Optimizing and Modeling Dynamics in Networks

Ibrahim Matta

March 29, 2013

Contents

1 Introduction

The Internet has grown very large. No one knows exactly how large but rough estimates indicate billions of users (around 1.8B in 2009 according to eTForecasts), hundreds of millions of web sites (over 180M in October 2008 according to netcraft), and hundreds of billions of web pages (around 150B according to the Internet archive).

The Internet is also very dynamic — users log in and out, new services get added, routing policies change, normal traffic gets mixed with DoS attack traffic, etc.

An important question is: *How do we manage such a huge and highly dynamic structure like the Internet?*

As a corollary, how can we build a network of the future unless we understand the *steady-state* and *dynamics* of what we build?

In these notes, we resort to two mathematical frameworks: *optimization theory* to study optimal steady states of networks, and *control theory* to study the dynamic behavior of networks as they evolve toward steady state. Our emphasis will be on *congestion control* using the notion of *prices* to model the level of congestion, e.g., delays, losses, etc. observed by users or traffic sources.

We assume minimal background in calculus and algebra, and rely on intuitive explanations and simple control applications, using examples from the Internet's congestion control. This material has been largely influenced by the work of Frank Kelly and R. Srikant, and control theory texts and notes, see for example:

- Frank Kelly. “Mathematical Modelling of the Internet.” In ”Mathematics Unlimited - 2001 and Beyond” (Editors B. Engquist and W. Schmid), Springer-Verlag, Berlin, 2001, pages 685-702. http://www.statslab.cam.ac.uk/˜frank/mmi.pdf
- R. Srikant. “The Mathematics of Internet Congestion Control.” Birkhauser, 2004. http://www.springer.com/birkhauser/mathematics/book/978-0-8176-3227-4
- Katsuhiko Ogata. “Modern Control Engineering.” Prentice Hall, 2010. books.google.com/books?isbn=0136156738
- Chenyang Lu. “Feedback Control Theory: A Computer System’s Perspective.” Tutorial at: http://www.cs.virginia.edu/˜cl7v/cs851-talks/control_tutorial.ppt

2 Network Control as an Optimization Problem

In this section, we describe Frank Kelly’s optimization framework which models the users’ expectations (requirements) with utility functions and the network congestion signals (e.g., loss, delay) as prices. The network is shown to allocate transmission rates (throughputs) to users (flows) in such a way as to meet some *fairness* objective.

The objective of a user, or what makes the user happy, can be mathematically modeled as a *utility function*. For example, drivers observe the“price” of transportation and make one of many possible decisions: drive, take the subway instead, walk, bike, or stay home. The decision may involve several factors like the price of gas, convenience, travel time, etc. For example, if it rains, you might decide to drive to work, or you might decide to walk to work to save money and can then afford to go to the movies later in the week. Of course, how much the driving a person does is affected by all sorts of factors and priorities is unknown to the the system of gas stations and oil companies. But, each driver has her own utility!

Figure 1 illustrates with a block diagram the closed-loop relationship between drivers (users), gas stations (where gas is sold to and consumed by users), and the market (which represents OPEC, the government, and oil companies that collectively produce gas and set market prices based on user demand). Drivers set the total demand by observing gas prices. Notice that the gas price includes at-the-pump gas prices, and possible other “exogenous” prices like tips for full service, fees for credit card payment, or additional local taxes. Observe also that the prices observed by users are *delayed* and do not typically represent the exact current state of the market given inherent delays in gas production, refinement, transportation, etc.

This kind of block diagram is typical of many closed-loop (feedback) control systems where the system is said to reach *equilibrium* if the demand (for gas by drivers) matches the supply (of gas in the market). In data networks, users drive the demand on the network and have different utilities (expectations) when downloading music, playing games, making skype (voice/video) calls, or denying others service by launching a DoS attack! In turn,

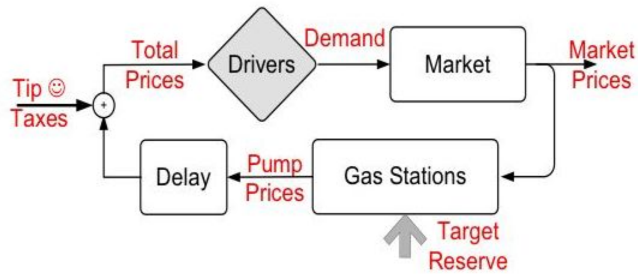


Figure 1: The gas control loop

the network observes that user demand and sets "prices", where the price could be real money, or it could be some measure (indication) of congestion (e.g., delay, loss), or it could represent additional resources that need to be allocated to avoid congestion.

An important question is: *What is the goal of network design?* Is it to make users happy? You hope so! Then, mathematically, we say the goal of the network is to maximize the sum of utilities for all its users. Figure 2 illustrates the data network equivalent of the gas control loop shown in Figure 1. We consider next the modeling of user utility and network behavior (resource allocation), before introducing the optimization framework to study the (optimal) steady state for the users and network.

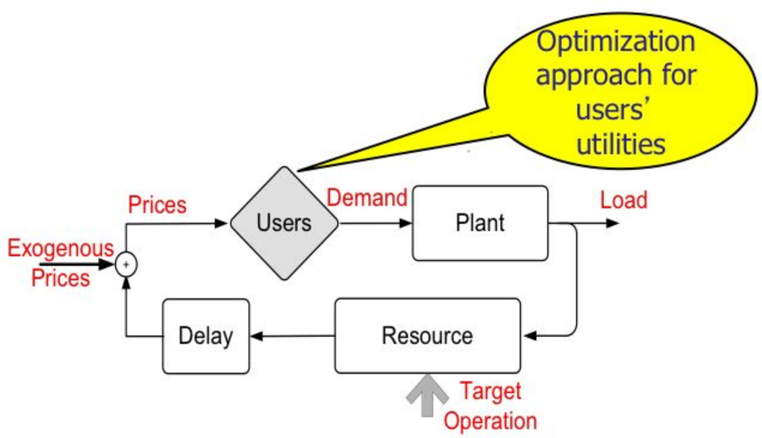


Figure 2: The network control loop

2.1 Modeling the User

Users typically have different utilities, i.e. different applications may perform differently based on the level of service (e.g. loss, delay) they get from the network. But, generally speaking, an application should perform better, the higher the rate (throughput) it is able to send at over the network. It is also generally the case that the gain (level of "happiness") from higher throughput (i.e. *marginal utility*) diminishes as the throughput increases.

Figure 3 shows such a utility function that is typical of what is called *elastic traffic*. Formally, user r has utility $U_r(x_r)$ when allocated rate $x_r > 0$. $U_r(x_r)$ is an increasing, strictly concave function of x_r (see Figure 4). And the derivative $U_r'(x_r) \to \infty$ as $x_r \to 0$, and $U_r'(x_r) \to 0$ as $x_r \to \infty$.

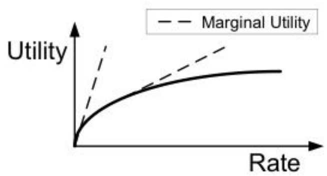


Figure 3: Concave utility function

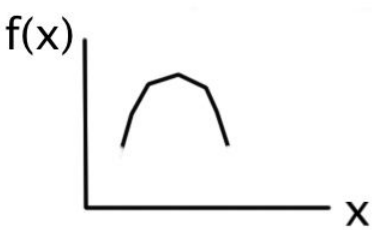


Figure 4: Concave function. A function $f(.)$ is said to be concave if $f(\alpha x_1 + (1-\alpha)x_2) \geq \alpha f(x_1) + (1-\alpha)f(x_2)$, i.e. for any two points x_1 and x_2, the straight line that connects $f(x_1)$ and $f(x_2)$ is always below or equal to the function $f(.)$ itself. Note that the function has a maximum value at some point x_{max}, and that the derivative $f'(x_{max}) = 0$. A strictly concave function would have a strict inequality, whether a convex function has a cup-like shape and has a minimum instead.

2.2 Modeling the Network

We consider a network on J resources, e.g. transmission links as they are typically considered the bottleneck. We denote by R the set of all possible routes, and we assume that each user (source-destination traffic flow) is assigned to exactly one route r (i.e. static single-path routing). We then define a 0-1 routing matrix A such that:

- $a_{ir} = 1$ if resource j is on route r
- $a_{ir} = 0$ otherwise

Figure 5 shows an example with three users (flows colored blue, green, and red) over a network of seven links, so the routing matrix has three columns and seven rows.

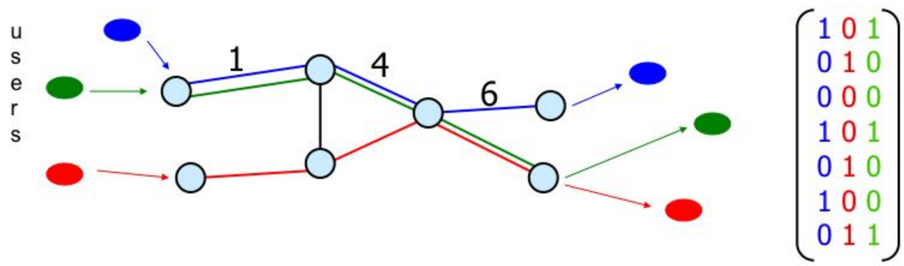


Figure 5: A network model

2.3 The Optimization Problem

Now we are ready to formulate an optimization problem that allows the network to allocate rates to users so that the sum of their utilities is maximized. We refer to this problem as $SYSTEM(U, A, C)$ where the inputs are the user utility functions $U_r(.)$, the routing matrix A, and the vector of link capacities C, and the output is the vector of allocated rates x.

$SYSTEM(U, A, C)$:

$$\max \sum_{r \in R} U_r(x_r)$$
$$\text{subject to } Ax \leq C$$
$$\text{over } x \geq 0$$

For such an optimization problem, it is known that there exists a unique solution. This is the case because the function to optimize is strictly concave and the link capacity inequality constraints $Ax \leq C$ form a so-called *convex set* (see Figure 6.)

The practical challenge in solving this problem however is that the network does not know the utilities of its users, let alone its centralized nature makes it computationally expensive to solve!

To address these challenges, we start by decomposing the problem into R problems, one for each user $r \in R$, and one problem for the network (we will later decompose this network problem further into individual resource problems). The network will present each

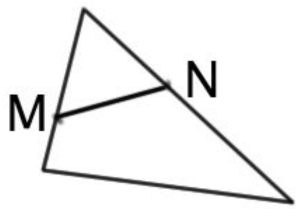


Figure 6: A convex set. A convex set means that any linear combination of any two points M and N located on the boundary of the region formed by the linear inequalities lies within the region itself.

user with a "price" λ_r (\$/bit). Through these prices, the network attempts to infer user utilities. Specifically, observing λ_r, user r will then choose an amount to pay w_r (\$/second) for the service (that maximizes the user's utility), which in turn determines how much rate x_r (bits/second) the user would get ($x_r = w_r/\lambda_r$). The network sets its prices λ_r based on the load x_r $\forall r$.

2.4 Introducing Prices

The decomposed optimization problem can then be stated in terms of the following user optimization problem, and network optimization problem.

$USER_r(U_r, \lambda_r)$:

$$\max U_r(x_r = \frac{w_r}{\lambda_r})$$
$$\text{over } w_r \geq 0$$

Given the network price λ_r and its own *private* utility function U_r, user r determines how much it is willing to pay w_r so as to maximize her own utility.

Knowing the vector $W = \{w_r, \forall r\}$, its routing and capacity matrices, the network allocates user rates x_r by optimizing some network function $f(x, W)$. Once x_r's are obtained, prices are obtained as $\lambda_r = \frac{w_r}{x_r}$.

$NETWORK(A, C, W)$:

$$\max \sum_{r \in R} f(x_r, w_r)$$
$$\text{subject to } Ax \leq C$$
$$\text{over } x \geq 0$$

2.5 Network Optimization

The choice of the network function $f(x, W)$ determines how the capacity of the network gets allocated to users, and so how *fair* we might consider this allocation to be! For example, consider the following function:

$$f = \sum_{r \in R} w_r \; x_r$$

Maximizing this function 1 results in maximizing the total weighted throughput for all users. As a special case, for unit weights, the network optimization problem maximizes the total throughput through the network. This might seem to fly in the face of what we think is fair! Consider the following simple example: In this example, given both links

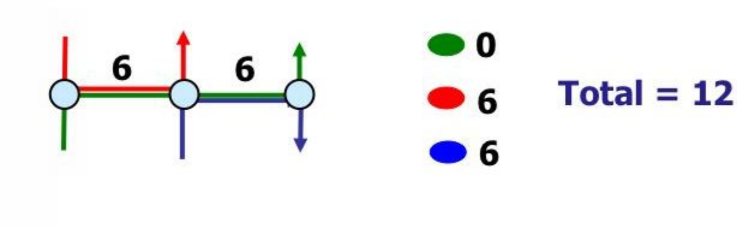


Figure 7: Greedy network allocation

have capacities of 6 units, the total throughput allocated to all users is the total network capacity of 12 units. This can be achieved by allocating 6 units of capacity to each of the 1-link flows (users): the "red" user and the "blue" user, leaving the 2-link ("green") flow with no capacity allocated to its user. That does not seem "fair"! A different function f would allocate rates to users differently and so it would provide a different notion of fairness.

But, the big question is: *how do (should) we define fairness?* The research literature introduces many notions of fairness, most notably the so-called *max-min fairness*.

2.5.1 Max-min Fairness

Intuitively, max-min fairness means we want to allocate resources (links) to users (flows) such that we are:

1. *fair:* all users get equal share of a link, as long as users have the demand that would fully consume their share, and

2. *efficient:* each link is utilized to the maximum load possible

In other words, if a user cannot consume its equal share of a link, the excess capacity must be (recursively) allocated equally among high-demanding users. So, the final outcome is

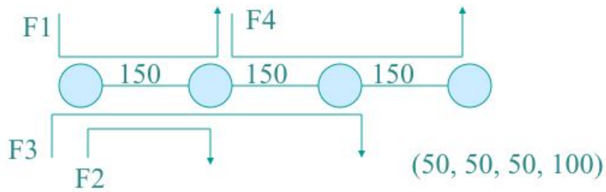


Figure 8: Max-min fair capacity allocation

that low-demanding users get exactly what they need, while high-demanding users get equal allocations. Consider the following multi-link network example: In this example, all links have capacities of 150 units and we assume elastic traffic sources, *i.e.* sources that would consume all what they can get. Starting from the first link, as it is used by most users so it is the most loaded one, each flow using that link gets allocated an equal share of $150/3 = 50$ units. Proceeding to the next loaded link, the middle one, each of its two flows should get an equal share of 75, however flow F_3 is limited by its first link to 50 units of throughput. Thus, flow F_4 gets the left-over from F_3 to a total allocation of $75 + 25 = 100$. The right-most link, at capacity of 150, does not limit the throughput of F_4, which ends up using only 100 units of that link, leaving 50 unused. At the end of this process, we say that the max-min fair allocation vector is $(50,\ 50,\ 50,\ 100)$.

Mathematically, max-min fairness is achieved when the network maximizes the following function:

$$f = \mathrm{min_{r \in R} x_r}$$

Intuitively, maximizing the minimum of allocated rates results in equalizing these rates, as long as users have enough demand that will consume these rates over the network.

2.5.2 Proportional Fairness

Another equally popular fairness definition is the so-called *(weighted) proportional fairness.* This notion of fairness is achieved when the network maximizes the following function:

$$f = \sum_{r \in R} w_r log(x_r)$$

Note that the log function is a concave, and strictly increasing function. Thus, given optimal rate allocation solution x^*, that is feasible, *i.e.* $x^* \geq 0$ and $A\ x^* \leq C$, any other feasible solution x will cause the aggregate proportional change $\sum_{r \in R} w_r \frac{x_r - x_r^*}{x_r^*}$ to be less than or equal zero. To show this, for simplicity, assume one user and unit weight, so $f(x) = log(x)$. Expanding $f(x)$ into its first-order (linear) Taylor's approximation around x^*, we obtain:

$$f(x) \approx f(x^*) + (x - x^*) f'(x^*)$$

Given $f'(x^*) = \frac{1}{x^*}$, we have:

$$f(x) \approx f(x^*) + \frac{(x - x^*)}{x^*}$$

Since f is maximized at x^*, $f(x^*) \geq f(x)$ and so the proportional fairness condition must hold:

$$\frac{x - x^*}{x^*} \leq 0$$

Note that the presence of weight w_r intuitively means that user (flow) r is equivalent to w_r users with unit weight each.

2.5.3 General Parameterized Utility

If the network function $f(x)$ is a function of the utilities of its users $U(x)$, then the network is in fact maximizing a function of the user utilities. Assuming each user r has unit weight w_r, $U_r(x_r)$ can be generalized as:

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

where α is a parameter that determines the fairness criterion of the network. More specifically, if $\alpha \to 0$, then a user's utility is linear in its allocated rate and the network is effectively maximizing the sum of user utilities $\sum_{r \in R} U_r(x_r) = \sum_{r \in R} x_r$, which in turn yields a greedy allocation that maximizes the total throughput over the network.

