

## Network Optimization and Control

Srinivas Shakkottai<sup>1</sup> and R. Srikant<sup>2</sup>

<sup>1</sup> *Texas A&M University, USA*

<sup>2</sup> *University of Illinois, Urbana-Champaign, USA, rsrikant@uiuc.edu*

### Abstract

We study how protocol design for various functionalities within a communication network architecture can be viewed as a distributed resource allocation problem. This involves understanding what resources are, how to allocate them fairly, and perhaps most importantly, how to achieve this goal in a distributed and stable fashion. We start with ideas of a centralized optimization framework and show how congestion control, routing and scheduling in wired and wireless networks can be thought of as fair resource allocation. We then move to the study of controllers that allow a decentralized solution of this problem. These controllers are the analytical equivalent of protocols in use on the Internet today, and we describe existing protocols as realizations of such controllers. The Internet is a dynamic system with feedback delays and flows that arrive and depart, which means that stability of the system cannot be taken for granted. We show how to incorporate stability into protocols, and thus, prevent undesirable network behavior. Finally, we consider a futuristic scenario where users are aware of the effects of their actions and try to game the system. We will see that the optimization framework is remarkably robust even to such gaming.

# Contents

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Network Utility Maximization</b>	<b>5</b>
2.1	Utility Maximization in Networks	7
2.2	Resource Allocation in Wireless Networks	12
2.3	Node-Based Flow Conservation Constraints	15
2.4	Fairness	16
2.5	Notes	19
<b>3</b>	<b>Utility Maximization Algorithms</b>	<b>21</b>
3.1	Primal Formulation	22
3.2	Dual Formulation	33
3.3	Extension to Multi-path Routing	38
3.4	Primal-Dual Algorithm for Wireless Networks	41
3.5	Notes	49
<b>4</b>	<b>Congestion Control Protocols</b>	<b>53</b>

ii *Contents*

4.1	TCP-Reno	55
4.2	Relationship with Primal Algorithm	58
4.3	A Generalization of TCP-Reno	59
4.4	TCP-Vegas: A Delay Based Algorithm	60
4.5	TCP-Vegas as a Resource Allocation Algorithm	62
4.6	Relation to Dual Algorithms and Extensions	64
4.7	Notes	66
<b>5</b>	<b>Network Stability</b>	<b>69</b>
5.1	Stability of the Primal Algorithm with Delays	71
5.2	Stability Under Flow Arrivals and Departures	80
5.3	Notes	84
<b>6</b>	<b>Game Theory and Resource Allocation</b>	<b>87</b>
6.1	VCG Mechanism	88
6.2	Kelly Mechanism	91
6.3	Strategic or Price-Anticipating Users	93
6.4	Notes	100
	<b>Conclusions</b>	<b>101</b>
	<b>Acknowledgements</b>	<b>105</b>
	<b>References</b>	<b>107</b>

# 1

---

## Introduction

---

The Internet has been one of the most conspicuous successes of the communications industry, and has permeated every aspect of our lives. Indeed, it is a testimony to its relevance in modern lifestyle that Internet connectivity is now considered an essential service like electricity and water supply. The idea that a best-effort data service could attain such significance was perhaps difficult to imagine a couple of decades ago. In fact, the Internet was initially envisioned as a decentralized data transmission network for military use. The argument was that if there were no centralized control, such as in a telephone network, and if much of the intelligence was at the edges of the system, it would make the network that much harder to destroy. Concurrent with the indestructibility requirements of the military was the need of scientific laboratories which required a network to exchange large data files of experimental results with each other. They envisioned high-speed links for transferring data between geographically distant data sources. The two requirements, coupled with statistical multiplexing ideas that illustrated the efficiency of using packetized data transmission gave rise to the Internet.

## 2 Introduction

As the network grew, it was clear that unrestricted data transfer by many users over a shared resource, i.e., the Internet, could be bad for the end users: excess load on the links leads to packet loss and decreases the effective throughput. This kind of loss was experienced at a significant level in the '80s and was termed *congestion collapse*. Thus, there was a need for a protocol to control the congestion in the network, i.e., control the overloading of the network resources. It led to the development of a congestion control algorithm for the Internet by Van Jacobson [1]. This congestion control algorithm was implemented within the protocol used by the end hosts for data transfer called the Transmission Control Protocol (TCP). Even though TCP is a lot more than just a congestion control algorithm, for the purposes of this monograph, we will use the terms “TCP” and “TCP congestion control algorithm” interchangeably.

Congestion control can also be viewed as a means of allocating the available network resources in some fair manner among the competing users. This idea was first noted by Chiu and Jain [2] who studied the relationship between control mechanisms at the end-hosts which use one-bit feedback from a link and the allocation of the available bandwidth at the link among the end-hosts. In fact, some of the details of Jacobson’s congestion control algorithm for the Internet were partly influenced by the analytical work in [2]. The resource allocation viewpoint was significantly generalized by Kelly et. al. [3] who presented an optimization approach to understanding congestion control in networks with arbitrary topology, not just a single link. The purpose of this monograph is to present a state-of-the-art view of this optimization approach to network control. The material presented here is complementary to the book [4]. While the starting point is the same, i.e., the work in [3], this monograph focuses primarily on the developments since the writing of [4]. The purpose of this monograph is to provide a starting point for a mature reader with little background on the subject of congestion control to understand the basic concepts underlying network resource allocation. While it would be useful to the reader to have an understanding of optimization and control theory, we have tried to make the monograph as self contained as possible to make it accessible to a large audience. We have made it a point to provide extensive

references, and the interested reader could consult these to obtain a deeper understanding of the topics covered. We hope that by providing a foundation for understanding the analytical approach to congestion control, the review will encourage both analysts and systems designers to work in synergy while developing protocols for the future Internet.

The monograph is organized as follows. We state the resource allocation objective in Chapter 2 and present the optimization formulation of the resource allocation problem. In Chapter 3, we will study decentralized, dynamic algorithms that solve the optimization problem. The chapter explains the framework for the design of such algorithms and proves the convergence of these algorithms. We then study current and recently proposed congestion control protocols, and their relationship to the optimization framework in Chapter 4. We then proceed to study the question of network stability in Chapter 5. We study two concepts of stability—that of convergence of algorithms to the fair allocation in the presence of feedback delays, and the question of whether the number of flows in the system would remain finite when flows arrive and depart. Finally, in Chapter 6, we study resource allocation from a game-theoretic viewpoint. In all the previous chapters, it was assumed that the users cooperate to maximize network utility. In Chapter 6, we study selfish users whose goal is to maximize their own utility and their impact on the total network utility.



# 2

---

## Network Utility Maximization

---

In this chapter we will study the problem of fair resource allocation in networks. The material presented here is central to the design of optimal congestion control algorithms. Before we consider what resources are available on the Internet, let us consider a simple example of resource allocation. Suppose that a central authority has a divisible good of size  $C$  that is to be divided among  $N$  different users, as illustrated in Figure 2.1. For example, suppose that the government has determined that a fixed quantity of ground water may be pumped in a certain region and would like to allocate quotas that may be pumped by different farms. One way of performing the allocation is to simply divide the resource into  $N$  parts and allocate  $C/N$  to each player. But such a scheme does not take into account the fact that each player might value the good differently. In our example, based on the type of crops being grown, the value of pumping a certain quantity water might be different for different farms. We refer to the value or “utility” obtained from an allocation  $x$  as  $U(x)$ . This utility is measured in any denomination common to all the players such as dollars. So a farm growing rice would have a higher utility for water than a farm growing wheat.

What would be the properties of a utility function? We would ex-



## 6 Network Utility Maximization

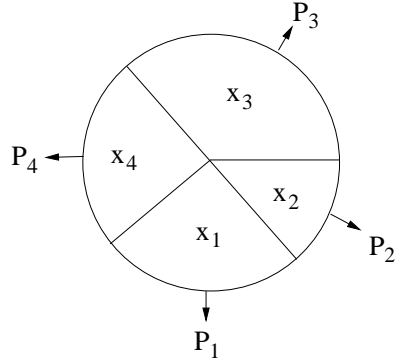


Fig. 2.1 An example of a resource that can be divided among multiple players. How can it be divided so as to maximize the social good?

pect that it would be increasing in the amount of resource obtained. More water would imply that a larger quantity of land could be irrigated leading to a larger harvest. We might also expect that a law of diminishing returns applies. In our example of water resources, the return obtained by increasing the quota from 10 units to 20 units would make a large difference in the crop obtained, but an increase from 100 units to 110 units would not make such a significant difference. Such a law of diminishing returns is modeled by specifying that the utility function is a strictly concave function since the second derivative of a strictly concave function is negative. Thus, the first derivative (which is the rate at which the function increases) decreases.

The objective of the authority would be to maximize the “system-wide” utility. One commonly used measure of system-wide utility is the sum of the utilities of all the players. Since the utility of each player can be thought of the happiness that he/she obtains, the objective of the central authority can be likened to maximizing the total happiness in the system, subject to resource constraints. We illustrate the sum-utility in Figure 2.2, with the utility functions of players 1 and 2, who receive an allocation  $x_1$  and  $x_2$  respectively being  $2 \log(x_1)$  and  $\log(x_2)$ . The marginals in the figure are strictly concave, which results in the sum being strictly concave as well.

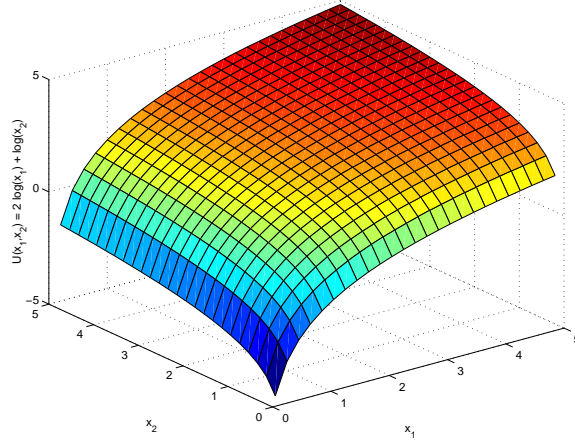


Fig. 2.2 Sum-utility of two strictly concave marginals.

## 2.1 Utility Maximization in Networks

We now consider the analog of the resource allocation problem in a communication network such as the Internet. Suppose we have a network with a set of traffic sources  $\mathcal{S}$  and a set of links  $\mathcal{L}$ . Each link  $l \in \mathcal{L}$  has a finite fixed capacity  $c_l$ . Each source in  $\mathcal{S}$  is associated with a route  $r \subset \mathcal{L}$  along which it transmits at some rate  $x_r$ . Note that we can use the index  $r$  to indicate both a route and the source that sends traffic along that route and we will follow this notation. Also since multiple sources could use the same set of links as their routes, there could be multiple indices  $r$  which denote the same subset of  $\mathcal{L}$ . The utility that the source obtains from transmitting data on route  $r$  at rate  $x_r$  is denoted by  $U_r(x_r)$ . We assume that the utility function is continuously differentiable, non-decreasing and strictly concave. As mentioned before, the concavity assumption follows from the diminishing returns idea—a person downloading a file would feel the effect of a rate increase from 1 kbps to 100 kbps much more than an increase from 1 Mbps to 1.1 Mbps although the increase is the same in both cases.

It is immediately clear that the scenario outlined is just a slightly

more complicated version of the water resource management example given earlier—instead of just one resource, multiple resources must be allocated and the same amount of resource must be allocated on all links constituting a route. It is straightforward to write down the problem as an optimization problem of the form

$$\max_{x_r} \sum_{r \in \mathcal{S}} U_r(x_r) \quad (2.1)$$

subject to the constraints

$$\sum_{r: l \in r} x_r \leq c_l, \quad \forall l \in \mathcal{L}, \quad (2.2)$$

$$x_r \geq 0, \quad \forall r \in \mathcal{S}. \quad (2.3)$$

The above inequalities state that the capacity constraints of the links cannot be violated and that each source must be allocated a non-negative rate of transmission. It is well known that a *strictly concave function has a unique maximum over a closed and bounded set*. The above maximization problem satisfies the requirements as the utility function is strictly concave, and the constraint set is closed (since we can have aggregate rate on a link equal to its capacity) and bounded (since the capacity of every link is finite). In addition, the constraint set for the utility maximization problem is convex which allows us to use the method of Lagrange multipliers and the Karush-Kuhn-Tucker (KKT) theorem which we state below [5, 6].

---

**Theorem 2.1.** Consider the following optimization problem:

$$\max_x f(x)$$

subject to

$$g_i(x) \leq 0, \quad \forall i = 1, 2, \dots, m,$$

and

$$h_j(x) = 0, \quad \forall j = 1, 2, \dots, l,$$

where  $x \in \mathbb{R}^n$ ,  $f$  is a concave function,  $g_i$  are convex functions and  $h_j$  are affine functions. Let  $x^*$  be a feasible point, i.e., a point that satisfies

all the constraints. Suppose there exists constants  $\lambda_i \geq 0$  and  $\mu_j$  such that

$$\frac{\partial f}{\partial x_k}(x^*) - \sum_i \lambda_i \frac{\partial g_i}{\partial x_k}(x^*) + \sum_j \mu_j \frac{\partial h_j}{\partial x_k}(x^*) = 0, \quad \forall k, \quad (2.4)$$

$$\lambda_i g_i(x^*) = 0, \quad \forall i, \quad (2.5)$$

then  $x^*$  is a global maximum. If  $f$  is strictly concave, then  $x^*$  is also the unique global maximum.

---

The constants  $\lambda_i$  and  $\mu_j$  are called Lagrange multipliers. The KKT conditions (2.4)-(2.5) can be interpreted as follows. Consider the Lagrangian

$$L(x, \lambda, \mu) = f(x) - \sum_i \lambda_i g_i(x) + \sum_j \mu_j h_j(x).$$

Condition (2.4) is the first-order necessary condition for the maximization problem  $\max_x L(x, \lambda, \mu)$ . Further, it can be shown that the vector of Lagrange multipliers  $(\lambda, \mu)$  solves  $\min_{\lambda, \mu} D(\lambda, \mu)$  where  $D(\lambda, \mu) = \max_x L(x, \lambda, \mu)$  is called the Lagrange dual function.

Next, we consider an example of such a maximization problem in a small network and show how one can solve the problem using the above method of Lagrange multipliers.

---

**Example 1.** Consider the network in Figure 2.3 in which three sources compete for resources in the core of the network. Links  $L_1$ ,  $L_3$  and  $L_5$  have a capacity of 2 units per second, while links  $L_2$  and  $L_4$  have capacity 1 unit per second. There are three flows and denote their data rates by  $x_0$ ,  $x_1$  and  $x_2$ .

In our problem, links  $L_3$  and  $L_4$  are not used, while  $L_5$  does not constrain source  $S_2$ . Assuming log utility functions, the resource allocation problem is given by

$$\max_x \sum_{i=0}^3 \log x_i \quad (2.6)$$



DRAFT

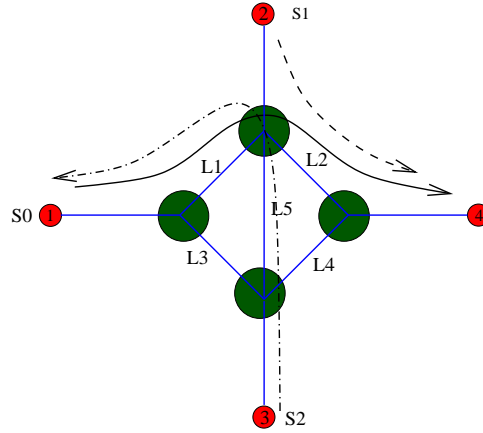


Fig. 2.3 Example illustrating network resource allocation. The large circles indicate routers that route packets in the core network. We assume that links  $L_1$ ,  $L_3$  and  $L_5$  have capacity 2, while  $L_2$  and  $L_4$  have capacity 1. The access links of the sources are unconstrained. There are three flows in the system.

with constraints

$$\begin{aligned} x_0 + x_1 &\leq 2, \\ x_0 + x_2 &\leq 1, \\ x &\geq 0, \end{aligned}$$

where  $x$  is the vector consisting of  $x_0, x_1$  and  $x_2$ . Now, since  $\log x \rightarrow -\infty$  as  $x \rightarrow 0$ , it means that the optimal resource allocation will not yield a zero rate for any source, even if we remove the non-negativity constraints. So the last constraint above is not active.

We define  $\lambda_1$  and  $\lambda_2$  to be the *Lagrange multipliers* corresponding to the capacity constraints on links  $L_1$  and  $L_2$ , respectively, and let  $\lambda$  denote the vector of Lagrange multipliers. Then, the *Lagrangian* is given by

$$L(x, \lambda) = \log x_0 + \log x_1 + \log x_2 - \lambda_1(x_0 + x_1) - \lambda_2(x_0 + x_2).$$

Setting  $\frac{\partial L}{\partial x_r} = 0$  for each  $r$  gives

$$x_0 = \frac{1}{\lambda_1 + \lambda_2}, \quad x_1 = \frac{1}{\lambda_1}, \quad x_2 = \frac{1}{\lambda_2}.$$

Letting  $x_0 + x_1 = 2$  and  $x_0 + x_2 = 1$  (since we can always increase  $x_2$  or  $x_3$  until this is true) yields

$$\lambda_1 = \frac{\sqrt{3}}{\sqrt{3} + 1} = 0.634, \quad \lambda_2 = \sqrt{3} = 1.732.$$

Note that (2.5) is automatically satisfied since  $x_0 + x_1 = 2$  and  $x_0 + x_2 = 1$ . It is also possible to verify that the values of the Lagrange multipliers actually are the minimizers of the dual function. Hence, we have the final optimal allocation

$$\hat{x}_0 = \frac{\sqrt{3} + 1}{3 + 2\sqrt{3}} = 0.422, \quad \hat{x}_1 = \frac{\sqrt{3} + 1}{\sqrt{3}} = 1.577, \quad \hat{x}_2 = \frac{1}{\sqrt{3}} = 0.577.$$

A few facts are noteworthy in this simple network scenario:

- Note that  $x_1 = 1/\lambda_1$ , and it does not depend on  $\lambda_2$  explicitly. Similarly,  $x_2$  does not depend on  $\lambda_1$  explicitly. In general, the optimal transmission rate for source  $r$  is only determined by the Lagrange multipliers on its route. We will later see that this feature is extremely useful in designing decentralized algorithms to reach the optimal solution.
- The value of  $x_r$  is inversely proportional to the sum of the Lagrange multipliers on its route. We will see later that, in general,  $x_r$  is a decreasing function of the Lagrange multipliers. Thus, the Lagrange multiplier associated with a link can be thought of the price for using that link and the price of a route can be thought of as the sum of the prices of its links. If the price of a route increases, then the transmission rate of a source using that route decreases.

---

In the above example, it was easy to solve the Lagrangian formulation of the problem since the network was small. In the Internet which consists of thousands of links and possibly millions of users such an approach is not possible. In the next chapter, we will see that there are distributed solutions to the optimization problem which are easy to implement in the Internet.



DRAFT

## 2.2 Resource Allocation in Wireless Networks

In the previous section, we considered the case of resource allocation in a wire-line network. Thus, we did not have to consider whether or not a particular link would be available for transmission. **This assumption need not hold in a wireless network** where scheduling one “link” for transmission might mean that some other “link” is unavailable due to interference. While the primary goal of this monograph is Internet congestion control, we will briefly show how the models and methods developed for the Internet can also be applied to wireless networks.

For ease of exposition, consider a simple model of a wireless network. We model the network as a graph with a link between two nodes indicating that the nodes are communication range of each other. When a link is scheduled, the transmitter can transmit at a fixed rate to the receiver. Further, a transmission from a transmitter, call it transmitter  $A$ , to a receiver is successful if there are no other transmitters (other than  $A$ ) within a pre-specified fixed radius around the receiver. There may be other constraints as well depending upon the details of wireless network protocols used.

We illustrate the model in Figure 2.4. There are three flows in progress and two of the nodes act as destinations. Around each destination, we draw a circle that is called either the reception range or interference region, depending on the context, associated with that destination. A transmitter has to lie within the reception range of a receiver for the transmission to be successful. In addition, no other node in the interference region can transmit if the receiver is to receive the transmitted data successfully. Thus, if node 2 is to successfully receive data, either node 3 or node 1 may transmit, but not both at the same time. In addition, many wireless networks employ an acknowledgment (ack) protocol on each link: when a receiver successfully receives a packet, it sends an ack back to the transmitter. Thus, for an ack to be successful, there cannot be any transmissions within the listening range of the transmitter as well. A transmission is not considered successful till the transmitter receives an ack from the receiver. In Figure 2.4, if node 3 is to receive a transmission from node 2 successfully, node 1 cannot transmit since it would interfere with the successful reception of an ack

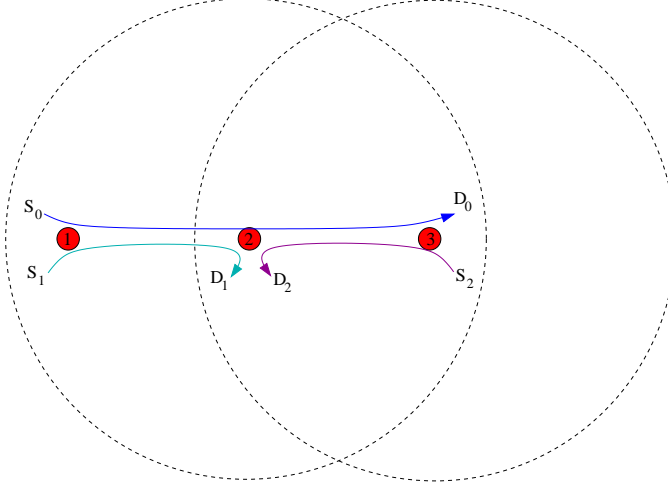


Fig. 2.4 Wireless Interference Model. The dashed circles indicate the interference regions of the two destination nodes. Due to interference, only one node can transmit at a time. We have to schedule the fraction of time allocated to the two possible hops as well as the rate allocated to each flow.

from node 3 to node 2. We assume that hops are of unit capacity, i.e., if scheduled, a hop could transfer packets at the rate of 1 packet per time unit. The figure shows three flows that share the wireless network. Clearly, we have a more constrained problem than we considered earlier, since we now have to consider both the scheduling of the links to **avoid interference as well as rate allocation to each flow.**

We incorporate the interference constraint by introducing new variables which indicate the fraction of time that each link is scheduled. Let  $R_{ij}$  be the fraction of time that node  $i$  transmits to node  $j$ . Also, define the vector  $R = [R_{12}, R_{23}, R_{32}]^T$ . Since nodes 1 and 2 cannot transmit simultaneously, we have  $R_{12} + R_{23} + R_{32} \leq 1$ . Thus, the network utility maximization problem becomes

$$\max_x \sum_{i=0}^2 U_i(x_i)$$



$$\begin{aligned}
x_0 + x_1 &\leq R_{12} \\
x_0 &\leq R_{23} \\
x_2 &\leq R_{32} \\
R_{12} + R_{23} + R_{32} &\leq 1 \\
x, R &\geq 0
\end{aligned}$$

We now use **Lagrange multipliers** to append the constraints to the objective. Thus, we can write the dual as

$$\begin{aligned}
\max_{x, R} \sum_{i=0}^2 U_i(x_i) - \lambda_1(x_0 + x_1 - R_{12}) - \lambda_2(x_0 - R_{23}) - \lambda_3(x_2 - R_{32}) \\
+ \sum_j \gamma_j x_j - \nu(R_{12} + R_{23} + R_{32} - 1) + \sum_{kl} \beta_{kl} R_{kl} \quad (2.7)
\end{aligned}$$

Observe (2.7) carefully. We see that there are no product terms involving  $x$  and  $R$  together. Hence we can decompose the problem into two separate sub-problems. The first one which we call the *utility maximization problem* is of the form

$$\max_{x \geq 0} \sum_{i=0}^2 U_i(x_i) - \lambda_1(x_0 + x_1) - \lambda_2 x_0 - \lambda_3 x_2, \quad (2.8)$$

while the second one which we call the *scheduling problem* is

$$\max_{\sum_{kl} R_{kl} \leq 1, R \geq 0} \lambda_1 R_{12} + \lambda_2 R_{23} + \lambda_3 R_{32}. \quad (2.9)$$

Note that in writing (2.8) and (2.9), we have chosen to write some of the constraints explicitly rather than in the Lagrange multiplier form, but these are equivalent to the Lagrange multiplier form and are easier to grasp intuitively.

We have seen expressions like (2.8) in the wired network example, but did not encounter link-scheduling issues. The expression in (2.9) uses the Lagrange multipliers ( $\lambda_i$ ) as weights to determine the fractions of time that a particular hop is scheduled. Such an algorithm to perform scheduling is called a **maximum-weight algorithm**, or *max-weight* algorithm for short.

## 2.3 Node-Based Flow Conservation Constraints

So far, we have posed the resource allocation problem as one where the goal is to maximize network utility subject to the constraint that the total arrival rate on each link is less than or equal to the available capacity. There is an alternative way to express the resource constraints in the network by noting that the total arrival rate into a node must be less than or equal to the total service rate available at the node. From a resource allocation point of view, both formulations are equivalent. However, there are subtle differences in the implementation and analysis of decentralized algorithms required to solve the two formulations. These differences will be discussed in the next chapter. As we will see in the next chapter, these differences seem to be more important in wireless networks and therefore, we illustrate the idea for wireless networks.

