INTERACTIVE CONTROL ENGINEERING
Principles and Applications

Volume 2 Practical PID Control

CHAPTER 7

Delay Differential Equations and the Stability Boundary in a TCP Congestion Simulator

Pages 171–195

Edited by John D. Hedengren

How to cite this chapter. "Delay Differential Equations and the Stability Boundary in a TCP Congestion Simulator," in *Practical PID Control*, John D. Hedengren, Ed., vol. 2 of *Interactive Control Engineering: Principles and Applications*. Advanced Process Solutions, LLC, 2026, ch. 7, pp. 171–195.

The page numbers in this file are those of the printed volume.

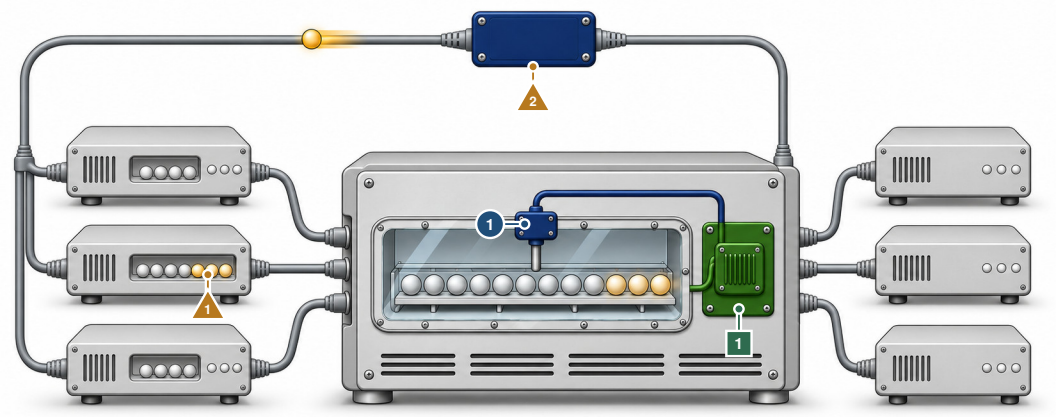
The app for this chapter
https://apmonitor.com/apps/tcp-congestion/

The volume online: contents, other chapters and citation records
https://apmonitor.com/apps/books/v02-practical-pid/#tcp-congestion

Copyright © 2026 Advanced Process Solutions, LLC. All rights reserved. Produced by Advanced Process Solutions, LLC.

CHAPTER 7

Delay Differential Equations and the Stability Boundary in a TCP Congestion Simulator



The controller moves 1 the share of packets the router marks

The controller reads 1 the length of the line of packets in the router

What disturbs it 1 more senders sharing the link 2 a longer route, which lengthens the round trip

Notice that a mark made at the router travels back over the upper cable, past the dark box that stands for the round-trip delay, before any sender can slow down. Illustration, not to scale.

IN PLAIN TERMS Holding a Router Queue Through a Delay

The main idea. Packets from many computers wait in a line at a router before one busy link. The router keeps that line near a target length by marking a few packets, and each sender that learns of a mark slows down.

The challenge. A mark reaches its sender only after a trip out and back, so the router acts on old news. On a longer trip each correction of a fixed setting comes later and hits harder, until the line swings.

Why it matters. A line that is too long delays every packet, even on a link that looks healthy.

THREE WAYS TO READ THIS CHAPTER

⊘	**Explore**	12 min	the first page, the history of congestion collapse and of the PI queue controller (Section 7.2), the fact on the queue setpoint, the try-it segments of scenarios 4 and 10 and the fact on the challenge (Section 7.8), and the key points
🕮	**Study**	50 min	the whole chapter with its three pause points and the twelve exercises at its end
⛰	**Practice**	15 min	the first page, the results (Section 7.8), Technology in Practice (Section 7.9), Limitations (Section 7.10) and the open challenge

The control question A router that marks packets to hold its queue acts on senders that learn of each mark one round trip later, and that round trip grows with the queue being held. At what round-trip time does a fixed PI tuning lose its phase margin, and does the simulated loop stop settling where the characteristic equation says it should?

Learning objectives

- Compute the operating point of the window and queue model and test whether it exists.
- Explain why the process gain rises as the cube of the round-trip time and falls as the square of the session count.
- Compute the phase margin of the PI loop behind a dead time and locate the critical round trip.
- Tell an actuator with no feasible operating point from a badly tuned loop.
- Explain how the queue setpoint sets the dead time of its own loop.

Prerequisites Sections 1.4.1 (dead time), 1.4.12 (phase margin), 1.4.3 (controller action), 1.6.2 and 1.4.9 (velocity form and windup). Chapter 6 is useful background on a gain that moves with the operating point.

The app https://apmonitor.com/apps/tcp-congestion/

See also Volume 5, Chapter 2 and Volume 5, Chapter 3 for control through a transport delay and a slow analyzer, and Volume 5, Chapter 8 for a loop closed over a delayed channel. Volume 7, Chapter 13 for a second delay differential equation with a stability boundary and Volume 7, Chapter 3 for a plant that the setpoint chooses.

7.1 Introduction

The internet moves data by running a feedback loop across every congested path. Senders raise their transmission rate until a router signals congestion, then cut back, and the equilibrium reached by that adjustment is what allocates a shared link among the flows using it [1, 2]. The congestion signal reaches a sender only after a full traverse of the path. A router that decides now to mark a packet cannot change any sender's rate until the marked packet has reached its destination and the acknowledgment has come back, and the effect of the changed rate does not appear at the router for a further round trip. The loop therefore carries a dead time equal to the round-trip time of the path, and no protocol design removes it.

Dead time appears in many process control problems, usually as a transport lag appended to a lumped model of the process, as in the first-order-plus-dead-time model of Section 1.4.1. This system is different in three ways that are worth separating. The delay is inside the state equations rather than in series with them, because the window adaptation at time t is driven by the marking probability the sender learned about one round trip earlier. The length of that delay is a function of the queue, which is the very state the controller regulates, so the delay is state dependent and the setpoint moves the plant. The steady-state process gain is proportional to the cube of the same round-trip time, so lengthening the delay subtracts phase and adds gain at the same instant. A loop with a fixed tuning can therefore be pushed across its stability boundary by a route change that touches nothing else.

The physical content of the model is a mass balance and a rate law. The router queue is an inventory whose derivative is the difference between the packet arrival rate and the link drain rate, which is the accumulation that governs any surge vessel, and the congestion window obeys a rate law whose growth is proportional to the fraction of acknowledgments that were not marked and whose decay is proportional to the fraction that were. A separation of time scales between a window that adapts over several round trips and a queue that fills in a fraction of one is what allows an entire congested path to be represented by two ordinary differential equations [3, 4]. Experiments built on those two states cover process order, transport delay, the phase a dead time removes at the crossover frequency, actuator saturation and anti-windup, feedforward from a computed operating point, and a process gain that varies with the load rather than staying constant [3, 5]. The same combination appears in the process industries wherever the measurement is remote from the actuator, and a queue managed badly costs interactive latency on a link that every throughput measurement reports as healthy.