On the other hand, if $\alpha \to 1$, then this is equivalent to a log utility, yielding proportional fair allocation. To see this, let's take the derivative of $U_r(x_r)$:

$$U_r'(x_r) = \frac{(1-\alpha)x_r^{-\alpha}}{1-\alpha} \to \frac{1}{x_r} \text{ as } \alpha \to 1$$

By integrating $U_r'(x_r)$, we get back $U_r(x_r) = log(x_r)$.

Similarly, it can also be shown that $\alpha \to \infty$ corresponds to a minimum utility, yielding a max-min fair allocation.

2.6 Solution to Optimization Problem

Consider the case where the network is maximizing the weighted sum of the log of user rates, i.e. the network is trying to solve the following optimization problem that would yield a weighted proportional fairness allocation:

$NETWORK(A, C, W)$:

$$\max \sum_{r \in R} w_r log(x_r)$$
$$\text{subject to } Ax \leq C$$
$$\text{over } x \geq 0$$

We can solve this problem using the theory of constrained convex optimization using the Lagrangian technique. Specifically, we move the constraints into the objective function that we want to optimize, thus making the optimization problem effectively unconstrained. We do so by introducing so-called "Lagrangian multipliers" into the new objective (Lagrangian) function L:

$$\max L = \sum_{r \in R} w_r log(x_r) + \lambda^T (C - Ax)$$

λ^T is a Lagrangian vector with a variable λ_j for each link j in the network. Note that L is a strictly convex function, thus a solution exists at which the derivatives of L with respect to each x_r and each λ_j are equal to zero:

$$\begin{aligned} \frac{\partial L}{\partial x_r} &= \frac{w_r}{x_r} - \sum_{j \in r} \lambda_j \\ \frac{\partial L}{\partial \lambda_j} &= (C_j - \sum_{r \in j} x_r) \end{aligned}$$

The notation $j \in r$ indicates all links j used by user (flow/route) r, whereas $r \in j$ denotes all flows r using link j, i.e. the total load on link j.

By equating the first set of equations to zero, we obtain the (weighted proportionally fair) solution[1]:

$$x_r = \frac{w_r}{\sum_{j \in r} \lambda_j}$$

We obtain λ_j by equating the second set of equations to zero. Note that λ_j and $(C_j - \sum_{r \in j} x_r)$ must be greater than or equal to zero since negative values do not maximize the objective function L! Furthermore, $(C_j - \sum_{r \in j} x_r) \geq 0$ ensures that the link capacity constraints $\sum_{r \in j} x_r \leq C_j$ are automatically satisfied. If $(C_j - \sum_{r \in j} x_r) = 0$ then λ_j can be greater than zero. On the other hand, if $\lambda_j = 0$, then the associated link may not be fully utilized, i.e. $\sum_{r \in j} x_r < C_j$. Intuitively, λ_j represents the "cost" associated with link j, so it is zero if the link is under-utilized, and positive if the link is allocated to capacity.

Example: Consider the example in Figure 7 but now assume the network's objective to proportionally allocate its capacity, i.e.,

$$\max f = log(x_0) + log(x_1) + log(x_2)$$

subject to:

$$\begin{aligned} x_0 + x_1 &\leq 6 \\ x_0 + x_2 &\leq 6 \\ x_0, x_1, x_2 &\geq 0 \end{aligned}$$

[1] In optimization theory, this is referred to as Karush-Kuhn-Tucker (KKT) conditions.

where x_0, x_1, and x_2 are the rates allocated to the two-link flow (user), the first-link flow, and the second-link flow, respectively. [2]

Using the Lagrangian's solution method, we obtain:

$$\max L = log(x_0) + log(x_1) + log(x_2) + \lambda_1(6 - (x_0 + x_1)) + \lambda_2(6 - (x_0 + x_2))$$

Taking derivatives, we obtain:

$$\begin{aligned}
\frac{\partial L}{\partial x_0} &= \frac{1}{x_0} - (\lambda_1 + \lambda_2) \\
\frac{\partial L}{\partial x_1} &= \frac{1}{x_1} - \lambda_1 \\
\frac{\partial L}{\partial x_2} &= \frac{1}{x_2} - \lambda_2 \\
\frac{\partial L}{\partial \lambda_1} &= 6 - (x_0 + x_1) \\
\frac{\partial L}{\partial \lambda_2} &= 6 - (x_0 + x_2)
\end{aligned}$$

Equating these derivatives to zero, the last two equations show full utilization of the link capacities and that $x_1 = x_2$, while the first three equations give the following values of x_i's:

$$x_1 = x_2 = \frac{1}{\lambda_1} = \frac{1}{\lambda_2} = \frac{1}{\lambda}$$

$$x_0 = \frac{1}{2\lambda}$$

Substituting in the capacity equations, we obtain the price of each link λ:

$$\frac{1}{2\lambda} + \frac{1}{\lambda} = 6$$

Thus, $\lambda = \frac{1}{4}$, and so $x_0 = 2$, and $x_1 = x_2 = 4$. Note that in this optimal case, each link is fully utilized to capacity, and the flow that traverses two links is charged twice for each link it traverses and so it gets allocated a lower rate.[3] **End Example.**

If the utility of each user r is a log function in its allocated rate x_r, then the (weighted proportionally fair) network solution $x_r = \frac{w_r}{\sum_{j \in r} \lambda_j}$ is in fact, a solution to the whole system optimization problem that includes the network, as well as all users possibly trying to

[2]Note that since the objective (log) function is strictly increasing, then the x_i's should be as large as possible to consume the total capacity of the links, so the two inequalities on link capacities could be turned to equalities.

[3]As we will later see, this proportional rate allocation is what TCP Vegas provides.

independently (in a distributed way) maximize their own log utilities. However, in a distributed setting, as noted earlier, even if the network knows the user utility functions, the network allocates user rates based on their willingness to pay, w_r, which might be unknown to the network. This lack of knowledge can be overcome by observing the demand behavior of the user x_r and the price $\lambda_r = \sum_{j \in r} \lambda_j$, and so w_r is computed as $w_r = x_r/\lambda_r$. Otherwise, the network can just assign some weights w_r to users based on some preference policy.

The moral of the story is that in practice, there is no central network controller that knows W and can then allocate rates to users. Each user and each resource (link) might have its own individual controller that will operate independently and so we need to study the collective behavior of such composite system and answer questions such as: Would the system converge (stabilize) to a solution in the long term (i.e., reaching steady state)? If so, is this solution unique and how far is it from the target (optimal) operating point? In general, if the system gets perturbed, is it stable, i.e. does it converge back to steady state, and how long does it take to converge and how smooth or rough was that? In control-theoretic terminology, we refer to the response to such perturbation until steady state is reached as the *transient response* of the system. We refer to how far the system is from being unstable, or the magnitude of perturbation that renders the system unstable, as *stability margin*.

To formally address these questions, we will resort to the modeling of user and network dynamic behaviors, in the form of differential (or difference) equations, then use well-known control-theoretic techniques to study the overall transient and steady-state behavior of the system.

3 The Control Problem

The basic control problem is to control the output of a system given a certain input. For example, we want to control the user demand (sending rate) given the observed network price (e.g., packet loss or delay). Similarly, we want to control the price advertised by a network resource given the demand (rates) of its users.

There is basically two kinds of control: open-loop control, and closed-loop (feedback) control. In open-loop control systems, there is no feedback about the state of the system and the output of the system is controlled directly by the input signal. This type of control is thus simple, but not as common as closed-loop control. An example of open-loop control system is a microwave that heats food for the input (specified) duration.

Feedback (closed-loop) control is more interesting and multiple controllers may be present in the same control loop. See Figure 2 where a user controller is present to control demand based on price, and a resource controller is also present to control price based on demand. Feedback control makes it possible to control the system well even if we can't observe or know everything, or if we make errors in our estimation (modeling) of the current state of the system, or if things change. This is because we can continually measure and correct (adapt) to what we observe (i.e., feedback signal). For example, in a congestion control system, we do not need to *exactly* know the number of users, the arrival rate of connections, or the service rate of the bottleneck resource, since each user would adapt its demand based on its own observed (measured, fed back) price, which reflects the current overall congestion of the bottleneck resource.

Associated with feedback control is a delay to observe the feedback (measured) signal, which is referred to as *feedback delay*. More precisely, feedback delay refers to the time taken from the generation of a control signal (e.g., updated user demand) until the process/system reacts to it (e.g. demand routed over the network), this reaction takes effect at each resource (e.g. load on each link), and this reaction is fed back to the controller (e.g. price observed by the user).

3.1 System Models

Models of controlled systems can be classified along four dimensions:

- *Deterministic versus stochastic models.* The latter models capture stochastic effects like noise and uncertainties.
- *Time-invariant versus time-varying models.* The latter models contain system parameters that change over time.
- *Continuous-time versus discrete-time models.* In the latter models, time is divided into discrete-time steps.
- *Linear versus non-linear models.* The latter models contain non-linear dynamics.

In most of our treatment, we consider the simplest kind of models that deterministic, time-invariant, continuous-time, and linear. In modeling a controlled system, we characterize the relationships among system variables as a function of time, i.e., dynamic equations. See Figure 9 where functions f and h are generally non-linear functions. As we will see later, for mathematical tractability, we often linearize dynamic non-linear models.

3.2 Modeling Source and Network Dynamics

Consider a source r with log utility, i.e. $U_r(t) = w_r log(x_r)$, and a network that allocates rates in a weighted proportional fashion. We saw earlier that in steady state, the (optimal) solution (referred to as the KKT condition) is:

$$x_r = \frac{w_r}{\lambda_r} \tag{1}$$

This can be re-written as $w_r - x_r\lambda_r = 0$. Also, we saw that the optimal solution ensures that each link l is fully utilized, i.e. the load (total input rate) on link l, denoted by $y_l = \sum_{s:l\in s} x_s$, equals the link capacity c_l.

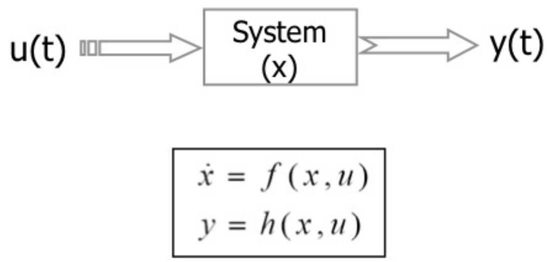


Figure 9: Typical system model.

The dynamics of the sources and links can then be modeled such that these steady-state user rates and link loads are achieved. Specifically, we can write the dynamic (time-dependent) source algorithm as:

$$\dot{x}_r(t) = k(w_r - x_r(t)\lambda_r(t)) \tag{2}$$

where k is a proportionality factor. Note that w_r represents how much user r is willing to pay, whereas $x_r(t)\lambda_r(t)$ represents the cost (price) of sending at that rate. Intuitively, the user sending rate increases (decreases) when the difference between these two quantities is positive (negative). And in steady state, $\dot{x}_r(\infty) \to 0$, and so we obtain the steady-state solution $x_r = \frac{w_r}{\lambda_r}$ (as expected).

Given that the derivative of $U_r(t)$, $U_r'(t) = \frac{w_r}{x_r}$, the source rate adaptation algorithm can be re-written as:

$$\dot{x}_r(t) = kx_r(t)(U_r'(t) - \lambda_r(t))$$

$$\dot{x}_r(t) = K(t)(U_r'(t) - \lambda_r(t)) \tag{3}$$

Intuitively, the user increases its sending rate if the marginal utility (satisfaction) is higher than the price that the user will pay, otherwise the user decreases its sending rate.

We can also write a dynamic equation for the adaptation in the link price $\lambda^l(t)$, called the link pricing algorithm:

$$\dot{\lambda}^l(t) = h(y_l(t) - c_l) \tag{4}$$

where h is a proportionality factor, and the total price, $\lambda_r(t)$, for user r, is the sum of the link prices along the user's route, i.e. $\lambda_r(t) = \sum_{l:l \in r} \lambda^l(t)$. Intuitively, the link price increases if the link is over-utilized (i.e. $y_l(t) > c_l$), otherwise the link price decreases. Note that at steady state, $\dot{\lambda}^l(\infty) \to 0$, and we obtain the steady-state optimal solution $y_l = c_l$ (as expected).

It turns out that the source and link algorithms, Equations 3 and 4, represent general user and resource adaptation algorithms that collectively determine the transient and steady behavior of the whole system. In what follows, we use the form of Equation 3 to reserve engineer different versions of TCP and deduce the utility function that the TCP source tries to maximize.

3.3 TCP and RED

Many analytical studies considered the network system composed of TCP sources over a network of queues that employ a certain queue management policy. Examples of TCP variants include Reno, SACK, NewReno, Vegas, FAST, etc. Examples of queue management policies include Drop Tail, RED, FRED, REM, PI, etc. One of the most widely studied instantiations is that of TCP sources over a RED bottleneck queue — see Figure 10.

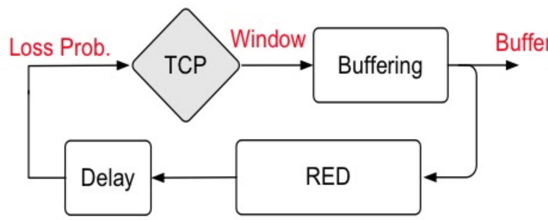


Figure 10: TCP over RED feedback control system.

First, consider the modeling of TCP Reno, where the congestion window *cwnd* is increased by $1/cwnd$ for every acknowledged TCP segment (i.e. non-loss), i.e. it is (roughly) increased by 1 every round-trip time, and decreased by half for every loss. Thus, we can write the following equation for changes in the congestion window of a single TCP flow, where p is the segment loss probability:

$$\Delta cwnd = \frac{1}{cwnd}(1-p) - \frac{cwnd}{2}p$$

Let x denote the sending rate, and T the round-trip time, thus $x = \frac{cwnd}{T}$. Assuming acknowledgments (ACKs) come equally spaced, the time between ACKs (or lack thereof) is given by $\frac{T}{cwnd}$. Thus, we can re-write the above equation in terms of change in sending rate as:

$$\Delta \frac{cwnd}{T} = \frac{(\frac{1}{cwnd}(1-p) - \frac{cwnd}{2}p)/T}{\frac{T}{cwnd}}$$

$$\frac{d}{dt}x(t) = \frac{(\frac{1}{x(t)T^2}(1-p(t)) - \frac{x(t)}{2}p(t))}{\frac{1}{x(t)}}$$

$$\frac{d}{dt}x(t) = \frac{1}{T^2}(1-p(t)) - \frac{x(t)^2}{2}p(t)$$

$$\frac{d}{dt}x(t) = \frac{1}{T^2}(1-p(t)) - \frac{x(t)^2}{2}p(t)$$

$$\frac{d}{dt}x(t) = \frac{1}{T^2} - (\frac{1}{T^2} + \frac{x(t)^2}{2})p(t) \tag{5}$$

Let's denote the loss probability $p(t)$ of TCP connection r as $p_r(t)$. $p_r(t)$ depends on the current load on the path r, and can be approximated by the sum of the loss probabilities experienced on the individual links $j \in r$ along the connection's path. More specifically,

$$p_r(t) = \sum_{i \in r} p_j(\sum_{s:j \in s} x_s(t))$$

Assuming small p such that $(1-p) \approx 1$, we can rewrite Equation 6 as follows:

$$\frac{d}{dt}x(t) = \frac{1}{T^2} - \frac{x(t)^2}{2}p(t)$$

$$\frac{d}{dt}x(t) = \frac{x(t)^2}{2}(\frac{2}{T^2 x(t)^2} - p(t)) \tag{6}$$

Comparing Equation 6 with Equation 3, we can deduce the utility function of a TCP Reno source:

$$\dot{U}(x) = \frac{2}{T^2 x^2}$$

Integrating $\dot{U}(x)$ we get:

$$U(x) = \frac{-2}{T^2 x}$$

Observe that maximizing Reno's utility results in minimizing the quantity $\frac{1}{x}$, which can be viewed as the "potential delay" as it is inversely proportional to the allocated rate x. Thus, this allocation is referred to as *minimum potential delay fair allocation.*

Example: Revisting the example in Figure 7 but now assume the network's objective is to allocate its capacity according to the minimum potential delay fair allocation, i.e.,

$$\max\ f = \frac{-1}{x_0} + \frac{-1}{x_1} + \frac{-1}{x_2}$$

subject to:

$$\begin{aligned} x_0 + x_1 &\leq 6 \\ x_0 + x_2 &\leq 6 \\ x_0, x_1, x_2 &\geq 0 \end{aligned}$$

where x_0, x_1, and x_2 are the rates allocated to the two-link flow (user), the first-link flow, and the second-link flow, respectively.

