flows.

We also assume that each user in the network must use one or more links. Otherwise it would never compete with the remaining users in the network and allocating bandwidth to such a user would be meaningless.

The objective of the resource allocation is to provide a fair and efficient allocation result. In our work, we define an efficient allocation as:

$$Objective : max \sum_i x_i \tag{1.1}$$

$$Subject\ to : \sum_{L_j \in R_i} x_i < C_j \quad \forall j \tag{1.2}$$

$$x_i > 0 \quad \forall i \tag{1.3}$$

where R_i is the sequence of links followed by the route of flow i.

According to our definition, the objective of an efficient bandwidth allocation is to utilize the network resources to maximum. The first constraint confines the allocations not exceeding the bandwidth capacity contraints. The second constraint requires that each user has positive allocation.

Next we define a fair allocation as follows: the network should not favor certain users while starving others. In other words, no user receives special treatment. By careful examining of the objective we defined above, it is easy to see that this objective is not fair. Hence a balance between fairness and efficiency is needed.

1.3 Dynamics of the Network

As we discussed in the background section, real network applications vary their bandwidth demands quite often. Therefore, a static allocation scheme lacks flexibility in the presence of dynamic loads, and leads to a potential waste of network resources.

Our work combines prior allocation with a flexible dynamic scheme, which adjusts the allocation according to the user's real-time demand for bandwidth. We use prior allocation to reserve some initial resources to each user. Then the central manager periodically collects feedback from each user and performs dynamic resource allocation.

The prior allocation is based on the concept of max-min fairness [1]. Max-min fair allocation guarantees both fairness and efficiency. We will see that max-min fairness guarantees that the most poorly treated user can receive as many of the resources as possible. In other words, the network is not biased to users with any special features.

The dynamic allocation includes two steps. The first one is the share/reclaim step. If the demand of bandwidth of user U_i can be satisfied and U_i has extra resources, then the user may be willing to share the unused resources with other users. On the other hand, if U_i shared any resources before and requires more resources, it starts to reclaim that bandwidth. After the share/reclaim step, the algorithm enters the borrow step. In this step, those users which are

still not satisfied try to borrow resources from others.

The share/reclaim/borrow steps in our dynamic allocation are novel because they allow collaboration among the users. Previous works assumed that network users always behave aggressively. The user does not consider the demands of other users. In our work, we give the user the ability to decide whether or not to share its unused bandwidth. Other users can take advantage of this unused bandwidth. Therefore the network resources can be utilized at maximum. Our scheme also has the ability to control the process of the share/reclaim/borrow steps. The user can share/reclaim its unused bandwidth quickly or slowly. Hence, sharp variance of network bandwidth might be avoided.

1.4 Handling Emergencies

Occasionally the user may face an emergency situation and want to transfer a large bulk of data within very short time. In our problem, when the VMHost recovers from failure, the CMS will start transfering the backup virtual machines back to the VMHost. In order to provide quick recovery, it requires peak bandwidth usage in a short time period. Therefore, emergency handling is a very important task that the central manager needs to take into consideration.

Few previous studies provide a solution for emergency handling. In our work, we show that there are multiple solutions to this problem due to the fact that, by using our algorithm, users in the network can collaborate with others. For example, the user may quickly raise its bandwidth demand, or the manager can suppress the bandwidth demands of the non-emergency users. We compare the pros and cons of these solutions in later chapters.

1.5 Related Work

There has been considerable work done on fair allocation. For example, serveral fair algorithms for sharing network resources are studied in [3], [7]. Kelly *et al* proposed the idea of proportional fairness [4], [15]. Their goal was

to maximize the total utility for all users. These users are assumed to offer elastic traffic, and can thus adjust their rates based on network feedback.

We chose max-min fair allocation as the initial static allocation scheme. Bertsekas and Gallager [1] adapted max-min fairness from its original use in econometrics to data networks. Recently considerable research has been devoted to the issue of providing optimal resource allocation in communication networks. For example, in [8] a unified treatment of max-min fairness applied to a network was proposed. We will give detailed discussion of max-min fairness in the next chapter.

References [5], [11], [12] describe distributed algorithms to achieve fair allocation. However, these are distributed algorithms which do not satisfy our requirement.

Moreover, a more general form of max-min fairness called weighted max-min fairness has also been extensively studied. In [6], White proposed an algorithm to achieve weighted max-min fairness. In [2], Marbach studied a pricing scheme which provides differentiated service by using priorities. The result leads to a weighted max-min fair allocation. We use the concept of weighted max-min fairness in our work. We show that we can achieve weighted max-min fairness in both the centralized and the distributed manners.

In [10], Chandra *et al* proposed a distributed dynamic resource allocation algorithm. In [16], Afek *et al* proposed a bandwidth allocation scheme which initially preserves some bandwidth for 'phantom' session on the link. However, a distributed algorithm does not always have full information about the network, and hence the allocation result is not as precise as that from a centralized computation.

In [13], Bar-Noy *et al* proposed a bandwidth allocation scheme with the ability to abort the data flow of some users to preserve bandwidth for others. However, this might not lead to a fair allocation because some users could be starved when their data flow is preempted.

In [14], Korilis *et al* introduced a distributed bandwidth allocation scheme. In their scheme, the users in the network are noncooperative. However, in our work we assume that the users cooperate with others to enable the share/reclaim/borrow

steps we discussed above.

In [9], a detailed discussion of rate adaptation, congestion control and fairness is provided. It provides knowledge such as how existing network protocols, *i.e.* TCP, would affect our algorithm.

Our work adjusts max-min fair allocation as well as dynamic resource allocation in both the centralized and distributed manners. It does not require synchronization of the network. However, in the centralized algorithm the users should continuously provide feedback to the central manager for the dynamic allocation procedure. We show that if the user flows remain stable, the allocation result will eventually be optimal and stable.

1.6 Outline

The rest of the thesis is organized as follows. In Chapter 2, we discuss the max-min fairness, and we extend the max-min fairness to weighted max-min fairness. We also provide an algorithm which can achieve weighted max-min fair allocation. In Chapter 3, we introduce the centralized version of our dynamic resource allocation algorithm. In Chapter 4, we extend the allocation algorithm to a distributed version. The distributed allocation algorithm is based on the centralized allocation algorithm. In Chapter 5, we discuss emergency handling. In Chapter 6, we give experimental results and discuss the pros and cons of both the centralized and the distributed allocation algorithms.

Chapter 2

Max-Min Fairness

In this chapter we discuss max-min fairness, which is the base of our work. We explain the reason why we adopt the max-min fairness into our work, and we propose the pros and cons of max-min fairness with other feasible solutions. In the first two sections we give definitions of max-min fairness and weighted max-min fairness from previous works. In the third section we give our algorithm which leads to weighted max-min fair allocation.

There exists many forms of fairness, *e.g.* max-min fairness, proportional fairness, *etc.* In [1] Bertsekas and Galager proposed an algorithm which can achieve max-min fairness. The result of max-min fairness guarantees that the most poorly treated user in the system is allocated with as much resource as possible. In other words, the increase of the allocation of one user U_p is at the cost of the decrease of the allocation of another user U_q, where x_q is equal to or greater than x_p; and the cost must be greater than the benefit that user U_p can gain from user U_q.

Kelly *et al* proposed the idea of proportional fairness [4]. Their goal was to maximize the total utility for all users. If allocation result vector $\{x_i\}$ is proportional fairness, and there exists another feasible allocation vector $\{x_{i'}\}$, then we have:

$$\sum_{i\in[1,n]} \frac{x_i - x_i'}{x_i} < 0 \tag{2.1}$$

These users are assumed to offer elastic traffic, and can thus adjust their rates based on network feedback. Proportionally fair allocation requires feed-

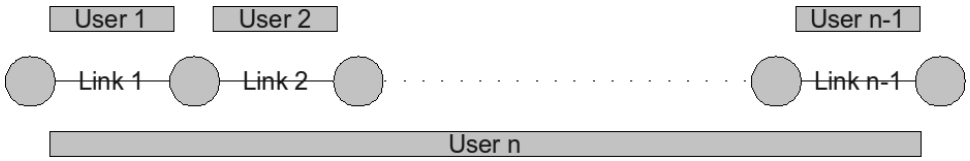


Figure 2.1: A simplified network to demonstrate the fairness of max-min

back from the network. In our work we require the central manager to collect information from users. The proportional fair allocation does not consider this. Therefore, we chose to use max-min fairness as the basis of our allocation scheme.

2.1 Definition of Max-Min Fairness

Max-min fairness is a classical sharing principle in the field of data networks. First we give a formal definition of max-min fairness.

Definition 2.1. *A vector of resource allocation* $\{x_i\}$ *is said to be max-min fair if, for any other feasible allocation* $\{x'_i\}$*, the following is true: if* $x'_p > x_p$ *for user* U_p*, then there exits another user* U_q *such that* $p, q \in [1, n]$*,* $x'_q < x_q$*, and* $x_q \leq x_p$*.*

This definition means that the result of max-min fair allocation is such that the result for the user with smallest allocation has been maximized over all feasible vectors. Then, given the user with the smallest allocation, the allocation of the user with second smallest allocation is maximized, etc.

Max-min fair allocation is more fair than equal-sharing allocation scheme. To illustrate this, consider a simplified network shown in Fig. 2.1. Assume that the number of the links, m, is one less than the number of users: that is, $m = n - 1$. We also assume that each link has the same capacity C. Let user $U_1, U_2, \ldots, U_{n-1}$ use only the single link $L_1, L_2, \ldots, L_{n-1}$, and user U_n uses all the links. Therefore, U_n is competing with all other users in the network. The bars in the figure denote the data flows. Assume that the direction of the data flows is from left to right. If U_n is allocated bandwidth x_n, then we have:

$$x_1 = x_2 = \ldots = x_{n-1} = C - x_n \tag{2.2}$$

We can easily see that in order to maximize the bandwidth usage, we should let $x_n = 0$ and $x_i = C$ where $i \in [1, n-1]$. Hence we have the sum of the bandwidth usage as:

$$\sum_{i \in [1,n]} x_i = (n-1)C \tag{2.3}$$

Therefore, user U_n gets starved because U_i, where $i \in [1, n-1]$, takes the whole available bandwidth.

If we apply max-min fair allocation to this problem, we will get:

$$x_1 = x_2 = \ldots = x_{n-1} = x_n = \frac{C}{2} \tag{2.4}$$

And therefore the sum of the bandwidth usage is:

$$\sum_{i \in [1,n]} x_i = \frac{nC}{2} \tag{2.5}$$

As we can see, the total sum of the bandwidth usage of max-min fair allocation is nearly half of that of simple equal-sharing allocation. However, bandwidth for users of longer routes are preserved. Max-min sacrifices efficiency for fairness.

In order to further understand the properties of max-min fairness, we introduce the concept of a bottleneck link.

Definition 2.2. *Link L_j is said to be the bottleneck link of user U_i if for all users $U_{i'}$ with $L_j \in R_{i'}$ where $i \in [1, n]$ and $U_{i'} \neq U_i$ we have:*

$$x_i + \sum x_{i'} = C_j \tag{2.6}$$

and

$$x_i \geq x_{i'} \tag{2.7}$$

Now we can define max-min fairness by using the concept of bottleneck link:

Definition 2.3. *A resource allocation vector $\{x_i\}$ is said to be max-min fair* iff *each user U_i has a bottleneck link.*

Proof. Suppose that x_i is a max-min fair allocation, but user U_k does not have a bottleneck link. Define:

$$X_j = \sum_{L_j \in R_i} x_i \tag{2.8}$$

Therefore, for each link $L_j \in R_k$, one of the following must be true: either $X_j < C_j$ or there exists user $U_{k'} \neq U_k$ such that $L_j \in R_{k'}$ and $x_{k'} > x_k$. For each link L_j, define the available bandwidth for U_k to be:

$$\Delta_j = C_j - X_j, \ if \ X_j < C_j \tag{2.9}$$

or

$$\Delta_j = x'_k - x_k, \ if \ X_j = C_j \tag{2.10}$$

Thus, if x_k is incremented by $max_{L_j \in R_k}(\Delta_j)$, then only users whose bandwidth allocation are larger than U_k will be affected, while all other users remain unaffected. This contradicts the definition of max-min fairness, and thus there must be a bottleneck link for each user.

In the reverse direction, assume that each user has a bottleneck link under allocation $\{x_i\}$. Suppose user U_k increases its allocation, then at least one of the users at each of its bottleneck links must have their allocation reduced to maintain feasibility. But all the users at a link have allocation less than or equal to x_k. Thus, $\{x_i\}$ must be a max-min fair allocation. □

Because we run the allocation algorithm in a centralized manner, the data about the users and the links are known. Therefore, the max-min fair allocation can be obtained by using a water-filling algorithm. We first denote the average bandwidth allocation of link L_j by $\tau_j = C_j/n_j$, where n_j is the number of users that use link L_j. If n_j is zero, we set τ_j to zero. The algorithm starts with all allocations equal to zero. It then finds the link L_{min} such that L_{min} has the minimum τ_{min}. Without loss of generality, we assume that users $U_1, U_2, \ldots, U_k$ are using link L_{min}. Next, we set $x_1, x_2, \ldots, x_k$ to τ_{min}. After the first step, we subtract users $U_1, U_2, \ldots, U_k$ from the user set U. Then we

update τ for the remaining links. Link L_{min} as well as any links with $\tau = 0$ are then removed from the link set L. We iterate until all users have bottleneck links.

2.2 Weighted Max-Min Fairness

Differentiated quality of service has received much attention because the simple max-min fair allocation is not capable of providing differential service to users based on variations in priority or other requirements. For example, consider a network with a single link L_1. Users $U_1, U_2, \ldots, U_n$ are competing for the bandwidth C_1 of link L_1. If some users pay more than others, it is reasonable to allocate more bandwidth to these users. Hence, it would be necessary to prioritize users by the amount they pay. In this section, we adjust the max-min fair allocation to accommodate this kind of problem. The name of the adjusted allocation scheme is weighted max-min fair allocation.

In weighted max-min fairness, a new parameter ω_i called 'priority' is assigned to each user U_i. The higher the priority of the user, the more bandwidth it can receive from the weighted max-min fair allocation. First we give a more general definition of max-min fairness.

Definition 2.4. *A vector of resource allocation $\{x_i\}$ is said to be max-min fair if, for any other feasible allocation $\{x'_i\}$, the following is true: if $f(x'_p) > f(x_p)$ for user U_p, then there exists some user U_q such that $p, q \in [1, n]$, $f(x'_q) < f(x_q)$ and $f(x_q) \leq f(x_p)$.*

Here we define $f(\cdot)$ as a function which maps the resource allocation of U_i to a finite number. For weighted max-min fairness, we have:

$$f(x_i) = \frac{x_i}{\omega_i} \tag{2.11}$$

Moreover, we modify the definition of the bottleneck link:

Definition 2.5. *Link L_j is said to be the bottleneck link of user U_i if for all users $U_{i'}$ with $L_j \in R_{i'}$ where $i' \in [1, n]$ and $U_{i'} \neq U_i$ we have:*

$$x_i + \sum x_{i'} = C_j \tag{2.12}$$

and

$$f(x_i) \geq f(x_{i'}) \tag{2.13}$$

2.3 Algorithm of Weighted Max-Min Fair Allocation

In order to modify the usual max-min fairness to become weighted max-min fair, we need to modify the parameters we used in the previous section. First, we introduce a new parameter for each link in the network. We define the total priority Ω_j of link L_j as the sum of the priorities of the users using the link:

$$\Omega_j = \sum_{L_j \in R_i} \omega_i \tag{2.14}$$

We also need to modify the definition of the average bandwidth allocation of link L_j. In the previous section, we defined the average bandwidth allocation of link L_j as the equal share among the users using such link. Since we are now introducing the concept of priority, we modify τ_j to:

$$\tau_j = \frac{C_j}{\Omega_j} \tag{2.15}$$

Then the definition of the average bandwidth allocation of link L_j changes to: τ_j **is the average bandwidth per unit of priority on link** L_j. With these definitions, max-min fair allocation is a specific case of weighted max-min fair allocation, where the priority of each user is set to one.

Next we give the algorithms of weighted max-min fair allocation. (see next page)

The complexity of Algorithm 2.2 is $O(n)$ because link L_j has to scan through all the users competing for its bandwidth. The complexity of Algorithm 2.3 is $O(n)$ because link L_{min} still needs to scan through the user list.

Algorithm 2.1 WeightedMaxMin(U, L)

```
allocatedUserNr = 0
while allocatedUserNr < n do
  for all L_j in L do
    UpdateLink(L_j)
  end for
  find L_min of minimum τ
  allocatedUserNr = allocatedUserNr + AllocateLink(L_min)
  remove L_min from L
  remove L_j with negative τ_j from L
end while
```

Algorithm 2.2 UpdateLink(L_j)

```
capacity = C_j
priority = Ω_j
for all U_i such that L_j ∈ R_i do
  if U_i has already been allocated then
    capacity = capacity-x_i
    priority = priority-ω_i
  end if
end for
if capacity == 0 then
  Ω_j = -1
else
  Ω_j = capacity / priority
end if
```

The main loop in Algorithm 2.1 will iterate at most m times. Therefore, the complexity of the algorithm is $O(mn)$.

We prove that Algorithm 2.1 results in a max-min fair allocation. We need to prove that after the allocation, each user has a bottleneck link in the network.

Proof. We denote step t of the while iteration in Algorithm 2.1 as $Iter(t)$. In each iteration, one and only one link would be chosen. We denote the link being chosen at $Iter(t)$ as $L_{min}(t)$. $L_{min}(t)$ has the minimum τ value among the links at $Iter(t)$. We denote the unallocated user set in which users compete for $L_{min}(t)$ as $U_{L_{min}(t)}$. Define $U_i \in U_{L_{min}(t)}$ as user U_i in the set. According to Algorithm 2.3, U_i's allocation is:

Algorithm 2.3 AllocateLink(L_{min})

```
allocatedUserNr = 0
for all U_i such that L_min ∈ R_i do
  if U_i has not been allocated then
    x_i = Ω_min * ω_i
    set U_i as been allocated
    allocatedUserNr = allocatedUserNr + 1
  end if
end for
return allocatedUserNr
```

$$x_i = \tau_{L_{min}(t)} * \omega_i \tag{2.16}$$

Assume that user $U_{i'}$ is allocated at $Iter(t')$, where $t > t'$. Because $L_{min}(t')$ is chosen before $L_{min}(t)$, we have:

$$\tau_{L(t')} \leq \tau_{L(t)} \tag{2.17}$$

Since we have:

$$f(x_i) = \frac{x_i}{\omega_i} = \tau_{L(t)}, \forall\ U_i \in U_{L(t)} \tag{2.18}$$

We know that $f(x_i) > f(x_{i'})$ for all $U_{i'}$ who is allocated before U_i. And for user U_k who is allocated in the same iteration with U_i, it is easy to see that $f(x_k) = f(x_i)$. Hence we claim that $L_{min}(t)$ is the bottleneck link of U_i. Because the iteration terminates only when all the users are allocated, it is guaranteed that each user has a bottleneck.