Fluid-flow models of TCP make all three effects computable. The window and queue equations of Misra, Gong and Towsley reduce a congested path to two states and one delay [6], and the control-theoretic linearization of Hollot and colleagues turns those two states into a second order transfer function behind a pure dead time [7–9]. The same model supports the classical feedback account of congestion control given by Low, Paganini and Doyle [10] and connects to the wider literature on time-delay systems, where stability is decided by the roots of a transcendental characteristic equation rather than a polynomial [11]. Volume 7, Chapter 13 treats a second delay differential equation, the regenerative chatter of a milling cutter, whose stability boundary moves with the spindle speed. However, the congestion material is normally presented either as networking research, in which the control content is a few paragraphs of Nyquist argument, or as delay-systems theory, in which the plant is an abstract example. Neither arrangement lets the reader adjust the delay and measure the phase margin that results.

This chapter works with a browser-based fluid-flow simulator of TCP and active queue management built around one adjustable quantity, the round-trip time, so that the collapse of a stability margin under a lengthening dead time becomes a measurement instead of an assertion.

The chapter is organized as follows. Section 7.2 recalls how congestion control and active queue management became feedback problems. Section 7.3 states the two-state delay differential model, its state-dependent delay, the operating point and every constant, completely enough for an independent reimplementation. Section 7.4 linearizes the model and solves the characteristic equation of the loop for a critical round trip of 297 ms. That prediction is then bracketed against the round trip at which the simulator itself stops settling. Section 7.5 describes the PI queue controller, which is the established design of the AQM literature and not a new one, and Section 7.6 gives the numerical machinery that advances a delay equation. Section 7.7 lists the ten scenarios and the graded challenge, arranged so that one mechanism is isolated at a time, and Section 7.8 reports the measured runs. Section 7.9 describes the devices that measure and mark a real queue. Sections 7.10 and 7.11 state the limitations of the model and its place in practice, Section 7.12 summarizes the results, and the exercises follow in Section 7.13.

7.2 Background

Congestion collapse was observed on the internet before any congestion controller existed, and the history box that follows tells how the first controller answered it [1]. The remedy was a control law. A sender increases its window by one packet per round trip while nothing is lost and halves it when a loss is detected, which is the additive-increase and multiplicative-decrease rule whose convergence to a fair and efficient allocation was analyzed by Chiu and Jain [2].

> ⧖ HISTORY **Congestion Collapse, 1984 to 1988**
>
> A technical note of January 1984 described congestion collapse. A sudden load can raise the round trip faster than the hosts can update their estimates of it, the hosts then send several copies of the same packets, the buffers of the switching nodes fill, and the condition is stable [12]. In October 1986 the internet had the first of a series of such collapses: throughput between two sites 400 yards apart fell from 32 kbit/s to 40 bit/s. The repair, published in 1988, made the sender's window a control law. The window is halved on congestion and grows by a small constant step otherwise, additive increase and multiplicative decrease. Its authors argued that a network is, to a very good approximation, a linear system built of integrators, delays and gain stages [1].

That first generation of congestion control placed the whole loop in the end hosts and left the router with a single signal, buffer overflow. A drop-tail router gives no indication of congestion until its buffer is full, by which time every packet in it is delayed by the full buffer drain time and every flow sharing the link receives the same synchronized loss. Random early detection was proposed to move the decision into the router, dropping or marking arriving packets with a probability computed

from an averaged queue length [13]. The Internet Architecture Board recommended active queue management for deployment on that basis in 1998, and named the determination of good parameter settings as an open problem [14]. It remained one, because RED was specified as a static mapping from average queue to drop probability rather than as a controller, and its parameters are not the quantities a control designer would choose.

The analysis that reframed the problem came from two directions at once. The optimization view of Kelly, Maulloo and Tan interpreted the congestion signal as a shadow price and the sender's rate law as a gradient step, which established what a network of such loops converges to and made the stability of the dynamics a separate question [15]. The fluid-flow view of Misra, Gong and Towsley wrote the coupled stochastic differential equations for a network of AQM routers and their TCP flows and reduced them to a small set of ordinary equations with delay [6]. Hollot and colleagues linearized that model, showed that RED is a low-pass filter followed by a proportional gain, and demonstrated that its published parameter guidance gives a loop whose stability depends on load and round-trip time in ways the guidance does not mention [7]. The same authors then designed a proportional-integral queue controller for the linearized plant and showed it holds the queue at a target with faster response than RED [8, 9]. The controller of this chapter is that one.

> ⧖ HISTORY **A PI Queue Controller in the Cable Modem, 2013**
>
> The PI queue controller left the research papers for the edge of the network. A design named Proportional Integral controller Enhanced (PIE) applies the classical PI controller, studied earlier for queue length, to the queueing delay, and its parameters are chosen by a stability analysis of the feedback loop [16]. Version 3.1 of the cable data service specification, first published in October 2013, requires cable modems to run a variant of it on their upstream queues. That technology gives broadband access to over one hundred million users [17]. Both designs were published as Requests for Comments in February 2017.

The consequences reached the public as latency rather than as loss. Cheap memory allowed edge devices to be built with buffers holding seconds of traffic, and a full buffer converts spare capacity into delay without reducing throughput, so every conventional measurement of a link reports that it is working while interactive use degrades. Gettys and Nichols named the condition bufferbloat and traced it through home routers, cable modems and operating system stacks [18]. The deployed answers are queue managers that target delay directly rather than occupancy [19]. Every one of them is a feedback controller operating around one round trip of dead time, so the phase margin argument of this chapter applies to each of them regardless of the marking rule.

7.3 Network Model

A single bottleneck router carries N long-lived TCP sessions from a group of senders to a group of receivers. The router transmits C packets per second, holds up to B packets while waiting, and marks or drops a fraction p of arriving packets to signal congestion. A marked packet is noticed by the receiver, reported in an acknowledgment and acted on by the sender, so a marking decision reaches the traffic one round trip after it is made. The sending hosts, the bottleneck, the marking block, the return path and every symbol of the model appear in Figure 7.1. The instantaneous queue length q is the controlled variable and the marking probability p is the manipulated variable.

7.3.1 Variables and Signal Classification

Table 7.1 classifies the quantities the operator and the controller see. Only two of them are integrated states. The round-trip time, the arrival rate, the equilibrium window and the loop gain are all algebraic functions of those two states and the parameters, and all of them are reported as outputs.

The queue setpoint is a latency setting as well as a control target. A queue of q packets on a link of C packets per second delays every packet that joins it by q/C seconds, so the setpoint is a latency

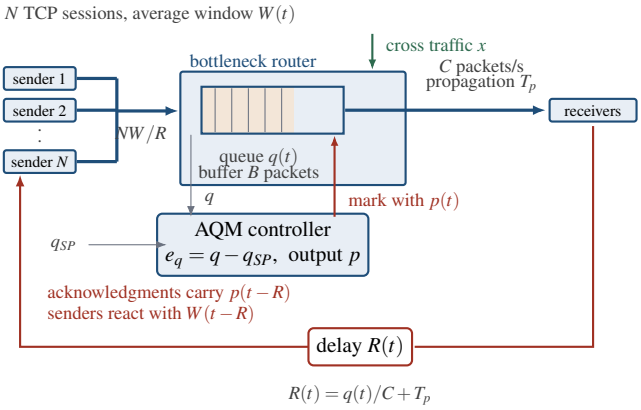


Figure 7.1 Congested path modeled by the application, with the marking decision and the traffic it controls separated by one round trip. The senders offer NW/R packets per second to a router that drains at C, and the AQM computes a marking probability from the queue error. The mark reaches a sender only after the return path, so the queue responds to $p(t-R)$ and the length of that delay depends on the queue itself. The error printed in the controller block is the convention of Equation (7.10) with the action flag already applied.