As we did in the previous section, we now consider the example with 3 flows that was shown in Figure 2.4. Again, let the rates associated with sources  $S_0$ ,  $S_1$  and  $S_2$  be  $x_0$ ,  $x_1$  and  $x_2$ , respectively. Let  $R_{ij}^{(d)}$  be the fraction of time that link  $(i, j)$  is dedicated to a flow destined for node  $d$ . The network utility maximization problem is

$$\max_{x, R \geq 0} \sum_{i=0}^2 U_i(x_i)$$

$$\begin{aligned} x_0 &\leq R_{12}^{(3)} \text{ (constraint at node 1 for destination node 3)} \\ x_1 &\leq R_{12}^{(2)} \text{ (constraint at node 1 for destination node 2)} \\ R_{12}^{(3)} &\leq R_{23}^{(3)} \text{ (constraint at node 2 for destination node 3)} \\ x_2 &\leq R_{32}^{(2)} \text{ (constraint at node 3 for destination node 2)} \\ R_{12}^{(3)} + R_{12}^{(2)} + R_{23}^{(3)} + R_{32}^{(2)} &\leq 1 \text{ (interference and node capacity constraint).} \end{aligned}$$

The Lagrange dual associated with the above problem is simple to write down, with one Lagrange multiplier  $\lambda_{id}$  at each node  $i$  for each



destination  $d$  (only the non-zero  $\lambda_{id}$  are shown below):

$$\begin{aligned} \max_{x, R \geq 0} \sum_{i=1}^2 U_i(x_i) - \lambda_{13}(x_0 - R_{12}^{(3)}) - \lambda_{12}(x_1 - R_{12}^{(2)}) \\ - \lambda_{23}(R_{12}^{(3)} - R_{23}^{(3)}) - \lambda_{32}(x_2 - R_{32}^{(2)}) \end{aligned} \quad (2.10)$$

subject to

$$R_{12}^{(3)} + R_{12}^{(2)} + R_{23}^{(3)} + R_{32}^{(2)} \leq 1.$$

Now, we see again that the maximization over  $x$  and  $R$  can be split up into two separate maximization problems:

$$\max_{x \geq 0} \sum_{i=0}^2 U_i(x_i) - \lambda_{13}x_0 - \lambda_{12}x_1 - \lambda_{32}x_2 \quad (2.11)$$

$$\begin{aligned} \max_{\sum_{d,i,j} R_{ij}^d \leq 1} R_{12}^3 (\lambda_{13} - \lambda_{23}) + R_{12}^2 (\lambda_{12} - 0) \\ + R_{23}^3 (\lambda_{23} - 0) + R_{32}^2 (\lambda_{32} - 0) \end{aligned} \quad (2.12)$$

Note that the scheduling problem (2.12) now not only decides the fraction of time that each hop is scheduled, but also determines the fraction of time that a hop is dedicated to flows destined for a particular destination. The scheduling algorithm is still a max-weight algorithm, but the weight for each hop and each flow is now the difference in the Lagrange multiplier at the transmitter and the Lagrange multiplier at the receiver on the hop. This difference is called the *back-pressure* since a link is not scheduled if the Lagrange multiplier at the receiver is larger than the Lagrange multiplier at the transmitter of the hop. Thus, the scheduling algorithm (2.12) is also called the **back-pressure algorithm**.

## 2.4 Fairness

In our discussion of the network utilization maximization, we have associated a utility function with each user. The utility function can be viewed as a measure of satisfaction of the user when it gets a certain data rate from the network. A different point of view is that a utility function is assigned to each user in the network by a service provider with the goal of achieving a certain type of resource allocation. For

example, suppose  $U(x_r) = \log x_r$ , for all users  $r$ . Then, from a well-known property of concave functions, the optimal rates which solve the network utility maximization problem,  $\{\hat{x}_r\}$ , satisfy

$$\sum_r \frac{x_r - \hat{x}_r}{\hat{x}_r} \leq 0,$$

where  $\{x_r\}$  is any other set of feasible rates. The left-hand side of the above expression is nothing but the inner product of the gradient vector at the optimal allocation and the vector of deviations from the optimal allocation. This inner product is non-positive for concave functions. For log utility functions, this property states that, under any other allocation, the sum of proportional changes in the users' utilities will be non-positive. Thus, if some User A's rate increases, then there will be at least one other user whose rate will decrease and further, the proportion by which it decreases will be larger than the proportion by which the rate increases for User A. Therefore, such an allocation is called *proportionally fair*. If the utilities are chosen such that  $U_r(x_r) = w_r \log x_r$ , where  $w_r$  is some weight, then the resulting allocation is said to be *weighted proportionally fair*.

Another widely used fairness criterion in communication networks is called *max-min* fairness. An allocation  $\{\hat{x}_r\}$  is called max-min fair if it satisfies the following property: if there is any other allocation  $\{x_r\}$  such a user  $s$ 's rate increases, i.e.,  $x_s > \hat{x}_s$ , then there has to be another user  $u$  with the property

$$x_u < \hat{x}_u \quad \text{and} \quad \hat{x}_u \leq \hat{x}_s.$$

In other words, if we attempt to increase the rate for one user, then the rate for a less-fortunate user will suffer. The definition of max-min fairness implies that

$$\min_r \hat{x}_r \geq \min_r x_r,$$

for any other allocation  $\{x_r\}$ . To see why this is true, suppose that exists an allocation such that

$$\min_r \hat{x}_r < \min_r x_r. \quad (2.13)$$

This implies that, for any  $s$  such that  $\min_r \hat{x}_r = \hat{x}_s$ , the following holds:  $\hat{x}_s < x_s$ . Otherwise, our assumption (2.13) cannot hold. However, this

implies that if we switch the allocation from  $\{\hat{x}_r\}$  to  $\{x_r\}$ , then we have increased the allocation for  $s$  without affecting a less-fortunate user (since there is no less-fortunate user than  $s$  under  $\{\hat{x}_r\}$ ). Thus, the max-min fair resource allocation attempts to first satisfy the needs of the user who gets the least amount of resources from the network. In fact, this property continues to hold if we remove all the users whose rates are the smallest under max-min fair allocation, reduce the link capacities by the amounts used by these users and consider the resource allocation for the rest of the users. The same argument as above applies. Thus, max-min is a very egalitarian notion of fairness.

Yet another form of fairness that has been discussed in the literature is called *minimum potential delay* fairness. Under this form of fairness, user  $r$  is associated with the utility function  $-1/x_r$ . The goal of maximizing the sum of the user utilities is equivalent to minimizing  $\sum_r 1/x_r$ . The term  $1/x_r$  can be interpreted as follows: suppose user  $r$  needs to transfer a file of unit size. Then,  $1/x_r$  is the delay in associated with completing this file transfer since the delay is simply the file size divided by the rate allocated to user  $r$ . Hence, the name *minimum potential delay* fairness.

All of the above notions of fairness can be captured by using utility functions of the form

$$U_r(x_r) = \frac{x_r^{1-\alpha}}{1-\alpha}, \quad (2.14)$$

for some  $\alpha_r > 0$ . Resource allocation using the above utility function is called  $\alpha$ -fair. Different values of  $\alpha$  yield different ideas of fairness. First consider  $\alpha = 2$ . This immediately yields minimum potential delay fairness. Next, consider the case  $\alpha = 1$ . Clearly, the utility function is not well-defined at this point. But it is instructive to consider the limit  $\alpha \rightarrow 1$ . Notice that maximizing the sum of  $\frac{x_r^{1-\alpha}}{1-\alpha}$  yields the same optimum as maximizing the sum of

$$\frac{x_r^{1-\alpha} - 1}{1-\alpha}.$$

Now, by applying L'Hospital's rule, we get

$$\lim_{\alpha \rightarrow 1} \frac{x_r^{1-\alpha} - 1}{1-\alpha} = \log x_r,$$

thus yielding proportional fairness in the limit as  $\alpha \rightarrow 1$ . Next, we argue that the limit  $\alpha \rightarrow \infty$  gives max-min fairness. Let  $\hat{x}_r(\alpha)$  be the  $\alpha$ -fair allocation. Then, by the property of concave functions mentioned at the beginning of this section,

$$\sum_r \frac{x_r - \hat{x}_r}{\hat{x}_r^\alpha} \leq 0.$$

Considering an arbitrary flow  $s$ , the above expression can be rewritten as

$$\sum_{r: \hat{x}_r \leq \hat{x}_s} (x_r - \hat{x}_r) \frac{\hat{x}_s^\alpha}{\hat{x}_r^\alpha} + (x_s - \hat{x}_s) + \sum_{i: \hat{x}_i > \hat{x}_s} (x_i - \hat{x}_i) \frac{\hat{x}_s^\alpha}{\hat{x}_i^\alpha} \leq 0.$$

If  $\alpha$  is very large, one would expect the third in the above expression to be negligible. Thus, if  $x_s > \hat{x}_s$ , then the allocation for at least one user whose rate satisfies  $\hat{x}_r \leq \hat{x}_s$  will decrease.

## 2.5 Notes

In this chapter we studied the utility maximization framework applied to network resource allocation. When network protocols were being developed, the relationship between congestion control and fair resource allocation was not fully understood. The idea that the congestion control is not just a means to prevent buffer overflows, but has the deeper interpretation of allocating link bandwidths in a way that system utility is maximized was presented in papers by Kelly et. al. [3, 7–9]. The utility maximization framework for wireless networks has been developed by Neely et. al., Lin and Shroff, Stolyar, and Eryilmaz and Srikant [10–14]. The reader is referred to [15, 16] for more detailed surveys of the optimization approach to wireless network resource allocation.

Many notions of fairness in communication networks have been proposed over the years. In particular, max-min fairness has been studied extensively in [17, 18]. Max-min fairness was originally developed in the context of political philosophy by Rawls [19]. Proportional fairness was introduced for communication networks in [3]. Nash studied this concept as a bargaining solution for players bargaining over the allocation of a common resource [20]. Hence, it is called the Nash bargaining solution in economics. Minimum potential delay fairness was introduced

in [21]. The concept of  $\alpha$ -fairness as a common framework to study many notions of fairness was proposed by Mo and Walrand in [22, 23].

In the following chapter, we will present decentralized algorithms that converge to the solution of the network utility maximization problem.

# 3

---

## Utility Maximization Algorithms

---

In our solution to the network utility maximization problem for a simple network in the previous chapter, we assumed the existence of a central authority that has complete information about all the routes and link capacities of the network. This information was used to fairly allocate rates to the different sources. Clearly, such a centralized solution does not scale well when the number of sources or the number of nodes in the network becomes large. We would like to design algorithms in which the sources themselves perform calculations to determine their fair rate based on some feedback from the network. We would also like the algorithms to be such that the sources do not need very much information about the network. How would one go about designing such algorithms?

In this chapter we will study three such algorithms called the Primal, Dual and Primal-Dual algorithms, which derive their names from the formulation of the optimization problem that they are associated with. We will show how all three approaches would result in a stable and fair allocation with simple feedback from the network. We assume in this chapter that there are no feedback delays in the system; we will study the effect of delays in Chapter 5. We start with the Primal



controller, which will also serve to illustrate some basic optimization concepts.

### 3.1 Primal Formulation

We relax the optimization problem in several ways so as to make algorithm design tractable. The first relaxation is that instead of directly maximizing the sum of utilities constrained by link capacities, we associate a cost with overshooting the link capacity and maximize the sum of utilities minus the cost. In other words, we now try to maximize

$$V(x) = \sum_{r \in \mathcal{S}} U_r(x_r) - \sum_{l \in \mathcal{L}} B_l \left( \sum_{s: l \in s} x_s \right), \quad (3.1)$$

where  $x$  is the vector of rates of all sources and  $B_l(\cdot)$  is either a “barrier” associated with link  $l$  which increases to infinity when the arrival rate on a link  $l$  approaches the link capacity  $c_l$  or a “penalty” function which penalizes the arrival rate for exceeding the link capacity. By appropriate choice of the function  $B_l$ , one can solve the exact utility optimization problem posed in the previous chapter; for example, choose  $B_l(y)$  to be zero if  $y \leq c_l$  and equal to  $\infty$  if  $y \geq c_l$ . However, such a solution may not be desirable or required. For example, the design principle may be such that one requires the delays on all links to be small. In this case, one may wish to heavily penalize the system if the arrival rate is larger than say 90% of the link capacity. Losing a small fraction of the link capacity may be considered to be quite reasonable if it improves the quality of service seen by the users of the network since increasing the link capacity is quite inexpensive in today’s Internet.

Another relaxation that we make is that we don’t require that the solution of the utility maximization problem be achieved instantaneously. We will allow the design of dynamic algorithms that asymptotically (in time) approach the required maximum.

Now let us try to design an algorithm that would satisfy the above requirements. Let us first consider the penalty function  $B_l(\cdot)$  that appears in 3.1 above. It is reasonable to require that  $B_l$  is a convex function since we would like the penalty function to increase rapidly as we approach or exceed the capacity. Further, assume that  $B_l$  is continu-

ously differentiable. Then, we can equivalently require that

$$B_l \left( \sum_{s:l \in s} x_s \right) = \int_0^{\sum_{s:l \in s} x_s} f_l(y) dy, \quad (3.2)$$

where  $f_l(\cdot)$  is an increasing, continuous function. We call  $f_l(y)$  the *congestion price function*, or simply the price function, associated with link  $l$ , since it associates a price with the level of congestion on the link. It is straightforward to see that  $B_l$  defined in the above fashion is convex, since integrating an increasing function results in a convex function. If  $f_l$  is differentiable, it is easy to check the convexity of  $B_l$  since the second derivative of  $B_l$  would be positive since  $f_l$  is an increasing function.

Since  $U_r$  is strictly concave and  $B_l$  is convex,  $V(x)$  is strictly concave. Further, we assume that  $U_r$  and  $f_l$  are such that the maximization of (3.1) results in a solution with  $x_r \geq 0 \forall r \in \mathcal{S}$ . Now, the condition that must be satisfied by the maximizer of (3.1) is obtained by differentiation and is given by

$$U'_r(x_r) - \sum_{l:l \in r} f_l \left( \sum_{s:l \in s} x_s \right) = 0, \quad r \in \mathcal{S}. \quad (3.3)$$

We require an algorithm that would drive  $x$  towards the solution of (3.3). A natural candidate for such an algorithm is the gradient ascent algorithm from optimization theory.

The idea here is that if we want to maximize a function of the form  $g(x)$ , then we progressively change  $x$  so that  $g(x(t + \delta)) > g(x(t))$ . We do this by finding the direction in which a change in  $x$  produces the greatest increase in  $g(x)$ . This direction is given by the gradient of  $g(x)$  with regard to  $x$ . In one dimension, we merely choose the update algorithm for  $x$  as

$$x(t + \delta) = x(t) + k(t) \frac{dg(x)}{dx} \delta,$$

where  $k(t)$  is a scaling parameter which controls the amount of change in the direction of the gradient. Letting  $\delta \rightarrow 0$

$$\dot{x} = k(t) \frac{dg(x)}{dx}. \quad (3.4)$$

The idea is illustrated in Figure 3.1 at some selected time instants. The value of  $x$  changes smoothly, with  $g(x)$  increasing at each time

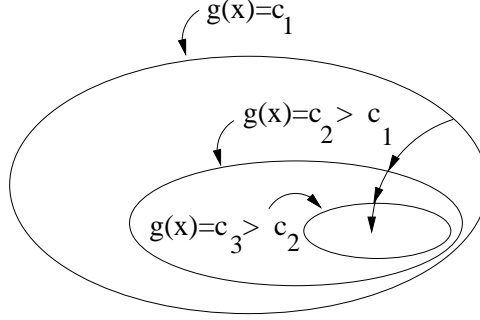


Fig. 3.1 Gradient ascent to reach the maximum value of a function  $g(x)$ . The gradient ascent algorithm progressively climbs up the gradient so as to reach the maximum value.

instant, until it hits the maximum value.

Let us try to design a similar algorithm for the network utility maximization problem. Consider the algorithm

$$\dot{x}_r = k_r(x_r) \left( U'_r(x_r) - \sum_{l:l \in r} f_l \left( \sum_{s:l \in s} x_s \right) \right), \quad (3.5)$$

where  $k_r(\cdot)$  is non-negative, increasing and continuous. We have obtained the above by differentiating (3.1) with respect to  $x_r$  to find the direction of ascent, and used it along with a scaling function  $k_r(\cdot)$  to construct an algorithm of the form shown in (3.4). Clearly, the stationary point of the above algorithm satisfies (3.3) and hence maximizes (3.1). The controller is called a *primal algorithm* since it arises from the primal formulation of the utility maximization problem. Note that the primal algorithm has many intuitive properties that one would expect from a resource allocation/congestion control algorithm. When the route price  $q_r = \sum_{l:l \in r} f_l(\sum_{s:l \in s} x_s)$  is large, then the congestion controller decreases its transmission rate. Further, if  $x_r$  is large, then  $U'(x_r)$  is small (since  $U_r(x_r)$  is concave) and thus the rate of increase is small as one would expect from a resource allocation algorithm which attempts to maximize the sum of the user utilities.

We must now answer two questions regarding the performance of the primal congestion control algorithm:

- What information is required at each source in order to implement the algorithm?
- Does the algorithm actually converge to the desired stationary point?

Below we consider the answer to the first question and develop a framework for answering the second. The precise answer to the convergence question will be presented in the next sub-section.

The question of information required is easily answered by studying (3.5). It is clear that all that the source  $r$  needs to know in order to reach the optimal solution is the sum of the prices of each link on its route. How would the source be appraised of the link prices? The answer is to use a feedback mechanism—each packet generated by the source collects the price of each link that it traverses. When the destination receives the packet, it sends this price information in a small packet (called the acknowledgment packet or *ack* packet) that it sends back to the source.

To visualize this feedback control system, we introduce a matrix  $R$  which is called the routing matrix of the network. The  $(l, r)$  element of this matrix is given by

$$R_{lr} = \begin{cases} 1 & \text{if route } r \text{ uses link } l \\ 0 & \text{else} \end{cases}$$

Let us define

$$y_l = \sum_{s: l \in s} x_s, \tag{3.6}$$

which is the load on link  $l$ . Using the elements of the routing matrix and recalling the notation of Section 2.1,  $y_l$  can also be written as

$$y_l = \sum_{s: l \in s} R_{ls} x_s.$$

Letting  $y$  be the vector of all  $y_l$  ( $l \in \mathcal{L}$ ), we have

$$y = Rx \tag{3.7}$$

Let  $p_l(t)$  denote the price of link  $l$  at time  $t$ , i.e.,

$$\begin{aligned} p_l(t) &= f_l \left( \sum_{s:l \in s} x_s \right) \\ &= f_l(y_l(t)). \end{aligned} \quad (3.8)$$

Then the price of a route is just the sum of link prices  $p_l$  of all the links in the route. So we define the price of route  $r$  to be

$$q_r = \sum_{l:l \in r} p_l(t). \quad (3.9)$$

Also let  $p$  be the vector of all link prices and  $q$  be the vector of all route prices. We thus have

$$q = R^T p \quad (3.10)$$

The above is more than just an intellectual exercise, since the expressions (3.7) and (3.10) provide linear relationships between the control at the sources and the control at the links that will help us later in analyzing the system further. The relationships derived above can be made clear using the block diagram in Figure 3.2.

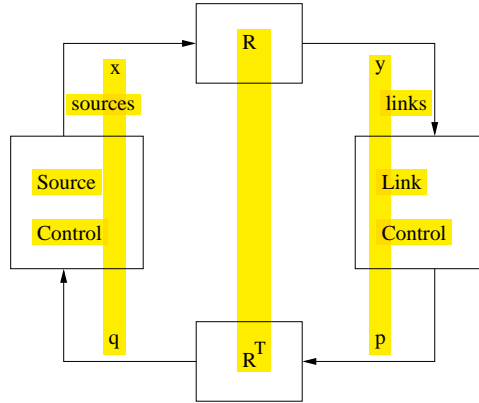


Fig. 3.2 A block diagram view of the congestion control algorithm. The controller at the source uses congestion feedback from the link to perform its action.

To answer the question of whether or not our controller actually achieves the desired allocation, we have to study the properties of the

controller dynamics. For this purpose, we next introduce the concept of a Lyapunov function, which is widely used in control theory.

### 3.1.1 Stability Notions for Dynamical Systems

We first give an intuitive interpretation of a Lyapunov function. Consider a surface such as that shown in Figure 3.3. Suppose we took a marble, placed it at any point on the concave side of the surface and let go. It would roll down due to gravity, would oscillate a few times, gradually lose energy due to friction, and finally settle in the state of lowest energy in the trough of the surface. Notice that in this example, the marble would rest in the trough regardless of the point from which it was released. The idea behind Lyapunov functions is similar.

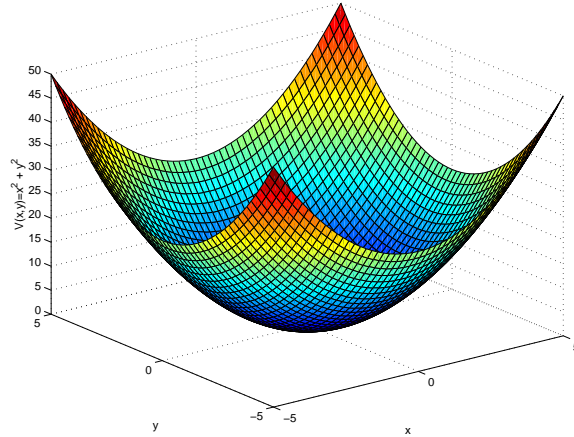


Fig. 3.3 Illustrating the nature of a Lyapunov function

A Lyapunov function can be thought of as a scalar characterization of a system similar to the energy of a system. If we can show that the energy is continuously decreasing, the system would finally settle down into the lowest-energy state. If we are able to find such a Lyapunov function for a control system with the lowest state being the equilibrium point, we would then be able to show that the system eventually settles at the equilibrium point. In this case, we say that the system is *asymptotically stable*. In some instances, the equilibrium point might be reached

only if the initial condition is in a particular set; we call such a set the *basin of attraction*. If the basin of attraction is the entire state space, then we say that the system is *globally asymptotically stable*. We now formalize these intuitive ideas. Although we state the theorems under the assumption that the equilibrium point is at zero, the results apply even when the equilibrium is non-zero by simply shifting the coordinate axes.

Consider a dynamical system represented by

$$\dot{x} = g(x), \quad x(0) = x_0, \quad (3.11)$$

where it is assumed that  $g(x) = 0$  has a unique solution. Call this solution 0. Here  $x$  and 0 can be vectors.

---

**Definition 3.1.** The equilibrium point 0 of (3.11) is said to be

- stable if, for each  $\varepsilon > 0$ , there is a  $\delta = \delta(\varepsilon) > 0$ , such that

$$\|x(t)\| \leq \varepsilon, \quad \forall t \geq 0, \quad \text{if } \|x_0\| \leq \delta.$$

- asymptotically stable if there exists a  $\delta > 0$  such that

$$\lim_{t \rightarrow \infty} \|x(t)\| = 0$$

for all  $\|x_0\| \leq \delta$ .

- globally, asymptotically stable if

$$\lim_{t \rightarrow \infty} \|x(t)\| = 0$$

for all initial conditions  $x_0$ .

---

We now state Lyapunov's theorem which uses the Lyapunov function to test for the stability of a dynamical system [24].

---

**Theorem 3.1.** Let  $x = 0$  be an equilibrium point for  $\dot{x} = f(x)$  and  $D \subset \mathbb{R}^n$  be a domain containing 0. Let  $V : D \rightarrow \mathbb{R}$  be a continuously differentiable function such that

$$V(x) > 0, \quad \forall x \neq 0$$

and  $V(0) = 0$ . Now we have the following conditions for the various notions of stability.

- (1) If  $\dot{V}(x) \leq 0 \forall x$ , then the equilibrium point is stable.
- (2) In addition, if  $\dot{V}(x) < 0, \forall x \neq 0$ , then the equilibrium point is asymptotically stable.
- (3) In addition to (1) and (2) above, if  $V$  is radially unbounded, i.e.,

$$V(x) \rightarrow \infty, \quad \text{when } \|x\| \rightarrow \infty,$$

then the equilibrium point is globally asymptotically stable.

Note that the above theorem also holds if the equilibrium point is some  $\hat{x} \neq 0$ . In this case, consider a system with state vector  $y = x - \hat{x}$  and the results immediately apply.

### 3.1.2 Global Asymptotic Stability of the Primal Controller

We show that the primal controller of (3.5) is globally asymptotically stable by using the Lyapunov function idea described above. Recall that  $V(x)$  as defined in 3.1 is a strictly concave function. Let  $\hat{x}$  be its unique maximum. Then,  $V(\hat{x}) - V(x)$  is non-negative and is equal to zero only at  $x = \hat{x}$ . Thus,  $V(\hat{x}) - V(x)$  is a natural candidate Lyapunov function for the system (3.5). We use this Lyapunov function in the following theorem.