Next we prove that the algorithm can terminate. Assume that user U_k never gets allocated. In line 3 of Algorithm 2.1, the for loop scans through the link set of the network. In Algorithm 2.3, unallocated users using link $L_{min}(t)$ will be marked as allocated if they are not marked before $Iter(t)$. If U_k never gets allocated, then it means that none of the link in the network can mark U_k. In other words, U_k uses none of the links. This contradicts the assumption that each user must use at least one of the link in the network. Therefore, all users must be allocated in the iteration of Algorithm 2.1 and the algorithm can terminate.

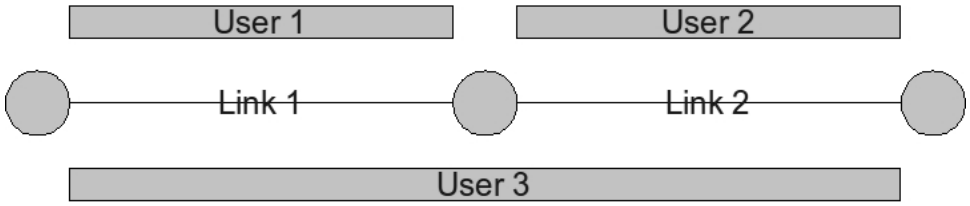


Figure 2.2: An example network to demonstrate weighted max-min fair allocation

In sum, each user has a bottleneck link and thus the allocation is max-min fair. □

To illustrate how this algorithm works, we give a simple example. Consider three users U_1, U_2, and U_3 sharing two links L_1 and L_2 in the network shown in Fig. 2.2. The priorities of the three users are:

$$\omega_1 = 1.0, \omega_2 = 2.0, \omega_3 = 3.0 \tag{2.19}$$

And the capacities of the two links are:

$$C_1 = 100Mbps, C_2 = 50Mbps \tag{2.20}$$

Referring to Algorithm 2.1, we have:

$$\tau_1 = \frac{100}{1.0 + 3.0} = 25Mbps \tag{2.21}$$

$$\tau_2 = \frac{50}{2.0 + 3.0} = 10Mbps \tag{2.22}$$

Therefore, link L_2 is chosen as L_{min}, and user U_2 and U_3 are allocated first. According to Algorithm 2.3, we have:

$$x_2 = 10 * \omega_2 = 20Mbps \tag{2.23}$$

$$x_3 = 10 * \omega_3 = 30Mbps \tag{2.24}$$

After the first iteration, only U_1 remains unallocated. Because U_3 has already been allocated with 30Mbps bandwidth, it would also consume 30Mbps

on L_1. Therefore, the bandwidth remaining for U_1 on L_1 is 70Mbps. Since U_1 is the only user using link L_1, we have:

$$\tau_1 = \frac{70}{1.0} = 70Mbps \tag{2.25}$$

$$x_1 = 70 * \omega_1 = 70Mbps \tag{2.26}$$

2.4 Summary

In this chapter, we give the definitions of max-min fairness and weighted max-min fairness from previous works. We also introduce our algorithm to compute a weighted max-min fair allocation. We use a simplified example to show the steps of our algorithm.

Chapter 3

Centralized Dynamic Resource Allocation Scheme

As we can see in the previous chapter, the weighted max-min fair allocation can only provide a prior bandwidth reservation for the users. Considering the dynamics of the network, a prior bandwidth reservation might lead to great waste of limited network resources. In order to deal this problem, we develop a new resource allocation scheme which takes advantage of weighted max-min fair allocation. In this chapter we introduce the centralized dynamic resource allocation scheme.

3.1 System Model

We have a network NET consisting of user set U and link set L. There exists a unique network node M that serves as the central manager. M is responsible for performing the allocation task, and for collecting information from each user. The system has two states: initial allocation state and dynamic allocation state. During the initial allocation, the manager assigns each user an initial allocation. Then the manager enters the dynamic allocation state. In this state, the manager periodically collects information from each user. Based on this information, the manager runs the dynamic resource allocation algorithm and sends each user an updated allocation. This manager stays in this state until the topology of the network changes. At that point the manager returns to the initial allocation state. A change of the topology of

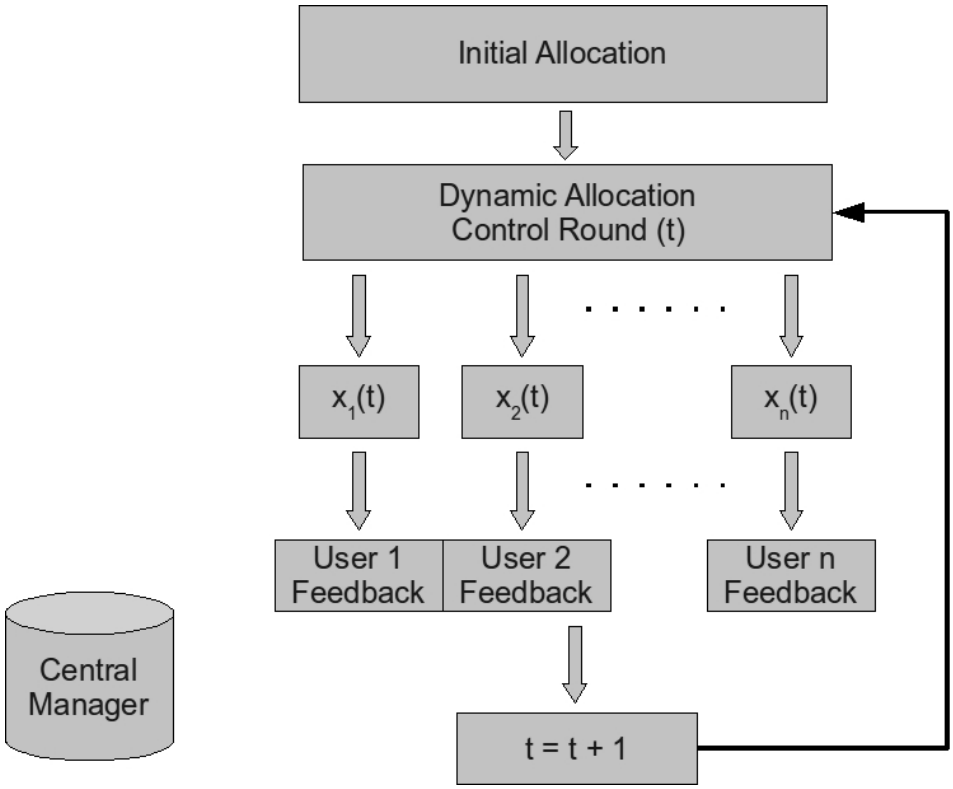


Figure 3.1: System model of the centralized dynamic resource allocation scheme

the network occurs when:

- a new user enters the network, or
- an existing user leaves the network, or
- a new link is added to the link set L, or
- a link is deleted from the link set L.

The central manager has a timer which it uses to time the duration of each 'control period' Γ_M. This is the interval between each round of sending the users their updated allocations. Let us consider Γ_M as one time unit. We use the term 'control round' to represent one complete iteration of the dynamic allocation procedure. Thus, when we say that $t = t + 1$, the actual time that has elapsed is Γ_M. The system model is shown in Fig.3.1.

Next we describe in detail of the two states of the manager.

3.2 Initial Allocation State

In the initial allocation state, the manager runs the weighted max-min fair algorithm. Each user records its initial allocation in the variable σ_i. This variable acts as the baseline for future dynamic adjustments. After the initial allocation, we have:

$$\sigma_i = x_i(0) \tag{3.1}$$

After this, the manager enters the dynamic allocation state.

3.3 Dynamic Allocation State

Our dynamic allocation algorithm uses the concepts of sharing, reclaiming and borrowing bandwidth with/from other users as required.

Share: If user U_i requires less bandwidth than its current allocation, then user U_i has some extra bandwidth. In order to utilize this unused bandwidth, user U_i can decide whether to share it with other users.

Reclaim: If user U_i requires more bandwidth than its current allocation, and if U_i has already shared some of its initial allocation, then user U_i can reclaim that shared bandwidth.

Borrow: This is related to Share and Reclaim. If user U_i's demand for bandwidth cannot be satisfied by its allocation even after the reclaim step, then it tries to borrow unused (shared) bandwidth from others.

In order to perform the dynamic allocation, we introduce several parameters:

θ_i records how much bandwidth user U_i has shared. If user U_i's demand for bandwidth decreases then it adds the unused bandwidth to θ_i. On the other hand, if user U_i's demand increases, and if the current allocation cannot satisfy its demand then U_i would start to reclaim bandwidth until θ_i is zero or the demand drops again.

δ_i is the share factor of user U_i. We use this parameter to control the speed of the sharing process. As δ_i increases, user U_i shares its unused bandwidth more quickly. The value of δ_i is bounded in $[0, 1]$. This is because U_i can only share a fraction of its unused bandwidth. If δ_i equals zero then U_i is not willing to share any of its unused bandwidth. If δ_i equals one then U_i is willing to share all of its unused bandwidth in one control round.

ρ_i is the reclaim factor of user U_i. We use this parameter to control the speed of the reclaim process. As ρ_i increases, user U_i reclaims its shared bandwidth more quickly. As with δ_i, ρ_i is bounded in $[0, 1]$. If ρ_i equals zero then U_i will not reclaim any of its shared bandwidth. If ρ_i equals one then U_i will reclaim all of its shared bandwidth in one control round.

η_i is user U_i's predicted demand for bandwidth in the next control round.

ϵ_i if η_i cannot be satisfied by merely using U_i's current bandwidth allocation, then U_i records the gap in ϵ_i. Later U_i will seek shared bandwidth to fill the gap.

κ_i records how much bandwidth U_i has borrowed from other users.

The new parameter for link L_j is:

Θ_j records the sum of the shared bandwidth of the users using L_j. That is:

$$\Theta_j = \sum_{L_j \in R_i} \theta_i \tag{3.2}$$

During each control round the manager sends $x_i(t)$ to each user U_i, and waits for $\eta_i(t+1)$. Once all the users finish reporting $\eta_i(t+1)$, the manager starts the dynamic allocation procedure. According to $\eta_i(t+1)$, there are two strategies that the manager can use to reallocate bandwidth for each user U_i.

3.3.1 Share Strategy

If we have:

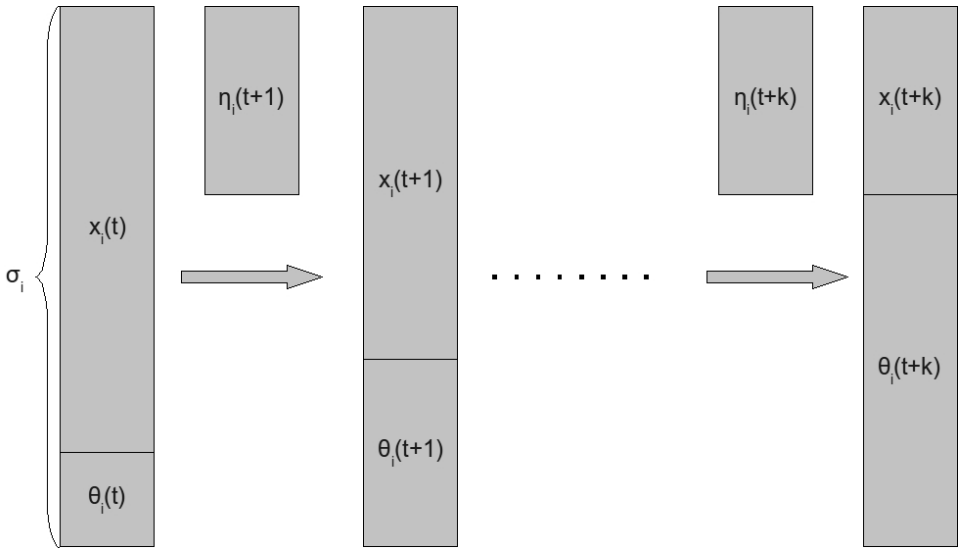


Figure 3.2: Share strategy. If the user's bandwidth demand remains stable, it will eventually share all of its unused bandwidth.

$$\eta_i(t+1) < x_i(t) - \kappa_i(t) \tag{3.3}$$

then user U_i is asking for less bandwidth in the next control round $t+1$. The user may be willing to share its unused bandwidth with other users. δ_i is used to control the amount of bandwidth that U_i can share. We have:

$$toShare_i(t+1) = (x_i(t) - \kappa_i(t) - \eta_i(t+1)) * \delta_i \tag{3.4}$$

$$\theta_i(t+1) = \theta_i(t) + toShare_i(t+1) \tag{3.5}$$

$$x_i(t+1) = \sigma_i - \theta_i(t+1) \tag{3.6}$$

$$\epsilon_i(t+1) = 0 \tag{3.7}$$

Fig. 3.2 shows the procedure of share strategy. If the user's demand remains stable, it would eventually share all the unused bandwidth.

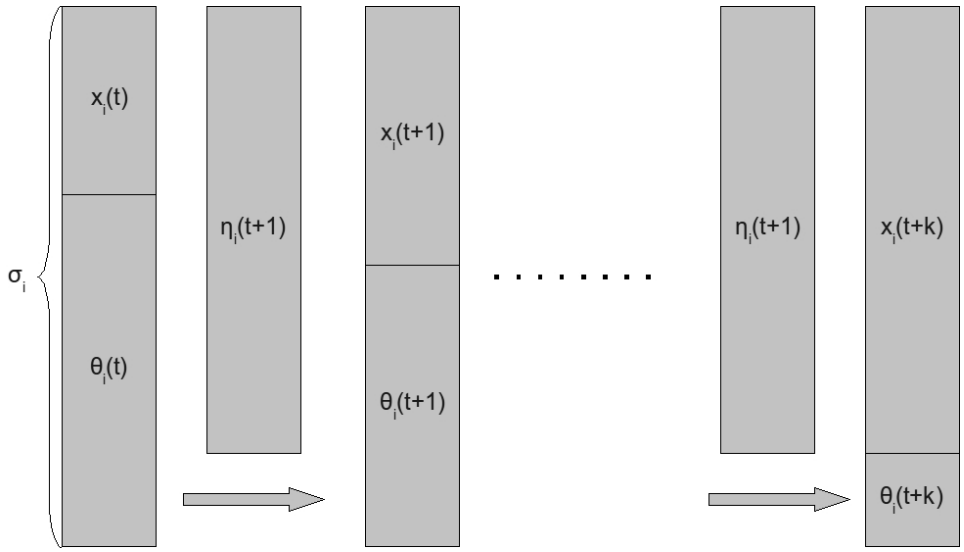


Figure 3.3: Reclaim strategy. If the user's bandwidth demand remains stable, it will eventually reclaim all the shared bandwidth.

3.3.2 Reclaim Strategy

If we have:

$$x_i(t) - \kappa_i(t) \leq \eta_i(t+1) \tag{3.8}$$

then user U_i is asking for more bandwidth in the next control round $t+1$. If the user shared any bandwidth in the previous control round then it may reclaim that bandwidth. Hence we have:

$$toReclaim_i(t+1) = \theta_i(t) * \rho_i \tag{3.9}$$

$$\theta_i(t+1) = \theta_i(t) - toReclaim_i(t+1) \tag{3.10}$$

$$x_i(t+1) = \sigma_i - \theta_i(t+1) \tag{3.11}$$

$$\epsilon_i(t+1) = \eta_i(t+1) - x_i(t+1) \tag{3.12}$$

Fig. 3.3 shows the procedure of the reclaim strategy. If the user's demand remains unchanged, it will eventually reclaim all its shared bandwidth.

3.3.3 Update Link Share Pool

Once the manager finishes the share/reclaim steps for all users, it refreshes the share pool of the links. The size of the share pool of link L_j is the sum of the shared bandwidth of each individual user U_i if $L_j \in R_i$. For each link L_j we have:

$$\Theta_j(t+1) = \sum_{L_j \in R_i} \theta_i(t+1) \tag{3.13}$$

3.3.4 Borrow Procedure

After the share/reclaim step, user U_i's demand for bandwidth might still not be satisfied. The manager scans through each user to check if $\epsilon_i(t+1)$ is greater than zero. If so, the borrow procedure is started for user U_i.

In the borrow procedure, user U_i scans the links on its route. The link L_{min} is chosen that has the minimum Θ_{min} among the links on U_i's route. This is because Θ_{min} is the upper bound of the possible bandwidth that U_i can borrow. Once we find L_{min}, we compare $\Theta_{min}(t+1)$ with $\epsilon_i(t+1)$. There are two possible cases.

