1 Utility Maximization

In this section 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. 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 expect 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.

2 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 $S$ and a set of links $L$. Each link $l \in L$ has a finite fixed capacity $c_l$. Each source in $S$ is associated with a route $r \subset L$ 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 $L$. 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

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

subject to the constraints

$$\sum_{r \in S} x_r \leq c_l, \forall l \in \mathcal{L},$$

$$x_r \geq 0, \forall r \in \mathcal{S}.$$  \hspace{1cm} (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.

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

$$\max_x f(x)$$

subject to

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

and

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

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

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

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

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 (4)-(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 (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 1 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 $x_2$.

![Figure 1: 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.](image)

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

$$\max_x \sum_{i=0}^{3} \log x_i$$

with constraints

$$x_0 + x_1 \leq 1,$$
$$x_0 + x_2 \leq 2,$$
$$x \geq 0,$$

where $x$ is the vector consisting of $x_0$, $x_1$ and $x_2$. Now, since $\log x \to -\infty$ as $x \to 0$, 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 (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 section, we will see that there are distributed solutions to the optimization problem which are easy to implement in the Internet.

### 3 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
\[
x_u < \hat{x}_u \quad \text{and} \quad x_u < \hat{x}_u.
\]
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
\[
\min_r \hat{x}_r < \min_r x_r. \tag{7}
\]
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 (7) 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 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
\[
U_r(x_r) = \frac{x_r^{1-\alpha}}{1-\alpha}, \tag{8}
\]
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 $\alpha \to 1$. Notice that maximizing the sum of $\frac{x_r^{1-\alpha} - 1}{1-\alpha}$ 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

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

thus yielding proportional fairness in the limit as $\alpha \to 1$. Next, we argue that the limit $\alpha \to \infty$ 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,

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

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.

4 Utility Maximization Algorithms

In our solution to the network utility maximization problem for a simple network in the previous section, 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 section 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 form the network. We assume in this section that there are no feedback delays in the system; the effect of delays will be considered later. We start with the Primal controller, which will also serve to illustrate some basic optimization concepts.
5 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

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

(9)

where \( x \) is the vector of rates of all sources and \( B_l(.) \) 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 section; 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 the algorithm that would satisfy the above requirements. Let us first consider the penalty functions \( B_l(.) \) that appears in 9 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 continuously differentiable. Then, we can equivalently require that

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

(10)

where \( f_l(.) \) 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, \( V(x) \) strictly concave. Further, we assume that \( U_r \) and \( f_l \) are such that the maximization of (9) results in a solution with \( x_r \geq 0 \ \forall r \in S \). Now, the condition that must be satisfied by the maximizer of (9) is obtained by differentiation and is given by

\[ U_r'(x_r) - \sum_{l \in r} f_l \left( \sum_{s : l \in s} x_s \right) = 0, \quad r \in S. \]

(11)

We now require an algorithm that would drive \( x \) towards the solution of (11). 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 from $g(x)$, 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, or letting $\delta \to 0$

$$\dot{x} = k(t)\frac{dg(x)}{dx}.$$ (12)

The idea is illustrated in Figure 2 at some selected time instants. The value of $x$ changes smoothly, with $g(x)$ increasing at each time instant, until it hits the maximum value.

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

$$\dot{x}_r = k_r(x_r) \left( U_r'(x_r) - \sum_{l \in r} f_l \left( \sum_{s: l \in s} x_s \right) \right),$$ (13)

where $k_r(.)$ is non-negative, increasing and continuous. We have obtained the above by differentiating (9) with respect to $x_r$ to find the direction of ascent, and used it along with a scaling function $k_r(.)$ to construct an algorithm of the form shown in (12). Clearly, the stationary point of the above algorithm satisfies (11) and hence maximizes (9). 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 $q_r = \sum_{l \in r} f_l(\sum_{s: l \in s} x_s)$ is large, then the congestion controller decreases its transmission rate. Further, if $x_r$ is large, then $U_r'(x_r)$ 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 (13). 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 apprised 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 \( \text{ack} \) 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

\[
y_l = \sum_{s, l \in s} x_s,
\]

which is the load on link \( l \). Using the elements of the routing matrix and recalling the notation of Section 2, \( y_l \) can also be written as

\[
y_l = \sum_{s, l \in s} R_{ls} x_s.
\]

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

\[
y = Rx
\]

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

\[
p_l(t) = f_l \left( \sum_{s, l \in s} x_s \right) = f_l(y_l(t)).
\]

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

\[
q_r = \sum_{l \in r} p_l(t).
\]

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

\[
q = R^T p
\]

The above is more than just an intellectual exercise, since the expressions (15) and (18) 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.