Using the Lagrangian's solution method, we obtain:

$$\max\ L = \frac{-1}{x_0} + \frac{-1}{x_1} + \frac{-1}{x_2} + \lambda_1(6 - (x_0 + x_1)) + \lambda_2(6 - (x_0 + x_2))$$

Taking derivatives, we obtain:

$$\begin{aligned} \tfrac{\partial L}{\partial x_0} &= \tfrac{1}{x_0^2} - (\lambda_1 + \lambda_2) \\ \tfrac{\partial L}{\partial x_1} &= \tfrac{1}{x_1^2} - \lambda_1 \\ \tfrac{\partial L}{\partial x_2} &= \tfrac{1}{x_2^2} - \lambda_2 \\ \tfrac{\partial L}{\partial \lambda_1} &= 6 - (x_0 + x_1) \\ \tfrac{\partial L}{\partial \lambda_2} &= 6 - (x_0 + x_2) \end{aligned}$$

Equating these derivatives to zero, the last two equations show full utilization of the link capacities and that $x_1 = x_2$, while the first three equations give the following values of x_i's:

$$x_1 = x_2 = \frac{1}{\sqrt{\lambda_1}} = \frac{1}{\sqrt{\lambda_2}} = \frac{1}{\sqrt{\lambda}}$$

$$x_0 = \frac{1}{\sqrt{2\lambda}}$$

Substituting in the capacity equations, we obtain the price of each link $\lambda = 0.08$, and so $x_0 \approx 2.5$, and $x_1 = x_2 \approx 3.5$.

Note that in this optimal case, each link is fully utilized to capacity, and the rate allocated to a flow is inversely proportional to the square-root of the price it observes along its path.

Note also that this captures the well-known *steady-state* relationship between the throughput of a TCP Reno source and the inverse of the square-root of the *loss probability* observed by the TCP source. A TCP Reno source adapting based on Equation 6 would converge to such steady-state throughput value. **End Example.**

Now, let us consider the modeling of another version of TCP — TCP Vegas. This version, unlike Reno, tries to *avoid* congestion, rather than inducing loss and adapting the transmission (congestion) window to it. The basic idea behind Vegas is to calculate the *actual* throughput of the connection as $\frac{w(t)}{R(t)}$, where $w(t)$ is the current window size, and $R(t)$ is the measured round-trip time (RTT) over the connection's path. This RTT includes queueing delay, as well as propagation delay D. Ideally, with no congestion, the ideal throughput can be computed by the source as $\frac{w(t)}{D}$, where D is estimated using the minimum RTT recently observed by the source. To ensure high utilization of the network, we want some queueing, i.e. the actual throughput is lower than the ideal one, but not too low to start causing congestion (i.e. buffer overflow at the bottleneck link and so losses). Vegas then adapts $w(t)$ based on some target difference, α, between the actual throughput and the ideal one. More specifically, the window increases if $(\frac{w(t)}{D} - \frac{w(t)}{R(t)}) < \alpha$, decreases if $(\frac{w(t)}{D} - \frac{w(t)}{R(t)}) > \alpha$, and stays the same otherwise. This dynamic source behavior, i.e. change in window over time, can be modeled as:

$$\frac{dw(t)}{dt} = k(\alpha - (\frac{w(t)}{D} - \frac{w(t)}{R(t)}))$$

This can be re-written as:

$$\frac{dw(t)}{dt} = \frac{k}{D}(\alpha D - (w(t) - \frac{w(t)}{R(t)}D))$$

Denoting the sending rate (throughput) by $x(t) = \frac{w(t)}{R(t)}$, and $\gamma = \frac{k}{D}$, we have:

$$\frac{dw(t)}{dt} = \gamma(\alpha D - (w(t) - x(t)D))$$

At steady state, as $\dot{w}(\infty) \to 0$, we have:

$$w - xD = \alpha D$$

$$xR - xD = \alpha D$$

Denoting the queueing delay by Q, we have $R = Q + D$, and so:

$$xQ = \alpha D$$

$$x = \frac{\alpha D}{Q}$$

Comparing with Equation 1, we can deduce that the willingness to pay w_r for a Vegas user r is αD and that the price experienced by the user is the queueing delay Q.

Now, to deduce the utility function that a Vegas user tries to maximize, let us write its rate adaptation equation following Equation 2:

$$\dot{x}_r(t) = k(\alpha D - x_r(t)Q(t))$$

$$\dot{x}_r(t) = K(t)(\frac{\alpha D}{x_r(t)} - Q(t))$$

Thus, comparing with Equation 3, we deduce:

$$U'_r(t) = \frac{\alpha D}{x_r(t)}$$

Integrating, we obtain:

$$U_r(t) = \alpha D \ log(x_r(t))$$

Recall that maximizing such user utilities results in a weighted proportional fair allocation.

Let us now consider the modeling of the buffer and associated RED queue management algorithm. Figure 11 shows how RED tries to avoid congestion by dropping (or marking) packets with probablity p_c as a (non-linear) function of the average queue length v. First,

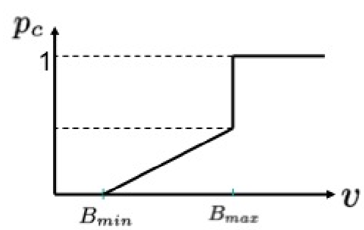


Figure 11: RED dropping (or marking) function.

we model the evolution of the queue length $b(t)$ as a function of the total input rate, $y(t) = \sum x_s(t)$, and (bottleneck) link capacity, C:

$$\dot{b}(t) = y(t) - C$$

Denoting by $v(t)$, the EWMA of the queue length:

$$v(t + \delta) = (1 - \alpha)v(t) + \alpha b(t)$$

$$v(t+\delta) - v(t) = \alpha(b(t) - v(t))$$

Given $v(t)$ gets updated at the link rate, i.e. $\delta = \frac{1}{C}$, and $\dot{v}(t) = \frac{v(t+\delta)-v(t)}{\delta}$, we have:

$$\dot{v}(t) = \alpha C(b(t) - v(t))$$

This last equation represents the dynamic model of RED averaging, which in turn determines the price $p_c(t)$ that users experience.

Ignoring RED averaging and the (hard) non-linearities of the RED function, and assuming the price is set in proportion to the actual queue length, we have:

$$p_c(t) = hb(t)$$

$$\dot{p}_c(t) = h\dot{b}(t) = h(y(t) - C)$$

Comparing with Equation 4, the packet dropping (congestion marking) probability, $\dot{p}_c(t)$, represents the "price", i.e. Lagrangian multiplier, observed by the users of this buffer. Note that at steady state, $\dot{p}_c(\infty) \to 0$, and so $y = C$, i.e. the link is fully utilized at steady state.

3.4 Solving the Feedback Control System

We have developed dynamic (time-dependent) models for users (sources), e.g. TCP, and the network (links), e.g. RED, and the interaction between them through prices. The next step is to solve for the transient and steady-state performance of such system. Solving such systems is challenging because of inherent non-linearilities, e.g. the "hard" non-linearities (discontinuities) in the RED pricing function, or the "soft" non-linearity of TCP where the sending rate changes quadratically in the current rate. Non-linear control theory becomes a useful tool as it deals directly with non-linear differential equations. Specifically, a method called *Lyapunov* allows one to study convergence (stability) by showing that the value of some positive function of the state of the system continuously decreases as the system evolves over time. Finding such a Lyapunov function can be challenging, and transient performance can often only be obtained by solving the system equations numerically.

To this end, a technique called *linearization* can prove more tractable where the non-linear system is approximated by a set of *linear* equations around a single operating point (state). See Figure 12. WIth linearization, we become concerned with *local stability* and study perturbations around the operating point using standard (linear) control theory. By local stability, we mean that if the system is perturbed within a small region around the operating point then the system will converge and stabilize back to that point. This is in contrast to *global stability* where the original (non-linear) system is shown to converge from *any* starting state. To linearize the non-linear system around an operating point, the

Figure 12: Linearization.

basic idea is to expand the non-linear differential equation into a Taylor series around that point and then ignore high-order terms.

In what follows, we briefly review some basics of classical control theory for linear systems, then we introduce non-linear control theory. We also show examples of control theoretic analysis for the dynamic models introduced above.

4 Linear Control Theory

In linear control theory, we transform differential equations in the *time domain* to algebraic equations in the so-called *frequency* or *Laplace* domains. Once this Laplace transformation is done, we use simple algebra to study the performance of the system without the need for going back to the (complicated) time domain. Specifically, we can transform a function $f(t)$ to an algebraic function $F(s)$, referred to as the Laplace transform of $f(t)$, as follows:

$$F(s) = \int_0^\infty f(t) e^{-st} dt$$

where s is a complex variable: $s = \sigma + j\omega$, σ is the real part of s, denoted by $Re(s)$, and ω is the imaginary part of s, denoted by $Im(s)$.

Example (Unit step function): The Laplace transform of a unit step function $u(t)$, where $u(t) = 1$ if $t > 0$, and $u(t) = 0$ otherwise, is given by:

$$U(s) = \int_0^\infty 1.e^{-st} dt = \frac{1}{s}$$

Example (Impulse function): The Laplace transform of a unit impulse function $\delta(t)$, where $\delta(t) = 1$ if $t = 0$, and $\delta(t) = 0$ otherwise, is given by:

$$U(s) = \int_0^\infty 1.e^{-st} dt = e^0 = 1$$

The following are basic Laplace transforms:

The following are basic composition rules, where $L[f(t)]$ denotes the Laplace transform of $f(t)$, i.e. $F(s)$.

Table 1: Basic Laplace transforms

Impulse input: $f(t) = \delta(t)$	$F(s) = 1$
Step input: $f(t) = a.1(t)$	$F(s) = a/s$
Ramp input: $f(t) = a.t$	$F(s) = a/s^2$
Exponential: $f(t) = e^{at}$	$F(s) = 1/(s-a)$
Sinusoid input: $f(t) = sin(at)$	$F(s) = a/(s^2 + a^2)$

Table 2: Composition rules

Linearity: $L[a\ f(t) + b\ g(t)] = aF(s) + bG(s)$
Differentiation: $L[df(t)/dt] = sF(s) - f(0) = sF(s)$ if $f(0) = 0$
Integration: $L[\int f(\tau)d\tau] = F(s)/s$
Convolution: $y(t) = g(t) * u(t) = \int_0^t g(t-\tau)u(\tau)d\tau \Rightarrow Y(s) = G(s)U(s)$

Example: Consider the following second-order linear, time-invariant differential equation, where $y(t)$ represents the output of a system, and $u(t)$ represents the input:

$$a_2\ddot{y}(t) + a_1\dot{y}(t) + a_0 y(t) = b_1\dot{u}(t) + b_0 u(t)$$

In the time domain, if we represent the system by $g(t)$, then $y(t)$ can be obtained by convolving $u(t)$ with $g(t)$, i.e. $y(t) = g(t) * u(t)$. This involves a complicated integration over the system responses, $g(t-\tau)$, to impulse inputs of magnitude $u(\tau)$, for all $0 < \tau < t$.

Assuming $y(0) = u(0) = 0$, taking the Laplace transform of both sides, we obtain:

$$a_2 s^2 Y(s) + a_1 s Y(s) + a_0 Y(s) = b_1 s U(s) + b_0 U(s)$$

$$Y(s) = \frac{(b_1 s + b_0)}{(a_2 s^2 + a_1 s + a_0)} U(s) = G(s)U(s)$$

Thus, in the Laplace domain, the output $Y(s)$ can be obtained by simply multiplying $G(s)$, called the *transfer function* of the system, with $U(s)$. We can then take the inverse Laplace transform, $L^{-1}[Y(s)]$, to obtain $y(t)$, or as we will later see, we can simply analyze the stability of the system by examining the roots of the denominator of the transfer function $G(s)$ and their location in the complex s-domain.

Note that because $Y(s) = G(s)$ for an impulse input, i.e. $U(s) = 1$, the transfer function $G(s)$ is also called *impulse response function*.

4.1 Modeling a Vegas-like System

Consider the system in Figure 13 where a controller C is used to match the queue length $b(t)$ to a target B_r by controlling the input window size $w(t)$. The output rate from the queue is denoted by $d(t)$. The goal is to first write down the differential equations that

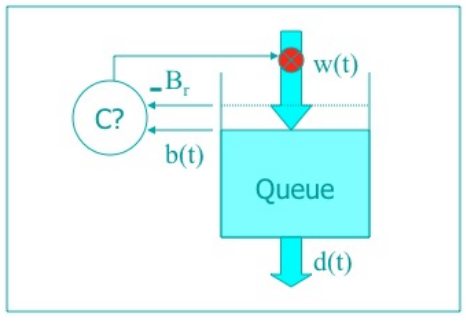


Figure 13: Vegas-like system

model the different components of the system, then instead of solving the equations in the time domain, we will transform them to the Laplace domain and analyze the stability of the system algebraically. This

We start by describing the buffer evolution as:

$$\frac{d}{dt}b(t) = w(t) - d(t)$$

Then, $w(t)$ is the output of convolving the error $e(t) = B_r - b(t)$ with the controller function $C(t)$, i.e.

$$w(t) = C(t) * e(t)$$

Now, taking the Laplace transforms, we obtain:

$$sB(s) = W(s) - D(s) \Rightarrow B(s) = \frac{W(s) - D(s)}{s}$$

$$W(s) = C(s)E(s) = C(s)(B_r(s) - B(s))$$

Figure 14 shows the system using its transfer functions and their input/output flows, where $G_0 = \frac{1}{s}$. This is called the *block diagram* and provides a powerful pictorial tool. From the block diagram, one can write the algebraic equation of the output in terms of the input(s). Dropping the "s" variable for convenience:

$$\frac{(B_r - B)C - D}{s} = B$$

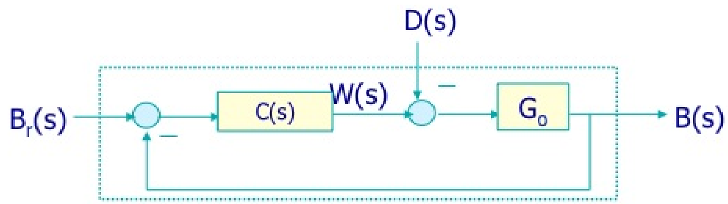


Figure 14: Block diagram of Vegas-like system

Rearranging, we get:

$$B = \frac{C}{s+C}B_r - \frac{1}{s+C}D \tag{7}$$

Note that the system has two inputs: $B_r(s)$ and $D(s)$, subjected to two transfer functions, $\frac{C(s)}{s+C(s)}$ and $-\frac{1}{s+C(s)}$, respectively, and adding their responses we obtain the output $B(s)$.

4.2 Proportional Control and Stability of Vegas-like System

One basic controller C is referred to as *Proportional (P)* controller where the controlled variable $w(t)$ is simply set in proportion to the error signal, i.e. $w(t) = K_p\ e(t)$. In this case, $C(s)$ is simply the constant K_p.

Substituting in Equation 7, we have:

$$B(s) = \frac{K_p}{s+K_p}B_r(s) - \frac{1}{s+K_p}D(s) \tag{8}$$

An important question is: *does the P-controller make the system stable?* More precisely, if we subject the system to impulse input(s), does the system converge back to a quiescent state? Control theory gives a systematic way to answer such stability question by examining the roots of the denominator of the system's transfer function, called the *characteristic equation*. In this case, the characteristic equation is:

$$s + K_p = 0 \Rightarrow s = -K_p$$

The system is stable if the roots (also called *poles*) lie in the lefthand side of the complex s-plane. Thus this system is stable if $-K_p < 0 \Rightarrow K_p > 0$.

Note that we did not need to go back to the time domain to analyze the stability of the system. But let's do that here to understand why poles on the lefthand side of the s-plane makes the system stable. Taking the inverse Laplace transform of Equation 7, and assuming impulse inputs, i.e. $B_r(s) = D(s) = 1$, we get:

$$b(t) = K_p e^{-K_p t} - e^{-K_p t}$$

We can then see that $b(t)$ decays exponentially over time, starting from $b(0) = (K_p - 1)$. We say the system is stable or exhibits *overdamped response*.

We can also analyze transient performance by noting that $b(0) = (K_p - 1)$ represents an *overshoot* response to the impulse input, and that this overshoot is lower for lower K_p. So by controlling K_p, referred to as the *controller gain*, we can also control the system's transient response.

4.3 Proportional Integral Control and Stability of Vegas-like System

Another type of controller is known as *Proportional Integral (PI)* controller where the controlled variable $w(t)$ is set in proportion to the integral of the error signal, i.e. $w(t) = K_i \int e(t)$. In this case, taking the Laplace transform, $C(s) = \frac{K_i}{s}$. Note that the integration means that the history of the error is used to control $w(t)$.

Substituting in Equation 7, we have:

$$B(s) = \frac{K_i}{s^2 + K_i} B_r(s) - \frac{s}{s^2 + K_i} D(s) \tag{9}$$

To analyze stability, we again examine the poles of the characteristic equation:

$$s^2 + K_i = 0 \Rightarrow s = \pm j\sqrt{K_i}$$

Given $K_i > 0$, the two imaginary conjugate poles lie in the lefthand side of the complex s-plane, and so the system is stable, though *critically* stable as we explain next.