Case 1:

$$\Theta_{min}(t+1) < \epsilon_i(t+1) \tag{3.14}$$

User U_i's demand cannot be fully satisfied because the maximum possible shared bandwidth is less than its gap. We assign all the shared bandwidth to user U_i, and set the gap to be the difference between the shared bandwidth and the original gap:

$$\kappa_i(t+1) = \Theta_{min}(t+1) \tag{3.15}$$

$$\epsilon_i(t+1) = \epsilon_i(t+1) - \kappa_i(t+1) \tag{3.16}$$

Case 2:

$$\Theta_{min}(t+1) \geq \epsilon_i(t+1) \tag{3.17}$$

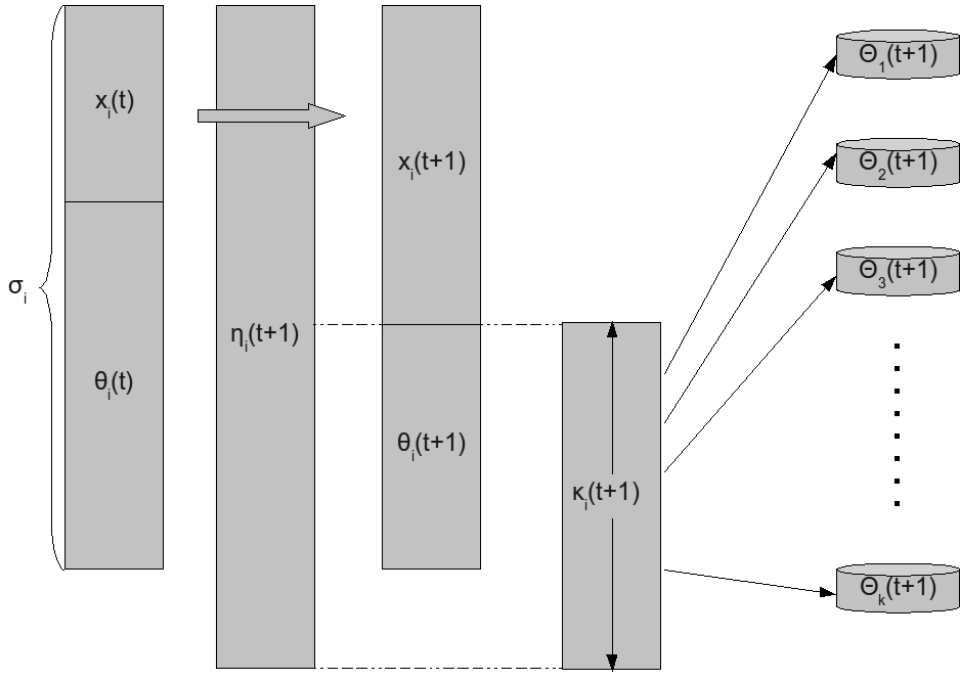


Figure 3.4: Borrow procedure. If the user's bandwidth demand cannot be satisfied even after reclaim, it will start to borrow bandwidth from other users.

User U_i's demand can be fully satisfied. We have:

$$\kappa_i(t+1) = \epsilon_i(t+1) \tag{3.18}$$

$$\epsilon_i(t+1) = 0 \tag{3.19}$$

Finally, we set the assignment of user U_i as:

$$x_i(t+1) = x_i(t+1) + \kappa_i(t+1) \tag{3.20}$$

After borrowing bandwidth, user U_i has to update its record of how much bandwidth it is using on each link on its route. This is because if U_i occupies shared bandwidth on link L_{min}, then this amount of bandwidth must also be occupied on the remaining links on U_i's route. Therefore, we have:

$$\Theta_j(t+1) = \Theta_j(t+1) - \kappa_i(t+1), \forall L_j \in R_i \tag{3.21}$$

Fig. 3.4 shows the borrow procedure.

3.4 Algorithm

In this section we describe our centralized resource allocation algorithm. We also analyze the complexity of the algorithm.

Algorithm 3.1 CentralizedAllocation(U, L)

```
WeightedMaxMin(U, L)
while manager is running do
  for all U_i in U do
    notify U_i with allocation x_i
    wait for U_i's feedback
  end for
  DynamicAllocate(U, L)
  sleep over one Γ_M
end while
```

Algorithm 3.2 DynamicAllocate(U, L)

```
for all U_i in U do
  UpdateUser(U_i)
end for
for all L_j in L do
  UpdateLink(L_j)
end for
for all U_i in U do
  if ε_i > 0 then
    Borrow(U_i)
  end if
end for
```

The complexity of the first step of Algorithm 3.1, WeightedMaxMin, is $O(mn)$. Line 3 to line 6 of Algorithm 3.1 takes $\Theta(n)$ to send allocation and receive feedback from users. Algorithm 3.2 has three for loops. The first loop calls Algorithm 3.3. Because in Algorithm 3.3 the user has to scan through its link list, the complexity of the first for loop is thus $O(mn)$. The second loop updates each link's share pool size, and it needs to scan through the user list of each link. Thus the complexity of the second loop is also $O(mn)$. The third loop is needed for users whose demand for bandwidth is not satisfied in the first loop. The complexity of this loop is $O(mn)$. Therefore, the complexity of Algorithm 3.1 is $O(mn)$.

Algorithm 3.3 UpdateUser(U_i)

if $\eta_i \geq x_i$ **then**
  $toReclaim_i = \theta_i$ * ρ_i
  $\theta_i = \theta_i$ - $toReclaim_i$
  $x_i = \sigma_i$ - θ_i
  $\epsilon_i = \eta_i$ - x_i
  if $\epsilon_i < 0$ **then**
    $\epsilon_i = 0$
  end if
else
  $toShare_i = (x_i$ - $\eta_i)$ * δ_i
  $\theta_i = \theta_i + toShare_i$
  $x_i = \sigma_i$ - θ_i
  $\epsilon_i = 0$
end if

Algorithm 3.4 UpdateLink(L_j)

Θ_j=0
for all U_i such that $L_j \in R_i$ **do**
  $\Theta_j = \Theta_j + \theta_i$
end for

Algorithm 3.5 Borrow(U_i)

find link L_{min} with minimum Θ_j, $forall L_j \in R_i$
toShare = Θ_{min}
if $\epsilon_i > \Theta_{min}$ **then**
  $\kappa_i = \Theta_{min}$
  $\epsilon_i = \epsilon_i$-κ_i
else
  $\kappa_i = \epsilon_i$
  $\epsilon_i = 0$
end if
$x_i = x_i + \kappa_i$
for all L_j such that $L_j \in R_i$ **do**
  $\Theta_j = \Theta_j$ - κ_i
end for

3.5 Discussion

Our algorithm leaves implementation flexibility to the central manager. In this section, we discuss how the algorithm can be applied to specific situations.

3.5.1 Control of the Share/Reclaim Process

In the real world, users with high priorities do not always have much data to transfer. Considering an ISP with a single 1.5Gbps link as an example. Two companies A and B are buying bandwidth from this ISP. Company A provides FTP service and pays $1,000. Company B provides e-mail service. Because the peak bandwidth requirement of the e-mail service is less than that of the FTP service, company B pays only $500 to the ISP. Because company A pays twice of company B, we have $\omega_A/\omega_B = 2$. According to the algorithm, we know that σ_A is 1Gbps, and σ_B is 500Mbps.

The separation of the priority and the share/reclaim factor leaves us great flexibility. High priority does not always mean high bandwidth usage. Because the number of customers using e-mail service is much more than that using FTP service, company A might always have some unused bandwidth. However, because company A pays much more than company B, it needs to control how company B utilize the unused bandwidth. Company A has two strategies:

Strategy 1 Company A can set its share factor δ_A very low so that the unused bandwidth would be shared very slowly. This provides a buffer time for company A if it decides to reclaim that bandwidth later, because only a relatively small fraction of the unused bandwidth has been shared.

Strategy 2 On the other hand, company A can also set its reclaim factor ρ_A very high. In that case, when company A starts the reclaim procedure, it can reclaim its shared bandwidth very fast.

The same strategies also apply to company B. The share factor and the reclaim factor control the speed of the allocation of the unused bandwidth. Small share factor and reclaim factor lead to slow share/reclaim procedures. This is desireble if we want to maintain a stable network environment. For

example, when a customer of company A starts downloading from the FTP server, the bandwidth allocated to company B would decrement very smoothly. The customers of company B would not suffer from sudden failure of sending e-mail due to the change of the bandwidth. On the other hand, if we set share/reclaim factors very large, the users would benefit from quick response according to their bandwidth demands.

3.5.2 Borrow Strategy

In our dynamic allocation algorithm, we did not propose any policy to control the borrow process. We simply search for users with gap. There are many possible strategies that we can take in the borrow step.

Random Selection As the name indicates, users with gaps are chosen randomly. This strategy is the most fair one because it does not depend on the priority or any other parameter of the user. No bias is introduced, and no user receives special treatment.

The disadvantage of this strategy is that users of paying a lot may not be satisfied because they pay more than others. Hence, once this kind of user suffers from resource shortage, they will anticipate being able to borrow bandwidth at first.

Priority Selection In this strategy, we sort the gap user list in descending order according to the priority. This way, users of higher priority have the chance to borrow first. However, the main disadvantage is that users of low priority may never have the chance to borrow because high priority users may starve them by borrowing all the available bandwidth.

Fair Selection In this strategy, we map the original network NET to a new system NET'. The new user set U' contains users in NET with positive gap value. The new link set L' contains links in NET with positive share pool size. Thus, we have:

$$U_i \rightarrow U_i', \forall i \; such \; that \; \epsilon_i > 0 \tag{3.22}$$

$$L_j \rightarrow L'_j, \forall j \; such \; that \; \Theta_j > 0 \tag{3.23}$$

Assume that L' contains m' links, and U' contains n' users. We define the link capacity of the mapped link L'_j to be the share pool size of L_j. That is:

$$C'_j = \Theta_j \tag{3.24}$$

The final goal is to produce a fair allocation, such that the capacity of the link set L'_j is fully utilized. Thus the objective is:

$$Objective : max \sum_{L'_j \in R'_i} x'_i \tag{3.25}$$

$$Subject \; to : \sum_{L'_j \in R'_i} x'_i < C'_j, \; \forall j' \tag{3.26}$$

$$and : x'_i > 0, \forall i' \tag{3.27}$$

As such, we can apply any fair allocation (*e.g.* max-min fair allocation, proportional fair allocation, *etc*) to this sub-system. Once we have the allocation result, we have:

$$\kappa_i = x'_i \tag{3.28}$$

However, there are two disadvantages to this strategy. First, performing fair allocation on the mapped system can introduce high overhead because such a strategy is much more complex than the aforementioned two strategies. Second, the fair allocation may not fully satisfy the gaps. For example, assume that there are two equal-priority users U_A and U_B sharing the single link L. Assume that at control round t, $\Theta_L(t)$ is 50Mbps, ϵ_A is 30Mbps and ϵ_B is 20Mbps. If we apply weighted max-min fair allocation to solve the problem, we have:

$$x'_A = 25Mbps, x'_B = 25Mbps \tag{3.29}$$

However, user U_A's gap is not satisfied and user U_B wastes 5Mbps when satisfying its gap. A better solution should be x'_A = 30Mbps and x'_B = 20Mbps. Therefore, an adjustment of the simple fair allocation is needed to solve this kind of problem. The goal is not only to provide a fair and efficient allocation result, but also to satisfy the gap of each user.

3.5.3 Prediction of the Bandwidth Demand

The share/reclaim/borrow procedure depends heavily on the value of η_i. If user U_i always sets η_i such that η_i is much higher than its actual demand, then U_i would always try to 'steal' bandwidth from other users. If we assume that there is no malicious user in the network, then there are two methods to determine η_i.

Precise Prediction If the user's application knows exactly how much data it wants to put on the wire, then η_i is the precise reflection of the user's demand for bandwidth. For example, when the user is downloading a file from the FTP server, it knows the size of the file. Thus, it can report the anticipated bandwidth requirement to the central manager.

Estimate If the user does not know the underlying application and cannot report the exact bandwidth demand, then there are several ways to estimate the bandwidth demand. One of these solutions is to monitor the bandwidth usage history and build a model to do the prediction. However, this may introduce overhead for the user and/or the central manager.

In our work to date, we do not record the bandwidth usage of U_i. If U_i is assigned bandwidth x_i, and it uses only a very small portion of that bandwidth, then some punishment should be applied on U_i in the next control round. However, we have not implemented this punishment and fake prediction detection. This leaves an open question for future work.

3.6 Summary

In this chapter, we give the definition of our centralized dynamic resource allocation algorithm. There are two states in our algorithm. The first state is initial allocation. We use weighted max-min fair allocation as the initial allocation method. After the initial allocation, our algorithm enters dynamic allocation state. In this state, we use the concept share/reclaim/borrow to adjust bandwidth allocation according to each user's bandwidth demand. We also make discussion about: 1) how the share/relcaim factor can influence the allocation result, 2) how to choose the borrowing order, and 3) how to predict bandwidth demand for each user.

Chapter 4

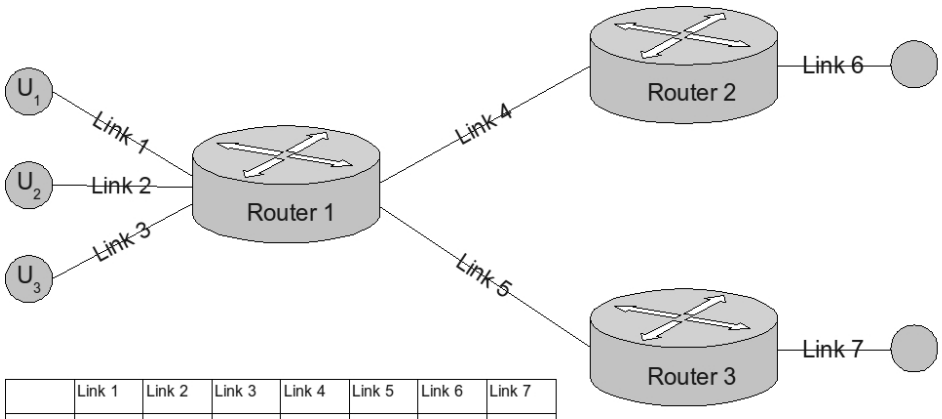
Distributed Dynamic Resource Allocation Scheme

In the previous chapter, we discussed the extension of the weighted max-min fair allocation in a centralized manner. In this chapter, we use a distributed system to implement the dynamic resource allocation algorithm.

The main reason that we try to convert the centralized method to a distributed version is because user information is not always available. Considering propagation delay and packet loss, reliable communications between the users and the central manager are hard to maintain. Therefore, we develop a router-based resource allocation scheme which takes advantage of our share/reclaim/borrow strategies.

4.1 System Model

In this section we discuss the system model that we use for the distributed version of the dynamic allocation scheme. In the network NET, there is a user set U and a link set L. The user set contains n users while the link set contains m links. In addition, there is a router set S which contains w routers. We use $S_k \in R_i$ to denote that router S_k is on the route of user U_i. If link $L_j \in R_i$ and $S_k \in R_i$, and if L_j is the egress link on S_k, then we say that router S_k is responsible for managing the bandwidth allocation of link L_j. We use $L_j \in S_k$ to denote this. To simplify our problem, we assume that each user only has one egress link on each router. An example system model is shown



	Link 1	Link 2	Link 3	Link 4	Link 5	Link 6	Link 7
User 1	●	○	○	●	○	●	○
User 2	○	●	○	○	●	○	●
User 3	○	○	●	○	●	○	●
Router 1	●	●	●	○	○	○	○
Router 2	○	○	○	●	○	○	○
Router 3	○	○	○	○	●	○	○

●: user uses the link, or the router manages the link
○: user does not use the link, or the router does not manage the link

Figure 4.1: Distributed system model

in Fig. 4.1.

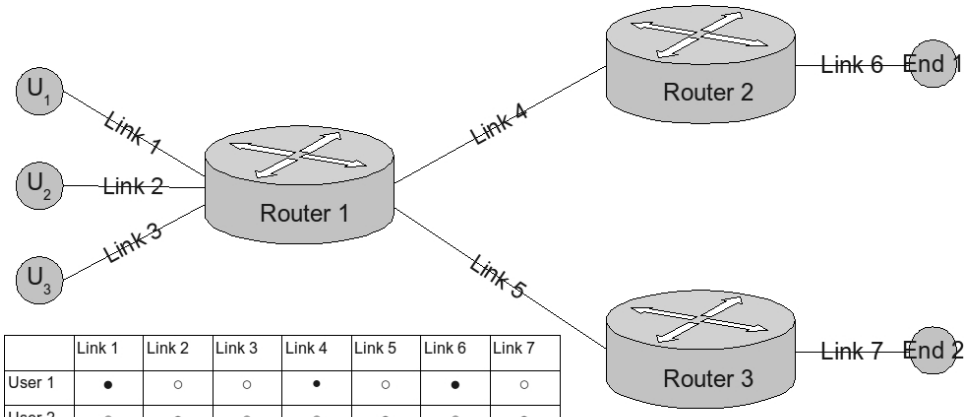
In Fig. 4.1, we can see that there is no router to manage the last two links, L_6 and L_7. To fix this, we assume that the end point of the data flow manages the egress links by running the some algorithm. The modified system model is shown in Fig. 4.2.

It is easy to see that each individual router takes the role of the central manager. If we denote the user set of router S_k as U^k, and the link set of S_k as L^k, we have:

$$U = U^1 \cup U^2 \cup \ldots \cup U^w \tag{4.1}$$

$$L = L^1 \cup L^2 \cup \ldots \cup L^w \tag{4.2}$$

If router S_k is on the route of user U_i, user U_i is in the user set of S_k. If link L_j is on the route of user U_i, and if L_j is managed by router S_k, we can use j and k as the headnote of each parameter of U_i interchangeably. For example, if we want to denote the bandwidth allocation of U_i on link L_j, we can use x_i^j. We can also use x_i^k because L_j is managed by S_k.



	Link 1	Link 2	Link 3	Link 4	Link 5	Link 6	Link 7
User 1	●	○	○	●	○	●	○
User 2	○	●	○	○	●	○	●
User 3	○	○	●	○	●	○	●
Router 1	●	●	●	○	○	○	○
Router 2	○	○	○	●	○	○	○
Router 3	○	○	○	○	●	○	○
End 1	○	○	○	○	○	●	○
End 2	○	○	○	○	○	○	●

●: user uses the link, or the router/end point manages the link
○: user does not use the link, or the router/end point does not manage the link

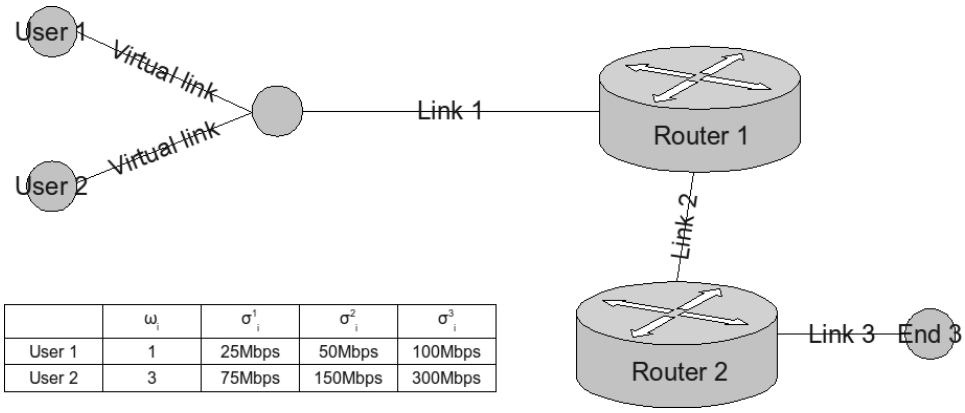
Figure 4.2: Modified distributed system model

There are many ways to identify a user, *e.g.* using its IP address in the routing table. The open question is how to determine the priority and the share/reclaim factor of each user. We will discuss this later in the chapter. Here we assume that each router S_k is pre-configured with the knowledge of the priority and other parameters needed for each user U_i such that $U_i \in U^k$.