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.

5.1 Stability Notions for Dynamical Systems

We first give an intuitive interpretation of a Lyapunov function. Consider a surface such as that shown in Figure 4. 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. 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

\[ \dot{x} = g(x), \quad x(0) = x_0, \]

where it is assumed that \( g(x) = 0 \) has a unique solution. Call this solution 0. Here \( x \) and 0 can be vectors.
Definition 5.1 The equilibrium point 0 of (19) 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
  \[ \lim_{t \to \infty} \|x(t)\| = 0 \]
  for all $\|x_0\| \leq \delta$.

- globally, asymptotically stable if
  \[ \lim_{t \to \infty} \|x(t)\| = 0 \]
  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.

Theorem 5.1 Let $x = 0$ be an equilibrium point for $\dot{x} = f(x)$ and $D \subset \mathbb{R}^n$ be a domain containing 0. Let $V : D \to \mathbb{R}$ 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.,

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

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.

5.2 Global Asymptotic Stability of the Primal Controller

We show that the primal controller of (13) is globally asymptotically stable by using the Lyapunov function idea described above. Recall that $V(x)$ 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 (13). We use this Lyapunov function in the following theorem.

**Theorem 5.2** Consider a network in which all sources follow the primal control algorithm (13). 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 $W(x) \to \infty$ as $\|x\| \to \infty$, $\hat{x}_i > 0$ for all $i$, and $V(x)$ is as defined in (9). Then, the controller in (13) is globally asymptotically stable and the equilibrium value maximizes (9).

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

$$\dot{W} = -\sum_{r \in S} \frac{\partial V}{\partial x_r} \dot{x}_r = -\sum_{r \in S} k_r(x_r) \left(U_r'(x_r) - q_r\right)^2 < 0, \forall x \neq \hat{x}, \quad (20)$$

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

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

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

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

5.3 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

\[
1 - e^{\sum_{l \in r} p_l}.
\]

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 \(\sum_{l \in r} p_l\). 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.

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

\[
1 - \frac{\rho}{1 - \rho B + \rho B} = 1 - \rho B + \rho B,
\]

where \(\rho = \sum_{l \in r} x_r \cdot c_l\). 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

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

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.

6 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

\[
\max_{x_r} \sum_{r \in \mathcal{S}} U_r(x_r)
\]  

(21)

subject to the constraints

\[
\sum_{r \in \mathcal{S}} x_r \leq c_l, \forall l \in \mathcal{L},
\]

(22)

\[
x_r \geq 0, \forall r \in \mathcal{S}.
\]

(23)

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

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

(24)

Here the \(p_l\)s are the Lagrange multipliers that we saw in the previous section. 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 (24), \(x_r\) must satisfy

\[
U'_r(x_r) = q_r,
\]

(25)

or equivalently,

\[
x_r = U_r^{-1}(q_r),
\]

(26)

where, as usual, \(q_r = \sum_{l : l \in r} p_r\), 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 (26) and (24) that

\[
\frac{\partial D}{\partial p_l} = \sum_{r : l \in r} \frac{\partial U_r(x_r)}{\partial p_l} - (y_l - c_l) - \sum_{l} p_l \frac{\partial y_l}{\partial p_l},
\]

(27)

where the \(x_r\) above is the optimizing \(x_r\) in (24). In order to evaluate the above, we first compute \(\partial x_r / \partial p_l\). Differentiating (25) with respect to \(p_l\) yields

\[
U''_r(x_r) \frac{dx_r}{dp_l} = 1
\]

\[
\Rightarrow \frac{\partial x_r}{\partial p_l} = \frac{1}{U''_r(x_r)}
\]

14
Substituting the above in (27) yields
\[
\frac{\partial D}{\partial p_l} = \sum_{r,l \in \mathcal{G}} \frac{U'_r(x_r)}{U''_r(x_r)} - (y_l - c_l) - \sum_l p_l \sum_{r,l \in \mathcal{G}} \frac{1}{U''_r(x_r)}
\]
(28)
\[
= c_l - y_l,
\]
(29)
where we have interchanged the last two summations in (28) and used the facts \(U'_r(x_r) = q_r\) and \(q_r = \sum_{l \in \mathcal{G}} p_l\). 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 (26) and (29), we have the following dual control algorithm:
\[
x_r = U'^{-1}_r(q_r) \quad \text{and} \quad p_l = h_l(y_l - c_l)^+_l
\]
(30)
\[
\dot{p}_l = h_l(y_l - c_l)^+_l
\]
(31)
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.