Table 7.1 Signal Roles on the Queue Loop. Each Nominal Entry Belongs to the Rest State That Opens Every Scenario.

Symbol	Role	Nominal	Range
q	CV, state	100 packets	10–400 packets SP
p	MV	1.556 %	0–100 %
W	state	11.25 packets	≥ 1 packet
R	output, delay	180 ms	T_p to $B/C + T_p$
N	DV, declared	20 sessions	1–200
ΔN	DV, operator	0 sessions	-15 to $+500$
x	DV, operator	0 packets/s	0–900 packets/s
C	DV, parameter	1250 packets/s	300–4000 packets/s
T_p	DV, parameter	100 ms	20–1000 ms
B	parameter	400 packets	60–1200 packets

budget written in a unit of storage. At the default settings the 100-packet target is 80 ms of queueing delay added to every packet of every session.

> **DID YOU KNOW? Five Milliseconds of Standing Queue**
>
> A delay-based queue manager published in 2018 sets its target to 5 ms of persistent queueing delay for normal internet traffic, the low end of a best range of 5 to 10 % of the round trip, and configures itself from a round-trip metric of 100 ms for the terrestrial internet [20]. The 100-packet target of this chapter adds 80 ms.

7.3.2 Window and Queue Equations

The average congestion window carried by the N sessions obeys the additive-increase and multiplicative-decrease rule of TCP Reno written as a rate, as shown in Equation (7.1).

$$\dot{W}(t) = \frac{1 - p(t-R)}{R(t)} - \frac{W(t)}{2}\,\frac{W(t-R)}{R(t-R)}\,p(t-R) \tag{7.1}$$

where W is the average window in packets, R is the round-trip time in seconds and p is the marking probability. The first term is additive increase, one packet per round trip, applied only for acknowledgments that were not marked. The second term is multiplicative decrease, in which marked acknowledgments arrive at $W(t-R)p(t-R)/R(t-R)$ per second and each one halves the window. Because W is a count of packets and is therefore dimensionless, both terms carry units of 1/s and the equation is dimensionally consistent. Retaining the $(1-p)$ factor is a small departure from the original fluid model [6], and its only visible consequence is stated in Section 7.3.4.

Every term carrying the marking probability in (7.1) is evaluated at $t-R$. That is the whole of the dead time in this application, and it is where the delay differential equation comes from. The packets whose acknowledgments are arriving now passed the router one round trip ago, so they were marked or not marked according to a decision the router has since revised hundreds of times.

The router queue is a mass balance on packets, as shown in Equation (7.2).

$$\dot{q}(t) = -C + \frac{NW(t)}{R(t)} + x, \qquad 0 \le q \le B \tag{7.2}$$

where C is the link capacity in packets per second, x is unresponsive cross traffic in packets per second and B is the buffer in packets. Each session holds W packets in flight over R seconds, so N of them deliver NW/R packets per second and the link removes C of them. Both sides carry units of packets per second. The two bounds on q are hard events and are applied after each integration step rather than inside the right-hand side, which is discussed in Section 7.6.

7.3.3 State-Dependent Round-Trip Time

The round-trip time is the propagation delay of the path plus the time a packet spends waiting in the bottleneck queue, as shown in Equation (7.3).

$$R(t) = \frac{q(t)}{C} + T_p \tag{7.3}$$

where T_p is the propagation delay in seconds. Equation (7.3) is the single line that sets this plant apart from a process with a fixed transport lag. The delay that limits the loop is computed from the state the loop is regulating. Raising the queue setpoint from 100 to 250 packets on the default link raises the round trip from 180 to 300 ms, which is enough on its own to destabilize the default tuning. A control problem in which the setpoint changes the plant is unusual, and it is the reason the modern answer to bufferbloat is to specify the target in units of time rather than in units of storage [19]. Volume 7, Chapter 3 treats a mechanical plant that its setpoint chooses in the same sense.

The propagation delay is floored at 20 ms. That is a numerical constraint rather than a physical one and its reason is given in Section 7.6. The window is floored at one packet, which is physical, because TCP always keeps one segment in flight.

Table 7.2 Model and Controller Parameters. All Are Visible and All Are Adjustable While a Run Is in Progress.

Symbol	Meaning	Value	Unit
Plant			
C	bottleneck link capacity	1250	packets/s
N	declared TCP sessions	20	sessions
T_p	propagation delay	0.100	s
B	router buffer	400	packets
L	packet size	1500	bytes
$W_{\min}$	minimum window	1	packet
Controller, default tuning			
K_c	proportional gain	3.5×10^{-4}	1/packet
τ_I	integral time	5.0	s
τ_D	derivative time	0	s
τ_f	derivative filter	0	s
p_0	bias, from (7.4)	0.015557	—
Numerics			
Δt	integration step	0.005	s
Δt_c	controller period	0.005	s
—	historian rate	20	samples/s
—	delay-history capacity	2048	entries
—	seeded history span	12	s

7.3.4 Operating Point

Requiring both derivatives to vanish pins down the rest state of the network, as given by Equation (7.4).

$$R_0 = \frac{q_{SP}}{C} + T_p, \quad W_0 = \frac{R_0 (C - x)}{N}, \quad p_0 = \frac{2}{2 + W_0^2} \tag{7.4}$$

where q_{SP} is the queue setpoint. The queue balance fixes the window the link can support and the window equation then fixes the loss rate that sustains it. At the default settings this is $R_0 = 180\,\text{ms}$, $W_0 = 11.25$ packets and $p_0 = 1.5557\,\%$. Dropping the $(1 - p)$ factor from (7.1) would give $p_0 = 2/W_0^2 = 1.5802\,\%$ instead, a difference of $1.6\,\%$, and the exact form is the one the application's first verification test checks by evaluating both derivatives at (7.4) and requiring them to vanish to within 10^{-9}.

Equation (7.4) is also a feasibility test. If W_0 comes out below one packet the operating point does not exist, because no marking probability can make a TCP session send less than one segment per round trip. Five hundred sessions on the default link need $W_0 = 0.45$ packets, and the application reports the requested operating point as infeasible.

7.3.5 Parameters

Table 7.2 lists every constant. Nothing is hidden from the user in this application, because the lesson is structural rather than one of identification. The link speed is the one quantity that is not entered directly: $C = 1250$ packets per second at $L = 1500$ bytes per packet is $15.0\,\text{Mbit/s}$.

The proportional gain is entered as $3.50\,\%$ per 100 packets and stored as 3.5×10^{-4} per packet. The exported column `ctrl_Kc_per_packet` carries the stored value, so the two representations differ by a factor of 10^4 and must not be interchanged.

7.3.6 Assumptions

The two-state description rests on several assumptions, and each one fails in a specific place.

Fluid approximation. W and q are continuous averages over many packets and many sessions. The model has no notion of an individual packet, of burstiness inside a round trip or of the discrete nature

of a drop. It applies for N of order ten or more and does not apply to a single connection, where the sawtooth of one window dominates the average.

One congestion control algorithm. Windows grow by one packet per unmarked round trip and halve on a mark, which is TCP Reno with no slow start and no retransmission timeouts. CUBIC, BBR and Vegas reach different equilibrium loss rates, so (7.4) would change. The argument about dead time and phase margin would not.