To convince ourselves, let us go back to the time domain by taking the inverse Laplace transform:

$$L^{-1}[\frac{K_i}{s^2 + K_i}] = L^{-1}[\frac{K_i}{(s - j\sqrt{K_i})(s + j\sqrt{K_i})}] = L^{-1}[\frac{A}{(s - j\sqrt{K_i})} + \frac{B}{(s + j\sqrt{K_i})}]$$

And some some values of A and B, this yields:

$$Ae^{j\sqrt{K_i}t} + Be^{-j\sqrt{K_i}t}$$

Given the fact that $e^{j\theta} = cos\theta + j\ sin\theta$, the function in the time domain oscillates in a sinusoidal fashion. Although the time function does not decay over time, it does *not* diverge, i.e it is not unstable! So, we consider such a system to have bounded oscillations in response to impulse input and we say it is *critically (or marginally) stable* or the system exhibits *undamped oscillatory response*.

Note that a higher value of K_i results in more oscillatory behavior.

4.4 Stability

More formally, a linear time-invariant system is stable if all poles of its transfer function are in the lefthand side of the s-plane, i.e. the real part of all poles is negative. Figure 15 shows the time response given the location of the poles.

Note that if the poles are complex conjugates and strictly in the lefthand side of the s-plane, the system is stable as oscillations in response to impulse input decay over time, and we say the system exhibits *underdamped response*.

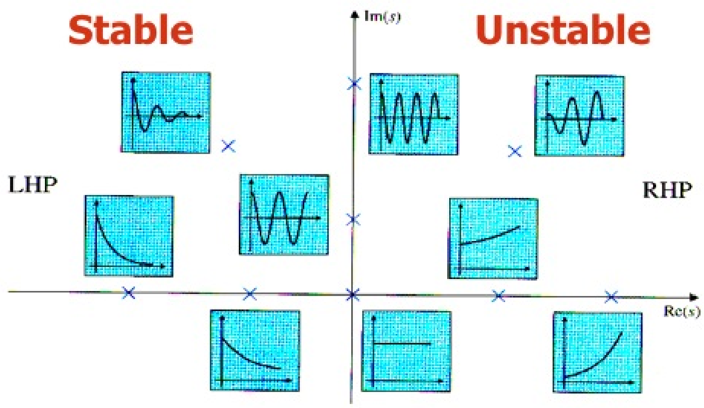


Figure 15: Time response depending on pole location.

4.5 Transient Performance and Steady-state Error

Besides stability, there are other performance metrics of interest that characterize the transient performance of the system and the quality of the steady state. Figure 16 shows several of these metrics, including the time for the controlled variable to reach its peak (maximum) value, the time to reach the target, the maximum overshoot over the steady-state value, and the error that remains at steady state when the system stabilizes away from the desired target value.

Figure 17 illustrates different configurations exhibiting different performance for our Vegas-like system. The controlled variable is the window size, i.e. number of packets allowed into the system. The response is the queue length, which we measure and compare to the target buffer size. A "good" system is one that converges quickly to the desired target with minimum oscillations (i.e., overshoots and undershoots) and with almost zero steady-state error.

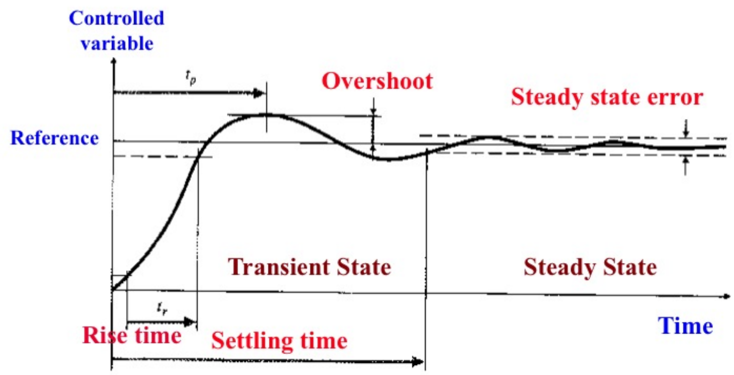


Figure 16: Performance specifications

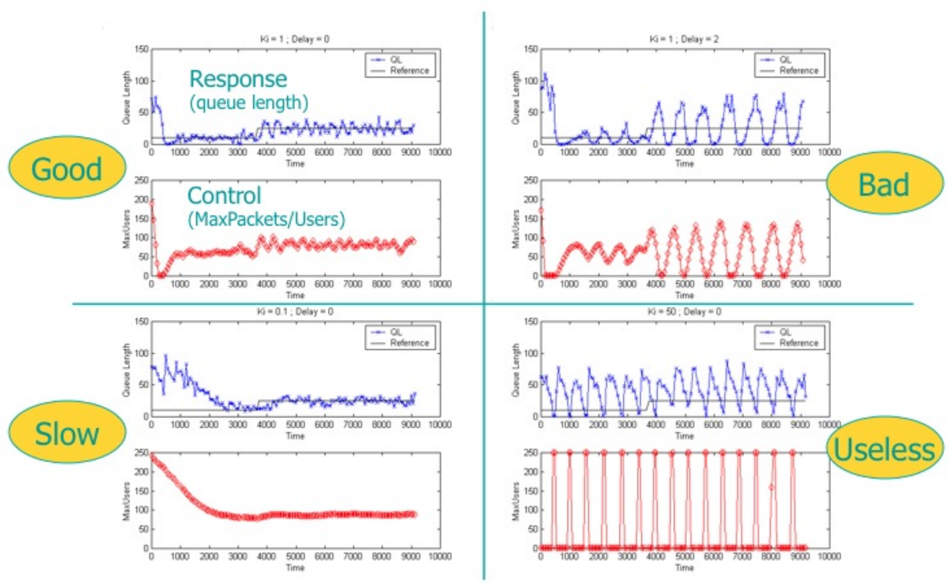


Figure 17: Control and response

4.6 Steady-state Error

In control theory, the steady-state error of a stable system is defined as:

$$e_{ss} = lim_{t \to \infty}\ e(t) = lim_{t \to \infty}\ (r(t) - y(t))$$

where $r(t)$ is the reference input, and $y(t)$ is the system output (response). This error reflects how accurately the system can achieve the desired target, which is chosen to be a step input.

We state without proof the Final Value Theorem:

$$e_{ss} = lim_{t \to \infty}\ e(t) = lim_{s \to 0}\ s\ E(s)$$

This theorem allows us to calculate e_{ss} algebraically in the Laplace domain.

Example (P-control of Vegas-like system):

$$E(s) = B_r(s) - B(s)$$

Substituting for $B(s)$ from Equation 8 and using the Final Value Theorem, we obtain:

$$e_{ss} = lim_{s \to 0}\ s\ (B_r(s) - \frac{K_p}{s + K_p} B_r(s) + \frac{1}{s + K_p} D(s))$$

Assuming step inputs, i.e. $B_r(s) = \frac{B_r}{s}$ and $D(s) = \frac{D}{s}$, we have:

$$e_{ss} = lim_{s \to 0}\ (B_r - \frac{K_p}{s + K_p} B_r + \frac{1}{s + K_p} D) = \frac{D}{K_p}$$

Recall that under the P-controller, the system is (over-damped) stable, i.e. $b(t)$ approaches the target B_r without oscillations, however, at steady state, $b(t)$ misses the target by $\frac{D}{K_p}$ and stabilizes at a value lower than B_r. Notice that the higher the service capacity D is, the larger the steady-state error. So, to decrease the steady-state error, the controller gain K_p could be increased. However, increasing K_p increases the overshoot. A tradeoff clearly exists between transient performance and steady-state performance, and one has to choose K_p to balance the two and meet desired operation goals. **End Example.**

Example (PI-control of Vegas-like system):

$$E(s) = B_r(s) - B(s)$$

Substituting for $B(s)$ from Equation 9 and using the Final Value Theorem, we obtain:

$$e_{ss} = lim_{s \to 0}\ s\ (B_r(s) - \frac{K_i}{s^2 + K_i} B_r(s) + \frac{s}{s^2 + K_i} D(s))$$

Assuming step inputs, i.e. $B_r(s) = \frac{B_r}{s}$ and $D(s) = \frac{D}{s}$, we have:

$$e_{ss} = lim_{s \to 0} \ (B_r - \frac{K_i}{s^2 + K_i} B_r + \frac{s}{s^2 + K_i} D) = 0$$

Although the steady-state error is zero under the PI-controller, recall that the system is critically stable, i.e. it converges to the target while oscillating. Decreasing the controller gain K_i decreases these oscillations, however at the expense of longer time to reach steady state. This illustrates again the inherent tradeoff between transient performance and the quality of the steady state.

5 Analyzing the Stability of Non-linear Systems

As we have seen, linear control theory can be applied to non-linear systems if we assume a small range of operation around which the system behavior is linear. This linear analysis is simple to use, and the system, if stable, has a unique equilibrium point.

On the other hand, most control systems are non-linear, and operates over a wide range of parameters, and multiple equilibrium points may exist. In this case, non-linear control theory could be more complex to use.

In what follows, we first consider a non-linear model of the adaptation of sources and network, and use a non-linear control-theoretic stability analysis method, called *Lyapunov method*. Then, we linearize the system and illustrate the application of linear control-theoretic analysis.

5.1 Solving Non-linear Differential Equations

Recall Vegas-like source adaptations from Equation 2:

$$\frac{dx_r(t)}{dt} = k(w_r - x_r(t)p_r(t))$$

where $p_r(t)$ represents the total price observed by user r along its path. Note that this differential equation is non-linear since $p_r(t)$ is a function of the rates $x_s(t)$:

$$p_r(t) = \sum_{\text{link } l \in \text{route } r} p_l(t) = \sum_{l \in r} p_l(\sum_{s:l \in s} x_s(t))$$

We assume that the pricing function $p_l(y)$ is monotonically increasing in the load y.

At steady state, if the system stabilizes, setting the derivatives to 0, we obtain the steady-state solution:

$$k(w_r - x_r(t)p_r(t)) = 0 \Rightarrow x_r = \frac{w_r}{p_r} = \frac{w_r}{\sum_{l \in r} p_l}$$

To prove stability, we use the non-linear method of Lyapunov. The basic idea is to find a positive scalar function $V(x(t))$, we call the Lyapunov function, and show that the function monotonically increases (or decreases) over time, approaching the steady-state solution.

Define $V(x)$ as follows:

$$V(x) = \sum_{r \in R} w_r log(x_r) - \sum_{j \in J} \int_0^{\sum_{s:j \in s} x_s} p_j(y) dy$$

Finding a suitable Lyapunov function that shows stability is tricky and more of an art! If you can't find one, it does *not* mean that the system is not stable. Note that this $V(x)$ has some special meaning: the first term represents the utility gain from making the user happy, while the second term represents the cost in terms of price. So $V(x)$ represents the net gain. Also, note that since the first term is concave because of the log function, and the second term is assumed to be monotonically increasing, so the resulting $V(x)$ is convex, i.e. it has a maximum value.

To show that $V(x(t))$ is strictly convergent, we want to show that $\frac{dV(x(t))}{dt} > 0$, which implies that $V(x(t))$ strictly increases (i.e. the net gain keeps increasing over time), until the system stabilizes and reaches steady state when $\frac{dV(x(t))}{dt} = 0$ (i.e. the net gain $V(x)$ reaches its maximum value).

First, we note:

$$\frac{\partial V(x)}{\partial x_r} = \frac{w_r}{x_r} - \sum_{j \in r} p_j(\sum_{s:j \in s} x_s)$$

Then:

$$\frac{V(x(t))}{dt} = \sum_{r \in R} \frac{\partial V(x(t))}{\partial x_r} \frac{dx_r(t)}{dt}$$

$$\frac{V(x(t))}{dt} = \sum_{r \in R} (\frac{w_r}{x_r} - \sum_{j \in r} p_j(\sum_{s:j \in s} x_s(t))) \; k(w_r - x_r(t)p_r(t))$$

$$\frac{V(x(t))}{dt} = \sum_{r \in R} \frac{k}{x_r(t)} (\frac{w_r}{x_r} - \sum_{j \in r} p_j(\sum_{s:j \in s} x_s(t)))^2 > 0$$

Observe that this non-linear stability analysis shows that the system is stable, no matter what the initial state $x(0)$ is. This property is referred to as *global stability*, which is in contract to *local stability* proved when the system is linearized locally around a certain operating point as we will see next.

5.2 Linearizing and Solving Linear Differential Equations

As noted earlier, finding Lyapunov functions to prove global stability of non-linear control systems, even for simple models, is challenging. For example, consider more sophisticated models with feedback delay, different regions of TCP operation (e.g., timeouts, slow-start), queue management with different operating regions (e.g. RED), and challenging or adversarial environments (e.g. exogenous losses over wireless links or due to DoS attacks).

Using linearization, we can separately study simpler linear models around the different points (regions) of operation. More specifically, we linearize the system around a single operating point x^* and study perturbations around x^*, i.e. if we perturb the system away from x^* to a point $x(0)$ such that the initial perturbation $\delta x(0) = x(0) - x^*$, we want to show that $\delta x(t)$ diminishes over time and the system returns to its original state x^*, i.e. $\delta x(t) \rightarrow 0$. In this case, we say that the system is *locally stable* around x^*.

Let's consider again the Vegas-like source adaptation and assume, for simplicity, a single user over a single resource:

$$\frac{dx(t)}{dt} = k(w - x(t)p(x(t)))$$

Define the perturbation $\delta x(t) = x(t) - x^*$. Then we can write:

$$\frac{d(\delta x(t) + x^*)}{dt} = k(w - (\delta x(t) + x^*)p((\delta x(t) + x^*)))$$

Expanding the non-linear term $p((\delta x(t) + x^*))$ into its first-order Taylor series:

$$p((\delta x(t) + x^*)) \approx p(x^*) + p'(x^*)\delta x(t)$$

Substituting with this linear approximation, we get:

$$\frac{d\delta x(t)}{dt} = k(w - (\delta x(t) + x^*)(p(x^*) + p'(x^*)\delta x(t)))$$

$$\frac{d\delta x(t)}{dt} = k(w - \delta x(t)p(x^*) - x^*p(x^*) - p'(x^*)\delta^2 x(t) - p'(x^*)x^*\delta x(t))$$

If x^* is the optimal steady-state point, we know that $w - x^*p(x^*) = 0$. Also, given small perturbation $\delta^2 x(t)$, $\delta^2 x(t) \approx 0$. Then, we have:

$$\frac{d\delta x(t)}{dt} = k(-\delta x(t)p(x^*) - p'(x^*)x^*\delta x(t))$$

$$\frac{d\delta x(t)}{dt} = -k(p(x^*) + x^*p'(x^*))\ \delta x(t)$$

Let $k(p(x^*) + x^* p'(x^*)) = \gamma$, we have:

$$\frac{d\delta x(t)}{dt} = -\gamma \; \delta x(t) \tag{10}$$

This is now a linear differential equation, which unlike the original non-linear differential equation, we can easily study using linear control-theoretic techniques, or in this simple case, solve by straightforward integration:

$$\int_0^t \frac{d \; \delta x(t)}{\delta x(t)} = -\gamma \; \int_0^t dt$$

$$log(\delta x(t)) - log(\delta x(0)) = -\gamma t$$

$$log(\frac{\delta x(t)}{\delta x(0)}) = -\gamma t$$

$$\delta x(t) = \delta x(0) e^{-\gamma t}$$

Note that from this time-domain analysis, the system is shown to be stable, i.e. the perturbation vanishes over time and the system returns to its original state $x(0)$. We also observe that the system response decays exponentially from its original perturbation $\delta x(0)$, i.e. without oscillations, and so the response is classified as *overdamped*.

If the linearized differential equation modeling the system were more complicated, it is much easier to transform it into the Laplace domain and analyze the system algebraically. Denoting $\delta x(t)$ by $u(t)$, the Laplace transform of $\delta x(t)$ by $U(s)$, and taking the Laplace transform of Equation 10, we get:

$$s \, U(s) - u(0) = -\gamma \; U(s)$$

$$U(s) = \frac{u(0)}{s + \gamma}$$

For stability analysis, we examine the location of the poles (roots) of the characteristic equation $s + \gamma = 0$, yielding the pole $s = -\gamma$. Since the pole is strictly in the left-side of the s-plane, given $\gamma > 0$, the system is stable and its response is overdamped.

To evaluate the steady-state error, we define the error as $e(t) = u(0) - u(t)$, and applying the Final Value Theorem with an impulse perturbation of magnitude $u(0)$, i.e. $U(0) = u(0)$, we obtain:

$$e_{ss} = lim_{s \to 0} \; sE(s) = lim_{s \to 0} \; s(u(0) - \frac{u(0)}{s + \gamma}) = 0$$

So, there is no steady-state error.

5.2.1 Effect of Feedback Delay and Nyquist Stability Criterion

As we just noted above, the power of solving the linearized model in the Laplace domain comes when the model is even slightly more complicated. For example, let us consider a feedback delay T such that Equation 10 looks like:

$$\frac{du(t)}{dt} = -\gamma \, u(t-T)$$

Taking the Laplace transform, and noting that the Laplace transform of a delayed signal $u(t-T)$ is $e^{-sT}U(s)$, we obtain:

$$sU(s) - u(0) = -\gamma \, e^{-sT}U(s)$$

$$U(s) = \frac{u(0)}{s + \gamma \, e^{-sT}}$$

Then, the characteristic equation is:

$$s + \gamma \, e^{-sT} = 0 \qquad (11)$$

which we need to solve to locate the poles and determine the stability of the system.