- Once 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.
6.1 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 (1). Due to the same concavity arguments used earlier, we know that the maximizer of (1) which we denote by ˆ\(x\) is unique. Suppose also that in the dual formulation (24) 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 6.1** 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
\[
V(p) = \sum_{l \in \mathcal{L}} (c_l - \hat{y}_l)p_l + \sum_{r \in \mathcal{S}} \int_{\hat{q}_r}^{q_r} (\dot{x}_r - (U'_r)^{-1}(\sigma))d\sigma.
\]
Then we have
\[
\frac{dV}{dt} = \sum_{l} (c_l - \hat{y}_l)\dot{p}_l + \sum_{r} (\dot{x}_r - (U'_r)^{-1}(q_r))\dot{\hat{q}}_r
\]
\[
= (c - \hat{y})^T \hat{p} + (\dot{x} - x)^T \dot{\hat{q}}
\]
\[
= (c - \hat{y})^T \hat{p} + (\dot{x} - x)^T R^T \hat{p}
\]
\[
= (c - \hat{y})^T \hat{p} + (\dot{y} - y)^T \hat{p}
\]
\[
= (c - y)^T \hat{p}
\]
\[
= \sum_{l} h_l(c_l - y_l)(y_l - c_l)^+ p_l
\]
\[
\leq 0.
\]
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 (1).
7 Extension to Multi-path Routing

The approaches considered in this section 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 5 indicates a simple scenario where this might be possible.

As before, the set of sources is called $S$. Each source $s \in S$ 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 $R$. As in single-path routing, note that two paths in the network that are otherwise identical can be associated with two different indices $r \in R$, if their sources are different. For instance, in Figure 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 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

$$
U(x) = \sum_s w_s \log \left( \sum_{r \in s} x_r \right) + \varepsilon \sum_{r \in s} \log(x_r) \\
- \sum_l \int_0^{\sum_{k \in l} x_k} f_l(y) \, dy.
$$

(32)
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 $r$:

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

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 $U(\hat{x}) - U(x)$ 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 section, 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.

#### 8 Congestion Control Protocols

We have seen in the previous sections 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 section, we explore the relationship between the algorithms discussed in the previous sections 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 sections. 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 the existing protocol to adapt to the evolution of the Internet as it continues to grow faster.

As mentioned in the first section, 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 receiver. TCP adapts the window size in response to congestion information. An illustration is presented in Figure 6. The window size is increased if the sender determines 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

![Figure 6: 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.](image-url)
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 section.

9 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 increases 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 cut 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 7.

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 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 sections.

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

$$W_r(t) = \frac{x_r(t - T_r)(1 - q_r(t))}{W_r(t)} - \beta x_r(t - T_r)q_r(t)W_r(t). \quad (34)$$

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 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

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

(35)

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

\[
\dot{x}_r = \sqrt{\frac{1 - \hat{q}_r}{\beta \hat{q}_r T_r}},
\]

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

\[
\dot{x}_r \propto 1/T_r \sqrt{q_r}.
\]

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

10 Relationship to Primal Algorithm

Now, consider the controller (35) again. Suppose that there were no feedback delay, but the equation is otherwise unchanged. So \( T_r^2 \) that appears in (35) 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

\[
\dot{x}_r = \frac{1}{T_r^2} - \beta x_r^2 q_r
\]

\[
= \beta x_r^2 \left( \frac{1}{\beta T_r x_r^2} - q_r \right).
\]

Comparing with (13), we find that the utility function of the source \( r \) satisfies

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

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

\[
U_r(x_r) = -\frac{1}{\beta T_r x_r}.
\]

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

\[
\dot{x}_r = \frac{1 - q_r}{T_r^2} - \beta x_r^2 q_r
\]

\[
= (\beta x_r^2 + 1/T_r^2) \left( \frac{1}{\beta x_r^2 + \frac{1}{T_r^2}} - q_r \right),
\]

Thus,

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

where the utility function is determined up to an additive constant.
Multirate multicast congestion control

In the previous sections, we only studied the congestion control problem for unicast sources, i.e., sources which transmit from an origin node to a single destination node. However, there are many applications, where the data from a single node has to be transmitted to many destinations simultaneously. An example would be a popular music concert that may be viewed by many people in the world. There are two ways for the network to perform this point-to-multipoint transmission:

- Unicast method: One can view the transmission from one source to say, $n$ endpoints, as $n$ separate unicast transmissions. In this case, the congestion control is no different from the problems considered in the previous sections.