**Theorem 3.2.** Consider a network in which all sources follow the primal control algorithm (3.5). Assume that the functions  $U_r(\cdot)$ ,  $k_r(\cdot)$  and  $f_l(\cdot)$  are such that  $W(x) = V(\hat{x}) - V(x)$  is such that  $W(x) \rightarrow \infty$  as  $\|x\| \rightarrow \infty$ ,  $\hat{x}_i > 0$  for all  $i$ , and  $V(x)$  is as defined in (3.1). Then, the controller in (3.5) is globally asymptotically stable and the equilibrium value maximizes (3.1).

*Proof.* Differentiating  $W(\cdot)$ , we get

$$\dot{W} = - \sum_{r \in \mathcal{S}} \frac{\partial V}{\partial x_r} \dot{x}_r = - \sum_{r \in \mathcal{S}} k_r(x_r) (U'_r(x_r) - q_r)^2 < 0, \quad \forall x \neq \hat{x}, \quad (3.12)$$

and  $\dot{W} = 0 \forall x = \hat{x}$ . Thus, all the conditions of the Lyapunov theorem are satisfied and we have proved that the system state converges to  $\hat{x}$ .  $\square$



In the proof of the above theorem, we have assumed that utility, price and scaling functions are such that  $W(x)$  has some desired properties. It is very easy to find functions that satisfy these properties. For example, if  $U_r(x_r) = w_r \log(x_r)$ , and  $k_r(x_r) = x_r$ , then the primal congestion control algorithm for source  $r$  becomes

$$\dot{x}_r = w_r - x_r \sum_{l:l \in r} f_l(y_l),$$

and thus the unique equilibrium point is  $w_r/x_r = \sum_{l:l \in r} f_l(y_l)$ . If  $f_l(\cdot)$  is any polynomial function, then  $V(x)$  goes to  $-\infty$  as  $\|x\| \rightarrow \infty$  and thus,  $W(x) \rightarrow \infty$  as  $\|x\| \rightarrow \infty$ .

### 3.1.3 Price Functions and Congestion Feedback

We had earlier argued that collecting the price information from the network is simple. If there is a field in the packet header to store price information, then each link on the route of a packet simply adds its price to this field, which is then echoed back to source by the receiver in the acknowledgment packet. However, packet headers in the Internet are already crowded with a lot of other information, so Internet practitioners do not like to add many bits in the packet header to collect congestion information. Let us consider the extreme case where there is only one bit available in the packet header to collect congestion information. How could we use this bit to collect the price of route? Suppose that each packet is *marked* with probability  $1 - e^{-p_l}$  when the packet passes through link  $l$ . Marking simply means that a bit in the packet header is flipped from a 0 to a 1 to indicate congestion. Then, along a route  $r$ , a packet is marked with probability

$$1 - e^{-\sum_{l:l \in r} p_l}.$$

If the acknowledgment for each packet contains one bit of information to indicate if a packet is marked or not, then by computing the fraction of marked packets, the source can compute the route price  $\sum_{l:l \in r} p_l$ . The assumption here is that the congestion control occurs at a slower time-scale than the packet dynamics in the network so that  $p_l$  remains roughly a constant over many packets.

Price functions were assumed to be functions of the arrival rate at a link. However, in practice, it is quite common to indicate congestion based on the queue length at a link. For example, consider the following congestion notification mechanism. Whenever the queue length at a node exceeds a threshold  $B$  at the time of a packet arrival, then the packet is marked. For the moment assume that there is only one link in the network. Further, again suppose that the congestion controller acts at a slower time compared to the queue length dynamics. For example, the congestion controller may react to the fraction of packets marked over a time interval. In this case, one can roughly view the price function for the link as the fraction of marked packets. To compute the fraction of marked packets, one needs a model for the packet arrival process at the link which could, in general, be a stochastic process. For simplicity, assume that the arrival process is a Poisson process of rate  $y_l$  which remains constant compared to the time-scale of the congestion control process. Further, suppose that packet sizes are exponentially distributed. Then, the fraction of marked packets is simply the probability that the number of packets in an  $M/M/1/$  queue exceeds  $B$ , which is given by  $p_l = f_l(y_l) = (y_l/c_l)^B$  if  $y_l < c_l$  and is equal to 1 otherwise [18]. If there is only one bit in the packet header to indicate congestion, then a packet is marked if it is marked on any one of the links on its route. Thus, the packet marking probability on a route  $r$  is given by  $1 - \prod_{l \in r} (1 - p_l)$ . If the  $p_l$  are small, then one can approximate the above by  $\sum_{l \in r} p_l$ , which is consistent with the primal formulation of the problem.

Instead of simply marking packets when the queue length exceeds a threshold, other price functions have been suggested in the literature as well. A well-known such mechanism is RED (random early detection) scheme. Let  $b_l$  be the number of packets in the buffer at link  $l$ . Then, the RED packet marking scheme is given by

$$g(b_l) = \begin{cases} 0, & \text{if } b_l \leq B_{min} \\ K(b_l - B_{min}), & \text{if } B_{min} < b_l \leq B_{max} \\ 1, & \text{if } b_l > B_{max} \end{cases}, \quad (3.13)$$

where  $K$  is constant, and  $B_{min}$  and  $B_{max}$  are fixed thresholds. This price function also lies strictly in  $[0, 1]$  and is illustrated in Figure 3.4.

Again, if the packet-level dynamics in the queue are faster than the

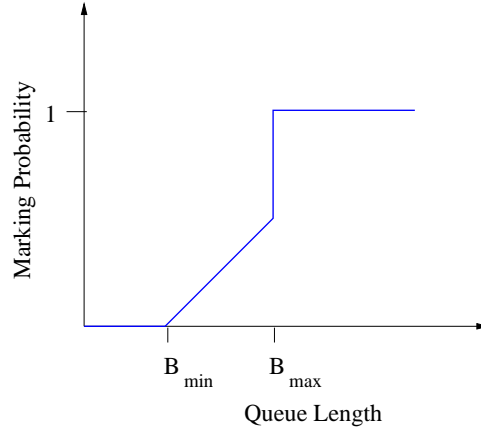


Fig. 3.4 The RED marking scheme, where the probability of marking a packet is a function of the queue size.

congestion-control dynamics, then one can convert the above marking scheme into a price function that depends on the link rate. Under the  $M/M/1$  assumption as before, the probability that there are  $n$  packets in the buffer is given by

$$r_n = \rho_l (1 - \rho_l)^n,$$

where  $\rho_l = y_l/c_l$ . Then, the equivalent price function for the RED scheme is given by

$$f_l(y_l) = \sum_{n=0}^{\infty} r_n g(n)$$

if  $\rho_l < 1$  and is equal to 1 if  $\rho_l \geq 1$ . In general, given any queue-based marking function  $g_l(b_l)$ , one can obtain an equivalent rate-based price function by computing  $E(g_l(b_l))$  where  $b_l$  is the steady-state queue length of a queueing model with arrival rate  $y_l$  and service rate  $c_l$ . In the above calculations, we have assumed that the queueing model is an  $M/M/1$  model, but one can use other, more complicated models as well. A queue-based marking scheme is also called an **active queue management** or AQM scheme.

Another price function of interest is found by considering packet dropping instead of packet marking. If packets are dropped due to the fact that a link buffer is full when a packet arrives at the link, then such a dropping mechanism is called a *Droptail* scheme. Assuming Poisson arrivals and exponentially distributed file sizes, the probability that a packet is dropped when the buffer size is  $B$  is given by

$$\frac{1 - \rho}{1 - \rho^{B+1}} \rho^B,$$

where  $\rho = \frac{\sum_{r:l \in r} x_r}{c_l}$ . As the number of users of the Internet increases, one might conceivably increase the buffer sizes as well in proportion, thus maintaining a constant maximum delay at each link. In this case,  $B \rightarrow \infty$  would be a reasonable approximation. We then have

$$\lim_{B \rightarrow \infty} \frac{1 - \rho}{1 - \rho^{B+1}} \rho^B = \begin{cases} 0, & \text{if } \rho < 1, \\ 1 - \frac{1}{\rho}, & \text{if } \rho \geq 1. \end{cases}$$

Thus, an approximation for the drop probability is  $\left(1 - \frac{1}{\rho}\right)^+$ , which is non-zero only if  $\sum_{r:l \in r} x_r$  is larger than  $c_l$ . When packets are dropped at a link for source  $r$ , then the arrival rate from source  $r$  at the next link on the link would be smaller due to the fact that dropped packets cannot arrive at the next link. Thus, the arrival rate is “thinned” as we traverse the route. However, this is very difficult to model in our optimization framework. Thus, the optimization is valid only if the drop probability is small. Further, the end-to-end drop probability on a route can be approximated by the sum of the drop probabilities on the links along the route if the drop probability at each link is small.

We now move on to another class of **resource allocation algorithms called Dual Algorithms**. All the tools we have developed for studying primal algorithms will be used in the dual formulation as well.

### 3.2 Dual Formulation

In the previous section, we studied how to design a stable and simple control mechanism to asymptotically solve the relaxed utility maximization problem. We also mentioned that by using an appropriate

barrier function, one could obtain the exact value of the optimal solution. In this section we consider a different kind of controller based on the dual formulation of the utility maximization problem that naturally produces the optimal solution without any relaxation. Consider the resource allocation problem that we would like to solve

$$\max_{x_r} \sum_{r \in \mathcal{S}} U_r(x_r) \quad (3.14)$$

subject to the constraints

$$\sum_{r: l \in r} x_r \leq c_l, \quad \forall l \in \mathcal{L}, \quad (3.15)$$

$$x_r \geq 0, \quad \forall r \in \mathcal{S}. \quad (3.16)$$

The Lagrange dual of the above problem is obtained by incorporating the constraints into the maximization by means of Lagrange multipliers as follows:

$$D(p) = \max_{\{x_r > 0\}} \sum_r U_r(x_r) - \sum_l p_l \left( \sum_{s: l \in s} x_s - c_l \right) \quad (3.17)$$

Here the  $p_l$ s are the Lagrange multipliers that we saw in the previous chapter. The dual problem may then be stated as

$$\min_{p \geq 0} D(p).$$

As in the case of the primal problem, we would like to design an algorithm that causes all the rates to converge to the optimal solution. Notice that in this case we are looking for a gradient descent (rather than a gradient ascent that we saw in the primal formulation), since we would like to minimize  $D(p)$ . To find the direction of the gradient, we need to know  $\frac{\partial D}{\partial p_l}$ .

We first observe that in order to achieve the maximum in (3.17),  $x_r$  must satisfy

$$U'_r(x_r) = q_r, \quad (3.18)$$

or equivalently,

$$x_r = U'^{-1}_r(q_r), \quad (3.19)$$

where, as usual,  $q_r = \sum_{l:l \in r} p_l$ , is the price of a particular route  $r$ . Now, since

$$\frac{\partial D}{\partial p_l} = \sum_{r:l \in r} \frac{\partial D}{\partial q_r} \frac{\partial q_r}{\partial p_l},$$

we have from (3.19) and (3.17) that

$$\frac{\partial D}{\partial p_l} = \sum_{r:l \in r} \frac{\partial U_r(x_r)}{\partial p_l} - (y_l - c_l) - \sum_i p_i \frac{\partial y_i}{\partial p_l}, \quad (3.20)$$

where the  $x_r$  above is the optimizing  $x_r$  in (3.17). In order to evaluate the above, we first compute  $\partial x_r / \partial p_l$ . Differentiating (3.18) with respect to  $p_l$  yields

$$\begin{aligned} U_r''(x_r) \frac{dx_r}{dp_l} &= 1 \\ \Rightarrow \frac{\partial x_r}{\partial p_l} &= \frac{1}{U_r''(x_r)} \end{aligned}$$

Substituting the above in (3.20) yields

$$\frac{\partial D}{\partial p_l} = \sum_{r:l \in r} \frac{U_r'(x_r)}{U_r''(x_r)} - (y_l - c_l) - \sum_i p_i \sum_{r:l \in r} \frac{1}{U_r''(x_r)} \quad (3.21)$$

$$= c_l - y_l, \quad (3.22)$$

where we have interchanged the last two summations in (3.21) and used the facts  $U_r'(x_r) = q_r$  and  $q_r = \sum_{l \in r} p_l$ . The above is the gradient of the Lagrange dual, i.e., the direction in which it increases. We are now ready to design our controller. Recalling that we need to *descend* down the gradient, from (3.19) and (3.22), we have the following dual control algorithm:

$$x_r = U_r'^{-1}(q_r) \quad \text{and} \quad (3.23)$$

$$\dot{p}_l = h_l(y_l - c_l)_{p_l}^+, \quad (3.24)$$

where,  $h_l > 0$  is a constant and  $(g(x))_y^+$  denotes

$$(g(x))_y^+ = \begin{cases} g(x), & y > 0, \\ \max(g(x), 0), & y = 0, \end{cases}$$

We use this modification to ensure that  $p_l$  never goes negative since we know from the KKT conditions that the optimal price is non-negative.

Note that, if  $h_l = 1$ , the price update above has the same dynamics as the dynamics of the queue at link  $l$ . The price increases when the arrival rate is larger than the capacity and decreases when the arrival rate is less than the capacity. Moreover, the price can never become negative. These are exactly the same dynamics that govern the queue size at link  $l$ . Thus, one doesn't even have to explicitly keep track of the price in the dual formulation; the queue length naturally provides this information. However, there is a caveat. We have assumed that the arrivals from a source arrive to all queues on the path simultaneously. In reality the packets must go through the queue at one link before arriving at the queue at the next link. There are two ways to deal with this issue:

- Assume that the capacity  $c_l$  is not the true capacity of a link, but is a fictitious capacity which is slightly less than the true capacity. Then,  $q_l$  will be the length of this virtual queue with the virtual capacity  $c_l$ . A separate counter must then be maintained to keep track of the virtual queue size, which is the price of the link. Note that this price must be fed back to the sources. Since the congestion control algorithm reacts to the virtual queue size, the arrival rate at the link will be at most  $c_l$  and thus will be less than the true capacity. As a result, the real queue size at the link would be negligible. In this case, one can reasonably assume that packets move from link to link without experiencing significant queueing delays.
- One can modify the queueing dynamics to take into account the route traversal of the packets. We will do so in a later section on the primal-dual algorithm. This modification may be more important in wireless networks where interference among various links necessitates scheduling of links and thus, a link may not be scheduled for a certain amount of time. Therefore, packets will have to wait in a queue till they are scheduled and thus, we will model the queueing phenomena precisely.

### 3.2.1 Stability of the Dual Algorithm

Since we designed the dual controller in much the same way as the primal controller and they both are gradient algorithms (ascent in one case and descent in the other), we would expect that the dual algorithm too will converge to the optimal solution. We show using Lyapunov theory that this is indeed the case.

We first understand the properties of the solution of the original network utility maximization problem (2.1). Due to the same concavity arguments used earlier, we know that the maximizer of (2.1) which we denote by  $\hat{x}$  is unique. Suppose also that in the dual formulation (3.17) given  $q$ , there exists a unique  $p$  such that  $q = R^T p$  (i.e.,  $R$  has full row rank), where  $R$  is the routing matrix. At the optimal solution

$$\hat{q} = R^T \hat{p},$$

and the KKT conditions imply that, at each link  $l$ , either

$$\hat{y}_l = c_l$$

if the constraint is active or

$$\hat{y}_l < c_l \quad \text{and} \quad \hat{p}_l = 0$$

if the link is not a fully utilized. Note that under the full row rank assumption on  $R$ ,  $\hat{p}$  is also unique.

---

**Theorem 3.3.** Under the assumption that given  $q$ , there exists a unique  $p$  such that  $q = R^T p$ , the dual algorithm is globally asymptotically stable.

---

*Proof.* Consider the Lyapunov function

$$V(p) = \sum_{l \in \mathcal{L}} (c_l - \hat{y}_l) p_l + \sum_{r \in \mathcal{S}} \int_{\hat{q}_r}^{q_r} (\hat{x}_r - (U'_r)^{-1}(\sigma)) d\sigma.$$



Then we have

$$\begin{aligned}
\frac{dV}{dt} &= \sum_l (c_l - \hat{y}_l) \dot{p}_l + \sum_r (\hat{x}_r - (U'_r)^{-1}(q_r)) \dot{q}_r \\
&= (c - \hat{y})^T \dot{p} + (\hat{x} - x)^T \dot{q} \\
&= (c - \hat{y})^T \dot{p} + (\hat{x} - x)^T R^T \dot{p} \\
&= (c - \hat{y})^T \dot{p} + (\hat{y} - y)^T \dot{p} \\
&= (c - y)^T \dot{p} \\
&= \sum_l h_l (c_l - y_l) (y_l - c_l)_{p_l}^+ \\
&\leq 0.
\end{aligned}$$

Also,  $\dot{V} = 0$  only when each link satisfies either  $y_l = c_l$  or  $y_l < c_l$  with  $p_l = 0$ . Further, since  $U'_r(x_r) = q_r$  at each time instant, thus all the KKT conditions are satisfied. The system converges to the unique optimal solution of (2.1).  $\square$

### 3.3 Extension to Multi-path Routing

The approaches considered in this chapter can be extended to account for the fact that there could be multiple routes between each source-destination pair. The existing protocols on the Internet today do not allow the use of multiple routes, but it is not hard to envisage a situation in the future where smart routers or overlay routers would allow such routing. Our goal is to find an implementable, decentralized congestion control algorithm for such a network with multi-path routing. Figure 3.5 indicates a simple scenario where this might be possible.

As before, the set of sources is called  $\mathcal{S}$ . Each source  $s \in \mathcal{S}$  identifies a unique source-destination pair. Also, each source may use several routes. If source  $s$  is allowed to use route  $r$ , we say  $r \in s$ . Similarly, if a route  $r$  uses link  $j$ , we say  $j \in r$ . For each route  $r$  we let  $s(r)$  be the unique source such that  $r \in s(r)$ . We call the set of routes  $\mathcal{R}$ . As in single-path routing, note that two paths in the network that are otherwise identical can be associated with two different indices  $r \in \mathcal{R}$ , if their sources are different. For instance, in Figure 3.5 suppose there were two sources  $s_1$  and  $s_3$  between the node pair  $(N_1, N_3)$ . Let  $s_1$  subscribe to the multi-path service. Then the collection of routes

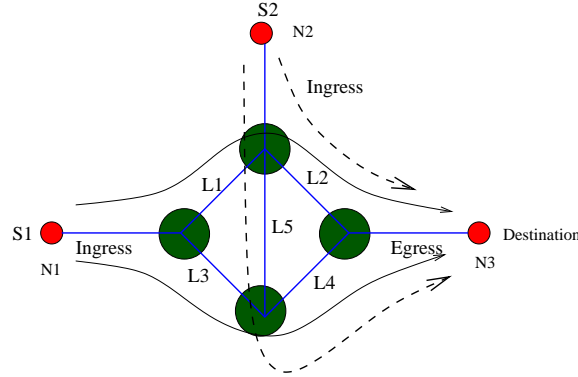


Fig. 3.5 In this example, two sources  $S_1$  and  $S_2$  are transferring data to a single destination using two paths each.

available to  $s_1$  is  $\{r_1, r_2\}$ , where  $r_1$  uses  $L_1$  and  $L_2$ , and  $r_2$  uses  $L_3$  and  $L_4$ . Also  $s(r_1) = s(r_2) = s_1$ . On the other hand let  $s_3$  subscribe to only a single path service. Then  $s_3$  is allowed to use only  $r_3$ , which uses  $L_1$  and  $L_2$  and  $s(r_3) = s_3$ . We emphasize that although  $r_1$  and  $r_3$  use the same links and are between the same node pair, they are indexed differently since they belong to the path-sets of different users.

We consider the primal formulation in this section. The utility function must also be appropriately modified in order to take into account the fact that the throughput of each source is now the *sum* rates obtained along each route. We seek a primal formulation of the problem and hence consider the (relaxed) system utility (using log utility for exposition) given by

$$\begin{aligned} \mathcal{U}(x) = & \sum_s \left( w_s \log \left( \sum_{r \in s} x_r \right) + \varepsilon \sum_{r \in s} \log(x_r) \right) \\ & - \sum_l \int_0^{\sum_{k: l \in k} x_k} f_l(y) dy. \end{aligned} \quad (3.25)$$

Here we denote the transmission rate on route  $r$  by  $x_r$ , and  $w_s$  is a weighting constant for all routes associated with source  $s$ . Also  $f_l(y)$  is the price associated with link  $l$ , when the arrival rate to the link is  $y$ , which may be considered to be the cost associated with congestion

on the link  $y$ . Note that we have added a term containing  $\varepsilon$  to ensure that the net utility function is strictly concave and hence has a unique maximum. The presence of the  $\varepsilon$  term also ensures that a non-zero rate is used on all paths allowing us automatically to probe the price on all paths. Otherwise, an additional probing protocol would be required to probe each high-price path to know when its price drops significantly to allow transmission on the path.

We then consider the following controller, which is a natural generalization of the earlier single-path controller to control the flow on route  $r$ :

$$\begin{aligned} \dot{x}_r(t) = & \kappa_r x_r \left( w_r - \left( \sum_{m \in s(r)} x_m(t) \right) q_r(t) \right) \\ & + \kappa_r \varepsilon \sum_{m \in s(r)} x_m(t), \end{aligned} \quad (3.26)$$

where  $w_r = w_s$  for all routes  $r$  associated with source  $s$ . The term  $q_r(t)$  is the estimate of the route price at time  $t$ . This price is the sum of the individual link prices  $p_l$  on that route. The link price in turn is some function  $f_l$  of the arrival rate at the link. The fact that the controller is globally asymptotically stable can be shown as we did in the primal formulation using  $\mathcal{U}(\hat{x}) - \mathcal{U}(x)$  as the Lyapunov function. The proof is identical to the one presented earlier and hence will not be repeated here.

So far, we have seen two types of controllers for the wired Internet, corresponding to the primal and dual approaches to constrained optimization. We will now study a combination of both controllers called the primal-dual controller in the more general setting of a wireless network. The wireless setting is more general since the case of the Internet is a special case where there are no interference constraints. It is becoming increasingly clear that significant portions of the Internet are evolving towards wireless components; examples include data access through cellular phones, wireless LANs and multi-hop wireless networks which are widely prevalent in military, law enforcement and emergency response networks, and which are expected to become common in the civilian domain at least in certain niche markets such as

viewing replays during football games, concerts, etc. As mentioned in the previous chapter, our goal here is not to provide a comprehensive introduction to resource allocation in wireless networks. Our purpose here is to point out that the simple fluid model approach that we have taken so far extends to wireless networks at least in establishing the stability of congestion control algorithms. On the other hand, detailed scheduling decisions require more detailed models which we do not consider here.

### 3.4 Primal-Dual Algorithm for Wireless Networks

In the discussion thus far, we saw that both controllers use rate-update functions at the source side, and price-update functions on each of the links. The exact form of the updates were different in the two cases. **It is possible to combine the two controllers by using the rate-update from the primal and the price-update from the dual.** Such a controller is called a primal-dual controller. While we could illustrate such a controller in the same wired-network setting as above, we will do so for a wireless network in this section.