To solve such characteristic equation, we resort to another control-theoretic method called the *Nyquist stability criterion*. To this end, we introduce, without proof, the *Cauchy's principle*, which states that given $F(s)$, and we plot $F(s)$ as we vary s along a certain contour in the s-plane — see Figure 18 — and denote the following:

- Z: the number of *zeros* of $F(s)$, i.e. the roots of the numerator of $F(s)$, inside the contour.
- P: the number of *poles* of $F(s)$, i.e. the roots of the denominator of $F(s)$, inside the contour.
- N: the number of encirclements of the plot of $F(s)$ around the origin in the $F(s)$-plane, such that an encirclement is negative if it is in the opposite direction of the s-contour.

Then the following relationship holds:

$$Z = P + N$$

The Nyquist method applies the Cauchy's principle as follows. Let's say we want to analyze the stability of a closed-loop control system whose forward transfer function is $G(s)$ and its feedback transfer function is $H(s)$—see Figure 19. Then, the closed-loop transfer

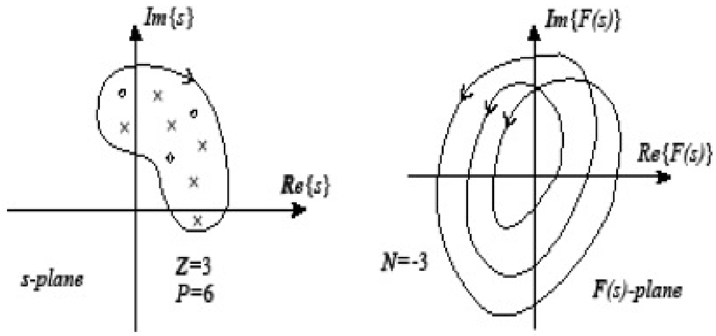


Figure 18: Cauchy's principle

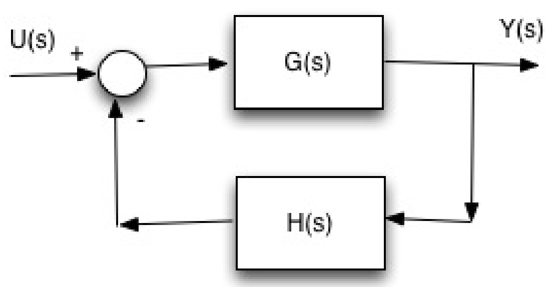


Figure 19: Typical closed-loop control system

function is given by $\frac{G(s)}{1+G(s)H(s)}$, where $G(s)H(s)$ is referred to as the *open-loop* transfer function. The characteristic equation is given by: $F(s) = 1 + G(s)H(s) = 0$. Observe that the zeros of $F(s)$ are the closed-loop poles, and the poles of $F(s)$ are the open-loop poles (i.e. poles of $G(s)H(s)$).

By taking the s-contour to be around the right-side (i.e. unstable side) of the s-plane (see Figure 20), and noting the number of *unstable* open-loop poles P and the number of encirclements N around the origin in the $F(s)$-plane, we determine the number of *unstable* zeros Z of $F(s)$, i.e. number of unstable closed-loop poles, using Cauchy's relationship: $Z = P + N$. If $P = 0$ and $N = 0$, then $Z = 0$ implies that there are no *unstable* closed-loop poles and so the closed-loop system is stable.[4]

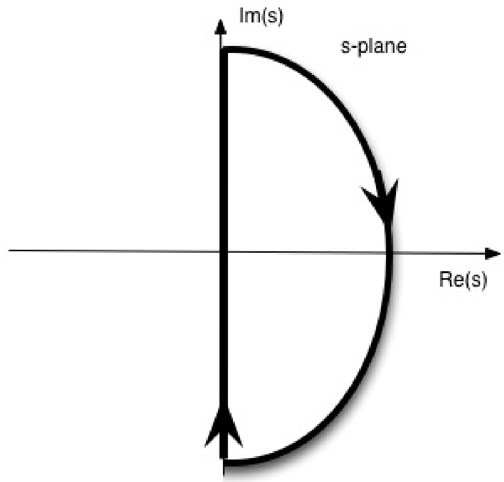


Figure 20: Contour around the unstable right-side of the s-plane

This process can be slightly simplified if instead of plotting $F(s)$, we instead plot the open-loop transfer function: $G(s)H(s)$ and observe its encirclements of the $(-1, j0)$ point in the $G(s)H(s)$-plane, instead of the origin $(0, j0)$ in the $F(s)$-plane. Given there are no poles of $G(s)H(s)$ in the right-side of the s-plane, i.e. $P = 0$, in order for the closed-loop system to be stable, the plot of $G(s)H(s)$ should not encircle -1 as we vary s along the contour enclosing the right-side of the s-plane. We are mostly interested in varying s along the imaginary axis, i.e. $s = j\omega$ where ω varies from 0 to ∞. This is because the plot for ω from $-\infty$ and 0 is symmetric, and the plot for the semi-circle as $s \to \infty$ maps to the origin in the $G(s)H(s)$-plane. Thus, we are interested in plotting $G(j\omega)H(j\omega)$ as ω varies from 0 to ∞.

[4] Note that a pole on the imaginary axis is not considered unstable and the s-contour avoids such a pole and we show it as a small circle around it.

Example: Let's go back to the characteristic equation in Equation 11:

$$s + \gamma\ e^{-sT} = 0 \Rightarrow F(s) = 1 + \frac{\gamma}{s} e^{-sT} \Rightarrow G(s)H(s) = \frac{\gamma}{s} e^{-sT}$$

Note that $G(s)H(s)$ does not have any unstable poles, i.e. $P = 0$. In particular, $s = 0$ is considered a (critically) stable pole.

Ignoring the constant factor γ for now, we want to plot:

$$\frac{e^{-j\omega T}}{j\omega} \qquad \omega : 0 \rightarrow \infty$$

Noting that $e^{j\theta} = cos\theta + j\ sin\theta$, we have:

$$e^{-j\omega T} = cos(\omega T) - j\ sin(\omega T)$$

Then,

$$\frac{e^{-j\omega T}}{j\omega} = -j\frac{cos(\omega T)}{\omega} - \frac{sin(\omega T)}{\omega}$$

Since we are interested in determining intercepts with the real-axis of $G(j\omega)H(j\omega)$ and whether they occur to the right or left of -1, we want to determine the values of ω for which the imaginary part of $G(j\omega)H(j\omega)$, i.e. $-\frac{cos(\omega T)}{\omega}$, is zero. Such intercepts occur when $\omega T = \frac{\pi}{2}, \frac{3\pi}{2}, \cdots$, when the cosine value is zero.

Now, at these values of ωT, we can determine the points of interception along the real-axis, i.e. the magnitude $|G(j\omega)H(j\omega)|$ when the plot intercepts the real-axis:

$$-\frac{sin(\omega T)}{\omega} = -\frac{1}{\frac{\pi}{2T}}, +\frac{1}{\frac{3\pi}{2T}}, \cdots$$

For the system to be stable, $|G(j\omega)H(j\omega)|$ must be less than 1, so the $G(s)H(s)$ plot does not encircle -1. This is the case if after restoring the constant factor γ we initially ignored, the following holds:

$$\gamma\frac{2T}{\pi} < 1$$

End Example.

Observe that T is the feedback delay so as T gets larger, it gets harder to satisfy the stability condition. Intuitively, this makes sense since a larger feedback delay results in outdated feedback (measurements) and it becomes impossible to stabilize the system. This is the fundamental reason why TCP over long-delay paths does not work, and architecturally, control has to be broken up into smaller control loops.

6 Routing Dynamics

So far, we assumed routes taken by flows to be static. In general, routes may also be adapted based on feedback on link prices (reflecting load, delay, etc.), albeit over a longer timescale of minutes, hours or even days compared to that of milliseconds for sending rate adaptation. Figure 21 shows a block diagram that includes both route and rate adaptation.

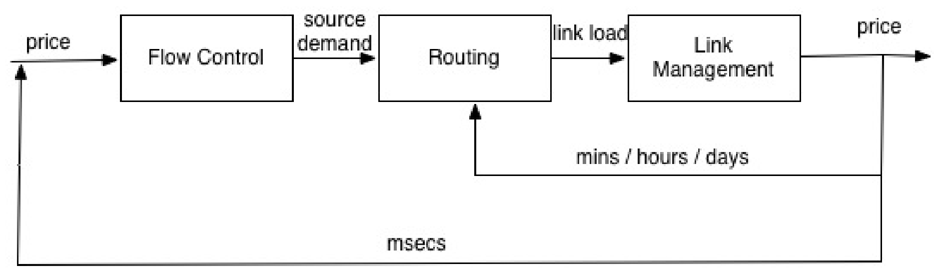


Figure 21: Block diagram with both flow and routing control

Figure 22 illustrates the general process of adaptation. Flow or routing control determines the amount of load directed to a particular link based on the link's observed price — relative to that of other possible links on alternate routes in the case of routing. We call this mapping from link price λ to link load x, the *response function* $H(\lambda)$. Given link load, a certain price is observed for the link. We call such load-to-price mapping, the *pricing (feedback) function* $G(x)$. The process of adaptation is then an iterative process:

$$\lambda = G(x)$$
$$x = H(\lambda)$$

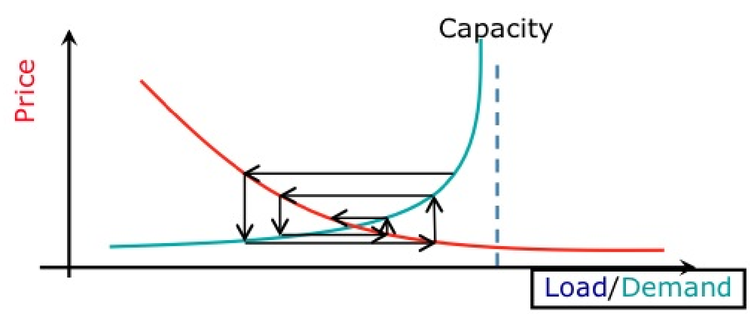


Figure 22: Convergence

We can then write:

$$\lambda = G(H(\lambda)) = F(\lambda)$$

where $F(\lambda)$ is an *iterative* function whose stable (fixed) point λ^* is the intersection of the response function and the pricing function. Figure 23 illustrates convergence to a fixed point. Starting from an initial λ_0, we find $F(\lambda_0)$, then projecting on the 45^o line we obtain $\lambda_1 = F(\lambda_0)$, which we use to find $F(\lambda_1)$, and this iterative process continues until we reach the fixed point.

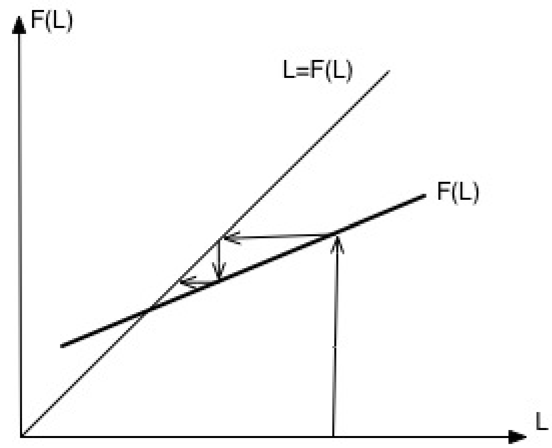


Figure 23: Contractive mapping

In order to converge to that fixed point, $F(\lambda)$ must be a so-called *contractive mapping*. $F(\lambda)$ is contractive iff its slope is less than 1, i.e. $|F(\lambda_2) - F(\lambda_1)| < \alpha|\lambda_2 - \lambda_1|$, $\alpha < 1$. Figure 24 illustrates a mapping that results in divergence.

Intuitively, the use of Lyapunov functions to prove convergence tests whether the iterative process describing the evolution of the system over time is a contractive mapping, i.e. the distance to the fixed point keeps shrinking at every iteration.

Example: Consider the adaptive routing of $N > 0$ unit-rate flows over two possible paths whose prices are given by *monotonically increasing* functions $p_1(x)$ and $p_2(N - x)$, where x represents the number of flows (or load) routed on the first path. Note that x completely defines the state of the system. Also, assume that routing to the least-loaded path is done gradually, to avoid wild oscillations, where $0 < \alpha < 1$ of the flows are re-routed. Using a discrete-time model where routes are adapted at discrete-time steps, we can write the following difference equations:

$$x(t+1) = x(t) + \alpha(N - x(t)), \quad \text{if } p_1(x(t)) \leq p_2(N - x(t))$$

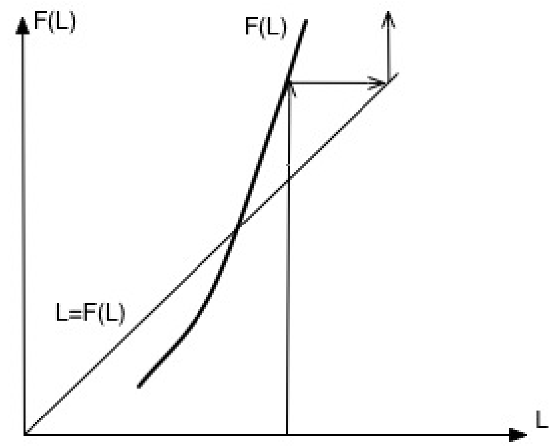


Figure 24: Divergent mapping

$$x(t+1) \;=\; x(t) - \alpha x(t), \quad \text{otherwise}$$

At steady state, this system might converge to one of two possible stable (fixed) points. One possibility is obtained when substituting with $x(t) \to x^*$ in the first difference equation: $x(t) \to x^* \Rightarrow x^* = x^* + \alpha(N - x^*) \Rightarrow x^* = N$, so all traffic will end up getting routed on the first path. A *necessary* condition to reach that $x^* = N$ fixed point is that $p_1(N) \leq p_2(0)$, i.e. the first path is least loaded (priced) even when all N flows are on it.

Another possibility is obtained when substituting with $x(t) \to x^*$ in the second difference equation: $x(t) \to x^* \Rightarrow x^* = x^* - \alpha x^* \Rightarrow x^* = 0$, so all traffic will end up getting routed on the second path. A *necessary* condition to reach that $x^* = 0$ fixed point is that $p_1(0) > p_2(N)$, i.e. the second path is least loaded (priced) even when all N flows are on it.

We can show convergence to one of these fixed points depending on which necessary condition holds: $p_1(N) \leq p_2(0)$ or $p_1(0) > p_2(N)$.

Let's assume $p_1(N) \leq p_2(0)$ holds. We want to define a Lyapunov function $V(x) \geq 0$ and show that $V(x(t+1)) \leq V(x(t))$ for some or all starting state $x(0)$, i.e. $V(x)$ monotonically decreases toward the $x^* = N$ fixed point where equality holds. If there are only certain values of the starting state $x(0)$ for which the system converges then such conditions must hold, in addition to the necessary condition, for convergence to happen. In this case, we say that the necessary condition by itself is not sufficient for convergence.

Define $V(x) = N - x$. Note that $V(x) \geq 0$ because $0 \leq x \leq N$, and $V(x) = 0$ when $x = N$, i.e. at the fixed point. So, under convergence, we expect $V(x)$ to monotonically decrease toward zero. Substituting for $x(t+1)$ in $V(x)$, we obtain:

$$V(x(t+1)) = N - x(t+1)$$

Given the pricing functions are monotonically increasing with load, $p_1(N) \leq p_2(0) \Rightarrow p_1(x(t)) \leq p_2(N-x(t)), \forall x(t)$, and we can only use the first difference equation to substitute for $x(t+1)$:

$$V(x(t+1)) = N - (x(t) + \alpha(N - x(t))) = (1-\alpha)(N - x(t)) = (1-\alpha)V(x(t)) \leq V(x(t))$$

So, we can conclude that the system is convergent regardless of the starting state $x(0)$ as long as $0 < \alpha < 1$.

Thus, $0 < \alpha < 1$, along with the necessary condition $p_1(N) \leq p_2(0)$, thus represent necessary and sufficient conditions for convergence.

A similar convergence proof can be obtained if on the other hand, the necessary condition $p_1(0) > p_2(N)$ holds. **End Example.**

7 Case Study: Class-based Scheduling of Elastic Flows

In this and the following section, we consider the modeling and control-theoretic analysis of two traffic control case studies. This first case study concerns the performance of elastic flows, i.e., rate-adaptive flows similar to TCP. The goal is to investigate the effect of class-based scheduling that isolates elastic flows into two classes (service queues) based on different characteristics, for example based on their lifetime (transfer size) or burstiness of arrivals/departures and sending rate (window) dynamics. We want to show the benefits of isolation, in terms of better predictability and fairness, over traditional shared queueing systems.

We formulate two control models. In the first model (Section 7.1), each flow controls its input traffic rate based on the *aggregate* state of the network due to all N flows. In the second model (Section 7.2), each flow (or class of homogeneous flows) controls its rate based on its own *individual* state within the network. We assume that the flows use PI control for adapting their sending rate.