- Multicast: The reason that unicast transmission is not desirable for point-to-multipoint transmission is that this results in wastage of network resources. Consider the Y-network shown in Figure 8. Suppose we need to transmit a packet from node $A$ to receivers at nodes $C$ and $D$. Then, under unicast transmission, two copies of the packet are made at node $A$ and each packet is transmitted from node $A$, one to node $C$ and the other to node $D$. Thus, the link $AB$ must process two packets. On the other hand, under multicast transmission, a single packet is transmitted from node $A$ and when the packet reaches node $B$, two copies of the packet are created and one is sent to node $C$ and the other is sent to $D$. Thus, each link only has to transmit one packet under multicast transmission. In general, if the source $A$ transmits at the rate of $x$ packets, then under unicast transmission, the load on link $AB$ would be $2x$, whereas, under multicast transmission, the load on link AB would only be $x$.

Once the decision has been made to transmit using multicast, there are still two further methods of multicast transmission that one can use to transmit, based on the application:

- Single-rate multicast: In this form of multicast, the intended application is online sharing of continuously-modified documents among a large group of people. For such applications, the data are useful only if all of the data are available at all the receivers. Further, the data should reach the receivers at nearly the same time. Thus, if $x$ is the transmission rate at node $A$ in Figure 8, then the received rates at nodes $C$ and $D$ are also $x$.

- Multirate multicast: This form of multicast is used for real-time transmission, where each receiver can receive data at different rates. For example, suppose that link BC is a 56 kbps link.
modem connection and link BD is a broadband home connection which receives at rate 1 Mbps, and the transmission from the source at node A is real-time video. Then, it would be appropriate to have a mechanism by which the transmitter transmits at a single rate, while the receivers receive at different rates with the high-speed receiver being able to receive a better-quality video than a slow-speed receiver. Such a mechanism for video transmission can be implemented by transmitting the packets carrying the video in “layers.” Packets in the lowest layer are minimally essential to be able to see anything at all at the receiver. Packets at a higher layer enhance the quality of the received video. Thus, low-speed receivers would subscribe to a few lower layers, and high-speed receivers can enhance their reception quality by subscribing to higher layers, in addition to the lower layers. In general, if a link is shared by many users and the transmission rates and the number of users vary with time, the available rate on a link may vary. Thus, a control algorithm is necessary for a receiver to adjust its received rate (by varying the number of video layers to which it subscribes), depending upon the level of congestion in the network. In this section, we will extend the results of the previous sections to cover the case of multirate, multicast algorithms.

Let $R_s$ be the set of receivers corresponding to any session $s \in S$. In the case of unicast sessions, $R_s$ is a singleton set, whereas, for multicast sessions, $|R_s| > 1$. We will use the term virtual session to indicate the connection from a multicast source to one of its receivers. Thus, there are $R_s$ virtual sessions corresponding to each session. We use the notation $(s, r)$ to denote a virtual session corresponding to session $s$. Let $L_{sr}$ be the set of all links in the route of a virtual session $(s, r)$. Let $S_l$ be the set of all sessions passing through link $l$ and let $V_{sl}$ be the set of all virtual sessions of the session $s$ using link $l$.

Let $x_{sr}(t)$ denote the rate at which the virtual session $(s, r)$ sends data at time $t$ and let $U_{sr}(x_{sr})$ be the utility that it derives from this data rate. The total flow in link $l$ at time $t$ is given by

$$\sum_{s \in S_l} \max_{(s, r) \in V_{sl}} x_{sr}(t).$$

The objective is to find rates for which the total utility is maximized. Thus, as in the unicast case, the problem can be posed as the following optimization problem.

$$\max \sum_{s \in S} \sum_{r \in R_s} U_{sr}(x_{sr})$$

subject to

$$\sum_{s \in S_l} \max_{(s, r) \in V_{sl}} x_{sr} \leq C_l, \quad \forall l \in L,$$

$$x_{sr} \geq 0, \quad \forall s \in S, r \in R_s.$$
can be equivalently written as

\[ x_{01} + x_1 \leq C_A \quad \text{and} \quad x_{02} + x_1 \leq C_A. \]