Like the central manager, each router S_k has a timer which it uses to time the duration of its 'control period' Γ_k. Γ_k can be different from router to router. These timers do not need to be synchronized. We will prove that if the bandwidth demands of all users are stable, then the bandwidth allocation will eventually be stable and optimum.

The distributed allocation scheme also has two states: the initial allocation state and the dynamic allocation state. However, because we do not have the central manager, the user would not receive the bandwidth allocation from the manager. Two possible alternatives are proposed here.

Passive Allocation Because the user has no method to notify the router its bandwidth usage, we redefine η_i as the transmission rate of user U_i. If the router finds that $\eta_i(t+1)$ is close to or more than $x_i(t)$, it can drop the



	ω_i	σ^1_i	σ^2_i	σ^3_i
User 1	1	25Mbps	50Mbps	100Mbps
User 2	3	75Mbps	150Mbps	300Mbps

Figure 4.3: An example of distributed weighted max-min fair allocation result

packet from that user. If the user uses protocols that reacts to packet loss (*e.g.* TCP protocol), it will reduce the transmission rate. The disadvantage is that the protocol which the user uses may not react as expected. For example, TCP protocol is an additive-increase-multiplicative-decrease protocol, which means that the increase rate of the transmission window is much slower than the decrease rate. If TCP detects a packet loss, the transmission rate of the user will decrease very sharply. However, the expectation is that η_i decreases to x_i. Therefore, the passive allocation loses control of the precise bandwidth allocation which the central manager does have.

Active Allocation If we insert several control bits into the packet to inform the user of the bandwidth allocation, then the user can control the bandwidth limitation process precisely. The disadvantage is that we need to modify the existing protocols to accommodate the new control bits.

4.2 Initial Allocation State

Router S_k performs the initial allocation when the following criteria are satisfied:

- a new user enters the router, or
- an existing user leaves the router, or

- a new link is added to the link set L^k, or
- a link is deleted from the link set L^k.

It is easy to see that changing the routing information of user U_i is a combinition of criteria 1 and criteria 2: U_i first leaves S_k by deleting its previous routing information, and U_i enters S_k by adding its new routing information. Adding/deleting a link on S_k may refer to physically plugging/unplugging the link or link failure. Note that adding/deleting links may always leads to adding/deleting existed users on router S_k.

Each router S_k runs the weighted max-min fair algorithm to do the initial allocation. Because each user has only one egrees link on router S_k, we can simplify the algorithm as:

Algorithm 4.1 WeightedMaxMin(U^k, L^k)

```
for all L_j in L^k do
  τ_j = C_j / Ω_j
  for all U_i in U^k such that L_j ∈ R_i do
    σ_i^k = τ_j * ω_i
  end for
end for
```

The differences between the simplified algorithm and the one in Chapter 2 are (1) the process of searching L_{min} is neglected, and (2) the initial allocation σ_i^k of user U_i is recorded on router S_k. If there are many routers on the route of user U_i, then the initial allocation on each router can be different.

To illustrate this, we use the network shown in Fig. 4.3 as an example. Assume that the network contains three links L_1, L_2 and L_3. The capacity of the three links are $100Mbps$, $200Mbps$, and $400Mbps$ respectively. The initial allocations are also shown in Fig. 4.3. It is easy to see that the allocation result is different on each router.

Because of this, the previous definition of the bottleneck link is meaningless. Assume that link L_j is on the route of user U_i, and router S_k manages L_j. Then for each user using link L_j we have:

$$f(\sigma_i^k) = \frac{\tau_j * \omega_i}{\omega_i} = \tau_j \tag{4.3}$$

It is easy to see that if user $U_{i'}$ is also using link L_j, then $f(\sigma_i^k) = f(\sigma_{i'}^k)$. Hence link L_j is the bottleneck of user U_i. The same applies to other links on the route of U_i. Therefore all of the links on U_i's route are bottleneck links of U_i. To fix this, we redefine the concept of bottleneck link as follows:

Definition 4.1. *Link L_j is said to be the bottleneck link of user U_i if for any other link $L_{j'}$ on the route of U_i, $\tau_j \leq \tau_{j'}$.*

By definition 4.1, if link L_j is the bottleneck link of user U_i and router S_k manages L_j, then σ_i^k is the minimum initial bandwidth allocation of U_i. Taking the network shown in Fig.4.3 as an example, link L_1 is the bottleneck link of both U_1 and U_2.

4.3 Dynamic Allocation State

After the initial allocation state, router S_k enters the dynamic allocation state. As discussed above, the user does not provide bandwidth demand to the router. Therefore, the router has to predict the bandwidth demand based on the user's transmission rate. If we denote the local transmission rate of user U_i on router S_k as TX_i^k, we have:

$$\eta_i^k(t+1) = g_k(TX_i^k(t)) \tag{4.4}$$

where $g_k(\cdot)$ is the bandwidth demand prediction function used on router S_k. If router S_p and S_q are both on the route of user U_i, we have:

$$TX_i = TX_i^p = TX_i^q \tag{4.5}$$

which means that the transmission rate of U_i is uniform on its route.

We apply the same share/reclaim strategies to the distributed dynamic allocation algorithm. However, it is worth noting that each router can set the share/relcaim factors independently. For example, for user U_i using router S_p and S_q, we can have:

$$\delta_i^p > \delta_i^q \; and \; \rho_i^p > \rho_i^q \tag{4.6}$$

This setting means that router S_p reacts more quickly than S_q to U_i's change of its transmission rate. On the other hand, the link managed by S_q is more stable than that of S_p. However, there are many other possible settings.

We modify the borrow step for the distributed algorithm. If link L_j is managed by router S_k and user U_i has $\epsilon_i^k(t+1)$ greater than zero, then U_i starts the borrow procedure. Router S_k distributes the share pool of L_j according to the priorites of users using L_j:

$$\kappa_i^k(t+1) = \omega_i * \frac{\Theta_j(t+1)}{sum\ of\ priorities\ of\ the\ users\ with\ positive\ gap} \tag{4.7}$$

4.4 Proof of Stability

It is important to show that the distributed algorithm can lead to a stable and optimal allocation if the bandwidth demand of each user is stable. Assume that link L_j is on the route of user U_i, and router S_k manages the link L_j. Therefore, we can use x_i^j and x_i^k interchangeably. To simplify the problem, we make several assumptions:

Assumption 4.1. *Assume that all the routers have same control period and start the initial allocations at the same time:*

$$\Gamma_1 = \Gamma_2 = \ldots = \Gamma_w \tag{4.8}$$

$$t_1(0) = t_2(0) = \ldots = t_w(0) \tag{4.9}$$

Assumption 4.2. *Assume that all the users are elastic users. Given bandwidth allocation $x_i^k(t)$ at control round t on router S_k, we have:*

$$TX_i(t) \leq min\{x_i^k(t)\}, \forall S_k \in R_i \tag{4.10}$$

Assumption 4.3. *Assume that the bandwidth prediction function is globally uniform. Specifically, we have:*

$$\eta_i^k(t+1) = \eta_i(t+1) = TX_i(t) * \upsilon \tag{4.11}$$

where υ is a factor that is greater than one.

This is an aggressive prediction model. The router assumes that each user requests more for bandwidth than its current allocation. With the uniform υ, all the users increase their bandwidth demands at the same pace.

Assumption 4.4. *Assume that for each user U_i, the share/reclaim factors are same.*

Besides the assumptions, we use the definition of constrained link to help prove the problem:

Definition 4.2. *User U_i is constrained by link L_j if:*

1. *$L_j \in R_i$, and*
2. *L_j is saturated, and*
3. *$x_i^j \leq x_i^{j'}$, $\forall L_{j'} \in R_i$ and $L_{j'} \neq L_j$.*

It is easy to see that when all the users are constrained, the allocation result is stable. The result is also optimal because there are at least one saturated link on the route of each user. To prove the stability of the problem, we need to prove two sub-problems:

- Each user can be constrained, and
- After the user has been constrained, its bandwidth allocation would be eventually stable.

Let us prove the first sub-problem. Consider the single link case. Assume that a link L_j is managed by a router S_k. Without loss of generality, users U_1, U_2, ..., U_r are competing for L_j. Because router S_k manages L_j, we can use x_i^j and x_i^k interchangeably. We divide the process of allocating L_j into four stages.

Stage 1 $[t_j(0), t_j(1))$: Within control round $[t_j(0), t_j(1))$, router S_k is in stage 1. The transmission rates of all of the users are below their initial allocation. For each user U_i we have:

$$x_i^k(t) < \sigma_i^k, where\ t \in [t_j(0), t_j(1)) \tag{4.12}$$

Because the inital allocation can satisfy the bandwidth demand, none of the users borrows bandwidth from others. The bandwidth allocation continues to increase for all of the users.

Stage 2 $[t_j(1), t_j(2))$: When the bandwidth allocation of one or more users reaches their initial allocation at control round $t_j(1)$, router S_k enters stage 2. There are two situations here. If all of the users have the same initial allocation, then their initial allocation would be saturated all at $t_j(1)$. This is because according to Assumption 4.3 the bandwidth demands of all of the users increase at the same pace. Thus for each U_i we have:

$$x_i^k(t) = \sigma_i^k, where\ t > t_j(1) \tag{4.13}$$

It is easy to see that after $t_j(1)$, none of the users can borrow from others because link L_j is saturated. Router S_k directly enters stage 4.

The second situation is that the users do not have same initial allocations. Then user U_i with minimum σ_i^k will saturate its initial allocation first. U_i starts to borrow bandwidth from unsaturated users.

In the second situation, the bandwidth allocation of all of the users still continues to increase during control round $t_j(1)$ to $t_j(2)$. We can divide the users into two sets $U(S)$ and $U(B)$. $U(S)$ contains users who share bandwidth, while $U(B)$ contains users who borrow bandwidth from users in $U(S)$.

Stage 3 $[t_j(2), t_j(3))$: When all of the users in $U(S)$ start to reclaim their shared bandwidth at control round $t(2)$, the bandwidth allocation of users

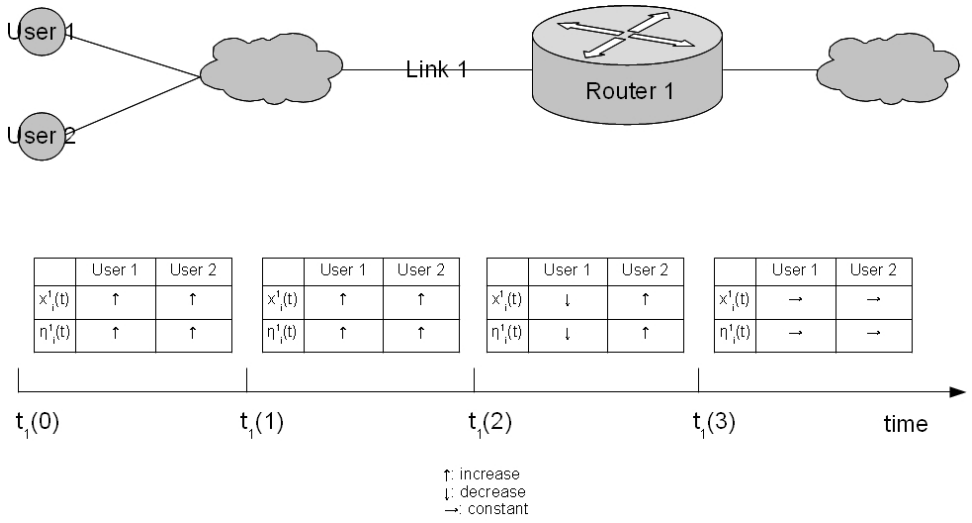


Figure 4.4: Bandwidth prediction on the router

in $U(B)$ start to decrease. Because all the users are elastic users, it is easy to see that link L_j is saturated:

$$\sum_{L_j \in R_i} x_i^j(t) = C_j, where\ t \in [t_j(2), t_j(3)) \tag{4.14}$$

Stage 4 $[t_j(3), \dots)$: At control round $t_j(3)$, all of the shared bandwidth of users in set $U(S)$ has been reclaimed. Then the bandwidth allocation of each user is the same as its initial allocation:

$$x_i^k(t) = \sigma_i^k, where\ t > t_j(3) \tag{4.15}$$

Hence link L_j is saturated and the allocation is stable.

To illustrate this, consider the network shown in Fig.4.4. Two users U_1 and U_2 share link L_1, which is managed by router S_1. The bandwidth of L_1 is 100Mbps. The priority of user U_1 is $\omega_1 = 1$, and the priority of user U_2 is $\omega_2 = 3$. Therefore we have $\sigma_1^1 = 25Mbps$, and $\sigma_2^1 = 75Mbps$. Initially U_1 and U_2 increase their bandwidth demand at the same pace. Assume that at control round t_1, U_1 and U_2's transmission rates increase to 25Mbps. Then U_1 starts to borrow bandwidth from U_2. At control round t_2, the transmission rate of U_1 and U_2 increases to 50Mbps. Then U_2 starts to reclaim its shared

bandwidth. Finally at control round t_3, the bandwidth demands of U_1 and U_2 become constant and equal to their respective initial allocations.

Next we consider the multiple links case. For link L_j, we define Λ_j as the maximum initial allocation among the users using L_j:

$$\Lambda_j = max\{\sigma_i\}, L_j \in R_i \tag{4.16}$$

Proposition 4.1. *If the routers start allocation at the same time, then link L_j with minimum Λ_j must be saturated first.*

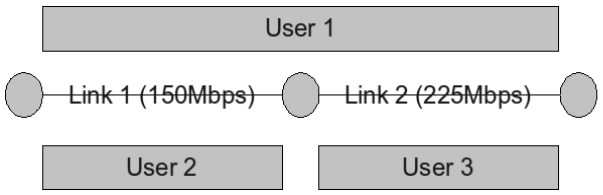
Proof. Assume link L_j is the link with minmum Λ_j, and user U_i is the user with maximum σ_i on link L_j. According to Assumption 4.3, the speed of increasing bandwidth demand is same for all the users. If the initial allocations are same for all the users using L_j, then L_j is saturated at the beginning of stage 2. If the initial allocations are not the same, then U_i must be in set $U(S)$ because U_i has max initial allocation. Because users share their bandwidth at the same pace, user U_i must be the last one to reclaim. Because link L_j enters stage 3 only after all the users in set $U(S)$ has been starting to reclaim the shared bandwidth, the time for link L_j to enter stage 3 is determined by σ_i. Because $\Lambda_j = \sigma_i$ is the smallest among the links, L_j is the first link that enters stage 3. According to Assumption 4.4, because the reclaim pace of all the users are the same, L_j enters stage 4 first. Thus, L_j is saturated first. □

It is easy to see that users sharing link L_j with $min\{\Lambda_j\}$ would be constrained first. Next we prove that each user can be constrained.

Corollary 4.1. *Each user will eventually be constrained.*

Proof. Assume that U_i is a user using L_j that has minimum Λ_j, and $U_{i'}$ is another user such that U_i and $U_{i'}$ are sharing links other than L_j. Then $U_{i'}$ has not been contrained when U_i is, and $U_{i'}$ would increase its bandwidth allocation by borrowing from U_i. Hence, $U_{i'}$ would eventually be constrained when all the extra bandwidth of U_i is borrowed. □

Next we prove that after the user has been constrained, its bandwidth allocation will eventually be stable.



	ωi	σ1i	σ2i
User 1	1	50Mbps	25Mbps
User 2	2	100Mbps	-
User 3	8	-	200Mbps

Figure 4.5: Example of Shifting Constrained Link

Proposition 4.2. *If user U_i is constrained by link L_j, then we have:*

$$TX_i = x_i^j \tag{4.17}$$

Proof. Because U_i is constrained by L_j, TX_i cannot be greater than x_i^j. If $TX_i < x_i^j$, this constradicts the definition of the contraint link that such link is saturated. Hence, TX_i equals to x_i^j. □

However, constraint shift can sometimes happen. We define constraint shift as follows: link L_p and L_q are both on the route of U_i. At time t_a, user U_i is constrained by link L_p. If at time t_b, bandwidth allocation changes on L_q and if the change results in that U_i is constrained by L_q, then we say that U_i shifts its constraint from L_p to L_q.

To illustrate this, consider the network shown in Fig.4.5. The settings are also shown in the figure. According to Proposition 4.1, we know that link L_1 will be saturated first. User U_1 is first constrained by link L_1. However, when U_3 starts to reclaim its shared bandwidth, the allocation for U_1 starts to decrease. Hence, the constraint of U_1 is finally shifted from L_1 to L_2.

Proposition 4.3. *If U_i shifts its constraint from L_p to L_q, then its bandwidth allocation on L_p is non-increasing.*

Proof. It is easy to see that before the shift, L_p is saturated, and hence there is no extra bandwidth for U_i to borrow to increase its allocation on L_p. After the shift, if $x_i^p = x_i^q$, then the bandwidth allocation for U_i remains unchanged on L_p. Otherwise, it decreases on L_p. □

Corollary 4.2. *If U_i shifts its constraint from L_p to L_q, then its allocation on L_q is non-increasing.*

Proof. Define the bandwidth allocation of U_i on link L_p before/after the shift as $x_i^p(t_a)$ and $x_i^p(t_b)$. Define the bandwidth allocation of U_i on link L_q before/after the shift as $x_i^q(t_a)$ and $x_i^q(t_b)$. According to Proposition 4.3, we have:

$$x_i^q(t_a) \geq x_i^p(t_a) \geq x_i^p(t_b) \geq x_i^q(t_b) \tag{4.18}$$

□

Corollary 4.3. *After user U_i has been constrained, its bandwidth allocation will eventually be stable.*

Proof. According to Corollary 4.2, after user U_i has been constrained, its bandwidth allocation is non-increasing. Because the lower bound of the bandwidth allocation of U_i is σ_i. Hence, U_i's bandwidth allocation will eventually be stable. □

4.5 Discussion

There are two major problems with the distributed version of the resource allocation algorithm. The first problem is how to identify each individual user. The second problem is how to set the parameters for each user. In this section, we discuss the possible solutions for these two problems.