In the aggregate control model, the packet sending rate of flow i, denoted by $x_i(t)$, is adapted based on the difference between a target *total* buffer space, denoted by B, and the current *total* number of outstanding packets, denoted by $q(t)$. In the individual control model, $x_i(t)$ is adapted based on flow (or class) i's target, denoted by B_i, and its current number of outstanding packets, denoted by $q_i(t)$. We denote by $c(t)$ the total packet service rate, and by $c_i(t)$ the packet service rate of flow/class i. In what follows, for each control model, we determine conditions under which the system stabilizes. We then solve for the values of the state variables at equilibrium, and show whether fairness (or a form of weighted resource sharing) can be achieved. Table 3 lists all system variables along with their meanings.

Table 3: Table defining system variables

Variable	Meaning
N	total number of flows (or classes of homogeneous flows)
$x_i(t)$	packet sending rate of flow/class i
$q_i(t)$	number of outstanding packets of flow/class i
$c_i(t)$	packet service rate of flow/class i
$q(t)$	total number of outstanding packets
$c(t)$	total packet service rate
B	target total buffer space
B_i	target buffer space allocated to flow/class i
α_i	parameter controlling increase and decrease rate of $x_i(t)$

7.1 Aggregate Control or Sharing

Under aggregate PI control, the evolution of the system state is described by the following differential equations:

$$\begin{aligned}\dot{x}_i(t) &= \alpha_i(B - q(t)) \\ \dot{q}(t) &= \sum_{i=1}^{N} x_i(t) - c(t) \qquad (12)\end{aligned}$$

Stability Condition: Without loss of generality, assume a constant packet service rate (i.e. $c(t) = C$ for all t), all flows start with the same initial input state (i.e. $x_i(0)$ is the same for all i), and that all flows adapt at the same rate (i.e. $\alpha_i = \alpha$ for all i). Then, equations (12) can be rewritten as:

$$\begin{aligned}\dot{x}_i(t) &= \alpha(B - q(t)) \\ \dot{q}(t) &= \sum_{i=1}^{N} x_i(t) - C \qquad (13)\end{aligned}$$

Since flows adapt their $x_i(t)$ at the same rate, then $x_i(t) = \frac{\sum_{j=1}^{N} x_j(t)}{N}$ for all i. Denote by $e(t)$ the error at time t, i.e. $e(t) = B - q(t)$, and let $y(t) = \sum_{j=1}^{N} x_j(t) - C$. Equations (13) can then be rewritten as:

$$\begin{aligned}\frac{\dot{y}(t)}{N} &= \alpha\ e(t) \\ \dot{q}(t) &= y(t) \qquad (14)\end{aligned}$$

Taking the Laplace transform of equations (14), we get:

$$\begin{aligned}\frac{1}{N}(sY(s) - y(0)) &= \alpha\ E(s)\\ s\ Q(s) &= Y(s)\\ E(s) &= B - Q(s)\end{aligned} \tag{15}$$

Solving equations(15), we obtain the closed-loop system's characteristic equation:

$$s^2 + \alpha\ N = 0 \Rightarrow s = \pm\, j\sqrt{\alpha\ N} \tag{16}$$

For $\alpha > 0$, this system is marginally stable. However, the magnitude of oscillations increases for higher α and/or higher N.

This indicates that the existence of flows that rapidly change their sending rates through high values of α_i can cause the system to have high oscillations. This suggests that elastic flows that aggressively change their sending rates, may affect the stability of other flows that change their sending rates cautiously, in a system that mixes both kinds of flows. Furthermore, in such a system, the value of N may be high so as to cause high oscillations.

We now derive the values of the state variables at equilibrium. Denote by $(x_i)_s$ and q_s the steady-state values of $x_i(t)$ and $q(t)$, respectively. Then, at equilibrium, we have from equations (12):

$$\begin{aligned}0 &= \alpha_i(B - q_s)\\ 0 &= \sum_{i=1}^{N}(x_i)_s - C\end{aligned} \tag{17}$$

Thus, at equilibrium, $q_s = B$ and $\sum_{i=1}^{N}(x_i)_s = C$. In other words, the system converges to a state where the total input rate matches the total service rate, and the total buffer space is full.

Note that if $\alpha_i = \alpha$ for all i, then $x_i(t)$ changes at the same rate for every flow i. Consequently, if we start the evolution of the system with $x_i(0)$ being the same for all flows, only then we have equal sharing of the network at steady-state, i.e. $(x_i)_s = \frac{C}{N}$, regardless of the initial value $q(0)$. However, in general, when $x_i(0)$ are not equal for all flows, the system converges to an *unfair* state, more precisely, to a state where

$$(x_i)_s = x_i(0) + \frac{C - \sum_{j=1}^{N} x_j(0)}{N} \tag{18}$$

To summarize, controlling several flows by observing the resulting aggregate state of the network may lead to high oscillations due to either the existence of flows which are rapidly adjusting their sending rates, or a high number of flows competing for the same *shared* resource. Furthermore, the system is highly likely to converge to an unfair state where flows receive unequal shares of the resource.

7.2 Individual Control or Isolation

Under individual PI control, the evolution of the system state is described by the following differential equations:

$$\begin{aligned}\dot{x}_i(t) &= \alpha_i(B_i - q_i(t)) \\ \dot{q}_i(t) &= x_i(t) - c_i(t)\end{aligned} \tag{19}$$

Recall that under individual control, flow/class i regulates its input, $x_i(t)$, based on its *own* number of outstanding packets. For simplicity, assume a constant packet service rate, i.e. $c_i(t) = C_i$ for all t. Following the same stability analysis as aggregate control, it is easy to see that flow/class i stabilizes and the poles of the closed-loop system are:[5]

$$s = \pm j\sqrt{\alpha_i} \tag{20}$$

Observe that, unlike aggregate control, flows/classes are *isolated* from each other. Therefore, the existence of flows/classes that rapidly change their sending rates through high values of α_i, does not affect the stability of other flows. This isolation can be implemented using, for example, a class-based queueing (CBQ) discipline. In such a CBQ system, each class of homogeneous flows can be allocated its own buffer space and service capacity.

We now derive the values of the state variables of flow/class i at equilibrium. Denote by $(x_i)_s$ and $(q_i)_s$ the steady-state values of $x_i(t)$ and $q_i(t)$, respectively. Then, at equilibrium, we have from equations (19):

$$\begin{aligned}0 &= \alpha_i(B_i - (q_i)_s) \\ 0 &= (x_i)_s - C_i\end{aligned} \tag{21}$$

Thus, at equilibrium, $(q_i)_s = B_i$ and $(x_i)_s = C_i$. In other words, each flow/class i converges to a state where its input rate matches its allocated service rate, and its allocated buffer space is full. We note that if the allocated buffers B_i and service capacities C_i are equal, then every flow receives an equal share of the resources, *regardless of* the initial values $x_i(0)$ and $q_i(0)$. One can also achieve a weighted resource sharing by assigning different B_i and C_i allocations. Thus, a flow/class with higher priority (e.g. short interactive TCP flows operating aggressively in slow start) can be allocated more resources, so as to receive better throughput/delay service than other flows (e.g. long TCP flows operating cautiously in congestion avoidance).

To summarize, controlling each flow (or class of homogeneous flows) by observing its own individual state within the network provides isolation between them. Thus, stability can be achieved for a flow/class regardless of the behavior and number of other flows/classes. Furthermore, the system can converge to a fair state where flows/classes receive a weighted share of the resources.

[5] We set $N = 1$ in equation (16).

8 Case Study: Elastic Transport Tunnel

Consider n regular user connections between sending and receiving end-hosts, all passing through two "gateways" — let's call them a source gateway and a destination gateway. Our main goal is to provide a soft-bandwidth-guaranteed tunnel for these user flows over an Internet path of bottleneck capacity C, which is also shared by another set of x flows, representing cross traffic. Consider that user and cross-traffic connections are all rate-adaptive connections (similar to TCP). These x cross-traffic connections present a challenge: as x keeps changing, the bandwidth allocation for the n user flows keeps changing in tandem. So an important question is whether it is possible to "counter" the change in x so as to ensure that the n user flows are able to maintain a desirable bandwidth.

Clearly without the intervention of the two gateways, the answer to the above question is *no*. When different flows share a link, the effect of each individual flow (or an aggregate of flows) affects the rest since all are competing for a fixed amount of resources. However, if the gateways dynamically maintain a number m of open rate-adaptive (e.g., TCP) connections between them, they can provide a positive pressure that would equalize the pressure caused by the cross-traffic connections, if the latter occurs. Since m will be changing over time, we describe the gateway-to-gateway tunnel, made of the m connections, as *elastic*. Note that the source gateway can decide to reduce m (i.e. relieve pressure) if x goes down—the reason is that as long as the tunnel is achieving its target bandwidth, releasing extra bandwidth should improve the performance of cross-traffic connections, which is in the spirit of best-effort networking.

To illustrate this scenario and the issues involved, consider a gateway-to-gateway tunnel going through a single bottleneck link. Under normal load, the behavior of the bottleneck can be approximated by Generalized Processor Sharing (GPS), i.e. each connection receives the same fair share of resources. Thus, each connection ends up with $\frac{C}{m+x}$ bandwidth. This, in turn, gives the m gateway-to-gateway rate-adaptive flows, or collectively the elastic gateway-to-gateway tunnel, a bandwidth of $\frac{Cm}{m+x}$. As the source gateway increases m by opening more rate-adaptive connections to the destination gateway, the tunnel can grab more bandwidth. If x increases, and the gateways measure a tunnel's bandwidth below a target value (say B^*), then m is increased to push back cross-traffic connections. If x decreases, and the gateways measure a tunnel's bandwidth above B^*, then m is decreased for the good of cross-traffic connections. It is important to note that the source gateway should refrain from unnecessarily increasing m, thus achieving a tunnel's bandwidth above B^*, since an unnecessary increase in the total number of competing rate-adaptive flows reduces the share of each connection and may cause flows to timeout leading to inefficiency and unfairness. The source gateway also has the responsibility of scheduling user packets coming on the n user connections over the tunnel, i.e. the m gateway-to-gateway connections. In this case study, we do not focus on scheduling but the control theoretic modeling and analysis of the tunnel's bandwidth adaptation. We study the effect of different types of controllers employed at the source gateway. Such controller determines the degree of elasticity of the gateway-to-gateway rate-adaptive tunnel, thus it determines the transient and steady-state behavior of the soft-bandwidth-guaranteed service.

Naive Control: This naive controller measures the bandwidth b' grabbed by the current m' gateway-rate-adaptive connections. Then, it directly computes the quiescent number $\hat{m}$ of gateway-rate-adaptive connections that should be open as:

$$\hat{m} = \frac{B^*}{b'} m' \tag{22}$$

Clearly, this controller naively relies on the previously measured bandwidth b' and adapts without regard to delays in measurements and possible changes in network conditions, e.g. changes in the amount of cross traffic. We thus investigate general well-known controllers which judiciously zoom-in toward the target bandwidth value. To that end, we develop a flow-level model of the system dynamics. The change in the bandwidth grabbed $b(t)$ by the $m(t)$ gateway-rate-adaptive flows (constituting the elastic gateway-to-gateway tunnel) can be described as:

$$\dot{b}(t) = \alpha[(C - B^*)m(t) - B^* x(t)] \tag{23}$$

Thus, $b(t)$ increases with $m(t)$ and decreases as the number of cross-connections $x(t)$ increases. α is a constant that represents the degree of multiplexing of flows and we choose it here to be the steady-state connection's fair share ratio of the bottleneck capacity. At steady-state, $\dot{b}(t)$ equals zero, which yields:

$$B^* = \frac{C\hat{m}}{(\hat{x} + \hat{m})} \tag{24}$$

where $\hat{m}$ and $\hat{x}$ represent the steady-state values for the number of gateway-rate-adaptive and cross-traffic flows, respectively. Based of the current bandwidth allocation $b(t)$ and the target bandwidth B^*, an error signal $e(t)$ can be obtained as:

$$e(t) = B^* - b(t) \tag{25}$$

P and PI Control: A controller would adjust $m(t)$ based on the value of $e(t)$. For a simple Proportional controller (P-type), such adjustment can be described by:

$$m(t) = K_p e(t) \tag{26}$$

P-type controllers are known to result in a non-zero steady-state error. To exactly achieve the target B^* (i.e. with zero steady-state error), a Proportional-Integral (PI-type) controller can be used:

$$m(t) = K_p e(t) + K_i \int e(t)\, dt \tag{27}$$

Figure 25 shows the block diagram of this elastic-tunnel model. In the Laplace domain, denoting the controller transfer function by $C(s)$, the output $b(s)$ is given by:

$$b(s) = \frac{C(s)G_1(s)}{1 + C(s)G_1(s)} B^*(s) + \frac{G_2(s)}{1 + C(s)G_1(s)} x(s) \tag{28}$$

where $G_1(s)$ is given by:

$$G_1(s) = \frac{\beta}{s} \tag{29}$$

where $\beta = \alpha(C - B^*)$. $G_2(s)$ is given by:

$$G_2(s) = \frac{-\alpha B^*}{s} \tag{30}$$

where $\gamma = -\alpha B^*$. For the P-controller, from Equation (26), $C(s)$ is simply K_p. For the PI-controller, from Equation (27), $C(s)$ equals $K_p + \frac{K_i}{s}$. Thus, the transfer function $\frac{b(s)}{B^*}$ in the presence of a P-controller is given by:

$$\frac{b(s)}{B^*} = \frac{K_p\beta}{s + K_p\beta} \tag{31}$$

The system with P-controller is always stable since the root of the characteristic equation (i.e. the denominator of the transfer function) is negative, given by $-K_p\beta$. In the presence of a PI-controller, the transfer function $\frac{b(s)}{B^*}$ is given by:

$$\frac{b(s)}{B^*} = \frac{K_p\beta s + K_i\beta}{s^2 + K_p\beta s + K_i\beta} \tag{32}$$

One can choose the PI-controller parameters K_p and K_i to achieve a certain convergence behavior to the target bandwidth B^*. We next define transient performance measures to assess such convergence behavior.

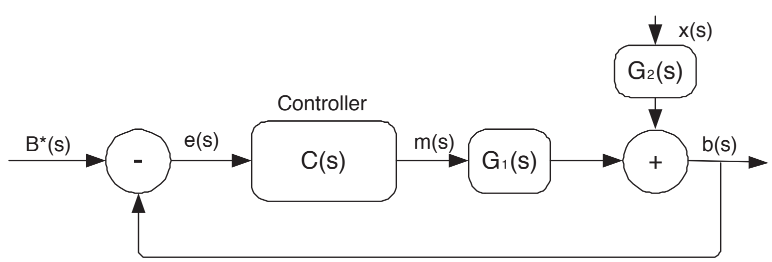


Figure 25: Block Diagram of the Elastic-Tunnel Model

8.1 Transient Performance Metrics

Transient behavior represents the system's response which decays with time. In the design of reliable systems, it is of extreme importance that transient response meets certain requirements such as reasonable settling time and overshoot. Often times, the transient response is obtained by subjecting the system to an impulse or a step input and observing the output(s). One has to guarantee that the response of the system to specific inputs does not render the system unstable or pushes it away from its intended target. For our specific elastic rate-adaptive tunneling system, we can define our performance metrics as follows:

- **Settling Time:** The time taken for our system to respond to a step input in the cross traffic or target bandwidth until it stabilizes once again.

  The system is assumed to have stabilized (in steady state) if the error (the difference between target and measured bandwidth) is bounded for at least 2 seconds. Specifically, if $e(t)$ is the error at time t then the system is considered in steady state if

  $$\forall t \in [t_0, t_k] \text{ where } t_k - t_0 \geq 2\text{secs}, \bar{b} - \delta \leq e(t) \leq \bar{b} + \delta,$$

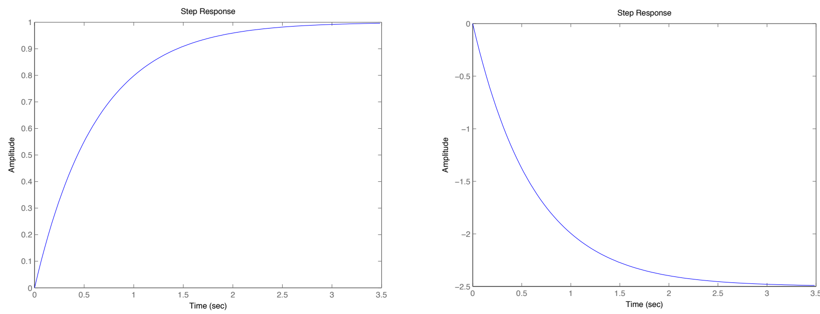
  where $\bar{b}$ is the average bandwidth measured during this period and δ is a constant. We choose here $\delta = \frac{B^*}{20}$.

- **Maximum Overshoot:** The largest overshoot value experienced by the controller in terms of extra gateway-rate-adaptive connections opened or extra bandwidth allocated.

- **Stability in Number of gateway-rate-adaptive Flows:** The variability in number of gateway-rate-adaptive flows reflects the overhead of setting up and tearing down gateway-rate-adaptive connections within the elastic tunnel.