Here \( C_A \) is the capacity of the link. We can thus split the link rate constraint into multiple linear constraints depending on the number of virtual sessions using the link. Suppose that we define a penalty function in a similar vein as in the solution to the unicast problem, and define the rate control mechanism of each of the virtual sessions using the partial derivative of the penalty function. It is easy to see that this would require each link to keep track of the rates of the individual virtual sessions (since each virtual session passing through a link will impose an additional constraint for that link) and thus, this approach is not scalable. To address this problem, we start by approximating the multicast rate control problem from which we can find a stable, decentralized solution for the approximate problem. We will then use this as a heuristic to solve the precise multicast congestion control problem.

Observe that the function \( \max_i (x_i) \) can be approximated by the function \( (\sum x_i^n)^{\frac{1}{n}} \) for large enough \( n \), when all \( x_i \)'s are non-negative real numbers. We now consider the following constraints, where the max functions in (37) are replaced by their differentiable approximations.

\[
\sum_{s \in S_l} \left( \sum_{(s,r) \in V_{sl}} x_{sr}^n \right)^{\frac{1}{n}} \leq C_l, \quad \forall l \in L, \\
x_{sr} \geq 0, \quad \forall s \in S, \; r \in R_s.
\]

(39)

Next, consider a penalty function formulation of the optimization problem (36) subject to (39) in which we maximize the following objective.

\[
V_n(x) = \sum_{s \in S} \sum_{r \in R_s} U_{sr}(x_{sr}) - \sum_{l \in L} \int_0^t \left( \sum_{m \in S_l} (\sum_{(m,j) \in V_{ml}} x_{mj}^n)^{\frac{1}{n}} \right)^{\frac{1}{n}} p_l(y) \, dy.
\]

(40)

Further, consider the following congestion control algorithm to maximize the cost function given by (40).

\[
\dot{x}_{sr} = \kappa_{sr}(x_{sr}) \left( U'_{sr}(x_{sr}(t)) - \sum_{l \in L_{sr}} p_l(\tilde{y}_l) G_{l,(sr)}^{(n)}(x) \right),
\]

(41)

where

\[
\tilde{y}_l = \sum_{m \in S_l} (\sum_{(m,j) \in V_{ml}} x_{mj}^n(t))^{\frac{1}{n}},
\]

and

\[
G_{l,(sr)}^{(n)}(x) = x_{sr}^{n-1} \sum_{(s,j) \in V_{sl}} x_{sj}^n(t)^{\frac{1}{n}-1}.
\]

We now show that the rate control differential equation given by (41) is stable for large but finite \( n \), and the function \( V_n(x) \) provides a Lyapunov function for the system of differential equations described in (41). Here \( x \) denotes the column vector of rates of all the virtual/unicast sessions.
Theorem 11.1 The function $V_n(.)$ is a Lyapunov function for the system described in (41) and this system is globally asymptotically stable. The stable point is the unique $x$ maximizing the strictly concave function $V_n(.)$.

Proof See homework exercise.

The stability of (41) for arbitrarily large $n$ thus motivates us to suggest a set of rate control equations for multicast networks. By considering the limit (see homework exercise)

$$
\lim_{n \to \infty} G^{(n)}_{L,sr}(x),
$$

we arrive at the following rate control equation for the virtual sessions in a multicast network.

$$
\dot{x}_{sr} = \kappa_{sr}(x_{sr}) \left( U'_{sr}(x_{sr}) - \sum_{l \in L_{sr}} p_l(y_l) \frac{I(x_{sr} = \max(s,j) \in V_{sl}(x_{sj}))}{\sum_{(s,j) \in V_{sl}} I(x_{sr} = x_{sj})} \right), \tag{42}
$$

where we recall that $y_l$ is the arrival rate at link $l$. The above rate control equation suggests that a link price is added to the path price of a virtual session only if that virtual session has a maximum rate among all the virtual sessions of its multicast session passing through that link. When more than one multicast receiver has a rate equal to the multicast session rate through the link, the link price should be split equally among all the multicast receivers with rate equal to the multicast session rate. So, from the point of view of a virtual session, it does not have to react to congestion on those links in which its rate is less than the multicast session rate.