One round trip for every session. Every session is assumed to share one bottleneck and to have the same propagation delay and the same average window. Real paths carry a spread of round-trip times, which acts as an additional low-pass filter and is mildly stabilizing. Real paths also carry multiple congested hops and reverse-path congestion, neither of which is represented and neither of which is stabilizing.

An ideal measurement. The AQM reads the instantaneous queue exactly, with no noise, no averaging and no sampling interval of its own, and applies its marking probability to arrivals immediately. Deployed queue managers average the queue and update on a timer, and both of those add lag to a loop that has none to spare. The delay in this application is entirely in the actuation path and none of it is in the instrument.

Window dilution when sessions join. W is an average over N sessions, so changing N changes what the state means. When the session count rises from N_{old} to N_{new} the average window is rescaled to $(N_{\text{old}}W + (N_{\text{new}} - N_{\text{old}}))/N_{\text{new}}$, which conserves the packets in flight apart from the one segment each newcomer brings. Without that correction a flash crowd would arrive with every new flow already at the incumbent window, which is an artifact of averaging rather than a property of TCP. Departing flows take their windows with them, so a falling N leaves the average alone.

No stochastic element anywhere. The physics contains no noise source, so two runs of the same scenario produce identical files. The only use of a random number generator in the application is in the display of individual packets, which does not feed back into the model. The seed of Section 1.6.1 therefore plays no part in the runs of this chapter.

7.4 Linearization and the Characteristic Equation

7.4.1 Linearized Plant

A first-order expansion of (7.1) and (7.2) around the rest state (7.4), taken with $x = 0$ and with the delay frozen at its rest value R_0, reproduces the two coupled equations the AQM literature works from [9]. These are Equations (7.5) and (7.6).

$$\delta\dot{W} = -\frac{2N}{R_0^2 C}\,\delta W - \frac{R_0 C^2}{2N^2}\,\delta p(t - R_0) \tag{7.5}$$

$$\delta\dot{q} = \frac{N}{R_0}\,\delta W - \frac{1}{R_0}\,\delta q \tag{7.6}$$

where δ denotes a deviation from the operating point. The window coefficient in (7.5) collects the sensitivity of the decrease term to both $W(t)$ and $W(t-R)$, which the linearization treats as one perturbation. The queue coefficient in (7.6) is exact, because $NW_0 = CR_0$ makes $\partial\dot{q}/\partial q$ equal to $-1/R_0$ with no approximation. Transforming and eliminating δW gives the plant the queue controller faces, as shown in Equation (7.7).

$$P(s) = \frac{\delta q(s)}{\delta p(s)} = -\frac{C^2}{2N}\,\frac{\mathrm{e}^{-sR_0}}{(s+a)(s+b)} \tag{7.7}$$

$$a = \frac{2N}{R_0^2 C}, \qquad b = \frac{1}{R_0}, \qquad |P(0)| = \frac{R_0^3 C^3}{4N^2} \tag{7.8}$$

where the two real poles are the TCP window dynamics at a and the queue at b, as shown in Equation (7.8). At the default operating point they are 0.988/s and 5.556/s, the numerator gain $C^2/2N$

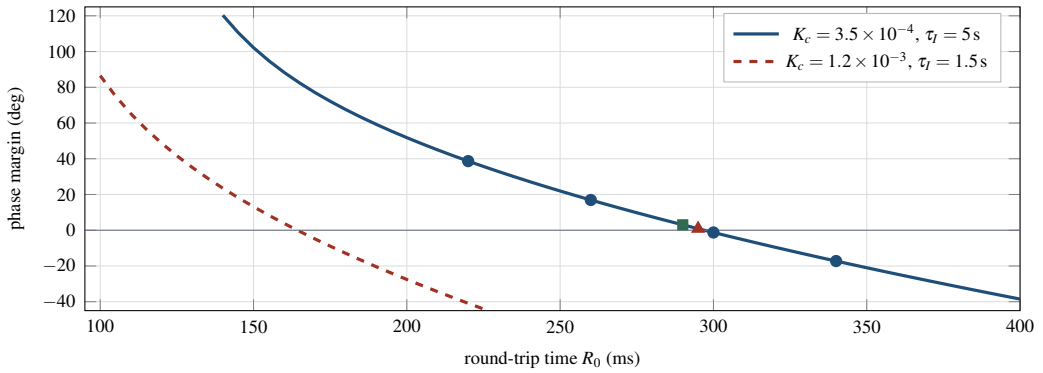


Figure 7.2 Phase margin against round-trip time computed from the characteristic equation (7.9) at the two tunings the application ships. The solid line is the default tuning and the dashed line is the faster one. Filled circles are the four operating points scenario 4 visits. On the 5 ms grid of the propagation-delay setting, the square at 290 ms is the longest round trip at which the simulator settles and the triangle at 295 ms is the shortest at which it sustains an oscillation, against a predicted crossing of the zero line at 297 ms.

is 3.906×10^4 packets per second squared per unit marking probability, and the DC gain is 7119 packets per unit marking probability. Retaining the $(1 - p)$ factor of (7.1) in the linearization moves the DC gain to 7346 packets, a difference of 3.2 %, and the application logs the simpler expression of (7.7) in the column `loop_dc_gain_packets`.

Three features of (7.7) and (7.8) matter more than the pole locations. The gain is negative, so a queue above target must be met with more marking rather than less. The gain varies as the cube of R_0, so a round trip twice as long multiplies it by eight. The gain varies as the inverse square of N, so the loop gain is highest on a lightly loaded path rather than on a busy one.

7.4.2 Phase Margin Against Round-Trip Time

Closing (7.7) with the PI controller of Section 7.5 gives a transcendental characteristic equation, as shown in Equation (7.9).

$$\tau_I s\,(s+a)(s+b) + \frac{K_c C^2}{2N}\,(\tau_I s + 1)\,\mathrm{e}^{-sR_0} = 0 \tag{7.9}$$

where the sign inversion of the process gain has been absorbed into the controller. Equation (7.9) has infinitely many roots because of the exponential, which is the defining property of a delay differential equation [11] and the reason a root-locus construction on a polynomial cannot be used here. What can be used is the frequency response, with the phase margin as Section 1.4.12 defines it. Dead time contributes a phase lag of ωR_0 radians and attenuates nothing at all, so it removes margin at the frequencies where the loop gain is near unity.

At the default tuning and the default operating point the loop gain crosses unity at $\omega_c = 2.09\,\mathrm{rad/s}$ with a phase margin of 67.6°, of which the dead time has already taken 21.6°. Holding the tuning fixed and lengthening R_0 does two things at once. The dead time takes more phase, and the cubic DC gain pushes crossover to a higher frequency where it takes more still. The phase margin against round-trip time computed from (7.9) is shown in Figure 7.2. Between $R_0 = 180\,\mathrm{ms}$ and $R_0 = 300\,\mathrm{ms}$ the margin falls from 67.6° to −1.3°, a loss of 68.9° with no gain touched, while crossover moves from 2.09 to 3.02 rad/s.

🕮 **Pause.** Try Exercise 7.1 before reading on.

7.4.3 Critical Round-Trip Time and Its Measurement

Solving (7.9) for the round trip at which the phase margin reaches zero gives $R_{0,\mathrm{crit}} = 297\,\mathrm{ms}$ at the default tuning, with a crossover of 3.01 rad/s and therefore an expected oscillation period of 2.09 s. Retaining the $(1-p)$ factor moves the prediction to 296 ms.