Let  $\mathcal{N}$  be the set of nodes and  $\mathcal{L}$  be the set of permissible hops. Let  $\mathcal{F}$  be the set of flows in the system. We do not associate flows with fixed routes, but routes would also be decided by the resource allocation algorithm. In principle, all flows could potentially use all nodes in the system as part of their routes. Of course, the results also apply to the special case where a route is associated with a flow. Each flow  $f$  has a beginning node  $b(f)$  and an ending node  $e(f)$ . We assume that each node maintains a separate queue for each source-destination pair. Let  $r_{ij}$  be the rate at which transmission takes place from node  $i$  to node  $j$ . Due to interference, the rates between the various nodes are inter-related. We consider a simple model to describe this interference although the results can be generalized significantly. Let  $\{A_m\}$   $m = 1, 2, \dots, M$  be a collection of subsets of  $\mathcal{L}$ . The set  $A_m$  is a set of hops that can be scheduled simultaneously given some interference constraints. Each  $A_m$  is called a feasible *schedule* and  $M$  is the number of possible feasible schedules. If schedule  $A_m$  is used, then  $r_{ij} = 1$  if  $(i, j) \in A_m$ , and  $r_{ij} = 0$  otherwise. Thus, all the scheduled links can

transmit at rate 1 while the unscheduled hops remain silent. The network has a choice of implementing any one of the schedules at each time instant. Suppose that  $\pi_m$  is the fraction of time that the network chooses to use schedule  $m$ . Then, the average rate allocated to hop  $(i, j)$  is given by

$$R_{ij} = \sum_{m:(i,j) \in A_m} \pi_m.$$

In this section, we will use the node-based resource constraint formulation used in Section 2.3. Thus, at each node, we keep track of the rate at which traffic arrives and leaves for a particular destination, and impose the constraint that the arrival rate is less than the departure rate. The reason we choose this formulation is that we will maintain a separate queue for each destination at each node, and the constraints ensure that each of the per-destination queues is stable. We will see that, under this formulation, one does not have to assume that arrivals occur to each queue along a path simultaneously as in the previous section. To formulate the node-based constraints, denote the inflow rate allocated for destination  $d$  at node  $i$  by  $R_{in(i)}^d$  and the outflow rate  $R_{out(i)}^d$ . Thus, at each node  $i$ ,

$$\sum_d R_{in(i)}^d = \sum_j R_{ji} \quad \text{and} \quad \sum_d R_{out(i)}^d = \sum_k R_{ik}.$$

For simplicity, we will only consider the case where  $U_r(x_r) = w_r \log x_r$ . Thus, the optimization problem for the resource allocation problem is given by

$$\max_{x, \pi, R \geq 0} \sum_f w_f \log x_f,$$

subject to the following constraints:

$$R_{in(i)}^d + \sum_{f:b(f)=i,e(f)=d} x_f \leq R_{out(i)}^d, \quad \forall \text{ nodes } i, d \neq i \quad (3.27)$$

$$R_{in(i)}^d = \sum_j R_{ji}^d, \quad \forall i, j, d \neq i \quad (3.28)$$

$$R_{out(i)}^d = \sum_j R_{ij}^d, \quad \forall i, j, d \neq i \quad (3.29)$$

$$\sum_d R_{ij}^d = R_{ij}, \quad \forall i, j, d \neq i \quad (3.30)$$

$$R_{ij} = \sum_{m:(i,j) \in A_m} \pi_m, \quad \forall i, j \quad (3.31)$$

$$\sum_{m=1}^M \pi_m = 1. \quad (3.32)$$

Let the Lagrange multiplier corresponding to constraint (3.27) be denoted by  $p_{id}$ . Appending the constraints (3.27) to the objective, we get

$$\max_{x,\pi,R \geq 0} \sum_f w_f \log x_f - \sum_i \sum_{d \neq i} p_{id} \left( R_{in(i)}^d + \sum_{f:b(f)=i} x_f - R_{out(i)}^d \right).$$

By manipulating the summations, the above expression can be rewritten as

$$\max_{x \geq 0} \left( \sum_f w_f \log x_f - \sum_f p_{ib(f)} x_f \right) + \max_{\pi, R \geq 0} \left( \sum_i \sum_{d \neq i} p_{id} (R_{out(i)}^d - R_{in(i)}^d) \right).$$

If the optimal values of the Lagrange multipliers are known, then we have to solve two problems: the congestion control problem

$$\max_{x \geq 0} \left( \sum_f w_f \log x_f - \sum_f p_{ib(f)} x_f \right),$$

and the scheduling problem

$$\max_{\pi, R \geq 0} \left( \sum_i \sum_{d \neq i} p_{id} (R_{out(i)}^d - R_{in(i)}^d) \right), \quad (3.33)$$

where the scheduling problem is subject to the constraints (3.28)-(3.32). To solve the congestion control problem, we use the primal algorithm at each source  $f$ :

$$\dot{x}_f(t) = \frac{w_r}{x_r} - p_{b(f)e(f)}, \quad (3.34)$$

and the dual algorithm for price update at each node  $i$ , for each destination  $d$ :

$$\dot{p}_{id}(t) = \left( \sum_{f:b(f)=i, e(f)=d} x_f(t) + R_{in(i)}^d(t) - R_{out(i)}^d \right)_{p_{nd}}^+. \quad (3.35)$$

In the price-update algorithm, the rates  $R_{in(i)}^d$  and  $R_{out(i)}^d$  are calculated at each time instant by solving (3.33). If the optimal solution to (3.33) is not unique, any one of the optima can be used.

To show convergence of the algorithm, we will use a theorem similar to the Lyapunov theorem that we saw earlier in the chapter. The theorem is designed for situations where the time-derivative of a Lyapunov function is zero at more than one point. Consider the differential equation  $\dot{y} = f(y(t))$ . Then we have the following theorem [24]:

---

**Theorem 3.4.** (LaSalle's Invariance Principle). Let  $W : D \rightarrow R$  be a radially unbounded (i.e.,  $\lim_{\|y\| \rightarrow \infty} W(y) = \infty$ ,  $y \in D$ ), continuously differentiable, positive definite function such that  $\dot{W}(y) \leq 0$  for all  $y \in D$ . Let  $\mathcal{E}$  be the set of points in  $D$  where  $\dot{W}(y) = 0$ . Let  $\mathcal{M}$  be the largest invariant set in  $\mathcal{E}$  (i.e.,  $\mathcal{M} \subseteq \mathcal{E}$  and if  $y(0) \in \mathcal{M}$ , then  $y(t) \in \mathcal{M} \ \forall t \geq 0$ ). Then, every solution starting in  $D$  approaches  $\mathcal{M}$  as  $t \rightarrow \infty$ .

---

In order to study the convergence of the controller, we need to construct an appropriate Lyapunov function. We will use the following Lyapunov function:

$$W(x, q) \triangleq \frac{1}{2} \sum_{f \in \mathcal{F}} (x_f - \hat{x}_f)^2 + \frac{1}{2} \sum_i \sum_{d \neq i} (p_{id} - \hat{p}_{nd})^2, \quad (3.36)$$

where the quantities with hats are the solutions to the network utility maximization problem. There can be multiple values of the Lagrange

multipliers that satisfy the KKT conditions; in that case, one can pick any one of them to construct the Lyapunov function. From now on, in double summations involving a node  $i$  and destination  $d$ , for ease of notation, we will assume that  $d \neq i$  without explicitly stating it. We are now ready to prove the following theorem:

---

**Theorem 3.5.** Starting from any  $x(0)$  and  $q(0)$ , the rate vector  $x(t)$  converges to  $\hat{x}$  as  $t \rightarrow \infty$  and

$$p_{b(f)e(f)} = \hat{p}_{b(f)e(f)} = w_f / \hat{x}_f \quad \forall f.$$

Further, the queue length vector  $p(t)$  approaches the bounded set

$$\left\{ p \geq 0 : \sum_{i,d} (p_{id} - \hat{p}_{id}) \left( R_{out(i)}^d - R_{in(i)}^d + \sum_{f:b(f)=i,e(f)=d} (x_f - \hat{x}_f) \right) = 0 \right\}.$$


---

*Proof.* Differentiating the Lyapunov function with respect to time we obtain

$$\begin{aligned} \dot{W} &= \sum_f (x_f - \hat{x}_f) \left( \frac{1}{x_f} - p_{b(f)e(f)} \right) \\ &\quad + \sum_{i,d} (p_{id} - \hat{p}_{id}) \left( \sum_{f:b(f)=i,e(f)=d} x_f + R_{in(i)}^d - R_{out(i)}^d \right)_{p_{id}}^+ \quad (3.37) \\ &\leq \sum_f (x_f - \hat{x}_f) \left( \frac{1}{x_f} - p_{b(f)e(f)} \right) \\ &\quad + \sum_{i,d} (p_{id} - \hat{p}_{id}) \left( \sum_{f:b(f)=i,e(f)=d} x_f + R_{in(i)}^d - R_{out(i)}^d \right) \quad (3.38) \end{aligned}$$

The last inequality follows by noting that (3.38) and (3.37) are equal if the projection in (3.37) is inactive and if the projection is active, the expression in (3.37) is zero, while the expression in (3.38) is positive due to the fact that  $p_{id} = 0$  and the term inside the parenthesis is negative (otherwise, the projection will not be active). Using the fact



$w_f/\hat{x}_f = \hat{p}_{b(f)e(f)}$  and adding and subtracting terms, we get

$$\begin{aligned} \dot{W} &= \sum_f (x_f - \hat{x}_f) \left( \frac{1}{x_f} - \frac{1}{\hat{x}_f} + \hat{p}_{b(f)e(f)} - p_{b(f)e(f)} \right) \\ &\quad + \sum_{i,d} (p_{id} - \hat{p}_{id}) \left( \sum_{f:b(f)=i,e(f)=d} x_f - \sum_{f:b(f)=i} \hat{x}_f \right) \\ &\quad + \sum_{i,d} (p_{id} - \hat{p}_{id}) \left( \sum_{f:b(f)=i,e(f)=d} \hat{x}_f + R_{in(i)}^d - R_{out(i)}^d \right). \end{aligned}$$

Noting that

$$\sum_{i,d} (p_{id} - \hat{p}_{id}) \sum_{f:b(f)=i,e(f)=d} (x_f - \hat{x}_f) = - \sum_f (x_f - \hat{x}_f) (\hat{p}_{b(f)e(f)} - p_{b(f)e(f)}),$$

we get

$$\begin{aligned} \dot{W} &= \sum_f (x_f - \hat{x}_f) \left( \frac{1}{x_f} - \frac{1}{\hat{x}_f} \right) \\ &\quad + \sum_{i,d} (p_{id} - \hat{p}_{id}) \left( \sum_{f:b(f)=i,e(f)=d} \hat{x}_f + R_{in(i)}^d - R_{out(i)}^d \right). \end{aligned}$$

Let us now examine each of the terms in the right-hand side of the above equation. It is easy to see that

$$(x_f - \hat{x}_f) \left( \frac{1}{x_f} - \frac{1}{\hat{x}_f} \right) \leq 0.$$

From constraint (3.27),

$$\sum_{f:b(f)=i,e(f)=d} \hat{x}_f \leq \hat{R}_{out(i)}^d - \hat{R}_{in(i)}^d,$$

where we recall that the quantities with hats are the optimal solution to the network utility maximization problem. Thus,

$$\sum_{i,d} p_{id} \left( \sum_{f:b(f)=i,e(f)=d} \hat{x}_f + R_{in(i)}^d - R_{out(i)}^d \right) \leq 0,$$

since the rates  $R_{in(i)}^d$  and  $R_{out(i)}^d$  solve (3.33) and  $\hat{R}_{out(i)}^d$  and  $\hat{R}_{in(i)}^d$  are feasible solutions to the outflow and inflow rates at node  $i$ , for destination  $d$ . From the KKT conditions

$$p_{id} \left( \hat{R}_{in(i)}^d + \sum_{f:b(f)=i,e(f)=d} \hat{x}_f - \hat{R}_{out(i)}^d \right) = 0.$$

Thus,

$$\begin{aligned} & - \sum_{i,d} \hat{p}_{id} \left( \sum_{f:b(f)=i,e(f)=d} \hat{x}_f + R_{in(i)}^d - R_{out(i)}^d \right) \\ &= - \sum_{i,d} \hat{p}_{id} \left( \sum_{f:b(f)=i,e(f)=d} \hat{R}_{out(i)}^d - \hat{R}_{in(i)}^d - R_{out(i)}^d + R_{in(i)}^d \right) \leq 0, \end{aligned}$$

since the  $\hat{R}$ 's solve the scheduling problem with  $\hat{p}$ 's as the weights in the scheduling algorithm. Thus,  $\dot{W} \leq 0$ . To apply LaSalle's invariance principle, let us consider the set of points  $\mathcal{E}$ , where  $\dot{W} = 0$ . The set  $\mathcal{E}$  is given by the set of points  $(x, p)$  that satisfy

$$\left\{ x_f = \hat{x}_f, \sum_{i,d} (p_{id} - \hat{p}_{id}) \left( \sum_{f:b(f)=i,e(f)=d} \hat{x}_f + R_{in(i)}^d - R_{out(i)}^d \right) = 0 \right\}.$$

We claim that the largest invariant set  $\mathcal{M} \subseteq \mathcal{E}$  is further characterized by  $p_{b(f)e(f)} = w_f / \hat{x}_f$ . To see this, note that if this condition is violated, then the congestion controller (3.34) will change the rate from  $\hat{x}_f$  and hence the system will move outside  $\mathcal{E}$ . Thus, LaSalle's invariance principle applies and the theorem is proved.  $\square$

We will now examine the scheduling problem (3.33) carefully with the goal of simplifying it. By rearranging the summation and using the definitions of  $R_{in(i)}^d$  from (3.28) and  $R_{out(i)}^d$  from (3.29), (3.33) can be rewritten as

$$\max \sum_{i,k} \sum_d R_{ik}^d (p_{id} - p_{kd}).$$

Using the fact that  $\sum_d R_{ik}^d = R_{ik}$ , the scheduling problem becomes

$$\max \sum_{i,k} R_{ik} \max_d (p_{id} - p_{kd}).$$

Using (3.31) and (3.32), the scheduling problem further reduces to

$$\max_{\sum_m \pi_m = 1} \sum_{i,k,m:(i,k) \in A_m} \pi_m \max_d (p_{id} - p_{kd}) \quad (3.39)$$

$$\begin{aligned} &= \max_{\sum_m \pi_m = 1} \sum_m \pi_m \sum_{(i,k) \in A_m} \max_d (p_{id} - p_{kd}) \\ &= \max_m \sum_{(i,k) \in A_m} \max_d (p_{id} - p_{kd}). \end{aligned} \quad (3.40)$$

Thus, the scheduling problem becomes one of finding a schedule with the maximum weight, where the weight of a link is given by

$$\max_d (p_{id} - p_{kd}).$$

This is called the *back-pressure* algorithm.

Let us examine the price-update equation (3.35). This equation closely resembles the queue dynamics in the queue holding packets for destination  $d$  at packet  $i$ . However, notice that the arrival rate into the queue includes the term

$$R_{in(i)}^d = \sum_j R_{ji}^d,$$

which is not the true arrival rate at the queue, but it is the potential rate at which packets could arrive from other nodes to node  $i$ . This rate may not be realizable if some of the other nodes have empty queues. However, note from the back-pressure algorithm (3.40) that  $R_{ji}^d = 0$  if  $p_{jd} = 0$ . Thus, hop  $(i, j)$  is not scheduled if  $p_{jd} = 0$ . Hence, one can indeed interpret the price as the queue length at node  $i$  for packets destined for node  $d$ . One can then simply use the lengths of the per-destination queues as the prices, and no price computation is necessary.

It may seem a little surprising at first that the congestion controller (3.34) for flow  $f$  uses only the ingress queue length to regulate the flow's arrival rate. After all, congestion could occur anywhere in the interior of the network. To understand how congestion is managed in the interior of the network, consider a wireless network with just one flow and three hops, where the interference constraint dictates that either hops 1 and 3 or 2 may be scheduled in a time slot (see Figure 3.6). Suppose that the queue at the third hop is large initially, then

the back-pressure algorithm will schedule this hop first (along with the first hop since it satisfies the scheduling constraint). The queue length at the second hop will increase, and eventually, the congestion controller will start throttling the rate which packets are injected into the first queue. Hops are scheduled depending upon the values of the back-pressures at these hops. In particular, at some point in time, the first hop will be scheduled. At this time, the queue length at the first hop will decrease and the source can increase its transmission rate. Thus, the back-pressure algorithm pushes the congestion towards the source node. This is illustrated in Figure 3.6.

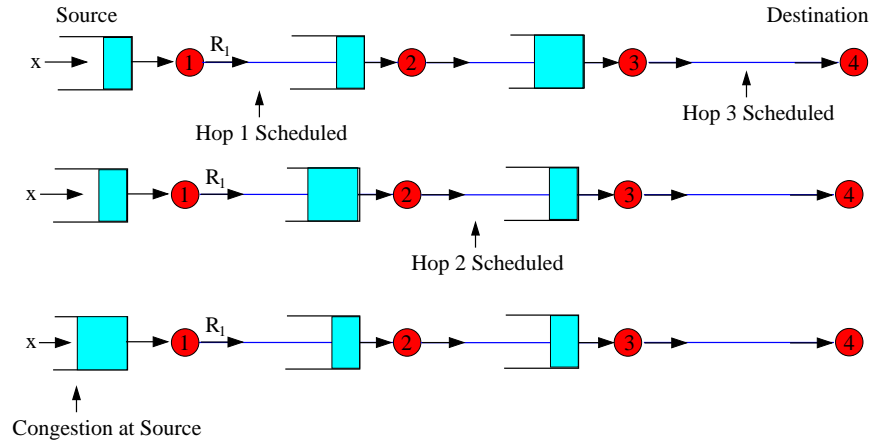


Fig. 3.6 Node based constraints yield the back-pressure algorithm that causes congestion effects to be progressively transferred towards the source.

### 3.5 Notes

Gradient algorithms are well known in optimization, but recognizing that the gradient algorithm for network utility maximization leads to distributed algorithms for congestion control is a relatively new development. The primal and dual algorithms for solving the network utility maximization problem were presented by Kelly et. al. in [7]. The version of the dual algorithm presented in this chapter is a continuous-time version of the algorithms proposed by Low and Lapsley [25], and Yaiche

et. al. in [26]. The Lyapunov proof we have used to show stability of the dual algorithm is an adaptation of a proof by Paganini [27]. The multi-path algorithm was also proposed in [7], while the minor addition of the  $\epsilon$  term was proposed in Han et. al. in [28]. A dual version of the multi-path congestion control algorithm was proposed by Low in [29] and its convergence was proved by Lin and Shroff [30].

The primal-dual algorithm for Internet congestion control was introduced by Kunniyur and Srikant in [31] and its convergence was proved in [32] although the algorithm at the nodes to compute the Lagrange multipliers is different from the one presented in this paper. The version of the primal-dual algorithm presented here is due to Wen and Arcak [33] and Alpcan and Başar [34]. The block diagram in Figure 3.2 that summarizes the relationships between the primal and dual variables is from [35]. The extension of the utility maximization approach to wireless networks is more recent and appears in [10–14]. The proof presented in this chapter is due to Eryilmaz and Srikant [14]. The results presented in this chapter can also be extended to deal with multi-rate, multicast flows; see [36–38].

We also note that the special case of distributed algorithms to achieve max-min fairness was considered by Fulton et al. [39], Lee and de Veciana [40].

We note that the models in this monograph assume that the flows in the network are elastic, i.e., they are able to adapt their rates. However, many real networks also have a large number of flows that request a fixed data rate from the network, also known as inelastic flows. When an inelastic flow arrives, the network first checks to see if it has enough spare capacity to accommodate the data rate request of an inelastic flow and the flow is admitted if there is sufficient available capacity. Once admitted, these flows are typically given higher priority compared to elastic flows. Thus, the elastic flows would be subjected to variable link capacities due to the arrivals and departures of inelastic flows. Such fluctuations can be incorporated into the stochastic models in [11–14]. The resource constraints in our model in this chapter and the previous one can be viewed as average capacity constraints where the averaging is done over a long time-scale.

One bit feedback for congestion control was proposed in Ramakrishnan et. al. in [41] and Chiu and Jain in [2]. The RED scheme was proposed by Floyd and Jacobson [42, 43]. The idea of using exponential functions to convert price information to one-bit information is due to Lapsley and Low [44]. The use of time-scale separation assumption to convert queue-based marking functions to rate-based marking functions was suggested by Kelly in [8] and a proof of this result was provided by Deb and Srikant in [45, 46]. The price function for the DropTail algorithm was proposed in Kunniyur and Srikant in [31]. A discussion of appropriate models of congestion control under different network conditions can be found in the paper by Raina and Wischik [47].

The optimization framework not only provides an algorithm for resource allocation but also reveals where computational functionalities required for resource allocation must reside in the network. For example, it tells us that the sources should adapt rates based on congestion signals from the network and suggests simple mechanisms for the resources (nodes and links) in the network to provide these congestion signals. In fact, communication networks are designed in this manner today. On the other hand, the optimization approach also reveals that certain traditional methods of designing networks may not be optimal. For example, in the primal-dual model with interference constraints, we showed that it is optimal for the network to implement scheduling and routing jointly using the back-pressure algorithm. However, in today's networks, routing and scheduling are done by different protocols, independently of each other. While the back-pressure algorithm is optimal, it is not implementable due to the fact that it requires a central authority with knowledge of the entire network state to solve the optimization problem. Thus, one may want to consider different architectures to strike a balance between optimality and implementability. The architectural implications of optimization-based design are discussed in detail in [48].

In the next chapter, we will study the relationship between the resource allocation framework that we have developed and the protocols used in the Internet today. We will see how the primal and the dual approaches bear strong resemblance to existing and proposed congestion-control protocols for the Internet.





# 4

---

## Congestion Control Protocols

---

We have seen in the previous chapters how resource allocation may be achieved in a decentralized fashion in a fair and stable manner. We emphasized that the algorithms could be in networks that are enabled with some kind of feedback that indicates congestion levels to users of each link. In this chapter, we explore the relationship between the algorithms discussed in the previous chapters and the protocols used in the Internet today. It is important to note that Internet congestion control protocols were *not* designed using the optimization formulation of the resource allocation problem that we have seen in the previous two chapters. The predominant concern while designing these protocols was to minimize the risk of congestion collapse, i.e., large-scale buffer overflows, and hence they tended to be rather conservative in their behavior. Even though the current Internet protocols were not designed with clearly-defined fairness and stability ideas in mind, they bear a strong resemblance to the ideas of fair resource allocation that we have discussed so far. In fact, the utility maximization methods presented earlier provide a solid framework for understanding the operation of these congestion control algorithms. Further, going forward, the utility maximization approach seems like a natural candidate framework to



modify existing protocols to adapt to the evolution of the Internet as it continues to grow faster.

As mentioned in the first chapter, the congestion control algorithm used in today's Internet is implemented within the *Transmission Control Protocol* (TCP). There are several different flavors of TCP congestion control, each of which operates somewhat differently. But all versions of TCP are *window-based* protocols. The idea is that each user maintains a number called a *window size*, which is the number of unacknowledged packets that it is allowed to send into the network. Any new packet can be sent only when an acknowledgment for one of the previous sent packets is received by the sender. TCP adapts the window size in response to congestion information. An illustration is presented in Figure 4.1. The window size is increased if the sender determines

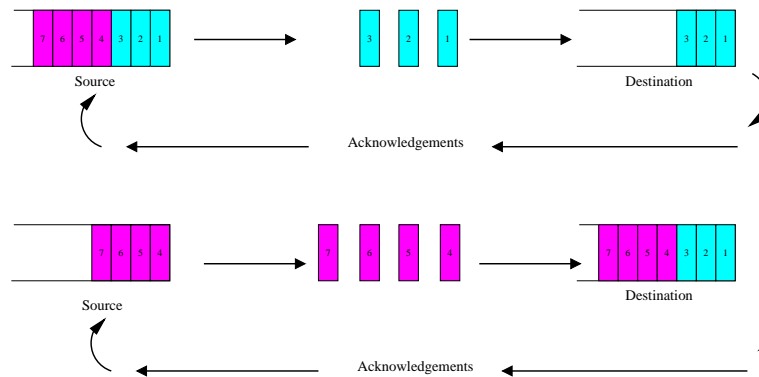


Fig. 4.1 An example of a window flow control. The sender's congestion window determines the number of packets in flight. The destination receives the packets sent and responds to them with acknowledgements. It takes one RTT for the source to be aware of a successful transmission and to change the window size. Note that this is a considerably simplified interpretation of TCP.

that there is excess capacity present in the route, and decreased if the sender determines that the current number of in-flight packets exceeds the capacity of the route. In the figure, the source has chosen to increase the window size from 3 to 4 packets. The amount of time that elapses between the sending of a packet and the reception of feedback from the destination is called the Round-Trip Time (RTT). We denote the

RTT by  $T$ . A decision on whether to send a new packet, and whether the window is to be increased or decreased, is taken upon reception of the acknowledgement packet. This means that the decision-making process has no periodicity that is decided by a clock of fixed frequency. TCP is therefore called *self-clocking*. The nearest approximation to a clock is the round-trip time  $T$ , which could potentially vary during the lifetime of a flow. The exact means of determining whether to increase or decrease the window size is what determines the difference between the congestion control mechanism of different TCP flavors which we discuss in the rest of this chapter.

## 4.1 TCP-Reno

The most commonly used TCP flavors used for congestion control in the Internet today are Reno and NewReno. Both of them are updates of the TCP-Tahoe, which was introduced in 1988. Although they vary significantly in many regards, the basic approach to congestion control is similar. The idea is to use successful reception packets as an indication of available capacity and dropped packets as an indication of congestion. We consider a simplified model for the purpose of exposition. Each time the destination receives a packet, it sends an acknowledgement (also called *ack*) asking for the next packet in sequence. For example, when packet 1 is received, the acknowledgement takes the form of a request for packet 2. If, instead of the expected packet 2, the destination receives packet 3, the acknowledgement still requests packet 2. Reception of three duplicated acknowledgments or *dupacks* (i.e., four successive identical acks) is taken as an indication that packet 2 has been lost due to congestion. The source then proceeds to cut down the window size and also to re-transmit lost packets. In case the source does not receive any acknowledgements for a finite time, it assumes that all its packets have been lost and times out.

When a non-duplicate acknowledgment is received, the protocol increases its window size. The amount by which the window size is increased depends upon the TCP transmission *phase*. TCP operates in two distinct phases. When file transfer begins, the window size is 1, but the source rapidly increases its transmission window size so as to

reach the available capacity quickly. Let us denote the window size  $W$ . The algorithm increases the window size by 1 each time an acknowledgement indicating success is received, i.e.,  $W \leftarrow W + 1$ . This is called the *slow-start* phase. Since one would receive acknowledgements corresponding to one window's worth of packets in an RTT, and we increase the window size by one for each successful packet transmission, this also means that (if all transmissions are successful) the window would double in each RTT, so we have an exponential increase in rate as time proceeds. Slow-start refers to the fact that the window size is still small in this phase, but the rate at which the window is increases is quite rapid. When the window size either hits a threshold, called the *slow-start threshold* or *ssthresh* or the transmission suffers a loss (immediately leading to a halving of window size), the algorithm shifts to a more conservative approach called the *congestion avoidance* phase. When in the congestion-avoidance phase, the algorithm increases the window size by  $1/W$  every time feedback of a successful packet transmission is received, so we now have  $W \leftarrow W + 1/W$ . When a packet loss is detected by the receipt of three dupacks, the slow-start threshold (*ssthresh*) is set to  $W$  and TCP Reno cuts its window size by half, i.e.,  $W \leftarrow W/2$ . Thus, in each RTT, the window increases by one packet—a linear increase in rate. Protocols of this sort where increment is by a constant amount, but the decrement is by a multiplicative factor are called *additive-increase multiplicative-decrease* (AIMD) protocols. When packet loss is detected by a time-out, the window size is reset to 1 and TCP enters the slow-start phase. We illustrate the operation of TCP-Reno in terms of rate of transmission in Figure 4.2.