4.5.1 Identifying User at the Router

The user does not need to know that there are routers between the source and the destination. Therefore, the user cannot register with a router by sending a registration message to that router. Another problem is that the router has limited memory and CPU resources. It would become expensive for the router to keep a list of all users whose traffic is passing through that router.

Since the router keeps a routing table in its memory, and the router can automatically add new destinations to the routing table, the router might use the routing table to record the users' identifications.

Notice that the router is only responsible for managing the outbound direction of each link it terminates. Therefore, when identifying the user, the

router only needs to consider incoming packets.

4.5.2 Setting Parameters

For a number of reasons, it would be difficult to let each user determine its priority and share/reclaim factors. Consequently, we propose two possible solutions to this problem. First, the router can have some users preconfigured, and leave the other users using some default values. For example, the administrator of the network can configure a router for some special users by setting their priority and share/reclaim factors, and set a default value for all other users. If the router detects a data flow of a special user, then it could use the predefined values for that user. On the other hand, if the data flow does not belong to any of the special users, then the router could use the default values.

A second solution is to use port numbers to differentiate users. For example, HTTP usually uses port 80, and FTP usually uses port 21. An FTP user can have higher priority and reclaim factor and lower share factor than an HTTP user. Thus the FTP user can use more bandwidth than the HTTP user. However, the major disadvantage is that only a few port numbers are well defined. Many network applications use random port numbers, which makes it hard to tell which application is more important than the other.

4.6 Summary

In this chapter, we extend the centralized resource allocation algorithm to a distributed version. We eliminate the CMS in the network. Instead, the functionality of the CMS is moved to each individual router in the network. We prove that under certain assumptions, the distributed algorithm can lead to a stable and optimal allocation result. We also discuss two potential problems: 1) how to identify users in the network, and 2) how to set parameters for each user.

Chapter 5

Emergency Handling

Emergency handling is important in real applications. For example, consider a network model which consists of one data center and multiple remote network users. The network user backs up their critical data to the data center. Normally all users are sharing the single ingress link of the data center. The backup action is a periodic process which requires a certain amount of bandwidth for synchronization between the data center and the user. If user U_i suffers from power failure or other disasters, it needs to quickly restart and transfer the backup data from the data center to itself. Hence, U_i requires peak bandwidth performance during failure or disaster recovery. We call this sort of action emergency handling.

A user can have either of two statuses:

Normal Status In this status, the user behaves exactly as we discussed in the previous sections. If the user has unused bandwidth, it can share this extra bandwidth with other users. If the user demands more bandwidth, then it can start the reclaim-and-borrow procedure to acquire more bandwidth.

Emergency Status In this status, the user is critical for bandwidth. Because of this, it does not share anymore. On the contrary, it will try to grab as much bandwidth from other users as possible. In reality, we can further define different emergency statuses as emergency level one, emergency level two, *etc,* so as to satisfy the customer's requirement.

In the emergency case, we divide the user set U into two categories: normal

user set $U(N)$ and emergency user set $U(E)$. We have:

$$U = U(N) \cup U(E) \tag{5.1}$$

We denote user U_i in Normal Status as $U_i \in U(N)$, and we denote U_i in Emergency Status as $U_i \in U(E)$. In order to record this status, we add a new parameter to the user:

ξ_i records the status of the user. $\xi_i = 1$ indicates that user U_i is in Normal Status. $\xi_i > 1$ indicates that U_i is in Emergency Status. Because the user can have multiple emergency level, ξ_i is in range $[1, \infty]$.

In this chapter, we discuss multiple ways to implement emergency handling. Furthermore, we discuss and compare the pros and cons of these handling methods. To simplify the problem, we assume that the allocation scheme runs in a centralized manner. That is, the central manager M is responsible for performing the dynamic allocation task as well as the emergency handling task.

5.1 Increase Bandwidth Demand of Emergency Users

The first method of emergency handling is to increase the bandwidth demand of the emergency user. Assume that the U_i is in emergency status, and therefore $\xi_i > 1$. Because of this, we can artificially increase the bandwidth demand of U_i. As discussed above, at control round t, the bandwidth demand of U_i is η_i. If we use:

$$\eta_i'(t) = \eta_i(t) * \xi_i \tag{5.2}$$

then it is guaranteed that U_i's demand for bandwidth is always larger than its need, and therefore can reclaim/borrow more from other users.

The advantage of this emergency handling method is that none of the users need to know the emergency status of other users in the network. The user

U_i simply sets its bandwidth demand to an artificial high value and requires the central manager to respond to that value. However, consider a case in which all users $U_1, U_2, \ldots, U_n$ are transfering data in full speed. If $U_i \in U(E)$ and η_i is artificially increased by multiplying ξ_i to η_i, U_i still cannot borrow any bandwidth from other users because none of the other users has extra bandwidth to share.

Another problem with this method exists when there are multiple emergency users in the network. For example, suppose that two users are in Emergency Status, *i.e.* $U_p, U_q \in U(E)$. As we discussed before, there are multiple choices for the borrow step. Assume that we let users with higher priority borrow first. We also assume that $\omega_p > \omega_q$, and $\xi_p < \xi_q$. In this case, U_q is much more critical for bandwidth than U_p is. However, because $\omega_p > \omega_q$, U_p always has the right to borrow before U_q. If the unused bandwidth can only satisfy U_p's critical demand, then U_q will never be served and thus the emergency handling fails for U_q. The situation can be even worse. If there exists a normal user $U_k \in U(N)$ such that U_k is also critical for bandwidth and $\omega_k > \omega_p$, then U_k will borrow first. U_k may deplete the share pool and prevent the emergency handling of both U_p and U_q.

Because of this, we need to be careful to choose the appropriate borrow strategy when we use this method for emergency handling. As discussed, the wrong choice of borrow strategy can lead to failure of emergency handling for certain emergency users.

5.2 Decrease Bandwidth Demand of Normal Users

In constrast with the method discussed in the previous section, we can artificially decrease the bandwidth demand of the normal user. To implement this, we need to add a new parameter to the central manager.

Ξ_M records the maximum emergency level among the emergency users. That is:

$$\Xi_M = max\{\xi_i : U_i \in U(E)\} \tag{5.3}$$

Next, we modify the η_i for each user U_i at control rount t as:

$$\eta_i'(t) = \eta_i(t) * \frac{\xi_i}{\Xi_M} \tag{5.4}$$

When there is no emergency user in the network, $\Xi_M = 1$ because $\xi_i = 1$ for each user U_i. Thus the bandwidth demand of each user U_i is not affected by dividing Ξ_M. If, on the other hand, there is an emergency user $U_p \in U(E)$, then we have:

$$\xi_p > 1 \; and \; \Xi_M = \xi_p \tag{5.5}$$

Because of this, the bandwidth demand of users other than U_p would be decreased by dividing by Ξ_M. However, eta_p is not affected because $\frac{\xi_p}{\Xi_M} = 1$. Thus the bandwidth demands of users other than U_p are suppressed, and U_p can borrow the extra bandwidth from other users.

The advantage of this scheme is that the central manager does not need to care about the bandwidth demand of the normal users, because the demands of the normal users are always suppressed. The emergency users are guaranteed that there is always extra bandwidth to borrow. Another advantage is that we can differentiate emergency users at different emergency levels.

For example, assume that we have two emergency users, U_p and U_q, in the network. If $\xi_p > \xi_q$, then we have:

$$\Xi_M = max\{\xi_p, \xi_q\} = \xi_p \tag{5.6}$$

$$\eta_p' = \eta_p * \frac{\xi_p}{\Xi_M} = \eta_p \tag{5.7}$$

$$\eta_q' = \eta_q * \frac{\xi_q}{\Xi_M} < \eta_q \tag{5.8}$$

Although U_p and U_q are both emergency users, the bandwidth demand of U_q is also suppressed by U_p because the emergency level of U_p is higher than U_q.

The first disadvantage of this emergency handling method is similar to the one we discussed in the previous section. The wrong borrow strategy can lead to failure of emergency handling. The second disadvantage is that eventually the normal users will get starved if the emergency user continuously suppresses the bandwidth demand of the normal users. Assume that normal user U_i can always get full bandwidth allocation. That is:

$$x_i(t+1) = \eta_i(t+1) \tag{5.9}$$

Then because the bandwidth demand of U_i is suppressed by dividing $\eta_i(t+1)$ by Ξ_M, eventually we have:

$$\lim_{t\to\infty} \eta_i(t) = 0 \tag{5.10}$$

and therefore

$$\lim_{t\to\infty} x_i(t) = 0 \tag{5.11}$$

Hence, the bandwidth allocation of this handling scheme is severely biased towards emergency users. In reality, normal users would not be satisfied when all of the bandwidth is grabbed by emergency users.

One solution to this problem is to set an upper bound for the bandwidth that each normal user can borrow. Then if the bandwidth shared by the normal user exceeds this upper bound, the normal user cannot share bandwidth anymore. Conversely, the emergency user can set a lower bound to limit how much bandwidth it can take from a normal user.

5.3 Emergency Policy

The emergency handling schemes discussed in the previous sections are capable of handling simple cases. However, the central manager does not have the flexibility to control each individual emergency user, nor to control the ratio of the allocation among them. In this section, we introduce an emergency policy which requires the central manager to provide a policy table to control the emergency users as well as normal users.

In Chapter 2, we assign each user U_i a priority ω_i. This parameter is used to control the initial allocation σ_i of U_i. We expand the concept of priority in order to deal with the emergency handling case. First we build an emergency policy table T. The table has A rows and n columns. Each row a represents one policy rule, where $a \in [1, A]$. Each column i represents the priority values of each user U_i in the network. Thus each entry of the table stores the priority value of user U_i at rule a. That is:

$$\omega_{ai} = T[a][i] \tag{5.12}$$

We use the first row of T to denote the priority values when all users are in Normal Status. From the second row, we define the priority values of U_i for each different case. For example, if the network has three users U_1, U_2, and U_3, then we have seven different cases in total:

- $U_1, U_2, U_3 \in U(N)$
- $U_1, U_2 \in U(N)$, $U_3 \in U(E)$
- $U_1 \in U(N)$, $U_2, U_3 \in U(E)$
- $U_2, U_3 \in U(N)$, $U_1 \in U(E)$
- $U_2 \in U(N)$, $U_1, U_3 \in U(E)$
- $U_3 \in U(N)$, $U_1, U_2 \in U(E)$
- $U_1, U_2, U_3 \in U(E)$

If we build a policy table according to these seven cases, we can say that, for example, **rule 2 of the policy table denotes the case in which users U_1 and U_2 are in Normal Status and U_3 is in Emergency Status**. We can also say that, for example, **the priority values of U_1, U_2, U_3 at rule 2 are $\omega_{21}, \omega_{22}, \omega_{23}$**.

By building such a policy table, when the emergency status of user U_i changes, we can look up the table entry and find the appropriate policy to do the allocation. Thus, we need to modify Algorithm 3.1 to Algorithm 5.1.

Algorithm 5.1 CentralizedAllocation

```
while manager is running do
  if emergency status changes then
    look up the emergency policy table T and find the appropriate policy
    rule a
    for all U_i in U do
      ω_i = T[a][i]
    end for
    WeightedMaxMin(U, L)
  end if
  for all U_i in U do
    notify U_i with its allocation x_i
    wait for U_i's feedback
  end for
  DynamicAllocate(U, L)
  sleep over one Γ_M
end while
```

Here we define that in the first instance that the algorithm runs, the emergency status is set to rule 1 of the policy table, in which all the users are in Normal Status.

The policy table is provided by the central manager. Therefore, the manager has more flexibility in handling the emergency by considering the status of each user. However, the disadvantage is that when the number of users grows, the policy table could be very large. Thus, it is hard to maintain such a table. One possible solution is to set only several critical rules and set one default rule. Taking the exmaple discussed above, we can reduce the seven cases to four cases:

- $U_1, U_2, U_3 \in U(N)$
- $U_1, U_2 \in U(N)$, $U_3 \in U(E)$
- $U_2, U_3 \in U(N)$, $U_1 \in U(E)$
- default rule

When the emergency status of the network does not satisfy rule 1, rule 2, or rule 3, the default rule is used to do the initial allocation.

5.4 Summary

In this chapter, we provide three methods for emergency handling. We compare the pros and cons of these methods, and discuss in what situation we can apply them accordingly.

Chapter 6

Experimental Result

In this chapter, we show the results from both numerical simulations and tests in a real network environment. These results demonstrate the feasibility and performance characteristics of our dynamic resource allocation algorithm. The remaining chapter is organized as follows: first we demonstrate how the priority can affect the initial allocation by running a set of numerical simulations. Next, we demonstrate the effectiveness of our centralized resource allocation algorithm. The results were generated from numerical simluation and real network experiment. We also show the feasibility of our emergency handling scheme. Finally, we give the result of the numerical experiment for our distributed resource allocation algorithm.

6.1 Experiment of the Initial Allocation

The first numerical simulation demonstrates how user priority can affect the initial allocation. The network consists of three users U_1, U_2 and U_3, and two links L_1 and L_2. We have $L_1 \in R_1, R_2$ and $L_2 \in R_1, R_3$. We set C_1 to 100Mbps, and C_2 to 50Mbps. The topology of the network is given in Fig. 5. The configuration and allocation results are given in Table 6.1.

Table 6.1: Configuration and Results of Simulation 1

	ω_1	ω_2	ω_3	σ_1	σ_2	σ_3
Test-1	1	1	1	25Mbps	75Mbps	25Mbps
Test-2	1	3	1	25Mbps	75Mbps	25Mbps
Test-3	1	9	1	10Mbps	90Mbps	40Mbps

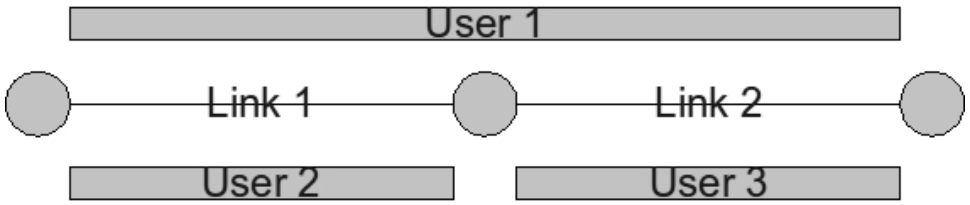


Figure 6.1: Network Topology of Simulation 1

The initial allocation results of Test-1 and Test-2 are the same. However, the procedures to achieve the results of the two tests are different. In Test-1, $\tau_1 = 50$ and $\tau_2 = 25$. The algorithm picks link L_2 as the first link to allocate bandwidth. Thus U_1 and U_3 get their initial bandwidth of 25Mbps first. Next, L_1 has $100 - 25 = 75$Mbps bandwidth left for U_2. On the other hand, in Test-2, $\tau_1 = \tau_2 = 25$. If link L_2 is picked first, then the procedure is similar to Test-1. If L_1 is picked first, then U_1 and U_2 get allocated first, and then U_3. This demonstrates that the procedure is insensitive to the order of allocation.

Comparing Test-1 and Test-3, we can see that as the priority of U_2 increases, U_2 can have a larger initial bandwidth allocation than U_1. This shows that weighted max-min fair allocation has the ability to provide differentiated service by using the priority to control this process. We note that the bottleneck links are fully allocated. Hence we can claim that weighted max-min fair allocation is fair and efficient.

6.2 Centralized Dynamic Resource Allocation

In the second experiment set we test the effectiveness of our dynamic resource allocation scheme. We continue to use the three-user, two-link network model. In this simulation, we configure the users as shown in Table 6.2. The initial allocation result is also shown in Table 6.2.

Table 6.2: Configuration and Initial Allocation of Simulation 2

	User 1	User 2	User 3
ω_i	4	3	2
δ_i	0.2	0.5	0.8
ρ_i	0.8	0.5	0.2
σ_i	33.33Mbps	66.67Mbps	16.67Mbps

6.2.1 Numerical Experiment

First we run a numerical experiment to show the effectiveness of the centralized dynamic resource allocation algorithm. We assume that the network is synchronized, as is possible if it is supported by, for example, a SONET optical transport network. We also assume that each user reports their bandwidth usage and demand at the same pace. We assume that each user makes a periodic 1.25GB bulk data transfer. The interval between two bulk transmissions is 30 seconds. The control period is 5 seconds. Therefore, the interval between two bulk transmissions is 6 control rounds. We assume that all users are best-effort users. In other words, if user U_i is assigned bandwidth $x_i(t)$ at control round t then it will fully utilize that bandwidth such that $u_i(t) = x_i(t)$. Moreover, the user's demand for bandwidth for the next control round is assumed to be $\eta_i(t+1) = u_i(t) * 110\%$. This is a simplified bandwidth prediction model but corresponds roughly to the increasing demands made over time by transport protocols like TCP. The dynamic allocation result is shown in Fig.6.2.

Although the priority of U_1 is higher than U_2, because σ_1 is smaller than σ_2, U_1 sends its data more slowly than U_2. When one user finishes its data bulk transmission, it gradually shares its unused bandwidth. Other users then start to borrow the shared bandwidth. For example, at control round 29, U_1 finished transmitting its data and U_2 started to borrow bandwidth from U_1. As a result the assigned bandwidth for user U_2 becomes greater than σ_2. The behaviour of U_3 is similar to U_2. Hence, we claim that our algorithm can dynamically adjust bandwidth allocations and increase bandwidth utilization in response to changing user needs. This dynamic adjustment is controllable via the share and reclaim factors.

For an illustration of the effects of the share factor, let us compare U_1 and U_3. Because the share factor of U_1 is smaller than that of U_3, the share process of U_1 is much slower than for U_3: within the 6 control rounds, U_1 shares only 2/3 of its bandwidth, while U_3 shares almost all of its bandwidth. The higher the share value, the faster the user shares its unused bandwidth.