That prediction is testable inside the application. Starting from the nominal network at rest, stepping the propagation delay at $t = 5\,\mathrm{s}$ and running to $t = 100\,\mathrm{s}$ gives a queue transient whose peak-to-peak excursion either decays or does not. At $R_0 = 285\,\mathrm{ms}$ the excursion falls from 15.2 packets in the window $t = 25$–$40\,\mathrm{s}$ to 0.37 packets in $t = 85$–$100\,\mathrm{s}$, and at 290 ms it still falls, from 29.1 to 8.8 packets over the same two windows. At $R_0 = 295\,\mathrm{ms}$ it holds between 37.4 and 37.6 packets in every 15 s window from $t = 25\,\mathrm{s}$ onward and does not decay at all. The propagation delay is set in steps of 5 ms, so the finest bracket the application allows places the measured boundary between 290 and 295 ms, against a predicted 297 ms, and the measured period of the sustained oscillation at 295 ms is 2.07 s against a predicted 2.09 s. The remaining disagreement is in the direction the approximations predict, because the linearization holds the delay fixed at R_0 while the simulator lets it move with the queue.

The same calculation gives the ultimate gain of Section 1.4.7. Holding $R_0 = 180\,\mathrm{ms}$ and $\tau_I = 5\,\mathrm{s}$ and raising the proportional gain drives the phase margin to zero at $K_c = 1.06 \times 10^{-3}$ per packet, which is 10.6 % per 100 packets in the entered units. The default gain of 3.5×10^{-4} is a factor of 3.0 below that.

7.5 Controller Design

7.5.1 Discrete Algorithm

One loop is closed, and it runs the velocity-form PID algorithm of Section 1.6.2, with the derivative taken on the measurement [21]. Equations (7.10) to (7.14) restate that algorithm in the symbols of the queue loop, because the results of Section 7.8 cite them. They are Equations (1.23) to (1.26) with $s_k = S_k$, $c_k = v_k$ and $\tau_t = T_t$.

$$e_k = a\,(q_{SP} - q_k), \qquad a = -1 \tag{7.10}$$

$$\Delta p_k^P = K_c\,(e_k - e_{k-1}) \tag{7.11}$$

$$\Delta p_k^I = \frac{K_c}{\tau_I}\,e_k\,\Delta t_c \tag{7.12}$$

$$\Delta p_k^D = -\,K_c\,\tau_D\,a\,\frac{q_k^f - 2q_{k-1}^f + q_{k-2}^f}{\Delta t_c} \tag{7.13}$$

$$\begin{aligned} s_k &= s_{k-1} + \Delta p_k^P + \Delta p_k^D \\ c_k &= p_0 + s_k + I_{k-1} \\ p_k &= \min(1,\ \max(0,\ c_k)) \\ I_k &= I_{k-1} + \Delta p_k^I + \frac{\Delta t_c}{\tau_t}\,(p_k - c_k) \end{aligned} \tag{7.14}$$

where a is the controller action flag, p_0 is the output bias of (7.4), s_k accumulates the proportional and derivative increments without limit, c_k is the marking probability the algorithm demands, p_k is that demand clipped to the physical range of a probability, I_k carries the integral contribution and $\tau_t = \tau_I$ is the tracking time of the back-calculation. Differentiation uses a backward difference on a filtered queue, $q_k^f = \alpha q_{k-1}^f + (1-\alpha) q_k$ with $\alpha = \exp(-\Delta t_c/\tau_f)$. That coefficient comes from an exponential and cannot go negative, so the filter is stable at whatever period it is run.

The action flag is the entire expression of the negative process gain in (7.7). Setting $a = -1$ makes the error $q_k - q_{SP}$, so a queue above target produces more marking. The controller output therefore rises as the measurement rises, which is a direct-acting controller in the convention of Section 1.4.3 [3, 21]. The sign is checked by one of the application's own verification tests.

🕮 **Pause.** Try Exercise 7.2 before reading on.

Anti-windup is the back-calculation term of Equation (7.14), which corrects the integral alone for the reasons Section 1.4.9 gives [21]. At the default $\tau_I = 5\,\mathrm{s}$ each execution removes $1/1000$ of the clipped excess, and with $\tau_t = \tau_I$ a loop held against a limit settles where $c_k - p_k = K_c e_k$. Away from the limits the recursion is algebraically identical to the positional law with the same three gains, so the tuning constants and the linear analysis of Section 7.4 carry over. Against a limit it is not, and the saturated runs of Section 7.8 are those of the back-calculation as coded.

Windup protection is not optional on this loop. The marking probability is bounded to $[0, 1]$ by its own definition, and both bounds are reached in ordinary operation, $p = 0$ whenever the queue sits well below target and $p = 1$ whenever more sessions are present than the link can serve at one packet per round trip each. No slew limit is applied, because a router can change its marking probability between one packet and the next. The controller executes every 5 ms, the same period as the physics step, which is 2.8 % of the default round trip.

7.5.2 Output Bias and Manual Operation

The output bias is p_0 from (7.4), evaluated at the declared session count and the declared capacity and recomputed on every controller execution. It is a feedforward term computed from the model rather than a constant. It places the marking probability at the value the target operating point requires on the first execution, and it moves with the queue setpoint, so a setpoint change of 100 to 160 packets steps the marking probability toward its new equilibrium immediately rather than waiting for the integral. The disturbances are excluded from it deliberately. Extra sessions and cross traffic do not enter the bias calculation, so the mismatch between the declared network and the real one is left entirely to the integral term.

Manual operation tracks the operator-set marking probability, and automatic operation copies the applied marking probability to the manual setting after every controller execution, so transfer is bumpless in both directions as Section 1.6.2 describes. The manual actuator is the marking probability itself, entered over 0 to 100 % at a resolution of 0.05 %. An equilibrium helper applies p_0 computed for the traffic actually present, and it reports the operating point as unreachable when $W_0 < 1$.

7.5.3 Derivative Action and Drop-Tail Operation

Derivative time is adjustable from 0 to 2 s and defaults to zero. Derivative action on a process whose dead time dominates its lag differentiates a queue slope that the senders committed to one round trip ago and can no longer change, so the prediction it forms is of an interval that is already determined. Raising τ_D is an instructive experiment rather than a recommended tuning. The derivative filter τ_f is relevant only when τ_D is nonzero.

The active queue manager can be switched off. The router then marks at the fixed nominal rate p_0 and the queue finds its own level, which is the drop-tail behavior that motivated active queue management in the first place [13]. The idle PI block keeps tracking the applied value while it is off, so switching it back on during a run produces no step.

7.6 Numerical Solution of the Delay Equations

A delay differential equation needs two things an ordinary initial value problem does not: a history of the solution reaching back at least one delay, and an initial function rather than an initial point.

The history is a ring buffer of 2048 entries, each holding the four numbers $[W, q, R, p]$ at one completed physics step. At 5 ms per step that is 10.24 s of stored past, which exceeds the longest reachable round trip of $B/C + T_p \le 5\,\mathrm{s}$. Reading the history at an arbitrary time performs a binary search for the bracketing pair of entries and interpolates linearly between them, and clamps to the oldest or newest stored value outside the range. The buffer is extended by exactly one entry per completed step, stamped at the new simulation time and holding the values after the hard constraints have been applied, so the value the senders react to is the value the router applied.