Now, the slow-start phase of a flow is relatively insignificant if the flow consists of a large number of packets. So we will consider only the congestion-avoidance phase. Let us call the congestion window at time  $t$  as  $W(t)$ . This means that the number of packets in-flight is  $W(t)$ . The time taken by each of these packets to reach the destination, and for the corresponding acknowledgement to be received is  $T$ . The RTT is a combination of propagation delay and queueing delay. In our modeling, we assume that the RTT is constant, equal to the propagation delay plus the maximum queueing delay. If one observes the medium between the source and destination for an interval  $[t, t + T]$ , and there are no

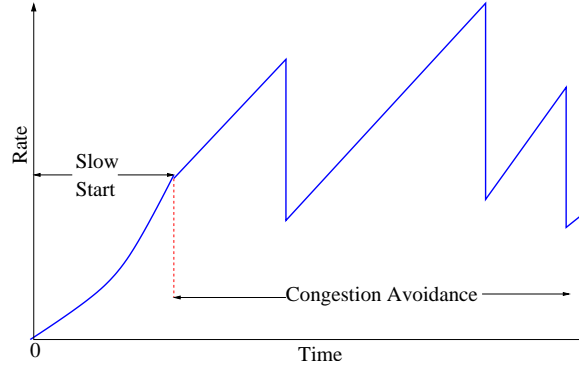


Fig. 4.2 Operation of TCP-Reno. There is an exponential increase in rate during the slow-start phase and a linear increase or a halving of rate in the congestion avoidance phase. If loss is experienced during slow start, the algorithm halves the sending rate and switches to congestion avoidance.

dropped packets, then the number of packets seen is  $W(t)$  since the window size changes rather slowly in one RTT. Thus, the average rate of transmission  $x(t)$  is just the window size divided by  $T$ , i.e.,  $x(t) = W(t)/T$ . Let us now write down what we have just seen about TCP Reno's behavior in terms of the differential equation models that we have become familiar with over the past few chapters.

Consider a flow  $r$ . As defined above, let  $W_r(t)$  denote the window size and  $T_r$  its RTT. Earlier we had the concept of the price of a route  $r$  being  $q_r(t)$ . We now use the same notation to denote the probability that a packet will be lost at time  $t$ . Notice that the loss of packets is the price paid by flow  $r$  for using the links that constitute the route it uses. We can model the congestion avoidance phase of TCP-Reno as

$$\dot{W}_r(t) = \frac{x_r(t - T_r)(1 - q_r(t))}{W_r(t)} - \beta x_r(t - T_r)q_r(t)W_r(t). \quad (4.1)$$

The above equation can be derived as follows:

- The rate at which the source obtains acknowledgements is  $x_r(t - T_r)(1 - q_r(t))$ . Since each acknowledgement leads to an increase by  $1/W(t)$ , the rate at which the transmission rate increases is given by the first term on the right side.

- The rate at which packets are lost is  $x_r(t - T_r)q_r(t)$ . Such events would cause the rate of transmission to be decreased by a factor that we call  $\beta$ . This is the second term on the right side. Considering the fact that there is a halving of window size due to loss of packets,  $\beta$  would naturally be taken to be  $1/2$ . However, studies show that a more precise value of  $\beta$  when making a continuous-time approximation of TCP's behavior is close to  $2/3$ .

To compare the TCP formulation above to the resource allocation framework, we write  $W_r(t)$  in terms of  $x_r(t)$  which yields

$$\dot{x}_r = \frac{x_r(t - T_r)(1 - q_r(t))}{T_r^2 x_r} - \beta x_r(t - T_r)q_r(t)x_r(t). \quad (4.2)$$

The equilibrium value of  $x_r$  is found by setting  $\dot{x}_r = 0$ , and is seen to be

$$\hat{x}_r = \sqrt{\frac{1 - \hat{q}}{\beta \hat{q}}} \frac{1}{T_r},$$

where  $\hat{q}_r$  is the equilibrium loss probability. For small values of  $\hat{q}_r$  (which is what one desires in the Internet),

$$\hat{x}_r \propto 1/T_r \sqrt{\hat{q}_r}.$$

This result is well-known and widely used in the performance analysis of TCP.

## 4.2 Relationship with Primal Algorithm

Now, consider the controller (4.2) again. Suppose that there were no feedback delay, but the equation is otherwise unchanged. So  $T_r^2$  that appears in (4.2) is just some constant now. Also, let  $q_r(t)$  be small, i.e., the probability of losing a packet is not too large. Then the controller reduces to

$$\begin{aligned} \dot{x}_r &= \frac{1}{T_r^2} - \beta x_r^2 q_r \\ &= \beta x_r^2 \left( \frac{1}{\beta T_r^2 x_r^2} - q_r \right). \end{aligned}$$

Comparing with (3.5), we find that the utility function of the source  $r$  satisfies

$$U'_r(x_r) = \frac{1}{\beta T_r^2 x_r^2}.$$

We can find the source utility (up to an additive constant) by integrating the above, which yields

$$U_r(x_r) = -\frac{1}{\beta T_r^2 x_r}.$$

Thus, TCP can be approximately viewed as a control algorithm that attempts to achieve weighted minimum potential delay fairness.

If we do not assume that  $q_r$  is small, the delay-free differential equation is given by

$$\begin{aligned} \dot{x}_r &= \frac{1 - q_r}{T_r^2} - \beta x_r^2 q_r \\ &= (\beta x_r^2 + 1/T_r^2) \left( \frac{1}{\beta T_r^2 x_r^2 + 1} - q_r \right), \end{aligned}$$

Thus,

$$U'_r(x_r) = \frac{1}{\beta T_r^2 x_r^2 + 1} \quad \Rightarrow \quad U_r(x_r) = \frac{1}{T_r \sqrt{\beta}} \tan^{-1} \left( \sqrt{\beta} T_r x_r \right),$$

where the utility function is determined up to an additive constant.

### 4.3 A Generalization of TCP-Reno

Instead of increasing the window size by  $1/W$  for each ack and decreasing the window by  $1/2$  upon detecting a loss, one could consider other increase-decrease choices as well. Consider a protocol where  $W \leftarrow W + a W^n$  when an acknowledgement is received, while a loss triggers a window decrease given by  $W \leftarrow W - b W^m$ . Setting  $a = 1$ ,  $n = -1$ ,  $b = 0.5$ , and  $m = 1$  would yield TCP-Reno type behavior. The equivalent rate-based equation describing the dynamics of such a protocol would be

$$\dot{x}_r = \frac{x_r(t - T_r)}{T_r} (a(x_r(t) T_r)^n (1 - q_r(t)) - b(x_r(t) T_r)^m q_r(t)). \quad (4.3)$$

Ignoring the feedback delay in obtaining the congestion information, the above differential equation becomes

$$\dot{x}_r = \frac{x_r(t)}{T_r} (a(x_r(t) T_r)^n (1 - q_r(t)) b(x_r(t) T_r)^m q_r(t)),$$

which can be rewritten as

$$\dot{x}_r = (a x_r(t)^{n+1} T_r^{n-1}) \left( 1 + \frac{b}{a} (x_r(t) T_r)^{m-n} \right) \left( \frac{1}{1 + \frac{b}{a} (x_r(t) T_r)^{m-n}} - q_r \right).$$

Note that

$$\frac{1}{1 + \frac{b}{a} (x_r(t) T_r)^{m-n}}$$

is a decreasing function (thus, its derivative will be negative) if  $m > n$  and hence one can view the above differential equation as the congestion controller for a source  $r$  with a concave utility function

$$\int_0^{x_r(t)} \frac{1}{1 + \frac{b}{a} (x T_r)^{m-n}} dx.$$

A special case of the above congestion control algorithm called *Scalable-TCP* which uses the parameters  $a = 0.01$ ,  $n = 0$ ,  $b = 0.125$ , and  $m = 1$  has been proposed for high-bandwidth environments.

#### 4.4 TCP-Vegas: A Delay Based Algorithm

We now consider another variation of TCP called TCP-Vegas. TCP-Vegas uses queueing delay to infer congestion in the network. The idea is to first identify the propagation delay of the route by assuming that it is equal to the smallest RTT seen by the source. This is a reasonable assumption if we assume that queues empty occasionally in which case the only delay is the propagation delay. Let us denote the estimated propagation delay by  $T_p$ . Any excess delay above this amount would be queueing delay and we denote it  $T_q$ . The objective of the algorithm is to calculate the value of window size such that there is no queueing delay. When this occurs, the rate of generation of packets is equal to the available capacity on the route. We now study the details of this algorithm.

If there is no queueing delay, the throughput would be approximately given by

$$e = \frac{W}{T_p},$$

where  $W$  is the window size. However, the actual throughput is the number of packets that make it successfully to the destination in a fixed amount of time. To calculate this quantity, the source sends a marked packet and waits for it to be acknowledged. The duration of time that it takes for this event to occur is the round-trip time  $T_p + T_q$ . Suppose during this time, the source receives  $S$  acknowledgments, then the actual throughput is estimated as

$$a = \frac{S}{T_p + T_q}.$$

Whenever we receive the acknowledgment for a marked packet at some time  $t$ , we have two values—the expected throughput  $e(t)$  and the actual throughput  $a(t)$ . If  $a(t)$  is less than  $e(t)$ , it means that our transmission rate is too high, so we should cut down the window size. On the other hand, if  $a(t)$  is greater than  $e(t)$ , it means that the estimate of the available rate is too low and we should increase the window size. Formally, we define constants  $\alpha$  and  $\beta$ , with  $\alpha \leq \beta$  and proceed as follows:

- if  $\alpha \leq (e(t) - a(t)) \leq \beta$ , then do nothing. This means that our estimate of the throughput is fairly accurate, and everything is as it should be.
- if  $(e(t) - a(t)) > \beta$ , decrease the window size by 1 for the next RTT. This means that our estimate is too high and the window size must be reduced.
- if  $(e(t) - a(t)) < \alpha$ , increase the window size by 1 for the next RTT. This means that our estimate is too low and the network can support more than we think.

Note that both the increase and decrease of window size are linear in nature. Also, the algorithm uses the usual slow start mechanism, with exponential growth initially. The behavior of TCP-Vegas under ideal conditions would look something like Figure 4.3.



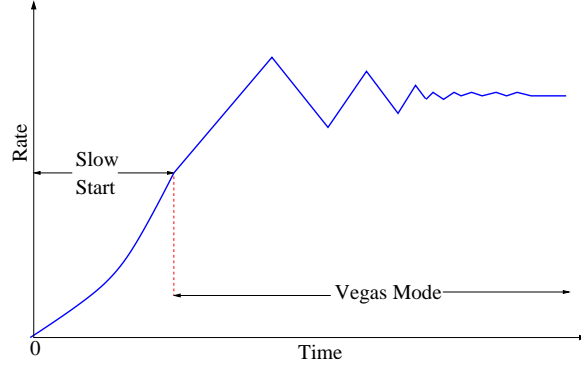


Fig. 4.3 Operation of TCP-Vegas. It preserves the slow-start phase of TCP-Reno, but switches to Vegas mode once it passes the slow start threshold (or a loss occurs). Both the window increase and decrease are linear. Ideally, the algorithm should converge to a stable window size.

#### 4.5 TCP-Vegas as a Resource Allocation Algorithm

In this section, we interpret TCP-Vegas as a resource allocation algorithm in the utility maximization framework. We assume  $\alpha = \beta$  and the propagation delay is estimated accurately, i.e., for source  $r$ ,  $T_{pr}(t) \equiv T_{pr}$  for all  $t$ .

At equilibrium, the estimated throughput is the same as the actual throughput, with the window size and the number of acknowledgements received in an RTT being the same. If we denote the equilibrium window size and queueing delay of source  $r$  by  $\hat{W}_r$  and  $\hat{T}_{qr}$  respectively (and by assumption, the propagation delay  $T_{pr}$  is correctly estimated), then

$$\frac{\hat{W}_r}{T_{pr}} - \frac{\hat{W}_r}{T_{pr} + \hat{T}_{qr}} = \alpha. \quad (4.4)$$

At equilibrium, the transmission rate  $\hat{x}$  is approximately

$$\hat{x}_r = \hat{W}_r / (T_{pr} + \hat{T}_{qr}),$$

which means that (4.4) can be simplified to

$$\frac{\alpha T_{pr}}{\hat{x}_r} = \hat{T}_{qr}. \quad (4.5)$$

Now that we know what the equilibrium transmission rate looks like, let us study what the equilibrium queueing delay  $\hat{T}_{qr}$  would look like. If the queue length at equilibrium is denoted by  $\hat{b}_l$ , then the equilibrium queueing delay at link  $l$  is  $\hat{b}_l/c_l$  (where  $c_l$  is the capacity of link  $l$ ). So we have

$$\hat{T}_{qr} = \sum_{l:l \in r} \frac{\hat{b}_l}{c_l}. \quad (4.6)$$

Also, if

$$\sum_{k:l \in k} \hat{x}_k < c_l,$$

then there is no queueing delay, i.e.,  $\hat{b}_l = 0$  in this case. Note that since the aggregate equilibrium transmission rate of all flows using link  $l$  cannot exceed the link capacity, we cannot possibly have that

$$\sum_{k:l \in k} \hat{x}_k > c_l.$$

Thus, if  $\hat{b}_l \neq 0$ , it means that

$$\sum_{k:l \in k} \hat{x}_k = c_l.$$

Thus, we have from the above and (4.5), that the equilibrium conditions are

$$\frac{\alpha T_{pr}}{\hat{x}_r} = \sum_{l:l \in r} \frac{\hat{b}_l}{c_l},$$

with

$$\frac{\hat{b}_l}{c_l} \left( \sum_{k:l \in k} \hat{x}_k - c_l \right) = 0 \quad \forall l.$$

However, these are the Karush-Kuhn-Tucker conditions for the utility maximization problem

$$\max_{\{x_r\}} \alpha T_{pr} \log x_r, \quad (4.7)$$

subject to

$$\sum_{r:l \in r} x_r \leq c_l, \quad \forall l$$

and  $x_r \geq 0, \forall r$ . Thus, TCP-Vegas is weighted-proportionally fair. If we let each flow have a different value of  $\alpha$ , i.e., we associate  $\alpha_r$  with route  $r$ , then the equilibrium rates will maximize  $\sum_r \alpha_r T_{pr} \log x_r$ .

#### 4.6 Relation to Dual Algorithms and Extensions

We now consider the relationship between TCP-Vegas and dual algorithms. A weighted proportionally fair dual algorithm would use the controller obtained by substituting  $w_r \log(x_r)$  as the utility function in (3.23) and (3.24), which yields

$$x_r = \frac{w_r}{p_r} \quad \text{and} \quad (4.8)$$

$$\dot{p}_l = h_l(y_l - c_l)_{p_l}^+, \quad (4.9)$$

To avoid confusion, we note that  $w_r$  is the weight assigned to source  $r$  and is unrelated to the window size  $W_r(t)$ . If we choose  $h_l = \frac{1}{c_l}$ , then the price function of a link becomes the queueing delay experienced by packets using that link, which, when added to a constant propagation delay, is the feedback that is used in the Vegas algorithm.

Let us study the source rates used in Vegas more closely to create a fluid model equivalent. From the algorithm description (with  $\alpha = \beta$ ) we have that the Vegas algorithm updates the window  $W_r(t)$  based on whether

$$\frac{W_r(t)}{T_{pr}} - \frac{W_r(t)}{T_{pr} + T_{qr}(t)} < \alpha \quad \text{or} \quad \frac{W_r(t)}{T_{pr}} - \frac{W_r(t)}{T_{pr} + T_{qr}(t)} > \alpha, \quad (4.10)$$

Using the approximation  $x_r(t) = \frac{W_r(t)}{T_{pr} + T_{qr}(t)}$ , we can rewrite the conditions as

$$x_r(t) T_{qr}(t) < \alpha T_{pr} \quad \text{or} \quad x_r(t) T_{qr}(t) > \alpha T_{pr}. \quad (4.11)$$

As in (4.6), we also have

$$T_{qr} = \sum_{l:l \in r} \frac{b_l(t)}{c_l} = \sum_{l:l \in r} p_l(t), \quad (4.12)$$

where  $b_l(t)$  is the queue length at link  $l$  at time  $t$ , and we have used  $p_l(t)$  to denote the queueing delay at link  $l$ , which acts as the price

function for Vegas. Combining the above expressions, the condition for increase/decrease becomes

$$x_r(t) \sum_{l:l \in r} p_l(t) < \alpha T_{pr} \quad \text{or} \quad x_r(t) \sum_{l:l \in r} p_l(t) > \alpha T_{pr}, \quad (4.13)$$

So the window control algorithm can be written as

$$W_r(t+1) = \left[ W_r(t) + \frac{1}{T_{pr} + T_{qr}(t)} \operatorname{sgn}(\alpha T_{pr} - x_r(t) T_{qr}) \right]_{W_r}^+, \quad (4.14)$$

where  $\operatorname{sgn}(z) = -1$  if  $z < 0$  and  $\operatorname{sgn}(z) = 1$  if  $z > 0$ . Thus, we can now write down the differential equations describing the TCP-Vegas algorithm as

$$\dot{p}_l(t) = \frac{1}{c_l} (y_l - c_l)_{p_l}^+ \quad (4.15)$$

$$\dot{W}_r(t) = \left[ \frac{1}{T_{pr} + T_{qr}(t)} \operatorname{sgn}(\alpha T_{pr} - x_r(t) T_{qr}(t)) \right]_{W_r}^+ \quad (4.16)$$

$$x_r(t) = \frac{W_r(t)}{T_{pr} + T_{qr}(t)}, \quad (4.17)$$

with  $T_{qr}(t) = \sum_{l:l \in r} p_l(t)$ . The above is not the same as the dual algorithm that we derived in the previous chapter. However, the price update dynamics are the same as the price update for the dual algorithm. Further, at the source, by attempting to increase or decrease the rate based on whether  $x_r$  is less than or greater than  $\alpha T_{pr}/T_{qr}$ , it is clear the source attempts to drive the system towards

$$x_r = \frac{\alpha T_{pr}}{T_{qr}} = \frac{\alpha T_{pr}}{\sum_{l \in r} p_l},$$

which is the desired source behavior for a dual congestion controller. Thus, one can interpret TCP-Vegas as an algorithm that approximates the dual congestion control algorithm.

A modification of TCP-Vegas called FAST-TCP has been suggested for very high-speed networks. In FAST-TCP, the window size is increased or decreased depending upon how far the window size is from

a desired equilibrium point. The fluid model describing the protocol is

$$\dot{p}_l(t) = \frac{1}{c_l} (y_l - c_l)_{p_l}^+ \quad (4.18)$$

$$\dot{W}_r(t) = \gamma_r (\alpha_r - x_r(t) T_{qr})_{W_r}^+ \quad (4.19)$$

$$x_r(t) = \frac{W_r(t)}{T_{pr} + T_{qr}(t)}, \quad (4.20)$$

where  $\alpha_r$  determines the desired equilibrium point and  $\gamma_r$  is a scaling constant. Replacing the sgn function in TCP-Vegas with the difference between the current operating point and the desired equilibrium allows FAST-TCP to rapidly approach the desired equilibrium point.

## 4.7 Notes

In this chapter we studied how Internet congestion control protocols can be viewed as decentralized resource allocation algorithms. TCP-Reno, NewReno and SACK are improvements on TCP-Tahoe proposed by Jacobson [1]. Details of the protocols studied are available in [49–51]. The differential equation for TCP without time delays was introduced by Kunniyur and Srikant in [31, 52] who also noted the connection to minimum potential delay fairness while a delay differential equation model was introduced by Misra et. al. in [53]. The choice of the arctan utility function for TCP was proposed by Kelly [9]. The differential equations models are approximations of stochastic, discrete models and the regimes under which a differential equation approximation hold have been explored in Baccelli et. al. [54], Shakkottai and Srikant [55, 56] and Tinnakornsriruphap and Makowski [57]. The model for Scalable TCP was proposed by Vinnicombe [58] and implemented by Tom Kelly [59]. TCP-Vegas was proposed by Bramko and Peterson in [60], while the analytical study was presented by Low et. al in [61]. A comparison of the throughput that TCP-Vegas and TCP-Reno sources achieve in a network may be found in [62] by Mo et. al. A description of how Vegas was modified to create FAST-TCP was presented by Wei et. al. in [63], and implementation details of FAST TCP are provided in [64]. The fluid models for TCP capture the behavior of TCP reasonably well in large networks with a large number of users. However, in some instances, one needs more detailed modeling of the window size evolution of TCP.

Models to capture the window size behavior in more detail can be found in the works of Baccelli and Hong [65], Wirth et al [66] and Kherani et al. [67]. Some of these models present properties of TCP resource sharing that are not always observable using the coarser differential equation models. Examples of protocols that are designed using these finer models are H-TCP [68] and TCP-Illinois [69].

One aspect of resource allocation algorithms that we have not considered so far is the impact of feedback delay on their convergence. Another aspect to be considered while studying real-life networks is the fact that the flows are not persistent. In fact, a user arrives with a file to be transferred over the network and departs when the file transfer is complete. We have not yet considered the question of whether the system would be stable in such a scenario. In the next chapter, we expand our models to incorporate delays and file arrivals and departures.



# 5

---

## Network Stability

---

In the previous chapters, we have studied networks with a fixed number of flows and assumed that packets traverse from one node to the next instantaneously. An exception was our discussion on TCP modeling, where we explicitly incorporated delay in the model. However, we did not discuss the impact of delay on network stability. In this chapter, we will explicitly consider the impact of delays and flow arrivals and departures on network stability. The two effects that we will study are shown in Figure 5.1. We motivate the two stability issues below, and consider their impact on network stability in the sections to follow.

In most of the models used in the previous chapters, flows receive feedback instantaneously. However, in real life there is always a delay between sending a packet and receiving the corresponding acknowledgment. We call this the Round Trip Time (RTT). The delay consists of both queueing and propagation delays. For example, the coast-to-coast distance of the United States is about 5000 kilometers. Since the speed of light in optical fiber is  $1.5 \times 10^8$  m/s, the round trip propagation delay is about 60 msec. Propagation delays can get even larger when packets have to traverse the globe. The queueing delay is a function of the load on the system, and with good design can be kept within



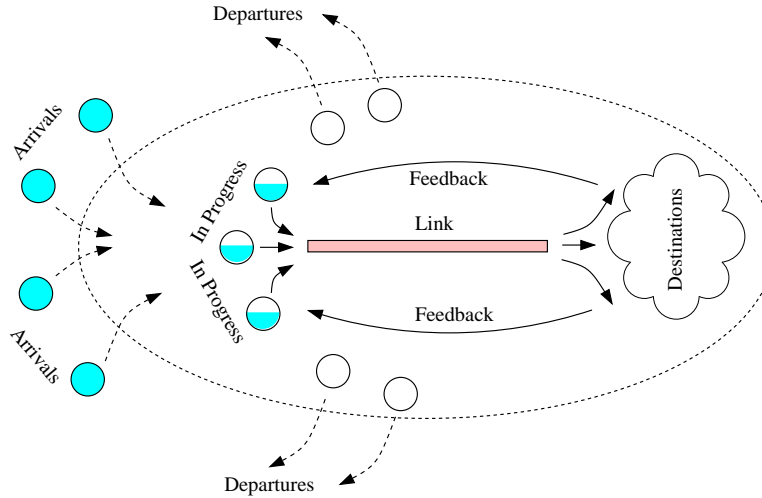


Fig. 5.1 Two separate phenomena in a network. Flows in progress compete with each other for bandwidth and keep the link filled. If their transfer rates converge, we say that the system is stable at a small time scale. Flows arrive bringing a file of fixed size to be transferred and leave when the transfer is complete. We say that the system is stable at a large time scale if the number of flows in progress is finite at all times.

limits. The RTT is considerably larger than the amount of time that it takes for a high-speed router to process a packet. For example, the size of an Internet packet is approximately 10,000 bits and when it goes through a 1 Gbps router, the processing time is  $10^5/10^9$  secs. or 0.1 msecs. Thus, one has to worry about the impact of a source reacting to congestion that happened some time ago. In particular, one has to make sure that the system parameters are designed to ensure the stability of a system of delay-differential equations.

Another effect that we need to include in our model is the fact that the flows in the system do not stay forever. Flows arrive with a file to be transferred, and leave once the transfer is complete. So if, on average, there were 2 flows arriving per second, each with a file transfer of size 10 MB, then the average load on the system would be 160 Mbps. If there is only one link in the network and the link's capacity is larger than 160 Mbps, then most reasonable resource allocation algorithms will ensure that the number of files in the network is bounded in some appropriate

stochastic sense. On the other hand, if the network comprises of many links and different flows have different sets of links on their route, then the interaction among the flows could be complex and the allocation of resources in one part of the network would have an impact on flows in another part of the network. In this case, the stability of the network is not obvious even if the total load on each link is less than its capacity. We will show that, under reasonable assumptions, the network is stable if one uses the optimization approach to resource allocation.