This same simulation shows that the reclaim factor is effective in controlling

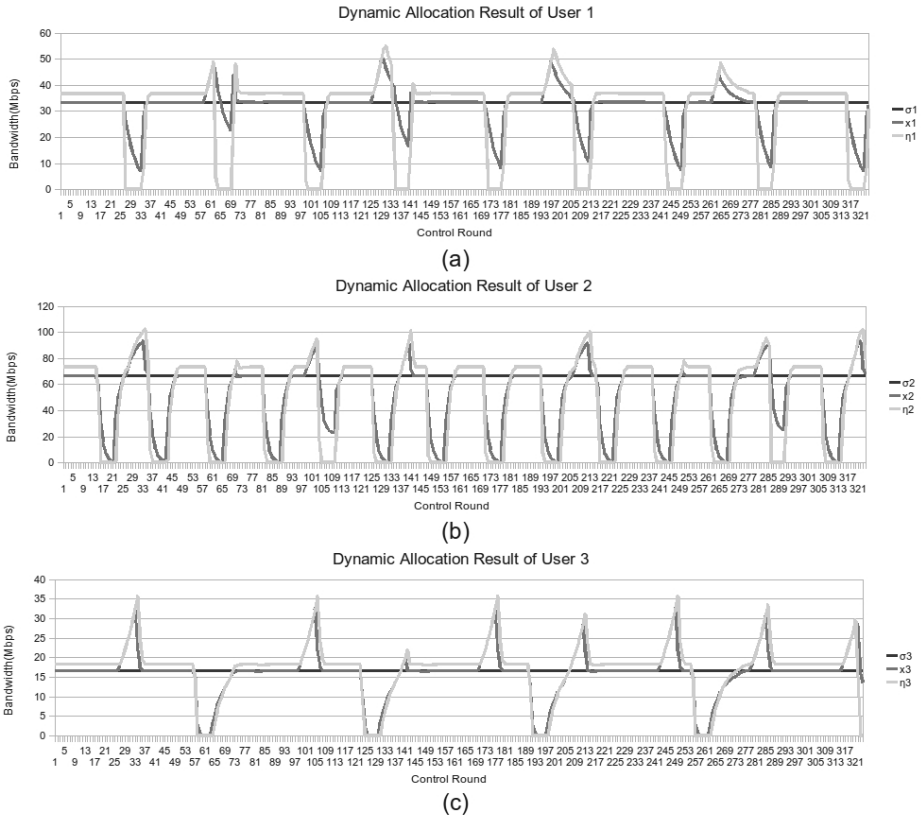


Figure 6.2: Mbps allocated to: (a) User 1 (b) User 2 (c) User 3

the reclaim process. For example, when U_1 begins its second bulk data transfer at control round 34, it starts to reclaim bandwidth. Comparing its behavior with the reclaim process for U_3 (see control round 65), the reclaim process is much faster for U_1. It takes 2 control rounds for U_1 to reclaim all of its shared bandwidth, while it takes 12 control rounds for U_3. Hence, the higher the reclaim factor, the faster the user can reclaim its shared bandwidth.

The dynamic behavior of link utilization is illustrated in Fig.6.3. We show both the capacity and share pool size of the two links. The average share pool size of L_1 is about 40Mbps, and that of L_2 is about 5Mbps. In other words, the average bandwidth not utilized on L_1 is 40Mbps, and about 5Mbps on L_2. This is because we set $\eta_i(t+1) = x_i(t) * 110\%$ for user U_i.

For example, consider when U_2 and U_3 borrow U_1's unused bandwidth. Because the increase of the bandwidth demand of U_2 and U_3 is 10% of its usage, they would not fully borrow the shared bandwidth from U1 within

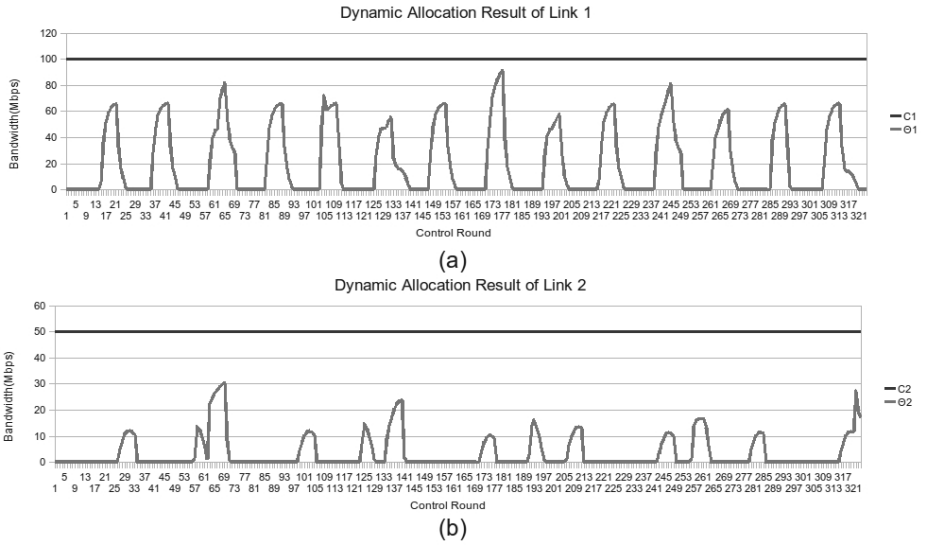


Figure 6.3: Share pool: (a) Link 1 (b) Link 2

6 control round. If we set the model more aggressively, the waste can be reduced. Another reason is that U_1 uses both links. Even if one link has extra bandwidth for U_1, if the other link is saturated, then this extra is wasted because we cannot increase bandwidth allocation for U_1.

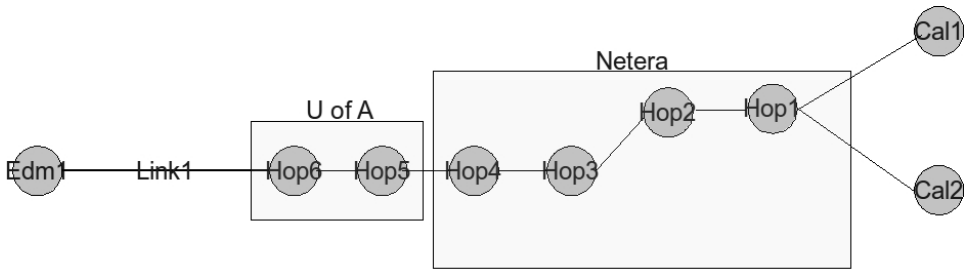
6.2.2 Prototype Test

In addition to simulation, we tested the behavior of a prototype implementation of our algorithm in a real network environment. We set up three physical machines. One machine was located in Edmonton, and served as both the management site and the destination for data transfers. The other two machines were in Calgary, and served as the origins of the data transfers. We will call the two Calgary machines Cal_1 and Cal_2, and the Edmonton machine Edm_1. There were six network hops in the network from Calgary to Edmonton, and the delay between Edm_1 and the two Calgary machines was 5.5ms. Cal_1 and Cal_2 shared a 100Mbps bottleneck link, denoted by L_1 with capacity $C_1 = 100$Mbps. The topology of the network is given in Fig. 8. Cal_1 and Cal_2 transmitted 1GB bulk data to Edm_1 every 30 seconds. We set Γ_M to 1 second. The remaining configuration details for Cal_1 and Cal_2 are shown in Table 6.3.

We first ran the system without the dynamic allocation algorithm (see

Table 6.3: Configuration of Prototype Test

	Cal_1	Cal_2
ω_i	4	2
δ_i	0.2	0.8
ρ_i	0.8	0.2



Hop	Name	IP
1	host136-159-94-1.netera.ca	136.159.94.1
2	209.115.239.141	209.115.239.141
3	209.115.228.166	209.115.228.166
4	host206-75-91-245.netera.ca	206.75.91.245
5	gsb175-c6509-1-canet.backbone.ualberta.ca	129.128.153.201
6	csc-carp1.gw.ualberta.ca	129.128.153.35

Figure 6.4: Topology of the Real Network

Fig. 6.5). Cal_1 and Cal_2 shared the link bandwidth equally. They started their bulk transfers at the same time, and stopped at almost at the same time too. Without the dynamic allocation algorithm, no differentiated service was provided and the users were merely relying on TCP to adjust their transmission rates. The equal sharing of the bottleneck link is an accidental consequence of the two TCP sessions having the same end-to-end delay. If Cal_1, for example, had been further away from Edm_1 than Cal_2, its relative bandwidth share would have been dictated by the uncontrolled and time-varying delay in the network.

Next we re-ran the experiment, but this time using our dynamic allocation algorithm. The results are shown in Fig. 6.6. The ratio of transmission rates between Cal_1 and Cal_2 is about $2:1$, which is equal to the ratio of their priorities. Thus, the allocation algorithm is controlling the link allocation as desired.

Cal_1 finished its first round of transmission at control round 126, while Cal_2 stopped at control round 257. Moreover, comparing the two quiet periods in

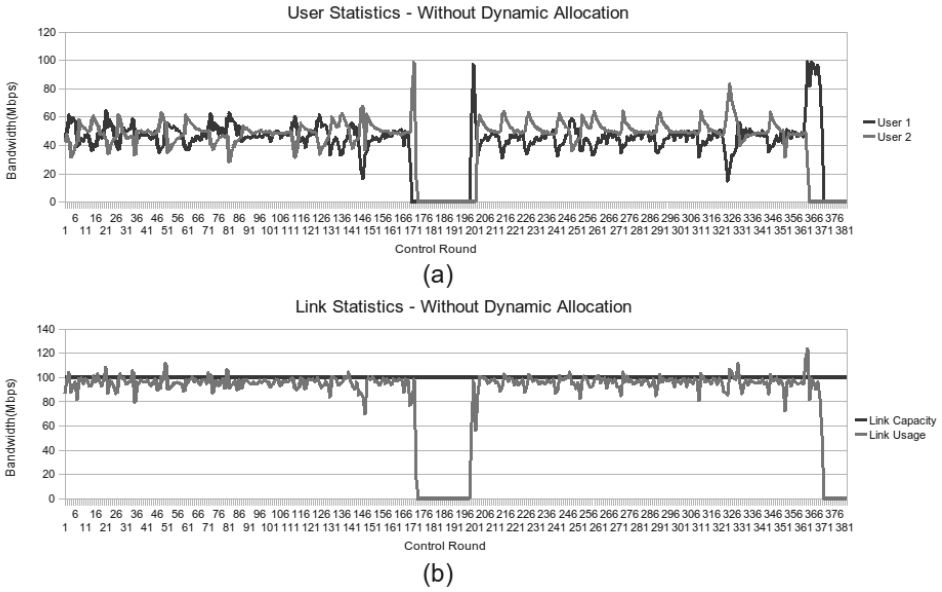


Figure 6.5: (a) User Allocations (b) Link Utilization

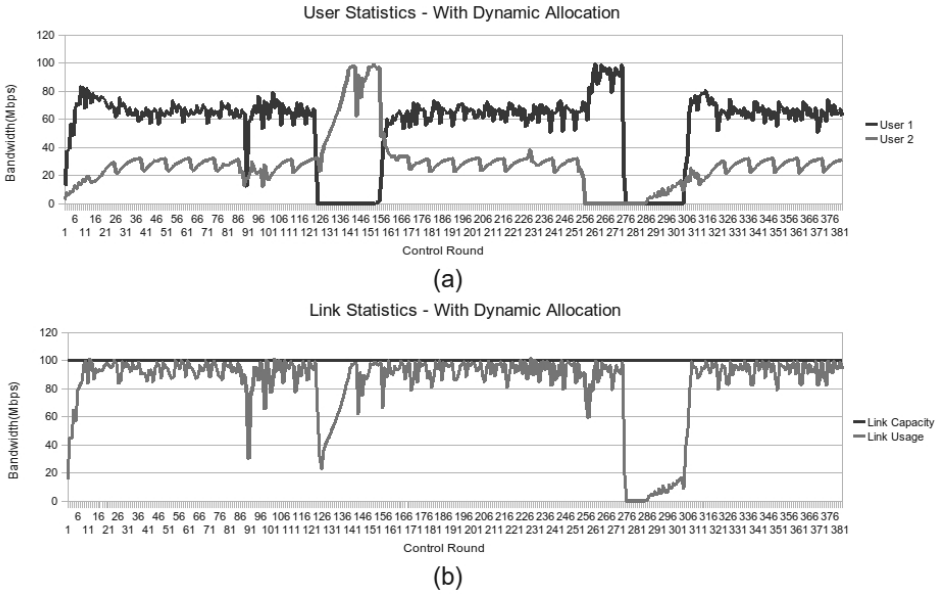


Figure 6.6: (a) User Allocations (b) Link Utilization

Fig.6.5and Fig.6.6, the rate at which utilization decreased and then increased is very sharp in Fig.6.5. That is, the link utilization suddenly dropped to zero at about time 172 , and then suddenly increased to saturation at about time 201. With dynamic allocation active, Fig.6.6 shows that the rate of increase was smooth at first, when Cal_2 started its second round of transmission at time

290. Because ρ_2 was 0.2, Cal_2 reclaimed its shared bandwidth quite slowly. Then, at time 310, the rate of utilization increase became quite high. This was because Cal_1 started its third round of transmission, and ρ_1 was 0.8 - much higher than for Cal_2. Compared to Fig.6.5, these changes are much smoother. It required only the equivalent of 2 control rounds in the uncontrolled case in Fig. 9 to fully saturate the link, while it required 22 control rounds in the controlled case in Fig.6.6.

Our share/reclaim scheme enables controllable changes in transmission rate. Users are not dependent on the uncontrolled behaviors of transport protocols like TCP. If user U_i wants to use the network aggressively, then it can set δ_i and ρ_i to relatively large values. On the other hand, if U_i desires a stable network environment, then it can set δ_i and ρ_i to smaller values, which will lead to smoother variation of the transmission window size.

6.3 Distributed Dynamic Resource Allocation

In the third experiment set we test the effectiveness of our resource allocation scheme running in a distributed manner. We setup a network scenario which includes six users and ten links. The priority values of the users and the optimal bandwidth allocation are shown in Table 6.4. The link configurations is shown in Table 6.5. The topology of the network is shown in Fig.6.7. The bandwidth of link L_1 is 10Gbps, which is much larger than other links. Since all the users share L_1, we have to make sure that L_1 would not be the bottleneck of the network, and therefore would not affect the experiment result. Without further notice, we will neglect the effect of L_1 below.

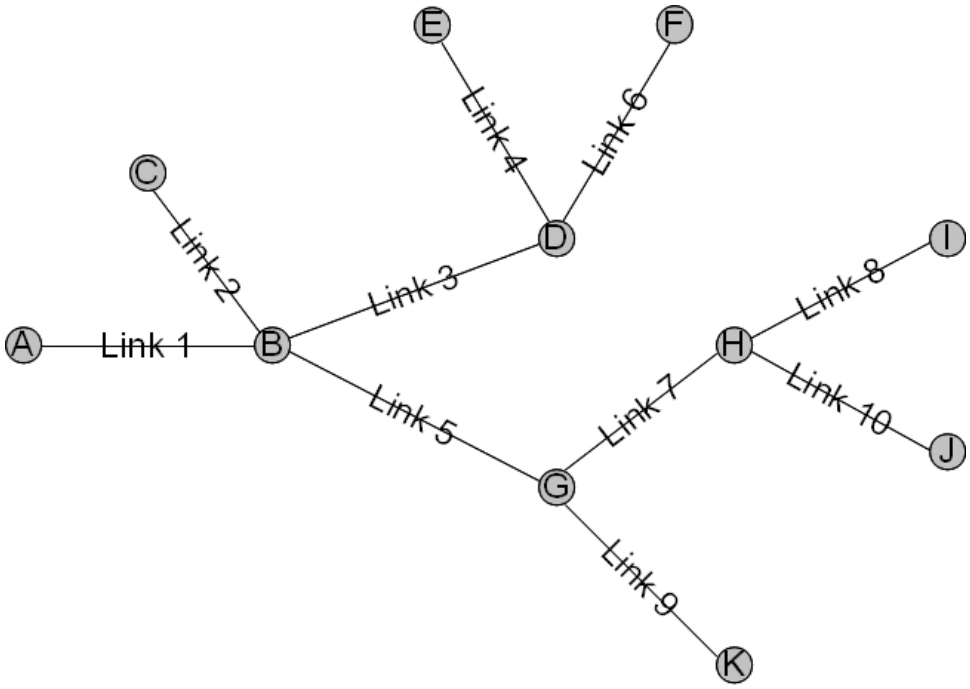
Table 6.4: User Configuration for the Test of Distributed Dynamic Resource Allocation

	User 1	User 2	User 3	User 4	User 5	User 6
ω_i	1	1	3	1	3	1
x_i	100Mbps	25Mbps	75Mbps	20Mbps	60Mbps	20Mbps

We use a simplified additive-increase-multiplicative-decrease (AIMD) function [17] [18] to simulate the behavior of TCP. Each user has a parameter WIN

Table 6.5: Link Configuration for the Test of Distributed Dynamic Resource Allocation

	Link 1	Link 2	Link 3	Link 4	Link 5
C_i	10Gbps	100Mbps	100Mbps	100Mbps	100Mbps
	Link 6	Link 7	Link 8	Link 9	Link 10
C_i	100Mbps	100Mbps	100Mbps	100Mbps	100Mbps



User	Data Flow Route
User 1	A → B → C
User 2	A → B → D → E
User 3	A → B → D → F
User 4	A → B → G → H → I
User 5	A → B → G → H → J
User 6	A → B → G → K

Figure 6.7: Network Topology for the Test of Distributed Dynamic Resource Allocation

called window. This is the same window as defined in TCP. Initially the window size is set to one packet. We also set a window threshold in our experiment, which is similar to TCP. If WIN of a user is lower than the threshold, then after each successful packet transmission, the window size is increased by two packets. If WIN is larger than the threshold, then after each successful packet

transmission, WIN is increased by $\frac{1}{WIN}$. On the other hand, if WIN exceeds the user's bandwidth allocation, WIN is halved.