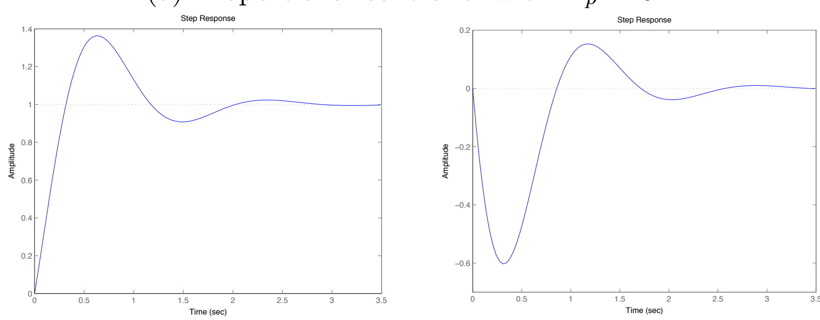
8.2 Transient Performance Results

Figure 26 shows the step response of the transfer function given in Equation (28). The left column shows the response to a step change in the target bandwidth, while the right column shows the response to a step change in the cross-traffic. Figure 26(a), for the P-controller, shows that while the response could be acceptable due to a step change in the reference bandwidth, it fails to remove the steady-state error (non-zero amplitude) obtained from the step change in the cross-traffic. Figures 26(a) and (b) show the response due to the PI-controller. One can see that through a careful choice of K_p and K_i, the transient response can be adjusted. Notice that with a PI-controller, the elastic-tunneling system can reach the target bandwidth with zero steady-state error in response to a step change in cross-traffic.



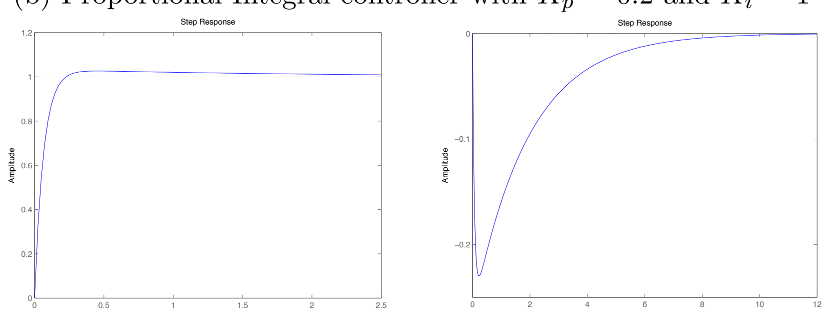
Target Bandwidth Step Response Cross-traffic Step Response

(a) Proportional controller with $K_p = 0.1$



Target Bandwidth Step Response Cross-traffic Step Response

(b) Proportional Integral controller with $K_p = 0.2$ and $K_i = 1$



Target Bandwidth Step Response Cross-traffic Step Response

(c) Proportional Integral controller with $K_p = 1$ and $K_i = 0.5$

Figure 26: Transient Analysis of the Elastic-Tunnel Model

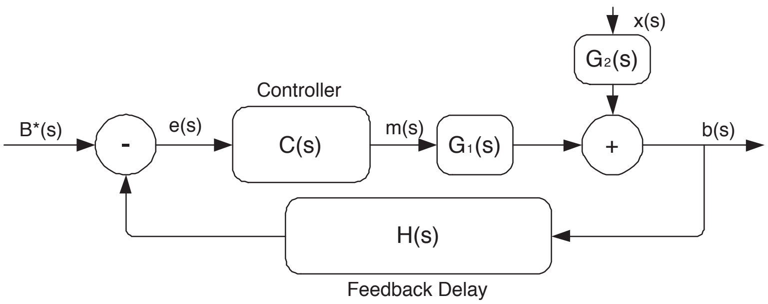


Figure 27: Elastic-Tunnel Model (with Feedback Delay)

8.3 Feedback Delay

So far in our analysis, we have ignored the feedback delay which is inherent in the design of any control system that tries to adjust its signal through a delayed feedback loop.

Figure 27 augments the block diagram of Figure 25 with feedback delay denoted by $H(s)$. This feedback delay arises either due to delayed mesurements of bandwidth and/or delayed response of the system as a result of applying new control. For example, when a new gateway-rate-adaptive connection is opened, it doesn't get its steady-state throughput instantaneously, rather after some delay (say τ). Thus, $H(s)$ is given by:

$$H(s) \quad = \quad e^{-\tau s} \tag{33}$$

where τ represents the feedback time delay. For small τ, the above equation can be approximated by:

$$H(s) \quad = \quad 1 - \tau s \tag{34}$$

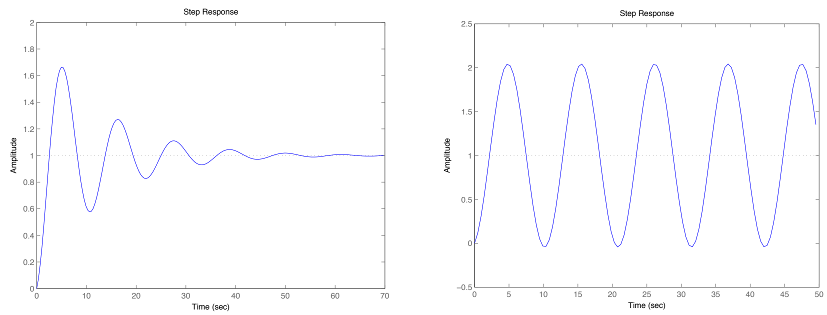
If we are using a PI-controller, the characteristic equation in the presence of feedback delay becomes:

$$s^2(1 - \beta\tau K_p) + s(K_p\beta - \beta\tau K_i) + \beta K_i \tag{35}$$

Figure 28 shows the response of the PI-controller to a step change in the target bandwidth. As the feedback delay τ increases, the system may not converge to the target bandwidth.

9 Exercises

1. Let r be a max-min fair rate vector corresponding to a given network and set of flows. This max-min fair allocation maximizes the allocation of each flow i subject to the constraint that an incremental increase in i's allocation does not cause a decrease in some other flow's allocation that is already as small as i's or smaller.



Target Bandwidth Step Response Response with Feedback Delay
(d) Proportional Integral controller with $K_p = 0.01$ and $K_i = 0.02$

Figure 28: Transient Analysis in the presence of Feedback Delay

(a) Suppose that some of the flows are eliminated and let $\bar{r}$ be a corresponding max-min fair rate vector. Show by example that we may have $\bar{r}_p < r_p$ for some of the remaining flows p.

(b) Suppose that some of the link capacities are increased and let $\bar{r}$ be a corresponding max-min fair rate vector. Show by example that we may have $\bar{r}_p < r_p$ for some flows p.

2. Consider two network links in tandem (one after the other) of capacities 6 Mbps each. For two different sets of elastic flows (i.e. the utility function of each flow/user is a log function of its allocated rate), you are asked to write down the corresponding network optimization problem, where the network tries to maximize the sum of flow utilities subject to link capacity constraints. For each set of flows, described in parts (a) and (b) below, rewrite the constrained optimization problem as an unconstrained optimization problem: write down the Lagrangian function and corresponding equations to solve for the optimal rate allocation. What are the optimal rates allocated to different flows for each one of these two settings?

   (a) Consider two flows: one flow using the first link only and another flow using both links.

   (b) Consider the same two flows from part (a), as well as a third flow using the second link only.

3. Consider two network links in tandem (one after the other) of capacities 6 Mbps each. Assume three flows: one flow using the first link only, another flow using the second link only, and a third flow using both links. Assume the utility function of each flow/user is a log function of its allocated rate, and that the two-link flow is given a weight of 2, while the two one-link flows are given a weight of 1.

You are asked to write down the corresponding network optimization problem, where the network tries to maximize the sum of the weighted flow utilities subject to link capacity constraints. Rewrite the constrained optimization problem as an unconstrained optimization problem: write down the Lagrangian function and corresponding equations to solve for the optimal rate allocation. What are the optimal rates allocated to each flow?

4. Consider a source adapting its sending rate $x(t)$ so the buffer size of its path's bottleneck $b(t)$ stays at a certain target value T. Denote the error signal by $e(t) = T - b(t)$. The sending rate is adapted according to one of the following three controllers:

   (a) $x(t) = K_p e(t)$

   (b) $x(t) = K_p e(t) + K_i \int_0^t e(t)dt$

   (c) $x(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t)$

   where K_p, K_i, K_d are constant parameters of the rate controllers.

   (a) Assume $c(t)$ is the capacity available to the source at time t, write down the differential equation for $b(t)$.

   (b) By transforming the system to the Laplace domain, determine the conditions under which the system is stable for each type of controller, i.e. does $b(t)$ converge to a given T for certain values of K_p, K_i, K_d. Do this by examining the roots (poles) of the characteristic equation of the system's transfer function. Draw the block diagram of the system that shows the relationships between the system variables in the (Laplace) s-domain for each type of controller.

   (c) Again by examining the roots (poles) of the characteristic equation of the system's transfer function and using the Final Value Theorem, compare the transient and steady-state performance under each type of controller.

   (d) Support your answers above by numerically solving the system's equations over time for each type of controller. Assume a small time step Δ, say $\Delta = 1$, and solve the discretized version of the equations at these time steps – you can then approximate the differentiation $\frac{d}{dt}b(t)$ by $b(t) - b(t-1)$.

5. Consider an adaptation of a transmission window w whose goal is to reach a target window T as follows:

$$w_{(k+1)} = w_k + \alpha(T - w_k)$$

   where $0 < \alpha < 1$.

   (a) Derive the necessary condition (if any) for convergence to the target window size.

(b) Use the Lyapunov method to show whether the system converges regardless of the initial window size.

6. Given a system with the following adaptation rules: $x(t)$, representing a sending rate, is adapted using an AIMD policy, whereas $p(t)$, representing the price, is adapted in proportion to how far $x(t)$ is from a target capacity c:

$$\frac{dx(t)}{dt} = 1 - x(t)p(x(t))$$

$$p(x(t)) = \alpha(x(t) - c)$$

   (a) Why is that system non-linear?

   (b) Linearize the system around a certain operating point x_0.

   (c) By transforming the linearized system to the Laplace domain, obtain the condition on x_0 under which the linearized system is stable.

7. This question is based on the paper *The Revised ARPANET Routing Metric, by Zinky and Khanna*. A unified way to model adaptive resource management—whether it is TCP adaptive to RED or routing adaptive to changing link costs or other examples—is through two functions: a feedback (pricing) function such as that of RED or link utilization metric, and an adaptation function such as that of TCP or utilization-based routing. Consider a resource that generates prices p and users that adapt their load λ based on the currently reported p. Assume the following pricing and load adaptation functions:

$$\lambda = 1 - p$$

$$p = \begin{cases} 0.1 & \text{if } 0 \le \lambda < 0.4 \\ 0.2 & \text{if } 0.4 \le \lambda \le 0.8 \\ 1.0 & \text{if } 0.8 < \lambda \le 1.0 \end{cases}$$

   Show the two functions on the (λ, p) plane. Trace one trajectory showing convergence to a fixed point, and another trajectory showing oscillations. *Hint:* consider initial values of $\lambda = 0.2$ and 0.6.

10 Solutions to Exercises

1. (a) Refer to Figure 8. Using max-min fairness the original flow assignment is: (F1=50, F2=50, F3=50, F4=100). Assume F2 is removed. The new assignment is: (F1=75, F2=0, F3=75, F4=75). F1 and F3 will share the extra capacity, while F4 will decrease its rate. This is only possible because F4's original assigned rate was not less than or equal to F3's rate, before F2 was removed, thus allowing F3 to increase its rate.

   (b) Refer to Figure 8. Using max-min fairness the original flow assignment is: (F1=50, F2=50, F3=50, F4=100). If the capacity of link 1 is increased to 225 then the new allocated rates will be: (F1=75, F2=75, F3=75, F4=75). The extra capacity on link 1 will be shared by F1, F2 and F3. Again, this is only possible because F4's original assigned rate is not less than or equal to F3's rate, before the capacity of link 1 was increased, thus allowing F3 to increase its rate.

2. Note that earlier in these notes, we did not explicitly cover the technique of "slack", which we use here to denote residual link capacity when solving the dual Lagrangian problem. Intuitively, if the link is fully utilized, i.e. it has zero slack, then its price (Lagrangian multiplier) is non-zero. On the other hand, a non-bottleneck link will have a non-zero slack and so a price of zero.

   (a) Let z_1^2 and z_2^2 denote the (non-negative) slack on links 1 and 2, respectively. Let x_1 and x_2 denote the rates assigned to flows 1 and 2, respectively. Let $f(x_1, x_2)$ denote the function to be maximized.

   $$f(x_1, x_2) = log\ x_1 + log\ x_2$$

   Constraints on links 1 and 2, respectively, are as follows:

   $$x_1 + x_2 + z_1^2 - c_1 = 0$$

   $$x_2 + z_2^2 - c_2 = 0$$

   Replacing c_1 and c_2 with their values we get:

   $$x_1 + x_2 + z_1^2 - 6 = 0$$

   $$x_2 + z_2^2 - 6 = 0$$

   The Lagrangian function to be differentiated is as follows:

$$F(x_1, x_2) = log\ x_1 + log\ x_2 - \lambda(x_1 + x_2 + z_1^2 - 6) - \mu(x_2 + z_2^2 - 6)$$

Taking the partial derivative with respect to each variable we get the following equations:

$$\frac{\partial F}{\partial x_1} = \frac{1}{x_1} - \lambda = 0$$

$$\frac{\partial F}{\partial x_2} = \frac{1}{x_2} - \lambda - \mu = 0$$

$$\frac{\partial F}{\partial \lambda} = -(x_1 + x_2 + z_1^2 - 6) = 0$$

$$\frac{\partial F}{\partial \mu} = -(x_2 + z_2^2 - 6) = 0$$

$$\frac{\partial F}{\partial z_1} = -2z_1\lambda = 0$$

$$\frac{\partial F}{\partial z_2} = -2z_2\mu = 0$$

Since F1 can utilize any left over capacity (slack) on link 1, $z_1 = 0$, which implies that $\lambda \neq 0$. Conversely, since F2 is limited by link 1, $z_2 \neq 0$, which implies that $\mu = 0$.

Replacing $z_1 = 0$ and $\mu = 0$ into the derived equations, we get:

$$\frac{1}{x_1} - \lambda = 0$$
$$\frac{1}{x_2} - \lambda - 0 = 0$$
$$x_1 + x_2 + 0 - 6 = 0$$

Solving for x_1 we get $x_1 = \frac{1}{\lambda}$
Solving for x_2 we get $x_2 = \frac{1}{\lambda}$. Hence, $x_1 = x_2$.
Replacing x_2 by x_1 in the capacity equation we get:

$$x_1 + x_1 = 6$$

Hence, $x_1 = x_2 = 3$.

(b) Let z_1^2 and z_2^2 denote the slack on links 1 and 2, respectively. Let x_1, x_2 and x_3 denote the rates assigned to flows 1, 2 and 3, respectively. Let $f(x_1, x_2, x_3)$ denote the function to be maximized.

$$f(x_1, x_2, x_3) = log\ x_1 + log\ x_2 + log\ x_3$$

Constraints on links 1 and 2, respectively, are as follows:

$$x_1 + x_2 + z_1^2 - c_1 = 0$$

$$x_3 + x_2 + z_2^2 - c_2 = 0$$

Replacing c_1 and c_2 with their values we get:

$$x_1 + x_2 + z_1^2 - 6 = 0$$

$$x_3 + x_2 + z_2^2 - 6 = 0$$

The Lagrangian function to be differentiated is as follows:

$$F(x_1, x_2, x_3) = log\ x_1 + log\ x_2 + log\ x_3 - \lambda(x_1 + x_2 + z_1^2 - 6) - \mu(x_3 + x_2 + z_2^2 - 6)$$

Taking the partial derivative with respect to each variable we get the following equations:

$$\frac{\partial F}{\partial x_1} = \frac{1}{x_1} - \lambda = 0$$

$$\frac{\partial F}{\partial x_2} = \frac{1}{x_2} - \lambda - \mu = 0$$

$$\frac{\partial F}{\partial x_3} = \frac{1}{x_3} - \mu = 0$$

$$\frac{\partial F}{\partial \lambda} = -(x_1 + x_2 + z_1^2 - 6) = 0$$

$$\frac{\partial F}{\partial \mu} = -(x_3 + x_2 + z_2^2 - 6) = 0$$

$$\frac{\partial F}{\partial z_1} = -2z_1\lambda = 0$$

$$\frac{\partial F}{\partial z_2} = -2z_2\mu = 0$$

From the equations above we can deduce the following:

$$x_1 = \frac{1}{\lambda}$$

$$x_2 = \frac{1}{\lambda + \mu}$$

$$x_3 = \frac{1}{\mu}$$

Since F1 and F3 can utilize any left over capacity (slack) on links 1 and 2, $z_1 = z_2 = 0$. This implies that $\lambda \neq 0$ and $\mu \neq 0$.