We note that there are two forms of stability that we consider in this chapter and these are quite different:

- (1) We first consider a fixed number of flows which exercise congestion control, but where the congestion feedback is obtained by the sources with some delay. In this case, we say that the system is stable if the set of arrival rates converges to the solution of the network utility maximization problem.
- (2) Next, we consider flow arrivals and departures and assume that the network uses the network utility maximization framework to allocate the resources. In this case, we say that the system is stable if the number of flows in the network stays bounded when the traffic load on each link is less than its capacity.

Let us first study the stability of the resource allocation algorithm in the presence of feedback delays.

## 5.1 Stability of the Primal Algorithm with Delays

We consider the primal algorithm with multi-path routing as we did in Section 3.3. We recall the notation introduced earlier for the reader's convenience. As before, let the set of links be  $\mathcal{L}$  and the set of sources be  $\mathcal{S}$ . Each source  $s \in \mathcal{S}$  identifies a unique source-destination pair. Also, each source has a collection of routes that it may use. If source  $s$  transmits on route  $r$ , we say  $r \in s$ . Similarly, if a route  $r$  uses link  $j$ , we say  $j \in r$ . For each route  $r$  we let  $s(r)$  be the unique source such that  $r \in s(r)$ . We call the set of routes  $\mathcal{R}$ . Note that two paths in the network that are otherwise identical can be associated with two

different indices  $r \in \mathcal{R}$ , if their sources are different, i.e., two sources that have the same origin and destination would have different indices for their routes, even if the physical paths they use are the same.

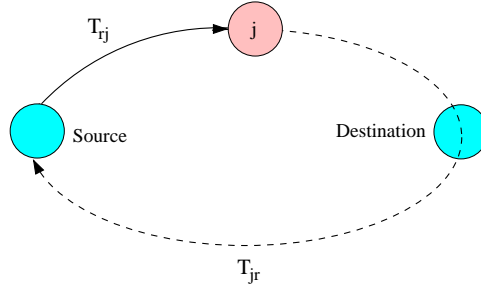


Fig. 5.2 The RTT is composed of two elements—the delay for a packet to get to an intermediate link, and the delay for feedback from that link back to the source.

In what follows, we will assume that queueing delays are negligible. We can also think of the queueing delays as constant, i.e., the sum of queueing and propagation delay is counted as the feedback delay. While it can be argued that the queueing delay is not constant with time, if we ensure that it remains bounded, we can take the maximum delay as the worst-case situation that the controller faces. This is only an approximation but it provides good qualitative insight into the design of congestion control algorithms. We let  $T_{rj}$  be the amount of time that it takes for a packet from source  $r$  to reach link  $j$ . We denote the feedback delay from link  $j$  to source  $r$  by  $T_{jr}$ . An illustration of these two delays is provided in Figure 5.2. The sum  $T_{rj} + T_{jr} = T_r$ , which is the RTT of the route  $r$ . Note that  $T_r = T_{rj} + T_{jr} \forall j$ .

The problem of deriving conditions for global asymptotic stability for this scenario is not straightforward and, at the time of writing this monograph, is still an open problem. We will linearize the system around its equilibrium point and obtain conditions for local asymptotic stability. Simulations suggest that the linearized stability conditions serve as good guidelines for designing stable congestion controllers. Once we linearize a system, we can bring into force tools from linear control theory in order to analyze it. In particular, we can use Laplace transforms and study the system in the complex domain. We will make

use of the Nyquist stability criterion which provides conditions for the stability of linear systems. In the next subsection, we will provide a brief introduction to the Nyquist criterion and use it to study our network in the following subsection.

### 5.1.1 Nyquist Stability Criterion

The stability of a linear system can be determined by studying the poles of the transfer function associated with it. If the poles lie in the left half of the complex plane, then the system is stable. While it is possible to compute the poles if the transfer function has an analytically tractable form, one has to resort to other tools to verify stability indirectly without calculating the poles. The mapping theorem (see [70]) given below is one such tool.

---

**Theorem 5.1.** Consider a Laplace transform  $G(s)$ . Let  $C$  be a closed contour in the complex plane such that no poles or zeroes of  $G(s)$  lie on  $C$ . Let  $Z$  and  $P$  denote the number of poles and zeroes, respectively, of  $G(s)$  that lie inside  $C$ . Then, the number of times that  $G(s)$  encircles the origin in the clockwise direction, as  $s$  traces out the entire contour  $C$  in the clockwise direction, is equal to  $Z - P$ .

---

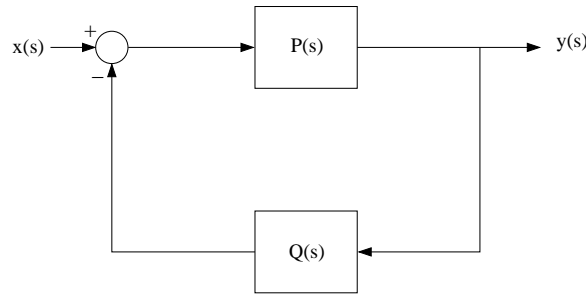


Fig. 5.3 A negative feedback loop. The return ratio is  $P(s)Q(s)$ .

Suppose we have an input-output system as shown in Figure 5.3. We would like to know if the system is stable. It is straightforward to

show that the transfer function  $L(s)$  of the system after taking Laplace transforms is

$$\frac{y(s)}{x(s)} = \frac{P(s)}{1 + P(s) Q(s)}. \quad (5.1)$$

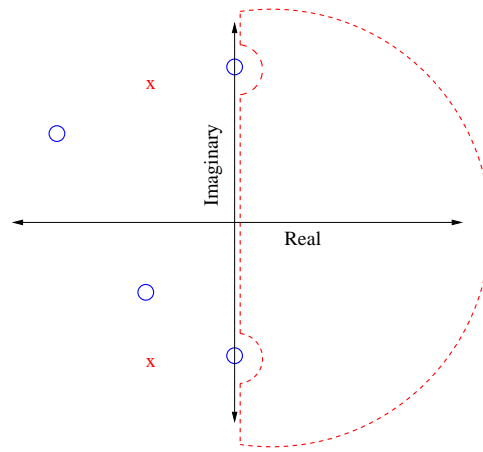


Fig. 5.4 Nyquist stability criterion. Zeroes are shown by an 'o' and poles by an 'x'. The zeroes of  $1 + P(s) Q(s)$  must lie in the left half plane. Also shown is the Nyquist contour used, which spans the right half of the plane with infinitesimal detours to take into account the zeroes on the imaginary axis.

Stability would be assured if the poles of the system were in the left half of the complex plane. Assuming that there are no pole-zero cancelations in the numerator and denominator of  $L(s)$ , the poles of  $L(s)$  are the roots of

$$1 + P(s)Q(s) = 0.$$

Thus, instead of checking if the poles of  $L(s)$  lies in the left-half plane, we can check if the zeroes of  $G(s) = 1 + P(s)Q(s)$  lie in the left-half plane. To apply the mapping theorem, consider a contour shown in Figure 5.5. The contour traverses the imaginary axis from  $-i\infty$  to  $+i\infty$  and maps out the semi-circle at  $\infty$ . Since we are not allowed to have any poles or zeroes of  $G(s)$  on the contour itself, if there are any poles of  $G(s)$  on the imaginary axis, we use infinitesimally small semi-circles to go around them as shown in Figure 5.5.

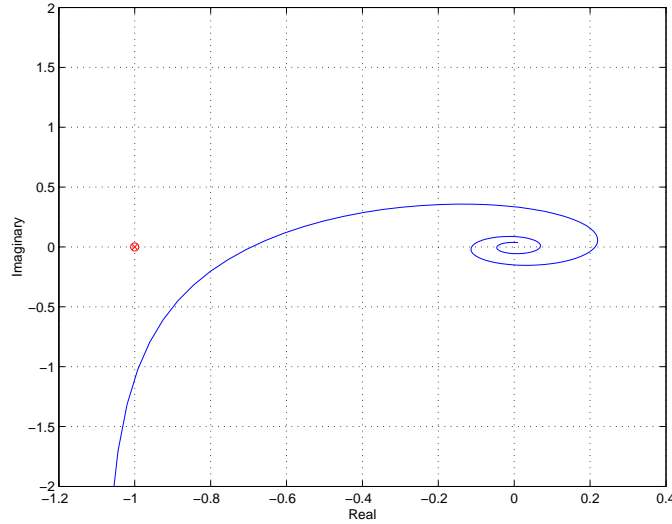


Fig. 5.5 A plot of the function  $\frac{T e^{-i\theta}}{i\theta(i\theta + T)}$  with  $T = 10$  for  $\theta \in [0, \infty]$ . The point  $-1 + i0$  is also shown. We see that the function has no zeroes to the left of this point.

A special case that is often encountered is one in which  $G(s)$  has no poles in the right-half of the plane. In this case, in terms of the Theorem 5.1, we want  $Z = 0$ , where  $Z$  is the number of zeroes of  $1 + P(s)Q(s)$  in the complex right-half plane. Applying the mapping theorem, the stability of the system is then equivalent to the condition that there are no encirclements of the origin by the plot of  $1 + P(i\theta)Q(i\theta)$  as  $\theta$  is varied from  $-\infty$  to  $\infty$ . Also notice that saying  $1 + P(i\theta)Q(i\theta)$  does not encircle origin, is equivalent to saying that  $P(i\theta)Q(i\theta)$  does not encircle  $-1 + i0$ . The product of  $P(s)$  and  $Q(s)$  is called the *return ratio*. If our original feedback loop given in (5.1) controller is stable, then the plot of its return ratio might look like Figure 5.5. Hence, given a return ratio function if we know that it has no poles in the right half plane, if its locus is such that all intersections with the real axis are to the right of  $-1 + i0$ , the system would be stable. In the case of multiple-input, multiple-output systems, such as a communication network, we have to consider the eigen-loci of the return ratio matrix as shown in [71]. We will do so in the next subsection to derive

linear stability conditions for the primal congestion control algorithms. The application of the mapping theorem to test for the stability of a linear system is called the Nyquist criterion.

### 5.1.2 Network Stability

We now have the tools required to tackle the question of multi-path linearized stability. The controller that we use in this section is as follows:

$$\frac{d}{dt}x_r(t) = \kappa_r x_r(t) \left( 1 - \frac{q_r(t)}{U'_{s(r)}(z_{s(r)}(t))} \right)_{x_r(t)}^+ \quad (5.2)$$

where

$$q_r(t) = \sum_{j \in r} p_j(t - T_r) \quad (5.3)$$

and

$$p_j(t) = f_j(y_j(t)), \quad y_j(t) = \sum_{r: j \in r} x_r(t - T_r) \quad (5.4)$$

and

$$z_s(t) = \sum_{r \in s} x_r(t - T_r) \quad (5.5)$$

The controller is slightly different from the one we saw in Section 3.3. In the absence of delay, it is easy to see that it is a gradient algorithm to maximize the following objective:

$$\mathcal{U}(x) = \sum_s w_s U_s \left( \sum_{r \in s} x_r \right) - \sum_l \int_0^{\sum_{k: l \in k} x_k} f_l(y) dy. \quad (5.6)$$

Even if  $U_s(\cdot)$  is strictly concave and  $f_l$  is a non-decreasing function, the above objective function need not be strictly concave in  $x$ . Thus, it may not have a unique maximum. On the other hand, the formulation in Section 3.3 would ensure strict concavity, but the resulting stability conditions are more complicated. So we assume the formulation in (5.6) but assume that the maximum is unique. We also assume that the optimal rates are non-zero along all paths.

Notice that (5.5) implies that the source must keep the rates it used one RTT ago in its memory. This can be easily implemented by putting the current rate in the header of each packet and echoing this rate back in the acknowledgments. Thus, by reading off the rate in the ack, a source knows that rate that it used on a particular path one RTT ago.

To motivate the stability result, consider a differential equation

$$\frac{dx}{dt} = f(x),$$

and assume that it is stable. Now, consider a delay-differential equation

$$\frac{dz}{dt} = \kappa f(z(t - T)).$$

Rescale time as  $\tau = Kt$  and define  $y(\tau) = z(\tau/K)$ . Straightforward differentiation using the fact that

$$\frac{dy(\tau)}{d\tau} = \frac{dz(t)}{dt} \frac{dt}{d\tau}$$

yields the following delay-differential equation for  $y$  in the  $\tau$  timescale:

$$\frac{dy}{d\tau} = f(y(\tau - \kappa T)).$$

If  $\kappa T = 0$ , then this is the same as the differential equation for  $x$ . Thus, it seems reasonable to expect the delay-differential equation to be stable if  $\kappa T$  is small. We will show that one can indeed derive such a stability condition for the network. Specifically, it will be shown that the following conditions are sufficient for linear stability:

$$\frac{\kappa_r T_r}{\hat{q}_r} \left( -\hat{U}''_{s(r)} \hat{z}_{s(r)} + \sum_{j \in r} \hat{y}_j \hat{f}'_j \right) < \frac{\pi}{2}, \quad (5.7)$$

where we have used the notation  $\hat{\cdot}$  to indicate equilibrium points.

We will prove the result by first linearizing the controller around its equilibrium, taking Laplace transforms and studying the spectrum of the return ratio in the complex domain using the Nyquist condition. The equilibrium condition of the controller is given by

$$\hat{U}'_{s(r)} \left( \sum_{r: r \in s} \hat{x}_r \right) = \sum \hat{f}_j \left( \sum_{a: j \in a} \hat{x}_a \right) \quad (5.8)$$



for each  $r$ . Let  $x_r(t) = \hat{x}_r + u_r(t)$ ,  $z_s(t) = \hat{z}_s + v_s(t)$ , and  $y_j(t) = \hat{y}_j + w_j(t)$ . Then linearizing the controller about the equilibrium yields

$$\frac{d}{dt}u_r(t) = -\frac{\kappa_r \hat{x}_r}{\hat{q}_r} \left( -\hat{U}_{s(r)}'' v_{s(t)}(t) + \sum_{j \in r} \hat{f}_j' w_j(t - T_{jr}) \right), \quad (5.9)$$

$$v_s(t) = \sum_{r:r \in s} u_r(t - T_r), \quad (5.10)$$

$$w_j(t) = \sum_{r:j \in r} u_r(t - T_j). \quad (5.11)$$

We now proceed to take Laplace transforms of the above linear system. The fact that we are now working in the complex domain is shown by  $t$  being replaced by  $\tilde{s}$  for all the variables. We use  $\tilde{s}$  to represent the Laplace domain variable to avoid any confusion with the source variable  $s$ .

$$\tilde{s}u_r(\tilde{s}) = -\frac{\kappa_r \hat{x}_r}{q_r} \left( -\hat{U}_{s(r)}'' v_s(\tilde{s}) + \sum_{j \in r} \hat{f}_j' e^{-\tilde{s}T_{jr}} w_j(\tilde{s}) \right), \quad (5.12)$$

$$v_s(\tilde{s}) = \sum_{r:r \in s} e^{-\tilde{s}T_r} u_r(\tilde{s}), \quad (5.13)$$

$$w_j(\tilde{s}) = \sum_{r:j \in r} e^{-\tilde{s}T_{rj}} u_r(\tilde{s}). \quad (5.14)$$

The above equations can be compactly written as

$$\begin{pmatrix} v(\tilde{s}) \\ w(\tilde{s}) \end{pmatrix} = F^{-1} R(-\tilde{s})^T X(\tilde{s}) R(\tilde{s}) F \begin{pmatrix} v(\tilde{s}) \\ w(\tilde{s}) \end{pmatrix}, \quad (5.15)$$

where  $X(w)$  is an  $|\mathcal{R}| \times |\mathcal{R}|$  diagonal matrix, with entries  $X_{rr}(\tilde{s}) = e^{-\tilde{s}T_r}/\tilde{s}T_r$  and  $F$  is a  $|\mathcal{S}| + |\mathcal{L}| \times |\mathcal{S}| + |\mathcal{L}|$  diagonal matrix whose entries are  $F_{ss} = 1$ ,  $F_{jj} = (\hat{f}_j')^{\frac{1}{2}}$ , and  $R(\tilde{s})$  is a  $|\mathcal{R}| \times (|\mathcal{S}| + |\mathcal{L}|)$  matrix with

$$R_{rs}(\tilde{s}) = \left( -U_{s(r)}'' T_r \frac{\kappa_r \hat{x}_r}{\hat{q}_r} \right)^{\frac{1}{2}}, \quad r \in s$$

$$R_{rj}(\tilde{s}) = e^{-\tilde{s}T_{jr}} \left( \frac{\kappa_r \hat{x}_r}{\hat{q}_r} T_r \hat{f}_j' \right)^{\frac{1}{2}}, \quad j \in r$$

and all other entries are 0.

The matrix  $F^{-1}R(-\tilde{s})^T X(\tilde{s})R(\tilde{s})F$  that appears in (5.15) is the return ratio of the controller, and we would like to study the eigenvalues of this matrix to see if they lie to the right of  $-1 + i0$ . Let  $\lambda$  be an eigenvalue of the above matrix. Then we have

$$\lambda y = R(i\theta)^* X(i\theta) R(i\theta) y,$$

where  $\star$  is used to indicate the matrix conjugate transpose. Thus we have

$$\lambda = y^* R(i\theta)^* X(i\theta) R(i\theta) y.$$

If  $d = R(i\theta)y$ , then since  $X$  is a diagonal matrix

$$\lambda = \sum_r |d_r|^2 X_{rr}(i\theta) = \sum_r |d_r|^2 \frac{e^{-i\theta T_r}}{i\theta T_r}.$$

So we can write  $\lambda = K\zeta$ , where  $K = \|R(i\theta)y\|^2$  and  $\zeta$  lies in the convex hull of

$$\left\{ \frac{e^{-i\theta T_r}}{i\theta T_r} : r \in s, s \in \mathcal{S} \right\}.$$

Now, as shown in Figure 5.6, the convex hull includes the point  $-2/\pi$  on its boundary, but has no point to the left of this on the real axis. This means that if  $\lambda$  is real, then  $\lambda \geq (-2/\pi)K$ .

It remains now to find a bound on  $K$ . Let  $P$  be the  $|\mathcal{S}| + |\mathcal{L}| \times |\mathcal{S}| + |\mathcal{L}|$  diagonal matrix taking the values  $P_{ss} = z_s \sqrt{-\hat{U}''}$  and  $P_{jj} = y_j \sqrt{\hat{f}_j'}$ . We let  $\rho(\cdot)$  denote the spectral radius and  $\|\cdot\|_\infty$  the maximum row sum matrix norm. So we now have

$$\begin{aligned} K &= y^* R(i\theta)^* R(i\theta) y \\ &\leq \rho(R(i\theta)^* R(i\theta)) \\ &= \rho(P^{-1} R(i\theta)^* R(i\theta) P) \\ &\leq \|P^{-1} R(i\theta)^* R(i\theta) P\|_\infty \\ &< \frac{\pi}{2}, \end{aligned}$$

where the last inequality follows from our choice made in (5.7). This means that any real eigenvalue  $\lambda$  must lie to the right of  $-1$  and the Nyquist condition, as outlined earlier is satisfied. Thus, the multi-path controller is stable around its equilibrium point with our choice of  $\kappa_r$ .

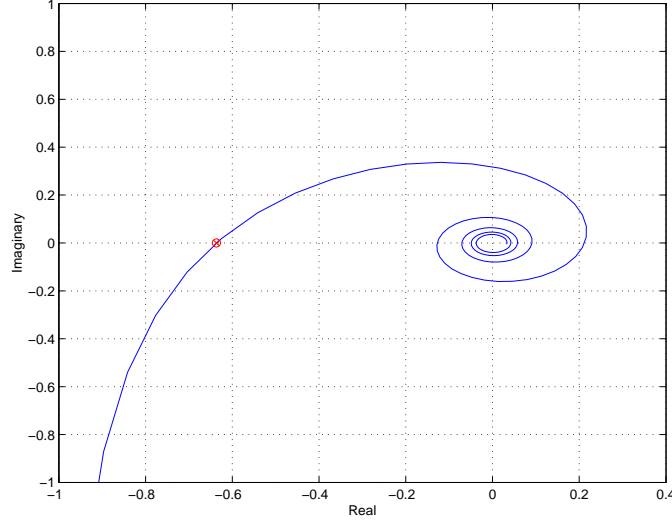


Fig. 5.6 A plot of the function  $\frac{\theta e^{-i\theta T_r}}{i\theta T_r}$  with  $T_r = 2$  for  $\theta \in [0, \infty]$ . The function has no zeroes to the left of  $(-\frac{2}{\pi}, 0)$ .

## 5.2 Stability Under Flow Arrivals and Departures

We will now consider stability under file arrivals and departures. We will consider single-path routing only, not multi-path routing as in the previous section although the results can be extended more generally. Assume that files (also known as connections) for route  $r$  arrive according to a Poisson process of rate  $\lambda_r$  and a file of type  $r$ . File sizes are independently and exponentially distributed, with the mean file size of a type  $r$  file denoted by  $1/\mu_r$ . Each file type  $r$  is associated with a collection of links known as its route, and we use the notation  $l \in r$  to indicate that a link belongs to the route of a file of type  $r$ . A file of type  $r$  generates data packets according to a (time-inhomogeneous) Poisson process of rate  $x_r(t)$ , where  $x_r(t)$  depends on the algorithm used by the network to allocate resources. To facilitate a simple Markov chain description, we assume that packet sizes are exponentially distributed with mean 1. The capacity of link  $l$  in the network is denoted by  $c_l$  and

we assume the following stability condition:

$$\sum_{r:l \in r} \frac{\lambda_r}{\mu_r} < c_l, \quad \forall l, \quad (5.16)$$

i.e., we assume that the load on each link is less than the link capacity. The goal is to show that the number of files in the network stays bounded. The reason for providing the reader with a stochastic description of the system is to let the reader know the model for which the results of this section applies. However, in keeping with the spirit of the rest of this monograph, we ignore the stochastic dynamics of the system and consider a heuristic deterministic differential equation description of the system and use a Lyapunov function method to show the boundedness of this system. In the next section, we will refer the interested reader to papers that provide the details of the proof of stochastic stability.

At each time instant  $t$ , there are  $n_r$  users on route  $r$ . We assume that all users in the network use the  $\alpha$ -fair utility function, with possibility different weights for different routes, i.e.,

$$U_r(x_r) = w_r \frac{x_r^{1-\alpha}}{1-\alpha}.$$

At each time instant, the network attempts to solve the utility maximization problem using the dual algorithm described in Section 3.2. Accordingly, the transmission rate  $x_r$  of a type  $r$  source, satisfies

$$\frac{w_r}{x_r^\alpha} = q_r,$$

where  $q_r$  is price of route  $r$  and is given by

$$q_r = \sum_{l:l \in r} p_l.$$

To ensure boundedness of  $x_r$  when  $q_r = 0$ , we modify the rate-update equation as

$$x_r = \max\left\{\left(\frac{w_r}{q_r}\right)^{1/\alpha}, M\right\},$$

where  $M$  is some large constant. The price  $p_l$  is computed using

$$\dot{p}_l = (y_l - c_l)_{p_l}^+, \quad (5.17)$$

where the arrival rate at each link is given by

$$y_l = \sum_{s:l \in s} n_s x_s.$$

Now, we describe the dynamics of the number of files in the network. Files on route  $r$  arrive at rate  $r$ . Since the average file size is  $1/\mu_r$  and  $x_r$  is the total rate assigned to a user on route  $r$ , the instantaneous rate at which a file is processed is  $x_r \mu_r$ . Since there are  $n_r$  files on route  $r$ , the total rate at which they are processed is  $\mu_r n_r x_r$ . Thus, we get the following differential equation for  $n_r(t)$  :

$$\dot{n}_r = \lambda_r - \mu_r n_r x_r. \quad (5.18)$$

Consider the Lyapunov function

$$V(n, p) = \frac{1}{1 + \alpha} \sum_r K_{1r} n_r^{1+\alpha} + \frac{1}{2} \sum_l K_{2l} p_l^2,$$

where the constants  $K_{1r}$  and  $K_{2l}$  will be chosen later. Note that

$$\frac{dV}{dt} = \sum_r K_{1r} n_r^\alpha (\lambda_r - \mu_r n_r x_r) + \sum_l K_{2l} p_l (y_l - c_l),$$

where we have used the fact  $p_l(y_l - c_l) = p_l(y_l - c_l)_{p_l}^+$ . From the stability condition, there exists an  $\epsilon > 0$  such that

$$\nu_l(1 + \epsilon) \leq c_l,$$

where

$$\nu_l = \sum_{s:l \in s} \rho_s, \quad \rho_s = \frac{\lambda_s}{\mu_s}.$$

Choose such an  $\epsilon$ , define  $\delta = \epsilon/2$  and rewrite  $\dot{V}$  as follows:

$$\begin{aligned}
\dot{V} &= -\delta \sum_r K_{1r} n_r^\alpha \lambda_r + \sum_r K_{1r} n_r^\alpha \mu_r (\rho_r(1+\delta) - n_r x_r) \\
&\quad + \sum_l K_{2l} p_l (y_l - c_l) \\
&= -\delta \sum_r K_{1r} n_r^\alpha \lambda_r \\
&\quad + \sum_r K_{1r} \mu_r n_r (\rho_r(1+\delta))^\alpha \left( \frac{1}{(\rho_r(1+\delta)/n_r)^\alpha} - \frac{1}{x_r^\alpha} \right) \\
&\quad \quad \times \left( \frac{\rho_r(1+\delta)}{n_r} - x_r \right) I_{n_r > 0} \\
&\quad + \sum_r K_{1r} \mu_r n_r \frac{(\rho_r(1+\delta))^\alpha}{x_r^\alpha} \left( \frac{\rho_r(1+\delta)}{n_r} - x_r \right) I_{n_r > 0} \\
&\quad + \sum_l K_{2l} p_l (y_l - c_l) \\
&\leq -\delta \sum_r K_{1r} n_r^\alpha \lambda_r + \sum_r K_{1r} \mu_r \frac{(\rho_r(1+\delta))^\alpha q_r}{w_r} (\rho_r(1+\delta) - n_r x_r) \\
&\quad + \sum_l K_{2l} p_l (y_l - c_l) + \sum_r K_{1r} \mu_r \frac{(\rho_r(1+\delta))^\alpha}{M^\alpha} \rho_r(1+\delta),
\end{aligned}$$

where the last inequality follows from the fact that  $1/x_r^\alpha$  is a decreasing function and  $q_r = w_r/x_r^\alpha$  if  $q_r \neq 0$  and  $x_r = M$  if  $q_r = 0$ . Now, choose  $K_{2l} = 1$  and  $K_{1r} \mu_r (\rho_r(1+\delta))^\alpha / w_r = 1$ . Then,

$$\dot{V} \leq -\delta \sum_r K_{1r} n_r^\alpha \lambda_r + \sum_r q_r (\rho_r(1+\delta) - n_r x_r) + \sum_l p_l (y_l - c_l) + K_3,$$

where  $K_3 = \sum_r K_{1r} \mu_r \frac{(\rho_r(1+\delta))^\alpha}{M^\alpha} \rho_r(1+\delta)$ . Noting that

$$\sum_r q_r \rho_r = \sum_l p_l \nu_l \text{ and } \sum_r q_r n_r x_r = \sum_l p_l y_l,$$

we have

$$\dot{V} \leq -\delta \sum_r K_{1r} n_r^\alpha \lambda_r - \delta \sum_l p_l \nu_l + K_3.$$

From the above equation, it is clear that we can find a bounded set  $B$  such that if  $(n, p) \in B$ ,  $V(t)$  decreases. Thus,  $(n, p)$  will be ultimately

bounded. One can translate this conclusion into an appropriate notion of stochastic boundedness.