We assume that all the routers in the network start allocation at the same time. We also assume that the control period of each router is the same, which is set to 1 second. To simplify the problem, we assume that the packet size of each user is uniformly 1500 kilobytes. We also assume that each user sends 100 rounds data within each control round. In other words, the transmission delay of each user for sending the packets is 10ms, and the queueing delays on the route are neglected. We call this 10ms "sending round" for each user. Here we define:

$$PACKET\_SIZE = 1500 kilobytes \tag{6.1}$$

$$WIN\_THRESHOLD = 10 * PACKET\_SIZE \tag{6.2}$$

$$\Gamma = 1s \tag{6.3}$$

$$DELAY = 10ms \tag{6.4}$$

Because the AIMD function is very aggressive, if we give all the initial allocation to the users in the first control round, then the users will consume all the available bandwidth in the first control round. It makes observing the share/reclaim process very hard. Therefore, in the first control round, for each user U_i we set:

$$x_i^j(0) = \sigma_i^j * \frac{1}{10}, \forall L_j \in R_i \tag{6.5}$$

$$\theta_i^j(0) = \sigma_i^j * \frac{9}{10}, \forall L_j \in R_i \tag{6.6}$$

6.3.1 Numerical Experiment 1

The purpose of the first numerical experiment is to show that our distributed dynamic allocation scheme can lead to a stable and convergent allocation result. We assume that for each user U_i, the share factor δ_i and the reclaim factor ρ_i are set to 0.5 uniformly. We assume that all users are elastic users.

There are two methods for the router to predict the bandwidth demand for each user. The first method is to record the maximum transmission rate and use that statistic as the bandwidth demand for each user. In other words, in each sending round u of control round t, the transmission rate of the user U_i is:

$$TX_i(u) = WIN(u) * PACKET\_SIZE/DELAY \tag{6.7}$$

Then the bandwidth demand of user U_i for the next control round $t+1$ is:

$$\eta_i(t+1) = max\{TX_i(u)\} \tag{6.8}$$

The second method is to use the average transmission rate as the bandwidth demand for each user. In other words, the bandwidth demand of user U_i is:

$$\eta_i(t+1) = \frac{\sum_u TX_i(u)}{\Gamma/DELAY} \tag{6.9}$$

First we run an experiment by using the maximum transmission rate as the bandwidth demand for each user. The bandwidth allocation and bandwidth utilization results are shown in Fig.6.8.

Because U_1 only uses L_2, its bandwidth allocation is not affected by other users. The bandwidth allocation of U_1 is gradually increased to 100Mbps, which equals to the capacity of L_2. U_2 and U_3 compete for L_3. At control round 9, the bandwidth allocation of U_2 is higher than 25Mbps. This is because U_3 has extra bandwidth at control round 9, and hence U_2 can borrow from U_3. However, from control round 11, U_3 starts to reclaim its shared bandwidth. Hence, the bandwidth allocation of U_2 decreases. Finally the bandwidth allocations of U_2 and U_3 are 25Mbps and 75Mbps, which equal to

the weighted max-min fair allocation result. This is similar to U_4, U_5, and U_6. At control round 6, the bandwidth allocations of U_4 and U_6 are higher than their optimal allocation. This is because U_5 has unused bandwidth. At control round 11, U_5 starts to reclaim its shared bandwidth, and hence the bandwidth allocations of U_4 and U_6 decrease.

However, the transmission rate does not equal to the bandwidth allocation for the user. This is because we use the simulated AIMD function. Each time the transmission rate exceeds the allocation, the transmission rate is halved. Hence, the average transmission rate shown in Fig.6.8 is lower than the bandwidth allocation for each user.

Next we show that the result proves Proposition 4.1. It is easy to see that L_2 is the bottleneck link of U_1. L_3 is the bottleneck link of U_2 and U_3. L_5 is the bottleneck link of U_4, U_5 and U_6. Hence, we concentrate on these three links and ignore the bandwidth allocation on other links. According to Table 6.4, the maximum initial allocation on L_2, L_3 and L_5 are:

$$\Lambda_2 = max\{\sigma_1\} = 100Mbps \tag{6.10}$$

$$\Lambda_3 = max\{\sigma_2, \sigma_3\} = 75Mbps \tag{6.11}$$

$$\Lambda_5 = max\{\sigma_4, \sigma_5, \sigma_6\} = 60Mbps \tag{6.12}$$

The bandwidth demands and utilizations of L_2, L_3 and L_5 are shown in Fig.6.9. As we discussed in Chapter 4, the link with minimum Λ is saturated first. According to Fig.6.9, L_2 and L_5 are both saturated at control round 10. This is because we use a different function for bandwidth demand prediction. In Chapter 4, we assume that the bandwidth demand of user U_i is $TX_i * \upsilon$, where υ is greater than one. However, in the experiment we use a simulated AIMD function, which would halve the transmission rate when it exceeds the bandwidth allocation. Since L_2 is used only by U_1, there is no competition on L_2. Hence L_2 is saturated at the same time as L_5, and is therefore saturated before L_3.

The changing of the bandwidth demand prediction function also explains that the total transmission rate on each bottleneck link does not equal to the bandwidth capacity. Only 80% bandwidth capacity of L_2 and L_3 is utilized, and only 70% bandwidth capacity of L_5 is utilized. .

Next we run an experiment to demonstrate the result of using the average transmission rate as the bandwidth demand for each user. The bandwidth allocation and bandwidth utilization results are shown in Fig.6.10

We can see that although the final result is stable, a great amount of the bandwidth is wasted. As we explained above, the average tranmission rate is always lower than the bandwidth allocation due to the fact that we are using the AIMD function for the experiment. Hence, the bandwidth demand is always smaller than the current bandwidth allocation, and hence makes the bandwidth allocation continuously decreasing. We claim that using the average transmission rate of the user for predicting its bandwidth demand is inappropriate and can lead to a great waste of the network resources.

6.3.2 Numerical Experiment 2

The purpose of the second numerical experiment is to show that even when a contraint shift happens, the allocation is still stable and optimal. The network we use in this experiment is the same as shown in Fig.4.5. The optimal bandwidth allocation is $x_1 = 25Mbps$, $x_2 = 125Mbps$ and $x_3 = 200Mbps$. We use the maximum transmission rate to predict the bandwidth demand in this experiment.

To make the constraint shift more observable, we set the share factor for each user to 0.9, and we set the reclaim factor for each user to 0.1. If we set the share factor too small, then U_1 would not have the chance to borrow from U_3. If we set the reclaim factor too large, then U_3 would reclaim its shared bandwidth from U_1 very quickly, and hence this process would be very hard to observe.

Then results are shown in Fig.6.11 and Fig.6.12. The bandwidth allocation of U_1 reaches 25Mbps at control round 9. However, since U_2 is not fully utilizing its allocated bandwidth, U_1 continues to borrow bandwidth from U_2.

At control round 31, the bandwidth allocation of U_1 is about 32Mbps, and the bandwidth allocation of U_2 is about 96Mbps. Because U_1 cannot borrow more bandwidth from U_3, this is the maximum bandwidth allocation that U_1 can have. Hence, at control round 31, link L_1 is nearly saturated, and U_1 and U_2 are constrained by L_1. At control round 33, the bandwidth allocation of U_1 starts to decrease. This is because U_3 starts to reclaim its shared bandwidth from U_1 on L_2. At control round 50, the bandwidth allocation of U_1 is about 26Mbps, the bandwidth allocation of U_2 is about 103Mbps, and the bandwidth allocation of U_3 is about 199Mbps. The constraint of U_1 shifts from link L_1 to L_2. After control round 50, the bandwidth allocation U_2 increases above 100Mbps. The extra bandwidth is from U_1. Therefore, we can claim that even when a constraint shift happens, the bandwidth allocation result of our distributed allocation scheme is stable and optimal.

6.3.3 Numerical Experiment 3

The purpose of the third experiment is to show that even if the network topology contains cycle, our distributed algorithm still converges to a stable result. The network we use in this experiment is shown in Fig. 6.13. There are four users and four links in the network. The four links form a cycle in the network. Each user uses two adjacent links. The link parameters are shown in Table 6.6. The user parameters are shown in Table 6.7. The allocation result is shown in Fig. 6.14, which is stable.

Table 6.6: User Configuration for the Test of Network with Cycle

	User 1	User 2	User 3	User 4
ω_i	1	1	1	1
δ_i	0.5	0.5	0.5	0.5
ρ_i	0.5	0.5	0.5	0.5

Table 6.7: Link Configuration for the Test of Network with Cycle

	Link 1	Link 2	Link 3	Link 4
C_j	100Mbps	200Mbps	400Mbps	400Mbps

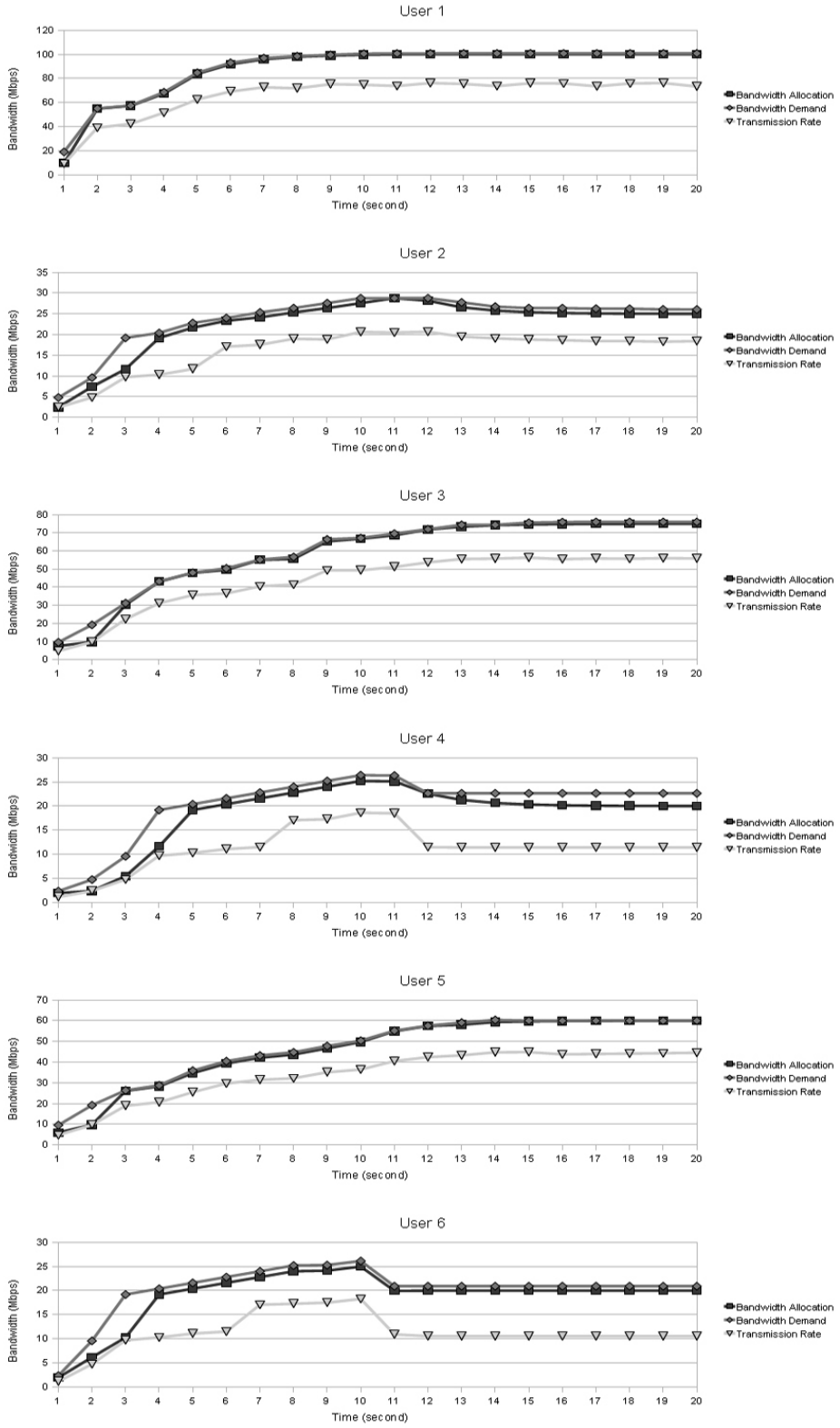


Figure 6.8: Bandwidth Allocation and Bandwidth Usage Result – Using Maximum Transmission Rate to Predict Bandwidth Demand

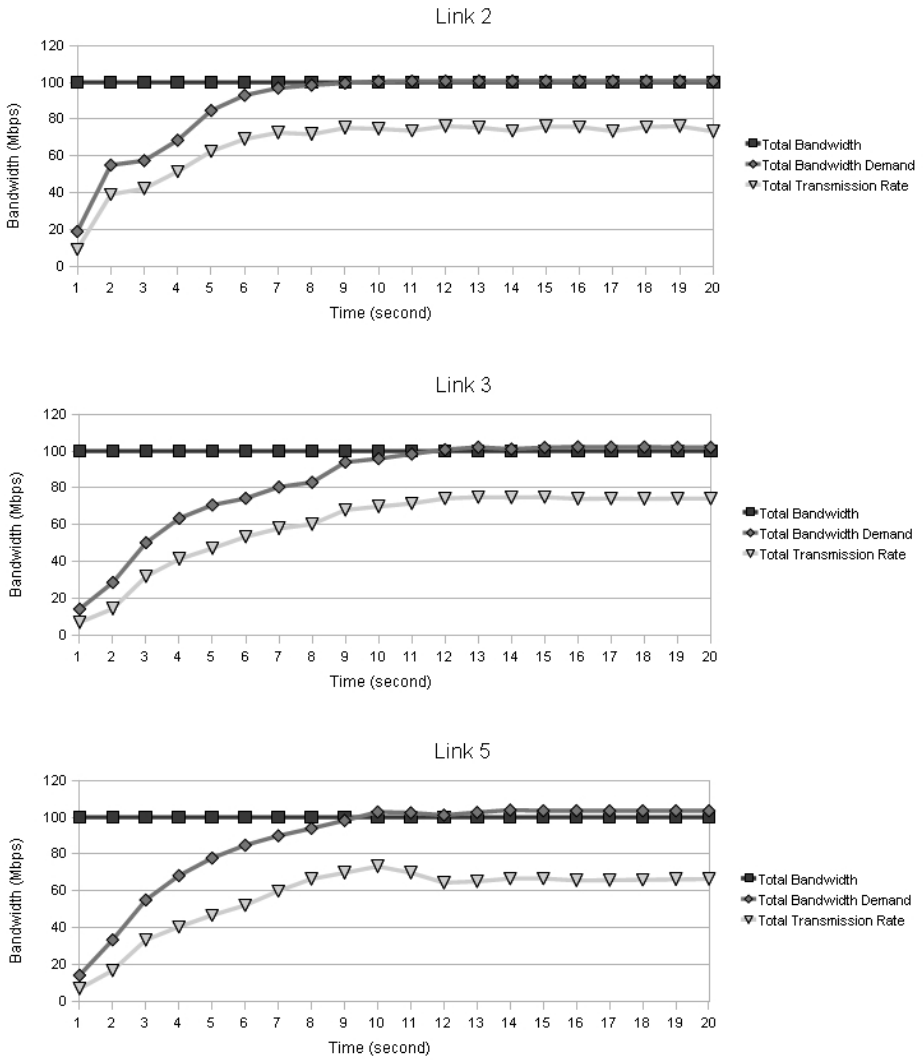


Figure 6.9: Bandwidth Usage on the Link – Using Maximum Transmission Rate to Predict Bandwidth Demand

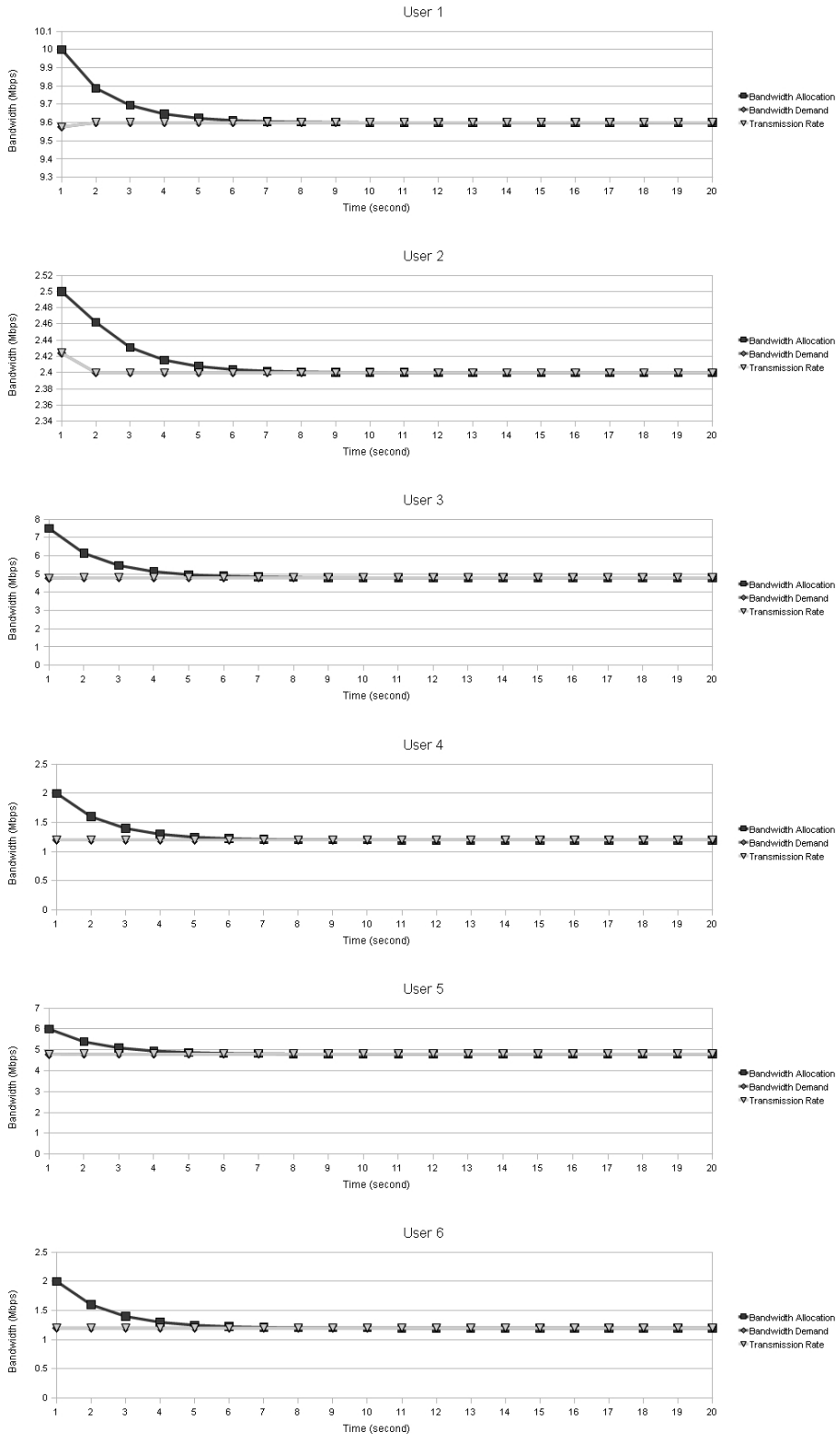


Figure 6.10: Bandwidth Allocation and Bandwidth Usage Result – Using Average Transmission Rate to Predict Bandwidth Demand