Replacing $z_1 = 0$ and $z_2 = 0$ into the derived equations we get:

$$x_1 + x_2 = 6$$

$$x_2 + x_3 = 6$$

Replacing x_1 by $\frac{1}{\lambda}$, x_2 by $\frac{1}{\lambda+\mu}$ and x_3 by $\frac{1}{\mu}$ into the equations above, we get:

$$\frac{1}{\lambda} + \frac{1}{\lambda + \mu} = 6$$

$$\frac{1}{\lambda + \mu} + \frac{1}{\mu} = 6$$

Multiplying the first equation by (-1) and adding it to second equation, results in the following:

$$\frac{1}{\mu} - \frac{1}{\lambda} = 0$$

Hence, $\lambda = \mu$

Thus, $x_1 = x_3 = \frac{1}{\lambda}$ and $x_2 = \frac{1}{2\lambda} = \frac{x_1}{2} = \frac{x_3}{2}$
Finally we get:

$$x_1 + \frac{x_1}{2} = 6$$

Hence, $x_1 = x_3 = 4$ and $x_2 = 2$.

3. The objective function we want to maximize is:

   $F(x) = log\ x_1 + log\ x_2 + 2\ log\ x_3$

   subject to the capacity constraints:

   $x_1 + x_3 \leq 6$
   $x_2 + x_3 \leq 6$
   $x_1, x_2, x_3 \geq 0$

   The Lagrangian (unconstrained) function that we want to maximize is:

   $L(x) = log\ x_1 + log\ x_2 + 2\ log\ x_3 - \lambda_1(x_1 + x_3 - 6) - \lambda_2(x_2 + x_3 - 6)$

   Note that we do not explicitly include the slacks for the link capacities here, since both links should be fully utilized as the one-link flows are not limited by any other link, so the slack values are zero.

   Taking the partial derivatives of $L(.)$ we obtain:

   $\frac{\partial L}{\partial x_1} = \frac{1}{x_1} - \lambda_1 = 0 \Rightarrow x_1 = \frac{1}{\lambda_1}$
   $\frac{\partial L}{\partial x_2} = \frac{1}{x_2} - \lambda_2 = 0 \Rightarrow x_2 = \frac{1}{\lambda_2}$
   $\frac{\partial L}{\partial x_3} = \frac{2}{x_3} - (\lambda_1 + \lambda_2) = 0 \Rightarrow x_3 = \frac{2}{\lambda_1+\lambda_2}$
   $\frac{\partial L}{\partial \lambda_1} = 0 \Rightarrow x_1 + x_3 - 6 = 0 \Rightarrow x_1 + x_3 = 6$
   $\frac{\partial L}{\partial \lambda_2} = 0 \Rightarrow x_2 + x_3 - 6 = 0 \Rightarrow x_2 + x_3 = 6$

   The last two equations yield $x_1 = x_2 \Rightarrow \lambda_1 = \lambda_2 = \lambda$

   From the capacity equation, we have:

   $x_1 + x_3 = \frac{1}{\lambda} + \frac{2}{2\lambda} = 6 \Rightarrow \lambda = \frac{1}{3}$, thus

   $x_1 = x_2 = 3$, and $x_3 = 6 - 3 = 3$.

4. We use the following notation:

   In the time domain we have:
   $b(t)$, buffer size at time t
   $x(t)$, sending rate at time t
   $c(t)$, service rate at time t
   T, target buffer size
   $e(t)$, error (difference between current buffer size and the target) at time t

In the frequency domain we have:
$B(s)$, current buffer size
$X(s)$, sending rate
$C(s)$, service rate
$T(s)$, target buffer size
$E(s)$, error signal
$D(s)$, controller (based on error signal computes sending rate)

We will assume that the target buffer size and the buffer's service rate are constant values.

(a) $e(t) = T - b(t)$
$\frac{d}{dt}b(t) = x(t) - c(t)$

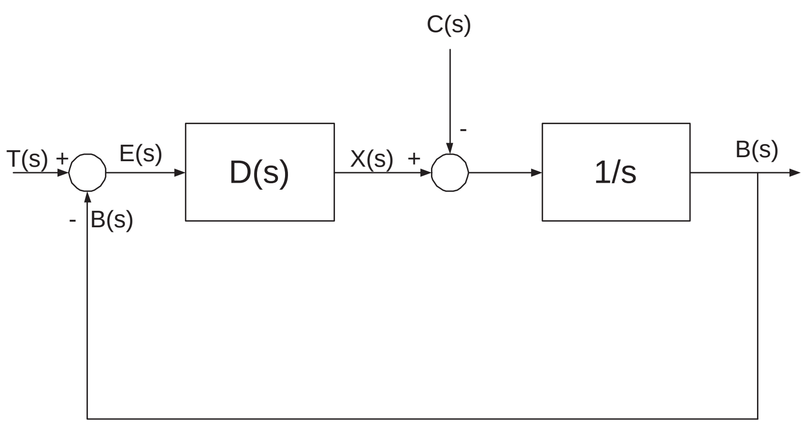


Figure 29: System Block Diagram

(b) The system's block diagram is depicted in Figure 29. Using the block diagram, we can formulate the following equation:

$$(\; [\; (T(s) - B(s)) \; D(s) \;] - C(s) \;) \; (\tfrac{1}{s}) = B(s)$$

$$(\; [T(s)D(s) - B(s)D(s)] - C(s) \;) \; (\tfrac{1}{s}) = B(s)$$

$$(T(s)D(s) - B(s)D(s) - C(s)) \; (\tfrac{1}{s}) = B(s)$$

$[\frac{T(s)D(s)}{s} - \frac{B(s)D(s))}{s} - \frac{C(s)}{s}] = B(s)$

$B(s) + \frac{B(s)D(s)}{s} = \frac{T(s)D(s)}{s} - \frac{C(s)}{s}$

$B(s)[1 + \frac{D(s)}{s}] = \frac{T(s)D(s)}{s} - \frac{C(s)}{s}$

$B(s)[\frac{s+D(s)}{s}] = \frac{T(s)D(s)}{s} - \frac{C(s)}{s}$

$B(s) = \frac{T(s)D(s)}{s+D(s)} - \frac{C(s)}{s+D(s)}$

P-Controller:

$x(t) = K_p e(t)$

By taking the Laplace transform we get:

$X(s) = K_p E(s) \Rightarrow D(s) = \frac{X(s)}{E(s)} = K_p$

Replacing $D(s)$ into the equation for $B(s)$ we get:

$B(s) = \frac{T(s)K_p}{s+K_p} - \frac{C(s)}{s+K_p}$

Roots of the system's characteristic equation are:

$s + K_p = 0 \Rightarrow s = -K_p < 0$

Thus, system is stable for all values of $K_p > 0$

PI-Controller:

$x(t) = K_p e(t) + K_i \int_0^t e(t)dt$

By taking the Laplace transform we get:

$X(s) = K_p E(s) + \frac{K_i}{s} E(s) \Rightarrow D(s) = \frac{X(s)}{E(s)} = K_p + \frac{K_i}{s}$

Replacing $D(s)$ into the equation for $B(s)$ we get:

$B(s) = \frac{(K_p + \frac{K_i}{s})T(s)}{s+(K_p + \frac{K_i}{s})} - \frac{C(s)}{s+(K_p + \frac{K_i}{s})}$

Roots of the system's characteristic equation are:

$s + K_p + \frac{K_i}{s} = 0$

$s^2 + K_p s + K_i = 0$

$a = 1,\ b = K_p,\ c = K_i$

$\Delta = b^2 - 4ac = K_p^2 - 4(1)(K_i) = K_p^2 - 4K_i$

$s = \frac{-b \pm \sqrt{\Delta}}{2a}$

$s = \frac{-K_p \pm \sqrt{K_p^2 - 4K_i}}{2}$

$s = -\frac{1}{2}K_p \pm \frac{1}{2}\sqrt{K_p^2 - 4K_i}$

System will be stable with two real roots (i.e., overdamped) if the following conditions are true:
$K_p > 0$
$K_p^2 - 4K_i > 0 \Rightarrow K_p^2 > 4K_i$
$\sqrt{K_p^2 - 4K_i} < K_p \Rightarrow K_p^2 - 4K_i < K_p^2 \Rightarrow K_i > 0$

System will be stable with two complex roots (i.e., underdamped) if the following conditions are true:
$K_p > 0$
$K_p^2 - 4K_i < 0 \Rightarrow K_p^2 < 4K_i$

PID-Controller:
$x(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt}e(t)$

By taking the Laplace transform we get:
$X(s) = K_p E(s) + \frac{K_i}{s}E(s) + K_d s E(s) \Rightarrow D(s) = \frac{X(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s$

Replacing $D(s)$ into the equation for $B(s)$ we get:
$B(s) = \frac{(K_p + \frac{K_i}{s} + K_d s)T(s)}{s + (K_p + \frac{K_i}{s} + K_d s)} - \frac{C(s)}{s + (K_p + \frac{K_i}{s} + K_d s)}$

Roots of the system's characteristic equation are:
$s + K_p + \frac{K_i}{s} + K_d s = 0$
$s^2 + K_p s + K_i + K_d s^2 = 0$
$(1 + K_d)s^2 + K_p s + K_i = 0$
$a = 1 + K_d,\ b = K_p,\ c = K_i$
$\Delta = b^2 - 4ac = K_p^2 - 4(1 + K_d)(K_i) = K_p^2 - 4K_i(1 + K_d)$

$s = \frac{-b \pm \sqrt{\Delta}}{2a}$

$s = \frac{-K_p \pm \sqrt{K_p^2 - 4K_i(1+K_d)}}{2(1+K_d)}$

$s = -\frac{K_p}{2(1+K_d)} \pm \frac{1}{2(1+K_d)}\sqrt{K_p^2 - 4K_i(1 + K_d)}$

System will be stable with two real roots if the following conditions are true:
$\frac{K_p}{1+K_d} > 0$
$K_p^2 - 4K_i(1+K_d) > 0 \Rightarrow K_p^2 > 4K_i(1+K_d)$
$\sqrt{K_p^2 - 4K_i(1+K_d)} < K_p \Rightarrow K_p^2 - 4K_i(1+K_d) < K_p^2 \Rightarrow 4K_i(1+K_d) > 0$

System will be stable with two complex roots if the following conditions are true:
$\frac{K_p}{1+K_d} > 0$
$K_p^2 - 4K_i(1+K_d) < 0 \Rightarrow K_p^2 < 4K_i(1+K_d)$

(c) Final Value Theorem
$\lim_{t\to\infty} f(t) = \lim_{s\to 0} sF(s)$

Thus, we have:
$\lim_{t\to\infty} b(t) = \lim_{s\to 0} sB(s)$

Note that,
$B(s) = \frac{T(s)D(s)}{s+D(s)} - \frac{C(s)}{s+D(s)}$

Let the service rate be constant and equal to C. We therefore have, $C(s) = \frac{C}{s}$. Similarly, let the target buffer size be a constant and equal to T. We therefore have, $T(s) = \frac{T}{s}$. Replacing these two values into the equation for $B(s)$, we get:
$B(s) = \frac{\frac{T}{s}D(s)}{s+D(s)} - \frac{\frac{C}{s}}{s+D(s)}$

Multiplying by s to get $sB(s)$, we get:
$sB(s) = \frac{TD(s)}{s+D(s)} - \frac{C}{s+D(s)}$

P-Controller:
$D(s) = K_p$

$sB(s) = \frac{TK_p}{s+K_p} - \frac{C}{s+K_p}$

$\lim_{s\to 0} sB(s) = \lim_{s\to 0}[\frac{TK_p}{s+K_p} - \frac{C}{s+K_p}] = T - \frac{C}{K_p}$

PI-Controller:
$D(s) = K_p + \frac{K_i}{s}$

$sB(s) = \frac{T(K_p+\frac{K_i}{s})}{s+(K_p+\frac{K_i}{s})} - \frac{C}{s+(K_p+\frac{K_i}{s})} = [\frac{T(K_ps+K_i)}{s^2+K_ps+K_i} - \frac{Cs}{s^2+K_ps+K_i}]$

$\lim_{s\to 0} sB(s) = \lim_{s\to 0}[\frac{T(K_ps+K_i)}{s^2+K_ps+K_i} - \frac{Cs}{s^2+K_ps+K_i}] = T$

PID-Controller:
$D(s) = K_p + \frac{K_i}{s} + K_ds$

$sB(s) = \frac{T(K_p+\frac{K_i}{s}+K_ds)}{s+(K_p+\frac{K_i}{s}+K_ds)} - \frac{C}{s+(K_p+\frac{K_i}{s}+K_ds)} = \frac{T(K_ps+K_i+K_ds^2)}{s^2+K_ps+K_i+K_ds^2} - \frac{Cs}{s^2+K_ps+K_i+K_ds^2}$

$sB(s) = \frac{T(K_ps+K_i+K_ds^2)}{(1+K_d)s^2+K_ps+K_i} - \frac{Cs}{(1+K_d)s^2+K_ps+K_i}$

$\lim_{s\to 0} sB(s) = \lim_{s\to 0}[\frac{T(K_ps+K_i+K_ds^2)}{(1+K_d)s^2+K_ps+K_i} - \frac{Cs}{(1+K_d)s^2+K_ps+K_i}] = T$

(d) We leave it to the reader to show the plots for $b(t)$.

5. (a) At steady state, $w_k \longleftarrow w^*$. This implies that $w^* = w^* + \alpha(T - w^*)$. Thus, $w^* = T$. This means that there is no necessary condition, since the system will always converge to the target window size.

(b) We have $w_{k+1} \longleftarrow w_k + \alpha(T - w_k)$. Re-writing this equation we get, $w_{k+1} \longleftarrow w_k(1-\alpha)+\alpha T$. Assume $w_0 < T$, let the Lyapunov function be $L(w) = T-w > 0$.

$L(w_{k+1}) = T - w_{k+1}$
$L(w_{k+1}) = T - [w_k + \alpha(T - w_k)]$
$L(w_{k+1}) = T - w_k - \alpha T + \alpha w_k$
$L(w_{k+1}) = (1 - \alpha)T - (1 - \alpha)w_k$
$L(w_{k+1}) = (1 - \alpha)(T - w_k)$
$L(w_{k+1}) = (1 - \alpha)L(w_k) < L(w_k)$

This is true since $(1 - \alpha) < 0$. In other words, $L(w)$ is indeed a decreasing function.

If instead we assume that $w_0 > T$, then the Lyapunov function would be $L(w) = w - T > 0$. Similarly we have,

$L(w_{k+1}) = w_{k+1} - T$
$L(w_{k+1}) = [w_k(1-\alpha) + \alpha T] - T$
$L(w_{k+1}) = w_k(1-\alpha) - (T - \alpha T)$
$L(w_{k+1}) = w_k(1-\alpha) - T(1-\alpha)$
$L(w_{k+1}) = (1-\alpha)(w_k - T)$
$L(w_{k+1}) = (1-\alpha)L(w_k) < L(w_k)$

This is true since $(1-\alpha) < 0$. In other words, $L(w)$ is indeed a decreasing function.

6. (a) $\frac{d}{dt}x(t) = 1 - x(t)p(x(t))$
$p(x(t)) = \alpha(x(t) - c)$
$\frac{d}{dt}x(t) = 1 - x(t)[\alpha(x(t) - c)]$
$\frac{d}{dt}x(t) = 1 - x(t)[\alpha x(t) - \alpha c)]$
$\frac{d}{dt}x(t) = 1 - \alpha x^2(t) + \alpha c x(t)$

The equation above is non-linear because it has an $x^2(t)$ term.

(b) Let $f(x) = 1 - \alpha x^2(t) + \alpha c x(t)$
$f'(x_0) = -2\alpha x_0 + \alpha c$
(Note that one may choose to linearize the x^2 term only. Here we choose to linearize the whole $f(x)$.)
$\Delta f = f'(x_0)\Delta x$
$\frac{d}{dt}\Delta x = f'(x_0)\Delta x$
Let $y(t) = \Delta x$
$\frac{d}{dt}y(t) = f'(x_0)y(t)$
$\frac{d}{dt}y(t) = [-2\alpha x_0 + \alpha c]y(t)$
Let $\beta = [-2\alpha x_0 + \alpha c]$
$\frac{d}{dt}y(t) = \beta y(t)$

(c) Transforming this equation to the s-domain, we get:

$sY(s) - y(0) = \beta Y(s)$
$sY(s) - \beta Y(s) = y(0)$
$Y(s)[s - \beta] = y(0)$
$Y(s) = \frac{y(0)}{s-\beta}$

The root of the characteristic equation $s - \beta$ is $s = \beta$. For the system to be stable, β must be less than zero. Assuming $\alpha > 0$, we need the following to be true:

$\beta < 0$
$-2\alpha x_0 + \alpha c < 0$
$-2\alpha x_0 < -\alpha c$
$2\alpha x_0 > \alpha c$
$x_0 > \frac{c}{2}$

7. We leave it to the reader to produce the plot. Notice that the fixed point is ($\lambda = 0.8, p = 0.2$) where the pricing curve and the load curve intersect. Also the system diverges for starting $\lambda = 0.2$, whereas it converges for $\lambda = 0.6$. Specifically, in the latter case, $\lambda = 0.6$ yields $p = 0.2$, which in turn yields $\lambda = 1 - p = 0.8$ and the system stabilizes at the fixed point.