### 5.3 Notes

Linear stability results for the primal controller were obtained by Vinicombe [58, 72], for the dual controller by Paganini et. al. [73], and for the primal-dual-type controllers by Kunniyur and Srikant [74] and by Liu et. al. [75]. Stability conditions for the primal multi-path controller were obtained by Han et. al. in [28] and by Kelly and Voice in [76]. The results in this chapter follow the derivation by Kelly and Voice. Global stability results for congestion controllers are available only for certain limited scenarios and are not treated in this chapter. For the interested readers, we provide some references here. Conditions for single-link stability were derived by Hollot and Chait for TCP-type controllers [77], by Deb and Srikant for proportionally-fair primal controllers [78], and by Wang and Paganini for dual controllers [79]. Global stability for a network of primal controllers were derived by Ying et. al. in [80–82] and for dual controllers by Papachristodoulou [83]. Independent of the utility maximization framework, stability results for delay-differential equation models of TCP and AQM schemes were obtained by Misra et. al. [53], Hollot et. al. [84] and Kunniyur and Srikant [85].

The flow-level model for congestion controllers for a fixed number of sources, but where the sources alternate between ON and OFF states were introduced by Heyman et. al. [86]. A single link version of the model considered in this chapter, but where the congestion control is assumed to operate instantaneously was considered by Das and Srikant [87]. The general network assuming congestion control operates instantaneously compared to the time-scale of file arrivals and departures was introduced by Roberts and Massoulié in [88]. The stability of this network was proved by de Veciana et. al. in [89] for max-min congestion controllers and by Bonald and Massoulié in [90]. A stochastic version of the stability proof of [90] is presented in [91] using an extension of Lyapunov’s theorem for Markov chains (see [92], for example). The stability of a model without the time-scale separation between flow dynamics and congestion control dynamics was shown by Lin, Shroff

and Srikant [93–95]. All of these results assume that the file sizes are exponentially distributed. This assumption has been relaxed by Massoulié [96] and Bramson [97] for proportionally-fair and max-min controllers under the time-scale separation. For general  $\alpha$ -fair controllers, partial stability results for a fraction of the possible capacity region have been reported by Chiang et. al. [98].

We now move away from the question of network stability and present a different take on network utility maximization. In the next chapter, we consider a network of rational, selfish users who are capable of modifying their protocols to some extent, and study their impact on system performance.





# 6

---

## Game Theory and Resource Allocation

---

In this chapter, we consider an alternative interpretation of the utility maximization problem that we considered earlier in Chapters 2 and 3. Consider a network planner who is interested in allocating resources to users with the goal of maximizing the sum of the user's utilities. The network planner can do this only if he knows the utility functions of all the users in the network, or if there is an incentive for the users to reveal their utility functions truthfully. In this chapter, we will first discuss an incentive mechanism called the Vickrey-Clarke-Groves (VCG) which makes it profitable for the users to reveal their true utilities to the central network planner. However, the amount of information that needs to be conveyed by the users and the amount of computation required on the part of the network planner make it difficult to implement the VCG mechanism. One can design a mechanism based on the resource allocation algorithms in Chapters 2 and 3 under which truth-telling is optimal. We call this the Kelly mechanism. However, this mechanism is truth-revealing only under the assumption that the network is very large so that it is difficult for each user to estimate its impact on the price decided by the network. Users that are unaware of their effect on the price are called *price taking*. On the other hand, in a small network

such as a single link, a user may be able to assess its impact on the network price. In such a case, the user may act *strategically*, i.e., act in such a manner that influences the price to maximize its own benefit. We will review recent results that show that, the inefficiency of the Kelly mechanism with strategic users is bounded by 25%, i.e., the Kelly mechanism loses at the most a factor of 1/4 compared the maximum possible network utility.

### 6.1 VCG Mechanism

Consider the problem of dividing a fixed resource among different users. Consider a network planner who wants to solve the utility maximization considered in Chapter 3, where each user is associated with a route:

$$\max_{x \geq 0} \sum_r U_r(x_r)$$

subject to

$$\sum_{r:l \in r} x_r \leq c_l, \forall l.$$

Here,  $x_r$  is the rate allocated to user  $r$ , who has a utility function given by  $U_r$  and  $c_l$  is the capacity of link  $l$ .

Suppose that the network planner asks each user to reveal their utilities and user  $r$  reveals its utility function as  $\tilde{U}_r(x_r)$ , which may or may not be the same as  $U_r(x_r)$ . Users may choose to lie about their utility function to get a higher rate than they would get by revealing their true utility function. Let suppose that the network solves the maximization problem

$$\max_{x \geq 0} \sum_r \tilde{U}_r(x_r)$$

subject to

$$\sum_{r:l \in r} x_r \leq c_l, \forall l$$

and allocates the resulting optimal solution  $\tilde{x}_r$  to user  $r$ . In return for allocating this rate to user  $r$ , the network charges a certain price  $p_r$ . The price is calculated as follows. The network planner calculates the reduction in the sum of the utilities obtained by other users in the network due to the presence of user  $r$ , and collects this amount as the

price from user  $r$ . Specifically, the network planner first obtains the optimal solution  $\{\bar{x}_s\}$  to the following problem:

$$\max_{x \geq 0} \sum_{s \neq r} \tilde{U}_s(x_s)$$

subject to

$$\sum_{s \neq r: l \in s} x_s \leq c_l, \forall l.$$

In other words, the network planner first solves the utility maximization problem without including user  $r$ . The price  $p_r$  is then computed as

$$p_r = \sum_{s \neq r} \tilde{U}(\bar{x}_s) - \sum_{s \neq r} \tilde{U}(\tilde{x}_s),$$

which is the difference of sum utilities of all other users without  $(\{\bar{x}\})$  and with  $(\{\tilde{x}\})$  the presence of user  $r$ . The network planner announces this mechanism to the users of the network, i.e., the network planner states that once the users reveal their utilities, it will allocate resources by solving the utility maximization problem and will charge a price  $p_r$  to user  $r$ . Now the question for the users is the following: what utility function should user  $r$  announce to maximize its *payoff*? The payoff is the utility minus the price:

$$U_r(\tilde{x}_r) - p_r.$$

We will now see that an optimal strategy for each user is to truthfully reveal its utility function. We will show this by proving that announcing a false utility function can only result in reduction in the payoff for user  $r$ .

Suppose user  $r$  reveals its utility function truthfully, while the other users may or may not. In this case, the payoff for user  $r$  is given by

$$\mathcal{U}^t = U_r(\tilde{x}_r^t) - \left( \sum_{s \neq r} \tilde{U}_s(\bar{x}_s^t) - \sum_{s \neq r} \tilde{U}_s(\tilde{x}_s^t) \right),$$

where  $\{\tilde{x}_s^t\}$  is the allocation given to the users by the network planner and  $\{\bar{x}_s^t\}$  is the solution of the network utility maximization problem when user  $r$  is excluded from the network. The superscript  $t$  indicates

that user  $r$  has revealed its utility function truthfully. Next, suppose that user  $r$  lies about its utility function and denote the network planner's allocation by  $\tilde{x}^l$ . The superscript  $l$  indicates that user  $r$  has lied. Now, the payoff for user  $r$  is given by

$$\mathcal{U}^l = U_r(\tilde{x}_r^l) - \left( \sum_{s \neq r} \tilde{U}_s(\tilde{x}_s^t) - \sum_{s \neq r} \tilde{U}_s(\tilde{x}_s^l) \right).$$

If truth-telling were not optimal,  $\mathcal{U}^l > \mathcal{U}^t$ . If this were true, by comparing the two expressions for  $\mathcal{U}^t$  and  $\mathcal{U}^l$ , we get

$$U_r(\tilde{x}_r^l) + \sum_{s \neq r} \tilde{U}_s(\tilde{x}_s^l) > U_r(\tilde{x}_r^t) + \sum_{s \neq r} \tilde{U}_s(\tilde{x}_s^t),$$

which contradicts the fact that  $\tilde{x}^t$  is the optimal solution to

$$\max_{x \geq 0} U_r(x_r) + \sum_{s \neq r} \tilde{U}_s(x_s)$$

subject to the capacity constraints. Thus, truth-telling is optimal under the VCG mechanism. Note that truth-telling is optimal for user  $r$  independent of the strategies of the other users. A strategy which is optimal for a user independent of the strategies of other users is called a *dominant strategy* in game theory. Thus, truth-telling is a dominant strategy under the VCG mechanism.

In the above discussion, note that  $\bar{x}$  is somewhat irrelevant to the pricing mechanism. One could have chosen any  $\bar{x}$  that is a function of the strategies of all users other than  $r$  in computing the price for user  $r$ , and the result would still hold, i.e., truth-telling would still be a dominant strategy. The reason for this is that the expression for  $\mathcal{U}^t - \mathcal{U}^l$  is independent of  $\bar{x}$  since the computation of  $\bar{x}$  does not use either  $U_r(\cdot)$  or  $\tilde{U}_r(\cdot)$ . Another point to note is that truth-telling is an optimal strategy. Given the strategies of all the other users, there may be other strategies for user  $r$  that are optimal as well. Such user strategies may result in allocations that are not optimal from the point of view of the network planner.

We have now established that truth-telling is optimal under a VCG mechanism. However, the VCG mechanism is not used in networks. The reason for this is two-fold:

- Each user is asked to reveal its utility function. Thus, an entire function has to be revealed by each user which imposes a significant communication complexity in the information exchange required between the users and the network planner.
- The network planner has to solve many maximization problems: one to compute the resource allocation and one for user to compute the user's price. Each of these optimization problems can be computationally quite expensive to solve in a centralized manner.

In the next section, we show how one can design a mechanism using the theory developed in Chapters 2 and 3.

## 6.2 Kelly Mechanism

One simple method to reduce the communication burden required to exchange information between the users and the network planner is to ask the users to submit bids which are amounts that the users are willing to pay for the resource, i.e., the link capacities in the network. We will show that if the users do not anticipate the effects of their bids on the unit price of the resource, then the controllers discussed in Chapter 3 follow as a rational consequence of users trying to maximize their payoffs. In other words, the schemes that we studied are not only optimal, they are also incentive compatible. We refer to the mechanism of submitting bids for resource in the context of network resource allocation as the Kelly mechanism. We will describe the Kelly mechanism only for the case of a single link with capacity  $c$ .

Let the bid of user  $r$  be denoted by  $w_r$ . Given the bids, suppose that the network computes a price per unit amount of the resource as

$$q \triangleq \frac{\sum_k w_k}{c}. \quad (6.1)$$

and allocates an amount of resource  $x_r$  to user  $r$  according to  $x_r = w_r/q$ . This is a weighted proportionally-fair allocation since it is equivalent to maximizing  $\sum_r w_r \log x_r$  subject to the resource constraint.

The payoff that the user obtains is given by

$$U_r \left( \frac{w_r}{q} \right) - w_r \quad (6.2)$$

Since the user is rational, it would try to maximize the payoff. We assume that users are price-taking and hence are unaware of the effect that their bids have on the price per unit resource. As far as they know, the central authority is selling them a resource at a price  $q$ , regardless of what their bid might be.

What would a user bid given that the price per unit resource is  $q$ ? Clearly, the user would try to maximize the payoff and tries to solve the problem

$$\max_{w_r \geq 0} U_r \left( \frac{w_r}{q} \right) - w_r.$$

Assuming, as we did earlier, that  $U_r(x_r) \rightarrow -\infty$  as  $x_r \rightarrow 0$ , the optimal value of  $w_r$  is strictly greater than zero and is given by

$$U'_r \left( \frac{w_r}{q} \right) = q. \quad (6.3)$$

Since we know that  $x_r = w_r/q$ , the above equation can be equivalently written in two other forms:

$$q = U'_r(x_r) \quad (6.4)$$

and

$$w_r = x_r U'_r(x_r). \quad (6.5)$$

Now the utility maximization problem that the network planner wants to solve is given by

$$\max_{x \geq 0} \sum_r U_r(x_r),$$

subject to  $\sum_r x_r \leq c$ . The solution to the problem satisfies the KKT optimality conditions given by

$$U'_r(x_r) = q, \sum_r x_r = c \quad (6.6)$$

The weighted proportionally-fair mechanism ensures that (6.6) is satisfied (we do not allocate more resource than is available). Also, we have

just seen that coupled with rational price taking users, the mechanism results in an allocation that satisfies (6.4), which is identical to (6.6). Thus, there exists a solution to the system of price taking users that we call  $x^T, q^T$  (using  $T$  to denote “taking”) that achieves the desired utility maximization by using the weighted proportionally fair mechanism.

### 6.2.1 Relation to Decentralized Utility Maximization

Now, suppose that the network wants to reduce its computational burden. Then, it can compute the price according to the dual algorithm:

$$\dot{q} = (\sum_r x_r - c)_q^+.$$

Clearly, this is not the optimal. The system merely responds to resource usage by giving back a price, it increases or decreases the price depending on whether the resource is over-utilized or under-utilized, respectively. User  $r$  is then allocated a rate  $w_r/q$  in proportion to its bid  $w_r$ . Given the price, we have already seen that the user  $r$ ’s rational bid  $w_r$  assuming that it is a price-taking user is given by

$$U'_r\left(\frac{w_r}{q}\right) = q,$$

which is the same as  $U'(x_r) = q$ . We have already seen that the dual price-update along with such a user response converges to the optimal solution of the network utility maximization problem. Thus, the decentralized algorithms studied in earlier chapters can be thought of as a decentralized implementation of the Kelly mechanism. Thus, the Lagrange multiplier is a pricing incentive for users to behave in a socially responsible manner assuming that they are price-taking.

## 6.3 Strategic or Price-Anticipating Users

In the price-anticipating paradigm, the users are aware of the effect that their bid has on the price of the resource. In this case the problem faced by the users is a game in which they attempt to maximize their individual payoffs *anticipating* the price change that their bid would



cause. Each user strategically tries to maximize its payoff given by

$$P_r(w_r; w_{-r}) = \begin{cases} U_r\left(\frac{w_r}{\sum_k w_k}c\right) - w_r & \text{if } w_r > 0 \\ U_r(0) & \text{if } w_r = 0, \end{cases} \quad (6.7)$$

where  $w_{-r}$  is the vector of all bids, except  $w_r$ .

Our game is a system wherein there are  $R$  users and each user  $r$  can make a bid  $w_r \in \mathbb{R}^+ \cup \{0\}$ , i.e., the game consists of deciding a non-negative real number. A *strategy* is a rule by which a user would make his bid. In our case, the strategy that users might use could be  $S_r = \text{“set the bid } w_r = w_r^S\text{”}$ , where  $w_r^S \in \mathbb{R}^+ \cup \{0\}$  is some constant. Since the strategy recommends playing the same bid all the time, it is called a *pure strategy*. A *strategy profile* is an element of the product-space of strategy spaces of each user denoted by  $\mathcal{S}$ . We denote the strategy profile corresponding to all users using the strategy of setting their bids based on the vector  $w^G$  by  $S \in \mathcal{S}$ . We would like to know if a particular strategy profile is stable in some sense. For example, we might be interested in knowing if everyone knows everyone else’s strategy, would they want to change in some way. The concept of the *Nash equilibrium* formally defines one kind of stability.

---

**Definition 6.1.** A pure strategy Nash equilibrium is a strategy profile from which no user has a unilateral incentive to change his strategy.

---

Note the term “unilateral”. This means that users do not collude with each other—they are interested solely in their own payoffs. If they find that there is no point changing from the strategy that they are currently using, they would continue to use it and so remain at equilibrium. How do we tell if the strategy profile  $S$  defined above is actually a Nash equilibrium? The answer is to check the payoff obtained by using it. We denote the bid recommended by strategy  $S_r$  to user  $r$  as  $w_r^S$  and similarly the bid recommended by any other strategy  $G$  by  $w_r^G$ . So the users would not unilaterally deviate from the strategy profile  $S$  if

$$P_r(w_r^S; w_{-r}^S) \geq P_r(w_r^G; w_{-r}^S), \quad (6.8)$$

which would imply that the strategy profile  $S$  is a Nash equilibrium. We would like to know if there exists a vector  $w^S$  such that strategy

profile  $S$  that recommends would be a Nash equilibrium, i.e., whether there exists  $w^S$  that satisfies (6.8). We first find the conditions that need to be satisfied by the desired  $w^S$  and then show that there indeed exists a unique such vector.

Now, our first observation is that  $w^S$  must have at least two positive components. On the one hand, if there were exactly one user with a positive bid, it would want to decrease the bid towards zero and yet have the whole resource to itself (so there can't be exactly one positive bid). On the other hand, if there were no users with a positive bid, there would be an incentive for all users to increase their bids to some non-zero value to capture the resource. The next observation is that since  $w^S$  has at least two positive components and since  $w_r/(\sum_{k \neq r} w_k + w_r)$  is strictly increasing in  $w_r$  if there are, the payoff function is strictly concave in  $w_r$ . Assuming that the utility functions are continuously differentiable, so this means that for each user  $k$ , the maximizer  $w^S$  of (6.7) satisfies the conditions

$$U'_r \left( \frac{w^S}{\sum_k w_k^S c} \right) \left( 1 - \frac{w_r^S}{\sum_k w_k^S} \right) = \frac{\sum_k w_k^S}{c}, \text{ if } w_k^S > 0 \quad (6.9)$$

$$U'_r(0) \leq \frac{\sum_k w_k^S}{c}, \text{ if } w_k^S = 0, \quad (6.10)$$

which are obtained by simply differentiating (6.7) and setting to 0 (or  $\leq 0$  if  $w_r^S = 0$ ) and multiplying by  $\sum_k w_k^S/c$ .

We now have the set of conditions (6.9)-(6.10) that must be satisfied by the bids that the Nash strategy profile suggests. But we don't know yet if there actually exists any such vector  $w^S$  that would satisfy the conditions. How do we go about showing that such a vector actually exists? Consider the conditions again. They look just like the KKT first-order conditions of a constrained optimization problem. Perhaps we could construct the equivalent optimization problem to which these are indeed the KKT conditions? Then if the optimization problem has a unique solution, the solution would be the desired vector of bids  $w^S$ . Consider the constrained optimization problem of maximizing

$$\sum_k \hat{U}_k(x_k), \quad (6.11)$$

subject to the constraints

$$\sum_k x_k \leq c, \quad x_k \geq 0 \quad \forall k = 1, 2, 3, \dots, R \quad (6.12)$$

where the utility function  $\hat{U}(\cdot)$  is defined as

$$\hat{U}_k(x_k) = \left(1 - \frac{x_k}{c}\right) U_k(x_k) + \left(\frac{x_k}{c}\right) \left(\frac{1}{x_k} \int_0^{x_k} U_k(z) dz\right). \quad (6.13)$$

It is easy to see that  $\hat{U}(\cdot)$  is concave and increasing in  $0 \leq x_k \leq c$  by differentiating it, which yields  $\hat{U}'_k(x_k) = U'_k(x_k)(1 - x_k/c)$ . Since  $U_k(\cdot)$  is concave and strictly increasing, we know that  $U'_k(x_k) > 0$ , and that  $U'_k(\cdot)$  is non-increasing. Hence, we conclude that  $\hat{U}'_k(x_k)$  is nonnegative and strictly decreasing in  $k$  over the region  $0 \leq x_k \leq c$  as required.

We verify that the KKT first-order conditions are identical in form to the conditions (6.9)-(6.10). Directly from the optimization problem above, we have that there exists a unique vector  $w$  and a scalar (the Lagrange multiplier)  $\lambda$  such that

$$U'_r(x_k) \left(1 - \frac{x_k}{c}\right) = \lambda, \quad \text{if } x_k > 0 \quad (6.14)$$

$$U'_k(0) \leq \lambda, \quad \text{if } x_k = 0 \quad (6.15)$$

$$\sum_k x_k = c \quad (6.16)$$

We check that at least two components of  $x$  above are positive. We have from (6.16), at least one of the  $x_k > 0$ . If only a single component  $x_r > 0$  with all others being 0, then from (6.14) we have  $\lambda = 0$ , which in turn means from (6.15) that  $U'_k(0) \leq 0$  for some  $k$ . This is impossible since  $U_k(\cdot)$  was assumed to be concave, strictly increasing for all  $k$ . Then we see that as desired, the above conditions are identical to the Nash conditions with  $\lambda = \sum_k w_k^S/c$  and  $x_k = cw_k/\sum_k w_k$ . Thus, we see that even in the case of price-anticipating users, there exists unique solution. We will denote the resulting allocation  $x^S$ , the  $S$  being used to denote “strategy”.

Notice that the solution found applies to an optimization problem that is different to the one in the price-taking case, so we would expect that in terms of utility maximization, the price-anticipating solution

would probably be worse than the price taking case. We will show how to bound the worst case performance of the price-anticipating case in the next section. Since in the price-anticipating case, all users play strategically with complete knowledge, and what the central authority does is to allocate resources in a weighted proportionally fair manner, the system can be likened to anarchy with users operating with minimal control. We refer to the inefficiency in such a system as *the price of anarchy*.

We now examine the impact of price-anticipating users on the network utility, i.e., the sum of the utilities of all the users in the network. We use the superscript  $T$  to denote solution to the network utility maximization problem (we use  $T$  since the optimal network utility is also the network utility assuming price-taking users), and the superscript  $S$  to denote the solution for the case of price-anticipating users. It will be shown that

$$\sum_r U_r(x_r^S) \geq \frac{3}{4} \sum_r U_r(x_r^T). \quad (6.17)$$

So the price of playing a game versus the optimal solution is no greater than  $1/4$  of the optimal solution. The proof of this theorem consists of two steps:

- Showing that the worst-case scenario for the price-anticipating paradigm is when the utility functions  $U_r$  are all linear.
- Minimizing the value of the game under the above condition.