The initial function is a constant. At every reset the history is seeded with two identical entries, one stamped at $t = -12\,\mathrm{s}$ and one at $t = 0$, both carrying the initial state and the equilibrium marking probability. Linear interpolation between two equal values is a constant over the whole span, and anything older clamps to the same value, so the run opens from a network that has held that condition over the whole seeded span. This is the correct initial condition for (7.1) and it is also what makes a rerun with the same settings produce an identical file, because the controller object and its history are rebuilt from scratch at every reset.

Integration uses the classical fourth-order Runge-Kutta method at a fixed step of 5 ms, the framework default of Section 1.6.1. That choice requires justification on a delay equation, because a Runge-Kutta stage evaluates the right-hand side at $t + \Delta t$ and therefore asks the history for the state at $t + \Delta t - R$, which would be an extrapolation if R were smaller than Δt. The propagation delay floor of 20 ms removes that possibility. It guarantees $R \geq 4\Delta t$ even with an empty queue, so the latest query any stage can make is $t - 3\Delta t$, which lands strictly inside the stored history and is served by ordinary interpolation. Nothing is extrapolated. An explicit Euler step would need no such floor, because it queries the history only at the start of the step, and the shared framework selects Euler for applications whose right-hand side is discontinuous. This application keeps the fourth-order method by moving its discontinuities out of the derivative function. The queue bounds of (7.2) and the one-packet window floor are applied between steps rather than inside them, which leaves the right-hand side smooth across every stage.

Two approximations remain, and both are documented in the application. The delay is resolved to within one step, which is 2.8 % of the round trip at the 180 ms default, about 5 % in the short round-trip scenario and at worst 25 % in the corner where T_p is set to its 20 ms minimum and the queue is held empty. Applying the bounds at step boundaries makes the first step after a boundary contact first-order accurate rather than fourth-order, which matters only at an empty queue and at the buffer limit.

Of the three clocks of Section 1.6.1, the physics and the controller both advance at 5 ms, an unusually fast controller relative to the process and one chosen so that no sampling artifact confuses the delay argument, while the log takes 20 rows per simulated second. Six checks are built into the application itself. They cover the rest condition of (7.4), the sign of the queue response to marking, the sign of the queue response to added sessions, the sign of the controller, the interpolation and clamping of the delay buffer against a known ramp, and the assertion that a step in marking moves the queue one round trip later rather than sooner.

7.7 Scenarios

Every scenario is built on the same nominal network of 20 sessions, 1250 packets per second, 100 ms of propagation delay and a 100-packet target, and is arranged so that each run isolates a single mechanism. Table 7.3 lists them. Two tunings recur. The default $K_c = 3.5 \times 10^{-4}$ with $\tau_I = 5\,\mathrm{s}$ was designed at the nominal 180 ms round trip. The alternative $K_c = 1.2 \times 10^{-3}$ with $\tau_I = 1.5\,\mathrm{s}$ was designed by the same procedure at a 100 ms round trip, where the plant is far less sensitive and the gain can be three times larger. Scenario 8 keeps the default gain and shortens the integral time to 1.5 s. Table 7.4 gives each scenario under the name the application shows, with the requirement for completing it under the rules of Section 1.6.4. The challenge is judged on the case as given, so changing a model parameter or a disturbance voids the attempt, and the footnote of the table gives its outcome under the shipped and the suggested tuning.

Two operator-set disturbances and four timed events are available in every scenario. The disturbances are extra TCP sessions, from -15 to $+500$ in steps of 5, and unresponsive cross traffic in packets per second, which consumes capacity and ignores marking entirely. The events are a flash crowd of 40 sessions, the departure of every added session, a halving of the link capacity and a route change adding 120 ms of propagation delay.

Table 7.3 Scenarios and the Mechanism Each One Exercises

#	Change applied	Mechanism exposed
1	none, network at rest	an equilibrium that needs no controller
2	N 20 to 40 at $t = 6$ s	operating point moves, CV does not
3	C 1250 to 600 packets/s at $t = 6$ s	same queue, longer latency
4	T_p to 140, 180, 220, 260 ms at $t = 8, 30, 52, 74$ s	the stability boundary at fixed tuning
5	+40 sessions at $t = 6$ s	a peak affected by dead time and tuning
6	q_{SP} 100 to 160 to 40 packets	the setpoint is the plant and the latency
7	$T_p = 20$ ms, fast tuning, route change to 160 ms at $t = 16$ s	a correct design commissioned elsewhere
8	+480 sessions at $t = 5$ s, $\tau_I = 1.5$ s	no feasible operating point, windup
9	$B = 140$, $q_{SP} = 60$, +30 sessions	overshoot larger than headroom
10	AQM off, +40 sessions at $t = 6$ s	bufferbloat with the loop open
C	$q_{SP} = 120$, aggressive tuning, crowd arrives and leaves	gain scheduling against session count

Two of the scenarios are deliberately extreme. Five hundred sessions arriving at once on a 15 Mbit/s link would not occur in this form on a physical network, and the scenario is there to show how an operating point that does not exist appears from inside the loop. A propagation delay stepped four times in 100 s is equally artificial, and it is the only arrangement that holds everything else constant while the delay moves. The scripted steps of scenario 4 overwrite any earlier setting of the propagation delay. The boundary is therefore explored by hand either after the last scripted step at $t = 74$ s or on the undisturbed network of scenario 1, as in Section 7.4.

7.8 Results

Table 7.5 collects the scorecard values at the stated simulated time in each case. The trajectories behind those numbers were taken from the 40-column comma-separated export at 20 samples per simulated second. The integrated measures accumulate over the whole run, while the last two columns are instantaneous readings, so for the three runs that end in a limit cycle (4, 7 and the shipped challenge tuning) those two columns depend on the phase of the cycle at the reading instant. The saturated fraction of scenario 4 is exactly 347 of its 2000 logged samples, so it is given to two decimals rather than rounded either way. Figure 7.3 draws three of its columns.

7.8.1 Dead Time Under a Flash Crowd

Scenario 5 makes the dead time visible in a single trace. Forty sessions open at $t = 6.00$ s and the controller responds from its next execution at 6.005 s, raising the marking probability from 1.56 to 2.99 % within 0.2 s. The column `mark_probability_delayed`, which is $p(t - R)$ and therefore the value the senders are reacting to, is still 1.556 % at $t = 6.20$ s and first moves between 6.20 and 6.25 s, one round trip after the event. No marking decision made in that interval had reached a sender. The queue peaks at 226.8 packets at $t = 7.18$ s at the 5 ms step, and the initial buildup is forced by the disturbance and the feedback delay. Tuning still changes the eventual peak and decides whether the recovery is monotone. The delay does not make that peak independent of the gain.

The steady state that follows is the one (7.4) predicts. Sixty sessions sharing the same link settle at $W_0 = 3.75$ packets, which is sustainable only at a marking probability of 12.45 %, and the integral term raises the output from 1.56 toward that value while the queue returns to target.

7.8.2 Stability Boundary Under Stepped Propagation Delay

Scenario 4 holds the tuning, the traffic, the capacity and the buffer fixed and raises the propagation delay in four steps. The round trip at the 100-packet target is $R_0 = q_{SP}/C + T_p$, which gives 220, 260,

300 and 340 ms. The behavior at each is summarized in Table 7.6 and is shown in Figure 7.4. The first two steps ring and recover, the third does not, and the transition falls where (7.9) places it.