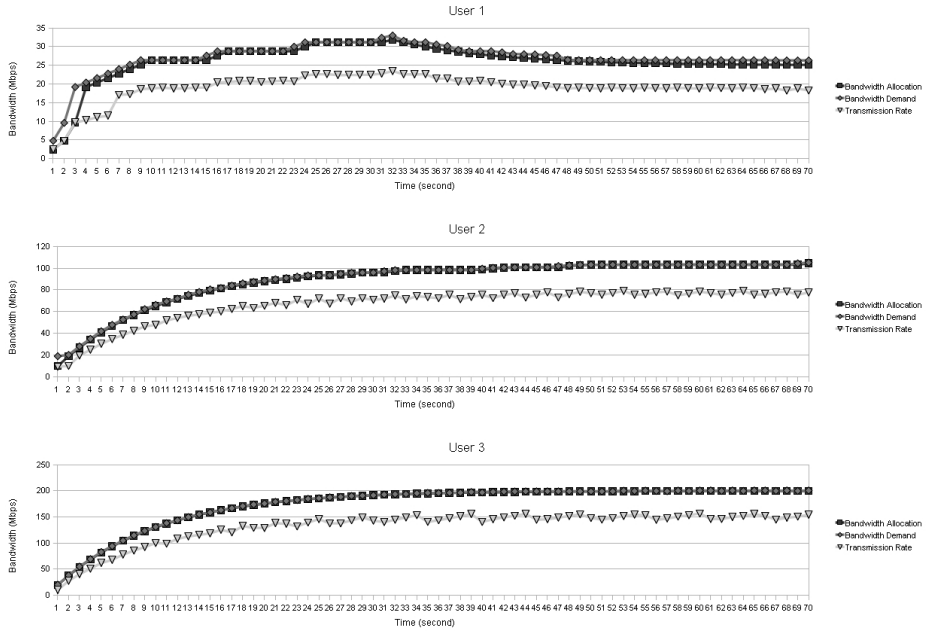


Figure 6.11: Bandwidth Allocation and Bandwidth Usage Result When Constraint Shift Occurs

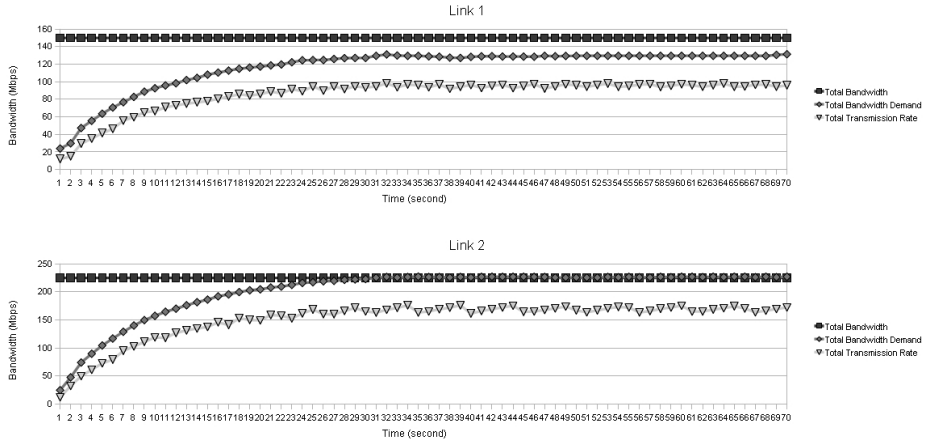


Figure 6.12: Bandwidth Usage on the Link When Constraint Shift Occurs

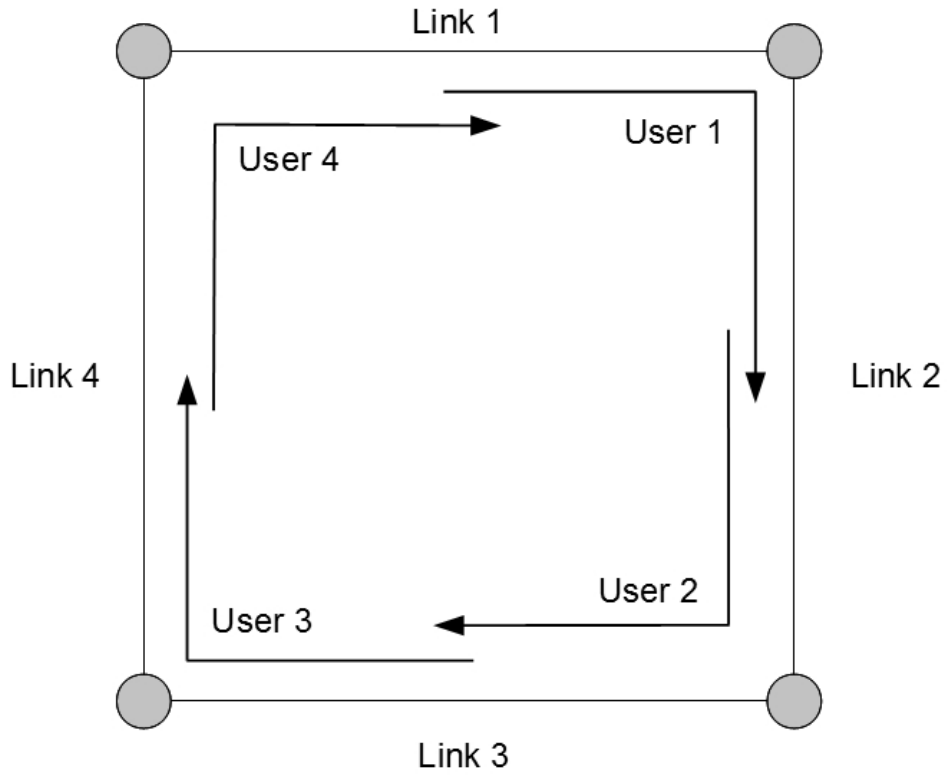


Figure 6.13: Network Topology Contains Cycle

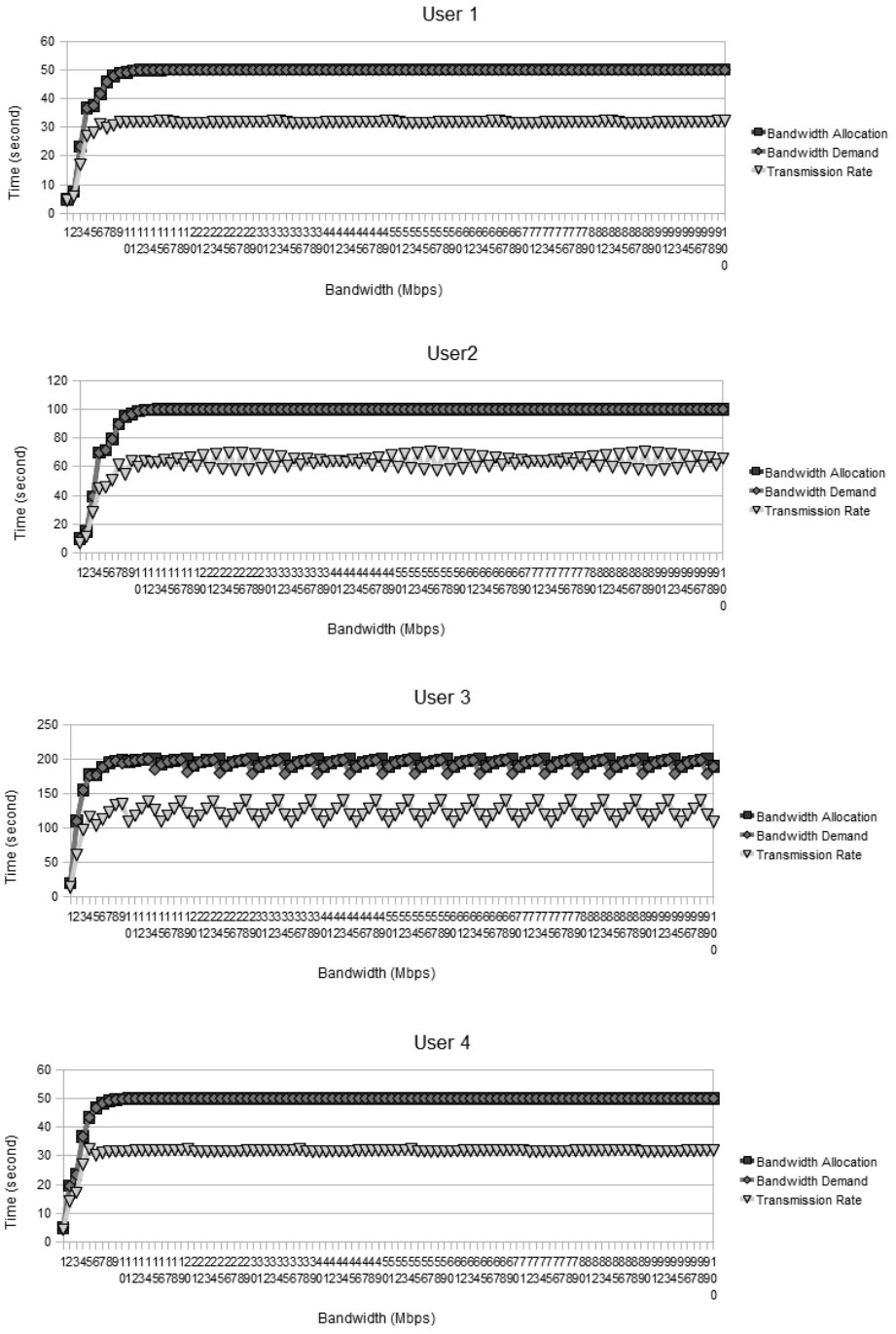


Figure 6.14: Allocation Result with Cycle in the Network

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we have proposed a resource allocation algorithm, which can be used in both centralized and distributed environments. We summarize our contributions as follows.

1 An algorithm which yields weighted max-min fair allocation has been proposed in Chapter 2. This algorithm can be used in both centralized and distributed environment. Even in the distributed environment, which lacks the ability to gather global information, if the users are elastic users, then the allocation result is still weighted max-min fair.

2 We proposed a new algorithm, which dynamically adjusts resource allocations according to the users' demands. We first implemented this algorithm in a centralized manner and applied it to a real problem. We compared the results of the network utilization with and without our algorithm. It showed that our algorithm can provide a fairer allocation result. It also showed that the allocation result is adjusted according to the users' demands.

3 We extended our centralized algorithm to a distributed version. We proved that the allocation result of the distributed algorithm is stable and optimal. The experimental result showed that if all the users in the network are elastic users, then the allocation result is stable and optimal.

4 We proposed several methods which deal with emergencies. Emergency handling was not investigated in any previous work.

7.2 Future Work

Resource allocation, especially network resource allocation, has been researched for decades. Many ideas have been proposed and applied in different fields. In this thesis, we proposed a dynamic resource allocation algorithm which can be applied in both centralied and distributed environments. However, there is still much left to do. Here is a list of possible future directions:

1 We use weighted max-min fair allocation as the initial allocation algorithm. However, there are many other fair algorithms, *e.g.* proportional fair allocation, that can be applied.

2 As we discussed in Chapter 3, there are several possible borrow strategies. We can see that a different borrow strategy can lead to a different allocation result. However, each of them has its own disadvantages. Therefore, more investigation is needed to generalize which borrow strategy applies to which situation.

3 Bandwidth demand prediction is another problem. In Chapter 3, the centralized version of the algorithm assumes that the network is a private network, such that the central manager has the knowledge of the link bandwidth and the users know exactly their bandwidth demands. However, this does not always apply. In Chapter 4, the router uses the user's peak bandwidth usage as its estimate of bandwidth demand for the next control round. The experimental result in Chapter 6 shows that using average bandwidth usage on the router as the bandwidth demand prediction can lead to performance degradation. Hence, a method that can accurately reflect the user's bandwidth demand is critical.

4 Our work assumes that the users in the network would collaborate with the central manager or the router. This means that the users are friendly

users. If there are one or more malicious users in the network, then they can send a fake bandwidth demand to the central manager, or can ignore the regulation on the router. This would severely decrease the performance of the allocation algorithm, and the result would not be fair any more. Hence, a robust and protected algorithm needs to be investigated.

5 The emergency handling methods leave many possible paths for future work. We compared the cons and pros of increasing/decreasing bandwidth method in Chapter 5. We also discussed the method of using a emergency handling policy table. How to apply them to different situations would be a very interesting topic. Moreover, other possible methods would be interesting to investigate.

Bibliography

[1] D.P. Bertsekas, R. Gallager, *Data Networks*, ch.6, Prentice Hall, 2nd Edition, 1994.

[2] P. Marbach, *Priority Service and Max-Min Fairness*, IEEE/ACM Transactions on Networking, vol.11, issue 5, pp.733-746, 2003.

[3] L. Massoulie and J. Roberts, *Bandwidth Sharing: Objectives and Algorithms*, IEEE/ACM Transactions on Networking, vol.10, issue 3, pp.320-328, 2002.

[4] F.P. Kelly, A.K. Maulloo, D.K.H. Tan, *Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability*, The Journal of the Operational Research Society, vol.49, no.3, pp.237-252, 1998.

[5] E.E. Graves, R. Srikant and D. Towsley, *Decentralized Computation of Weighted Max-Min Fair Bandwidth Allocation in Networks with Multicast Flows*, Evolutionary Trends of the Internet, vol.2170, 2001.

[6] D.J. White, *A Heuristic Approach to a Weighted Max Min Dispersion Problem*, IMA Journal of Mathematics Applied in Business and Industry, vol.7, pp.219-231, 1996.

[7] T. Bonald, L. Massoulie, A. Proutiere, J. Virtamo, *A Queueing Analysis of Max-Min Fairness, Proportional Fairness and Balanced Fairness*, Journal of Queueing Systems, vol.53, no.1-2, pp.65-84, 2006.

[8] B. Radunovic, J.L. Boudec, *A Unified Framework for Max-Min and Min-Max Fairness With Applications*, IEEE/ACM Transactions on Networking, vol.15, issue 5, pp.1073-1083, 2007

[9] J. LeBoudec, *Rate Adaption, Congestion Control and Fairness: A Tutorial*, http://icapeople.epfl.ch/leboudec.

[10] A. Chandra, W. Gong, P. Shenoy, *Dynamic Resource Allocation for Shared Data Centers Using Online Measurements*, Quality of Service-IWQoS, vol.2707, pp.381-398, 2003.

[11] S. Sarkar, L. Tassiulas, *Distributed Algorithms for Computation of Fair Rates in Multirate Multicast Trees*, IEEE INFOCOM 2000, vol.1, pp.52-61, 2000.

[12] S. Sarkar, L. Tassiulas, *Fair Allocation of Discrete Bandwidth Layers in Multicast Networks*, IEEE INFOCOM 2000, vol.3, pp.1491-1500, 2000.

[13] A. Bar-Noy, R. Cannetti, S. Kutten, Y. Mansour, B. Schieber, *Bandwidth Allocation with Preemption*, Proceedings of the 27th ACM Symposium on Theory of Computing, pp.616-625, 1995.

[14] Y.A. Korilis, A.A. Lazar, A. Orda, *Capacity Allocation under Noncooperative Routing*, IEEE Transactions on Automatic Control, vol.42, no.3, pp.309-325, 1997.

[15] F.P. Kelly, *Charging and Rate Control for Elastic Traffic*, European Transactions on Telecommunications, vol.8, no.1, pp.33-38, 1997.

[16] Y. Afek, Y. Mansour, Z. Ostfeld, *Phantom: A Simple and Effective Flow Control Scheme*, Proceedings on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp.169-182, 1996.

[17] B. Behsaz, P. Gburzynski, M. MacGregor, *Transport-Independent Fairness*, Computer Networks, Vol. 53, Issue. 14, pp. 2444-2457, Sept. 2009.

[18] A. Charny, *An Algorithm for Rate Allocation in A Packet-switching Network with Feedback*, http://dspace.mit.edu/handle/1721.1/12000

Appendix A

Symbol List

NET: The network model.

U: The user set in the network. U_i represents the user i in the user set U.

L: The link set in the network. L_j represents the link j in the link set L.

S: The router set in the network. S_k represents the router k in the router set S.

R: The router of a user. R_i represents the data flow path of user U_i. If $L_j \in R_i$, then user U_i uses link L_j. If $S_k \in R_i$, then user U_i uses router S_k.

M: The central manager in the network.

C: The link capacity. Link L_j has link capacity of C_j.

x_i**:** The bandwidth allocation of user U_i.

ω_i**:** The priority of user U_i.

Ω_j**:** The total priority of the users using link L_j.

τ_j**:** The average bandwidth per unit of priority on link L_j.

Γ_M**:** The control period of the central manager M.

σ_i**:** The initial bandwidth allocation of user U_i.

θ_i**:** The shared bandwidth of user U_i.

δ_i**:** The share factor of user U_i.

ρ_i**:** The reclaim factor of user U_i.

η_i**:** The bandwidth demand of user U_i.

ϵ_i**:** The bandwidth gap of user U_i.

κ_i**:** The borrowed bandwidth of user U_i.

Θ_j**:** The total shared bandwidth on link L_j.

Λ_j**:** The maximum initial allocation among the users using L_j.