### Step 1

Using concavity of  $U_r(\cdot)$  for any user  $k$  we have for the general allocation vector  $z$  with  $\sum_k z_k \leq c$ , that  $U_r(x_k^T) \leq U_k(z_r) + U'_k(z_k)(x_k^T - z_k)$ . This means that

$$\frac{\sum_r U_r(z_r)}{\sum_r U_r(x_r^T)} \geq \frac{\sum_r (U_r(z_r) - U'_r(z_r)z_r) + \sum_r U'_r(z_r)z_r}{\sum_r (U_r(z_r) - U'_r(z_r)z_r) + \sum_r U'_r(z_r)x_r^T}$$

Since we know that  $\sum_k x_k^T = c$ , we know that  $\sum_k U'_k(z_k)x_k^T \leq$

$(\max_k U'_r(z_k))c$ . Using this fact in the above equation, we obtain

$$\frac{\sum_k U_k(z_k)}{\sum_k U_k(x_r^T)} \geq \frac{\sum_k (U_k(z_k) - U'_k(z_k)z_k) + \sum_k U'_k(z_k)z_k}{\sum_k (U_k(z_k) - U'_k(z_k)z_k) + (\max_k U'_k(z_k))c}$$

The term  $\sum_k (U_k(z_k) - U'_k(z_k)z_k)$  is non-negative by concavity of  $U_k$  which means that

$$\frac{\sum_k U_k(z_k)}{\sum_k U_k(x_r^T)} \geq \frac{\sum_k U'_k(z_k)z_k}{(\max_k U'_k(z_k))c} \quad (6.18)$$

The above inequality is of interest as it compares the utility function with its linear equivalent. If we substitute  $z = x^S$  in (6.18), we obtain

$$\frac{\sum_k U_k(x_k^S)}{\sum_k U_k(x_k^T)} \geq \frac{\sum_k U'_k(x_k^S)x_k^S}{(\max_k U'_k(x_k^S))c} \quad (6.19)$$

Now, we notice that the left-hand side of the expression above is the price of anarchy that we are interested in, while the right-hand side of the expression looks like the price of anarchy for the case where the utility function is linear, i.e., when  $U_r(x_r) = U'_r(x_r^S)x_r \triangleq \bar{U}_r(x_r)$ . We verify this observation by noting that since the conditions in (6.14)–(6.16) are linear, the numerator is the aggregate utility with price-anticipating users when the utility function is  $\bar{U}(\cdot)$ . Also, the denominator is the maximum aggregate that can be achieved with the utility function  $\bar{U}(\cdot)$ , which means that it corresponds to the price taking case for utility  $\bar{U}(\cdot)$ . Thus, we see that the linear utility function of the sort described above necessarily has a lower total utility than any other type of utility function.

## Step 2

Since the worst-case scenario is for linear utility functions, we may take  $U_r(x_r) = \alpha_r x_r$ . From Step 1, the price of anarchy, i.e., the ratio of aggregate utility at the Nash equilibrium to the aggregate utility at social optimum is then given by

$$\frac{\sum_k \alpha_k x_k^S}{\{\max_k \alpha_k\}c}. \quad (6.20)$$

Without loss of generality, we may take the  $\max_k \alpha_k = 1$  and  $c = 1$ . Since this means that the denominator of the above expression is 1, to find the worst-case ratio we need to find  $\alpha_2, \alpha_3, \dots, \alpha_R$  such that the numerator is minimized. This would directly give the price of anarchy. So the objective is to

$$\text{minimize } x_1^S + \sum_{r=2}^R \alpha_r x_r^S \quad (6.21)$$

$$\text{subject to } \alpha_k(1 - x_k^S) = 1 - x_1^S, \text{ if } x_k^S > 0 \quad (6.22)$$

$$\alpha_k \leq 1 - x_1^S, \text{ if } x_k^S = 0 \quad (6.23)$$

$$\sum_k x_k^S = 1 \quad (6.24)$$

$$0 \leq \alpha_k \leq 1, \quad k = 2, \dots, R \quad (6.25)$$

$$x_k^S \geq 0, \quad k = 1, \dots, R \quad (6.26)$$

Notice that the constraints on the above optimization problem ensure that  $x^S$  is a allocation vector that a Nash strategy profile would suggest. Since only the users with non-zero allocations contribute to the utility, we can consider the system with  $N \leq R$  users, with every user getting a non-zero allocation. Equivalently we could just assume that all  $R$  users get a non-zero allocation and observe what happens as  $R$  increases. Then  $\alpha_k(1 - x_k^S) = 1 - x_1^S$  holds for all users in the new system, which implies  $\alpha_k = (1 - x_1^S)/(1 - x_k^S)$ . Let us fix  $x_1^S$  for now, and minimize over the remaining  $x_r^S$ . Then we have

$$\text{minimize } x_1^S + \sum_{k=2}^R R \frac{x_k^S(1 - x_1^S)}{1 - x_k^S} \quad (6.27)$$

$$\text{subject to } \sum_{k=2}^R R x_k^S = 1 - x_1^S \quad (6.28)$$

$$0 \leq x_k^S \leq x_1^S, \quad k = 2, \dots, R \quad (6.29)$$

The above problem is well defined only if  $x_1^S \geq 1/R$  (otherwise the last constraint will be violated upon minimization). If we assume this condition, by symmetry, the minimum value occurs for all users  $2, \dots, R$  getting the same allocation equal to  $1 - 1 - x_1^S/(R - 1)$ . Substituting this value into the problem, we only have to minimize over  $x_1^S$ , i.e., the

problem is now given by

$$\text{minimize } x_1^S + (1 - x_1^S)^2 \left( 1 - \frac{(1 - x_1^S)}{R - 1} \right) \quad (6.30)$$

$$\text{subject to } 0 \leq x_1^S \leq 1. \quad (6.31)$$

The objective above is decreasing in  $R$  and so the lowest value would occur as  $R \rightarrow \infty$ . So we finally have very simple problem to solve, namely

$$\text{minimize } x_1^S + (1 - x_1^S)^2 \quad (6.32)$$

$$\text{subject to } 0 \leq x_1^S \leq 1 \quad (6.33)$$

By differentiation we can easily see that the solution to the above problem is for  $d_1^S = 1/2$ , which yields a worst case aggregate of  $3/4$ . Thus we see that the aggregate utility falls by no more than 25% when the users are price anticipating.

## 6.4 Notes

The VCG mechanism was developed by Clarke [99] and Groves [100] as a generalization of an auction called the second-price auction due to Vickrey [101]. The Kelly mechanism is due to Kelly [3]. The price of anarchy for strategic users using the Kelly mechanism was computed by Johari and Tsitsiklis in [102]. Our interest in the Kelly mechanism is due to the fact that it has a simple decentralized implementation when the users are price-taking. If one is more generally interested in truth-revealing mechanisms using one-dimensional bids, then there is recent work on the design of such mechanisms: the interested reader is referred to the works of Maheswaran and Basar [103], Yang and Hajek [104] and Johari and Tsitsiklis [105].

## Conclusions

---

In this monograph we have attempted to present the essence of network resource allocation through utility maximization. We started with the basic ideas of what the resource is and how allocations could be made. As we mentioned earlier, the concept that bandwidth is a divisible good that must be shared equitably is a concept that only recently has found favor in the research community. We explained how such allocations could be achieved in a decentralized fashion—an aim that is central to the idea of the Internet. In particular, we went into details of how network algorithms can be thought of as consisting of two controllers—the rate-update at source and price-update at links. We studied several different ways in which these two controllers could be chosen based on the type of formulation of the optimization problem. We also studied multiple different meanings of fairness.

Perhaps the most important part of our study was the relationship between congestion control protocols and the utility maximization framework. Although the details of the protocols cannot always be mapped to a congestion control algorithm derived from utility maximization, it certainly shows how the goals of the protocols are precisely what we seek to do by decentralized optimization. We also saw how



more recent developments in the protocol arena make use of this connection in order to design protocols with specific operating conditions and fairness ideals in mind.

Another important section of our study dealt with system stability. We showed how control parameters should be chosen to ensure stability in the presence of delays. We also emphasized the fact that a network system is dynamic, with flows arriving and departing. We showed that the utility maximization is robust to these fluctuations in the sense that the number of users in the network stays bounded despite fluctuations in flow arrivals and departures.

The Internet continues to grow as these words are being written. That the system has scaled so well is testament to the vision of its designers. Much has changed since the first links were laid between laboratories. Bandwidths are much higher than conceived at the time. Quite often it is not the bandwidth of the links, but the line-rate and memory access speeds of the hardware using these links that become the bottle neck. As the Internet spreads geographically wide, distances to be traversed by packets are no longer insignificant, even at the speed of light. RTTs of the order of a couple of hundred milliseconds are not uncommon. Much work remains to be done to design protocols that would operate in such high bandwidth-delay product environments.

To add to the question of how to harness new technology, there is also the question of the economics of the system. Bandwidth allocation auctions are not uncommon today, and there may come a time when users might have to bid for the resources they want. We demonstrated the inefficiencies that creep into the system when users become self-aware, and a major challenge as the Internet becomes increasingly corporatized would be to design schemes that use the fact that end-users might be intelligent to achieve efficient allocation.

In keeping with our objective of presenting only the fundamental principles, we tried to keep the material as simple as possible, so as to not bog down the reader in fine details and multiple extensions of a basic idea. We also focussed extensively on the fluid model approach, which, although an approximation, provides a clear insight into the workings of the real system which may have other discrete-event and stochastic components.

Some of the audience for this work would be from the network controls community. For them, this monograph might serve as a refresher or repository of basic results and proofs. However, we hope that interested audience of this monograph also includes those working actively in protocol development in both the wired and wireless networks. We have endeavored to answer questions like what goals are feasible? What should a good algorithm look like? What does stability of the system mean? We hope that we have succeeded in providing guidelines on how to answer these questions to a systems designer besides serving the needs of analysts.



## Acknowledgements

---

The authors gratefully acknowledge funding from NSF, DARPA and AFOSR which supported our own research and the time that we spent in understanding the work of many others who have contributed to this field.



## References

---

- [1] V. Jacobson, "Congestion avoidance and control," *ACM Computer Communication Review*, vol. 18, pp. 314–329, August 1988.
- [2] D. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, pp. 1–14, 1989.
- [3] F. P. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, pp. 33–37, 1997.
- [4] R. Srikant, *The Mathematics of Internet Congestion Control*. Birkhauser, 2004.
- [5] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.
- [6] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
- [7] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.
- [8] F. P. Kelly, "Models for a self-managed Internet," *Philosophical Transactions of the Royal Society*, vol. A358, pp. 2335–2348, 2000.
- [9] —, "Mathematical modelling of the Internet," in *Mathematics Unlimited - 2001 and Beyond (Editors B. Engquist and W. Schmid)*. Berlin: Springer-Verlag, 2001, pp. 685–702.
- [10] X. Lin and N. Shroff, "Joint rate control and scheduling in wireless networks," in *Proceedings of the IEEE Conference on Decision and Control*, 2004.
- [11] M. J. Neely, E. Modiano, and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," in *Proceedings of IEEE INFOCOM*, 2005.

- [12] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," in *Proceedings of IEEE INFOCOM*, 2005.
- [13] A. L. Stolyar, "Maximizing queueing network utility subject to stability: greedy primal-dual algorithm," *Queueing Systems*, vol. 50, pp. 401–457, 2005.
- [14] A. Eryilmaz and R. Srikant, "Joint Congestion Control, Routing and MAC for Stability and Fairness in Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1514–1524, August 2006.
- [15] X. Lin, N. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE Journal on Selected Areas in Communications*, pp. 1452–1463, August 2006.
- [16] L. Georgiadis, M. J. Neely, and L. Tassiulas, *Resource Allocation and Cross-Layer Control in Wireless Networks*. Foundations and Trends in Networking, NOW publishers, 2006.
- [17] J. M. Jaffe, "A decentralized "optimal" multiple-user flow control algorithm," *IEEE Transactions on Communications*, pp. 954–962, 1981.
- [18] D. Bertsekas and R. Gallager, *Data Networks*. Englewood Cliffs, NJ: Prentice Hall, 1987.
- [19] J. Rawls, *A Theory of Justice*. Belknap Press, 1971.
- [20] J. Nash, "The bargaining problem," *Econometrica*, vol. 18, pp. 155–162, 1950.
- [21] L. Massoulié and J. Roberts, "Bandwidth sharing: Objectives and algorithms," in *Proceedings of Infocom*, New York, NY, March 1999.
- [22] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," in *SPIE International Symposium*, Boston, MA, 1998.
- [23] —, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556–567, October 2000.
- [24] H. Khalil, *Nonlinear Systems*. Upper Saddle River, NJ: 2nd edition, Prentice Hall, 1996.
- [25] S. H. Low and D. E. Lapsley, "Optimization flow control, I: Basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, pp. 861–875, December 1999.
- [26] H. Yaiche, R. R. Mazumdar, and C. Rosenberg, "A game-theoretic framework for bandwidth allocation and pricing in broadband networks," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 667–678, October 2000.
- [27] F. Paganini, "A global stability result in network flow control," *Systems and Control Letters*, vol. 46, no. 3, pp. 153–163, 2002.
- [28] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, "Multi-Path TCP: A Joint Congestion Control and Routing Scheme to Exploit Path Diversity on the Internet," vol. 14, no. 6, December 2006.
- [29] S. H. Low, "Optimization flow control with on-line measurement or multiple paths," in *Proceedings of the 16th International Teletraffic Congress*, Edinburgh, UK, 1999.
- [30] X. Lin and N. Shroff, "The multi-path utility maximization problem," in *Proceedings of the Allerton Conference on Communications, Control and Computing*, Monticello, IL, October 2003.

- [31] S. Kunniyur and R. Srikant, "End-to-end congestion control: utility functions, random losses and ECN marks," in *Proceedings of IEEE Infocom*, Tel Aviv, Israel, March 2000.
- [32] —, "A time-scale decomposition approach to adaptive ECN marking," in *Proceedings of IEEE Infocom*, Anchorage, AK, April 2001.
- [33] J. Wen and M. Arcak, "A unifying passivity framework for network flow control," in *Proceedings of IEEE Infocom*, April 2003.
- [34] T. Alpcan and T. Başar, "A utility-based congestion control scheme for internet-style networks with delay," in *Proceedings of IEEE Infocom*, San Francisco, California, March-April 2003.
- [35] S. H. Low, F. Paganini, and J. C. Doyle, "Internet congestion control," *IEEE Control Systems Magazine*, vol. 22, pp. 28–43, February 2002.
- [36] S. Deb and R. Srikant, "Congestion control for fair resource allocation in networks with multicast flows," in *Proceedings of the IEEE Conference on Decision and Control*, 2001.
- [37] K. Kar, S. Sarkar, and L. Tassiulas, "A scalable low-overhead rate control algorithm for multirate multicast sessions," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1541–1557, October 2002.
- [38] L. Bui, R. Srikant, and A. L. Stolyar, "Optimal resource allocation for multicast flows in multihop wireless networks," in *Proceedings of the IEEE Conference on Decision and Control*, December 2007.
- [39] C. Fulton, S.-Q. Li, and C. S. Lim, "Ut: Abr feedback control with tracking," in *IEEE Infocom*, April 1997.
- [40] T.-J. Lee and G. de Veciana, "A decentralized framework to achieve max-min fair bandwidth allocation for atm networks," in *Proceedings of IEEE GLOBECOM*, 1998, pp. 1515–1520.
- [41] K. K. Ramakrishnan and R. Jain, "A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer," in *Proceedings of ACM Sigcomm*, 1988, pp. 303–313.
- [42] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, pp. 397–413, August 1993.
- [43] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, pp. 10–23, October 1994.
- [44] D. E. Lapsley and S. H. Low, "Random early marking for Internet congestion control," in *Proceedings of IEEE GLOBECOM*, 1999, pp. 1747–1752.
- [45] S. Deb and R. Srikant, "Rate-based versus queue-based models of congestion control," in *Proceedings of ACM Sigmetrics*, 2004.
- [46] —, "Rate-based versus queue-based models of congestion control," *IEEE Transactions on Automatic Control*, April 2006.
- [47] G. Raina and D. Wischik, "Buffer sizes for large multiplexers: TCP queueing theory and instability analysis," in *Proceedings of EuroNGI conference on Next Generation Internet Networks*, 2005.
- [48] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proceedings of IEEE*, January 2007.



- [49] R. Stevens, *TCP/IP Illustrated, Volume 1*. Addison-Wesley, 1994.
- [50] L. L. Peterson and B. Davie, *Computer Networks: A Systems Approach*. Morgan-Kaufman, 1999.
- [51] S. Keshav, *An Engineering Approach to Computer Networks*. Reading, MA: Addison-Wesley, 1997.
- [52] S. Kunniyur and R. Srikant, "End-to-end congestion control: utility functions, random losses and ECN marks," *IEEE/ACM Transactions on Networking*, vol. 7, no. 5, pp. 689–702, October 2003.
- [53] V. Misra, W. Gong, and D. Towsley, "A fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *Proceedings of ACM Sigcomm*, Stockholm, Sweden, September 2000.
- [54] F. Baccelli, D. R. McDonald, and J. Reynier, "A mean-field model for multiple TCP connections through a buffer implementing red," *Performance Evaluation*, vol. 49, no. 1/4, pp. 77–97, 2002.
- [55] S. Shakkottai and R. Srikant, "How good are fluid models of internet congestion control?" in *Proceedings of IEEE Infocom*, June 2002.
- [56] —, "Mean FDE models for internet congestion control under a many-flows regime," *IEEE Transactions on Information Theory*, pp. 1050–1072, June 2004.
- [57] P. Tinnakornsrisuphap and A. M. Makowski, "Limit behavior of ECN/RED gateways under a large number of TCP flows," in *Proceedings of Infocom*, San Francisco, CA, April 2003, pp. 873–883.
- [58] G. Vinnicombe, "On the stability of networks operating TCP-like congestion control," in *Proceedings of the IFAC World Congress*, Barcelona, Spain, 2002.
- [59] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," December 2002.
- [60] L. Bramko and L. Peterson, "TCP Vegas: end-to-end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications*, pp. 1465–1480, October 1995.
- [61] S. H. Low, L. Peterson, and L. Wang, "Understanding vegas: A duality model," *Journal of ACM*, vol. 49, no. 2, pp. 207–235, March 2002.
- [62] J. Mo, R. J. La, V. Anantharam, and J. Walrand, "Analysis and comparison of TCP Reno and Vegas," in *Proceedings of Infocom*, March 1999, pp. 1556–1563.
- [63] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: motivation, architecture, algorithms, performance," *IEEE/ACM Transactions on Networking*, Dec. 2006.
- [64] C. Jin, D. Wei, S. H. Low, G. Buhrmaster, J. Bunn, D. H. Choe, R. L. A. Cottrell, J. C. Doyle, W. Feng, O. Martin, H. Newman, F. Paganini, S. Ravot, and S. Singh, "FAST TCP: From theory to experiments," April 2003.
- [65] F. Baccelli and D. Hong, "A.I.M.D., fairness and fractal scaling of TCP traffic," in *Proceedings of IEEE INFOCOM*, 2002.
- [66] F. Wirth, R. Stanojevic, R. Shorten, and D. Leith, "Stochastic equilibria of AIMD communication networks," *SIAM J. Matrix Theory and Its Applications*, 2006.

- [67] A. Kherani, B. Prabhu, K. Avrachenkov, and E. Altman, "Comparative study of different adaptive window protocols," *Telecommunication Systems*, p. 321350, 2005.
- [68] R. N. Shorten and D. J. Leith, "H-TCP: TCP for high-speed and long-distance networks," in *Proc. PFLDnet*, 2004.
- [69] S. Liu, T. Basar, and R. Srikant, "TCP-Illinois: A loss and delay-based congestion control algorithm for high-speed networks," in *Proc. First International Conference on Performance Evaluation Methodologies and Tools (VALUE-TOOLS)*, Pisa, Italy, October 11-13 2006.
- [70] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Prentice-Hall, 2002.
- [71] C. A. Desoer and Y. T. Wang, "On the generalized Nyquist stability criterion," *IEEE Transactions on Automatic Control*, vol. 25, pp. 187–196, April 1980.
- [72] G. Vinnicombe, "On the stability of end-to-end congestion control for the Internet," 2001, university of Cambridge Technical Report CUED/F-INFENG/TR.398. Available at <http://www.eng.cam.ac.uk/~gv>.
- [73] F. Paganini, J. Doyle, and S. Low, "Scalable laws for stable network congestion control," in *Proceedings of the IEEE Conference on Decision and Control*, Orlando, FL, December 2001, pp. 185–190.
- [74] S. Kunniyur and R. Srikant, "Stable, scalable, fair congestion control and AQM schemes that achieve high utilization in the internet," *IEEE Transactions on Automatic Control*, vol. 48, pp. 2024–2028, 2003.
- [75] S. Liu, T. Başar, and R. Srikant, "Exponential RED: A stabilizing AQM scheme for low- and high-speed tcp protocols," *IEEE/ACM Transactions on Networking*, pp. 1068–1081, October 2005.
- [76] F. Kelly and T. Voice, "Stability of end-to-end algorithms for joint routing and rate control," *Computer Communication Review*, vol. 35, 2005.
- [77] C. V. Hollot and Y. Chait, "Nonlinear stability analysis for a class of TCP/AQM schemes," in *Proceedings of the IEEE Conference on Decision and Control*, December 2001.
- [78] S. Deb and R. Srikant, "Global stability of congestion controllers for the internet," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 1055–1060, June 2003.
- [79] Z. Wang and F. Paganini, "Global stability with time-delay in network congestion control," in *Proceedings of the IEEE Conference on Decision and Control*, 2002.
- [80] L. Ying, G. Dullerud, and R. Srikant, "Global stability of Internet congestion controllers with heterogeneous delays," in *Proceedings of the American Control Conference*, Boston, MA, June 2004.
- [81] —, "Global stability of internet congestion controllers with heterogeneous delays," *IEEE/ACM Transactions on Networking*, June 2006.
- [82] P. Ranjan, R. J. La, and E. H. Abed, "Characterization of global stability conditions for rate control with an arbitrary communication delay," *IEEE/ACM Transactions on Networking*, pp. 94–107, February 2006.

- [83] A. Papachristodoulou, "Global stability analysis of a TCP/AQM protocol for arbitrary networks with delay," in *Proceedings of the IEEE Conference on Decision and Control*, December 2004.
- [84] C. Holot, V. Misra, D. Towsley, and W. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proceedings of IEEE Infocom*, Anchorage, Alaska, April 2001, pp. 1726–1734.
- [85] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue algorithm for active queue management," in *Proceedings of ACM Sigcomm*, San Diego, CA, August 2001, pp. 123–134.
- [86] D. Heyman, T. V. Lakshman, and A. Niedhart, "A new method for analyzing feedback-based protocols with applications to engineering web traffic over the Internet," in *Sigmetrics 97*, 1997, pp. 24–38.
- [87] A. Das and R. Srikant, "Diffusion approximations for models of congestion control in high-speed networks," in *IEEE Conference on Decision and Control*, Tampa, FL, December 1998.
- [88] J. Roberts and L. Massoulié, "Bandwidth sharing and admission control for elastic traffic," in *Proc. ITC specialists seminar*, Yokohama, Japan, 1998.
- [89] G. de Veciana, T.-J. Lee, and T. Konstantopoulos, "Stability and performance analysis of networks supporting elastic services," *IEEE/ACM Transactions on Networking*, vol. 9, no. 1, pp. 2–14, February 2001.
- [90] T. Bonald and L. Massoulié, "Impact of fairness on Internet performance," in *Proceedings of ACM Sigmetrics*, 2001.
- [91] R. Srikant, "Models and methods for analyzing Internet congestion control algorithms," in *Advances in Communication Control Networks in the series Lecture Notes in Control and Information Sciences (LCNCIS)*, C.T. Abdallah, J. Chiasson and S. Tarbouriech (eds.). Springer, 2004.
- [92] S. P. Meyn, *Control Techniques for Complex Networks*. Cambridge University Press, 2007.
- [93] X. Lin and N. Shroff, "On the stability region of congestion control," in *Proceedings of the Allerton Conference on Communications, Control and Computing*, 2004.
- [94] R. Srikant, "On the positive recurrence of a markov chain describing file arrivals and departures in a congestion-controlled network," October 2004, presented at the IEEE Computer Communications Workshop.
- [95] X. Lin, N. B. Shroff, and R. Srikant, "On the connection-level stability of congestion-controlled networks," 2006, to appear in the *IEEE Transactions on Information Theory*.
- [96] L. Massoulié, "Structural properties of proportional fairness: stability and insensitivity," preprint.
- [97] M. Bramson, "Stability of networks for max-min fair routing," 2005, presented at INFORMS Applied Probability Conference.
- [98] M. Chiang, D. Shah, and A. Tang, "Stochastic stability of network utility maximization: General file size distribution," in *Proceedings of the Allerton Conference on Communication, Control and Computing*, 2006.
- [99] E. H. Clarke, "Multipart pricing of public goods," *Public Choice*, 1971.
- [100] T. Groves, "Incentives in teams," *Econometrica*, pp. 617–631, 1973.

- [101] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *Journal of Finance*, pp. 8–37, 1961.
- [102] R. Johari and J. N. Tsitsiklis, "Efficiency loss in a network resource allocation game," *Mathematics of Operations Research*, vol. 29, pp. 407–435, March 2004.
- [103] R. T. Maheswaran and T. Basar, "Efficient signal proportional allocation (espa) mechanisms: Decentralized social welfare maximization for divisible resources," *IEEE Journal on Selected Areas in Communications*, pp. 1000–1009, May 2006.
- [104] S. Yang and B. Hajek, "VCG-Kelly mechanisms for allocation of divisible goods: Adapting VCG mechanisms to one-dimensional signals," *IEEE Journal of Selected Area in Communications, Special Issue on Non-Cooperative Behavior in Networking*, 2007, to appear.
- [105] R. Johari and J. N. Tsitsiklis, "Communication requirements of VCG-like mechanisms in convex environments," in *Proceedings of the Allerton Conference on Control, Communications and Computing*, 2005.