The decay of the ringing tracks the predicted margin. At $R_0 = 220$ ms the successive queue peaks above target are 19.4, 5.3, 1.9 and 1.0 packets, a first decay ratio of 0.27. At $R_0 = 260$ ms they are 23.0, 13.0, 7.4, 4.2, 2.4 and 1.4 packets, a ratio of 0.57 held on every cycle and a longer settling. At $R_0 = 300$ ms the ratio reaches unity and the queue holds a 40-packet swing with the marking probability against its lower limit 22 % of the time. Nothing about the controller changed at any of those steps. The oscillation period is 2.0 s at 260 ms and 2.11 s at 300 ms, against the 2.09 s that ω_c at the boundary predicts.

Table 7.4 Scenarios and Completion Requirements

#	Scenario	Kind	Required to complete
1	Nominal network	Experiment	Hold Auto for 10 s with no added sessions, then switch to Manual, add 40 sessions and hold the queue within ±25 packets of 100 for 5 s by setting the marking probability by hand.
2	Twice the TCP sessions	Experiment	Run in Auto through the step from 20 to 40 sessions at $t = 6$ s until the queue has held within 3 packets of target for 5 s.
3	Link capacity cut in half	Experiment	Run in Auto through the cut to 600 packets/s at $t = 6$ s until the queue has held within 3 packets of 100 for 5 s.
4	Propagation delay, step by step	Experiment	Run the full 100 s in Auto through the steps of T_p to 140, 180, 220 and 260 ms at $t = 8$, 30, 52 and 74 s.
5	Sudden traffic arrival (flash crowd)	Experiment	Run in Auto past the 40-session arrival at $t = 6$ s to at least $t = 10$ s, then Reset, switch to Manual before the arrival, move the marking probability and stay in Manual for 12 s ending by $t = 18$ s.
6	Moving the queue target	Experiment	Run in Auto through the target steps to 160 and 40 packets at $t = 8$ and 35 s to $t = 50$ s, then Reset and, after the step to 160 packets, hold a 250-packet target for 20 s.
7	Tuned for a short RTT, run at a long one	Demonstration	Run in Auto with the fast tuning through the route change at $t = 16$ s until link utilization falls below 95 %, and on to $t = 60$ s, then Reset and hold $K_c = 3.5$ % per 100 packets and $\tau_I = 5$ s from before the route change for 45 s.
8	Packet marking saturated at 100 %	Demonstration	Run in Auto past the 480-session arrival at $t = 5$ s until the marking probability has held at 100 % for 5 s, then remove the extra sessions and run 5 s with the marking probability off its upper limit.
9	Small router buffer	Demonstration	Run in Auto to $t = 40$ s with the shipped tuning and more than 0.05 s of buffer overflow, then after Reset retune K_c or τ_I before $t = 6$ s and run to 40 s with no overflow, then after a second Reset hold a 30-packet target from before $t = 6$ s to 40 s.
10	Controller off (tail drop only)	Demonstration	With the AQM off, run in Auto until the queue has held at the 400-packet buffer limit for 3 s, then switch the PI controller on and run until the queue has held within 5 packets of 100 for 3 s.
C	Challenge, ride out a flash crowd	Challenge	With the case as given and q_{SP} left at 120 packets, run 75 s with the queue within ±25 packets of target from $t = 49.5$ s, mean link utilization of at least 99 % and no more than 0.05 s of buffer overflow.[a]

[a] The shipped tuning, $K_c = 15$ % per 100 packets and $\tau_I = 1.2$ s, fails with a worst queue error of 46 packets over the judged window, 21 packets outside the band, at 100.0 % mean utilization and no overflow. The suggested tuning, $K_c = 3.5$ % per 100 packets and $\tau_I = 5$ s, passes with a worst error of 9 packets, a margin of 16 packets, at 100.0 % mean utilization and no overflow, whether it is entered before Run or carried through Reset.

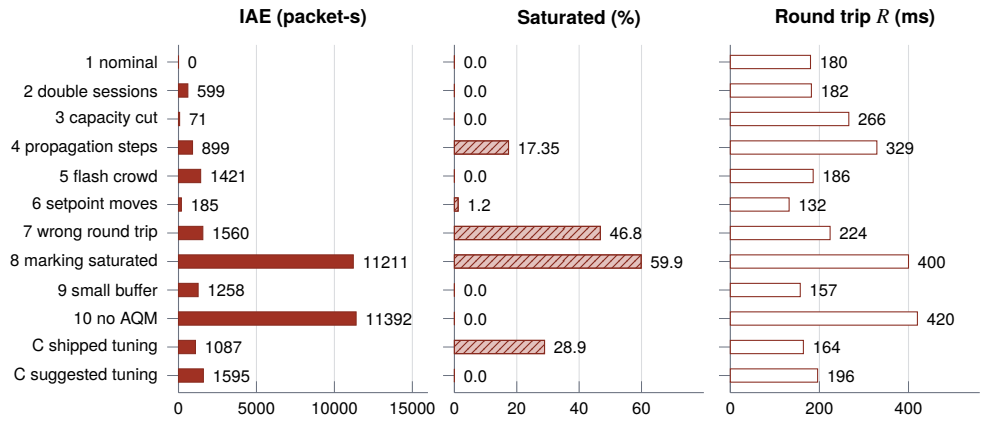


Figure 7.3 Three columns of Table 7.5 at the end of each run: integrated absolute error (solid bars), the share of the run spent against a marking limit (hatched) and the round trip at the reading instant (open). Notice that scenarios 8 and 10 carry almost the same error from opposite causes, one with the marking probability saturated for most of the run and the other with no marking change at all, and that the drop-tail router of scenario 10 holds the longest round trip on the chart.

Table 7.5 Scorecard Values Read at the End of Each Run. IAE Is in Packet-Seconds and Marking Travel Is $\sum_k |p_k - p_{k-1}|$.

Scenario	t (s)	IAE	travel	sat. (%)	$\lvert e_q \rvert$	R (ms)
1 nominal	25	0	0.00	0.0	0	180
2 double sessions	40	599	0.05	0.0	2	182
3 capacity cut	40	71	0.05	0.0	0	266
4 propagation steps	100	899	0.64	17.35	13	329
5 flash crowd	45	1421	0.10	0.0	8	186
6 setpoint moves	60	185	0.16	1.2	0	132
7 wrong round trip	60	1560	1.81	46.8	50	224
8 marking saturated	45	11211	0.98	59.9	275	400
9 small buffer	40	1258	0.09	0.0	12	157
10 no AQM	45	11392	0.00	0.0	300	420
C shipped tuning	75	1087	3.60	28.9	39	164
C suggested tuning	75	1595	0.13	0.0	0	196

Table 7.6 Scenario 4. Predicted Margin and Measured Queue Swing at Each Propagation-Delay Step. The Swing and the Saturated Fraction Are Both Taken Over the Second Half of Each Segment, After the Transient.

T_p (ms)	R_0 (ms)	$\lvert P(0) \rvert$	PM (deg)	swing (pkt)	sat. (%)
140	220	12 998	38.7	0.3	0
180	260	21 455	16.9	2.0	0
220	300	32 959	−1.3	40.4	22
260	340	47 979	−17.3	54.4	42

TRY IT IN THE APP **4 · Propagation Delay, Step by Step**

Load the case and press Run. Watch the router queue trend through the four steps of the propagation delay. After the first two steps the queue rings and settles. After the third, at a round trip of 300 ms, it holds a swing of about 40 packets, although nothing about the controller has changed.

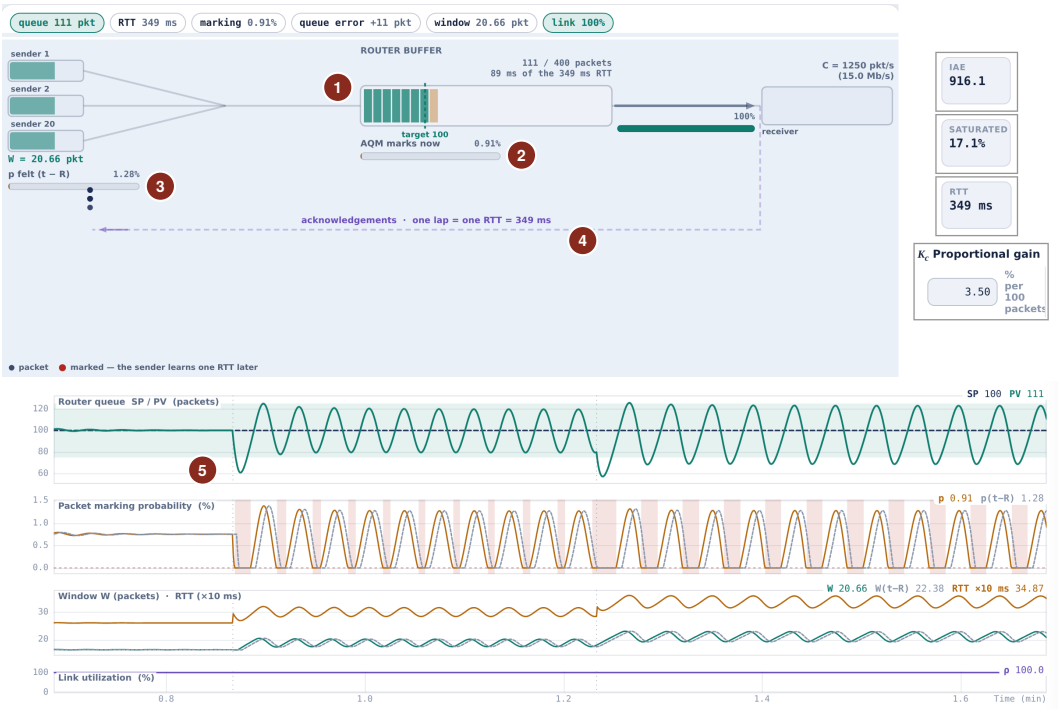


Figure 7.4 Crossing a stability boundary with nothing but the propagation delay. The screenshot was taken at $t = 101.2$ s of scenario 4, so its 60-second trend window opens on the settled tail of the 260 ms round trip, contains the step to 300 ms at $t = 52$ s and the step to 340 ms at $t = 74$ s. The queue strip changes from flat to a sustained swing at the first of those steps and to a larger one at the second. The marking strip shows the shaded bands where the probability sits at zero, which is 17.1 % of the run, and the window strip carries W, the same window one round trip earlier and the round-trip time itself, so the dead time appears as a horizontal displacement between two curves of the same shape. The scorecard reads IAE 916.1 packet-seconds and a round trip of 349 ms at this instant, and the proportional-gain field still holds the $K_c = 3.50$ % per 100 packets it started with. The figure is a composite: the readout chips and the plant view above the trend strips, with three scorecard tiles and the proportional-gain field moved into gray frames beside the plant view. Callouts: (1) the router queue against its 100-packet target, (2) the marking probability the queue manager applies now, (3) the marking the senders feel, $p(t - R)$, (4) the acknowledgment path, one lap of which is one round trip, and (5) the queue strip at the step to 300 ms.

7.8.3 A Tuning Commissioned at a Shorter Round Trip

Scenario 7 starts on a 20 ms path with the faster tuning, which has a phase margin of 86.5° at the 100 ms round trip it was designed for, and handles a setpoint move from 100 to 130 packets with a 48.9-packet excursion and no sustained motion. At $t = 16$ s the route changes and the propagation delay becomes 160 ms. The critical round trip for that tuning is 164 ms by (7.9), and the new operating point is 264 ms, so the predicted margin is −67°. The queue swings from empty to 159.4 packets in the first cycles after the route change and then settles into a limit cycle between 60.9 and 156.1 packets with a period of 2.23 s. The marking probability sits against a limit for 46.8 % of the run. The queue empties only once, from $t = 16.26$ to 16.64 s at the bottom of the first swing, and the link idles there to 76.2 % utilization at the 5 ms step, a dip the 20-sample-per-second log records as 79.0 %. Once the cycle is established the link stays fully used. The tuning is correct for the path it was designed on and incorrect for this one, and nothing about it was changed.

7.8.4 Marking Saturation and Drop-Tail Operation

Scenarios 8 and 10 produce almost the same integrated error from opposite causes. In scenario 8 five hundred sessions arrive on a link that can carry 1250 packets per second. The target operating point needs $W_0 = 0.45$ packets per session, TCP cannot go below one, and the marking probability runs to 100 % and stays there for 59.9 % of the run to $t = 45$ s. The queue then finds its own level. With every window pinned at one packet the offered load is N/R, which falls as the queue grows because the queue is inside the round trip, and the system settles where $N/R = C$, at $q = N - CT_p = 375$ packets and $R = 400$ ms. The controller has no authority left and a permanent error of 275 packets. The evidence that anti-windup is working is the pre-clamp command in `mark_probability_cmd`, which holds at 1.096, the value $1 + K_c e_k$ at which the back-calculation balances the integral increment, rather than ramping without limit. When the extra sessions are removed at $t = 45$ s the pre-clamp command falls at the first logged sample at $t = 45.05$ s, while the applied probability remains at its bound, and the marking probability leaves 100 % at the execution at $t = 45.235$ s, as soon as the draining queue crosses its target.

In scenario 10 the AQM is switched off and 40 sessions arrive. With p frozen at 1.556 % the window equation has no idea the link is now shared sixty ways, so each session returns to the same 11.25-packet window it had before. Sixty of them hold 675 packets in flight, the queue would have to reach $NW - CT_p = 550$ packets to accommodate them, and the 400-packet buffer stops it first. The queue parks against the limit for 36.3 s, the link runs at 100 % utilization throughout, and the round-trip time goes from 180 to 420 ms. With no active queue management the standing round trip is $B/C + T_p$, so the latency is set by the buffer size the operator specified and by nothing the loop does, which is the bufferbloat argument stated as an equation. Throughput and loss both report a healthy link throughout. Switching the PI controller back on mid-run returns the queue to within 25 packets of the 100-packet target after 26.4 s and within 5 packets after 53.0 s, from above and with the link at 100 % utilization throughout, whatever the time of the switch once the queue has parked. That recovery with no loss of throughput is the entire case for active queue management [14, 18].

> **TRY IT IN THE APP** **10 · Controller Off (Tail Drop Only)**
>
> Load the case and run it. The queue manager is off when 40 sessions arrive. Watch the round-trip tile climb from 180 to 420 ms while the link stays fully used. Then switch the PI controller on and watch the queue come back to its 100-packet target with no loss of throughput.

7.8.5 Load-Dependent Plant Gain in the Graded Challenge

The challenge ships with $K_c = 1.5 \times 10^{-3}$ and $\tau_I = 1.2$ s, and asks the user to hold a 120-packet target within ±25 packets through the arrival of 30 sessions at $t = 8$ s and their departure at $t = 32$ s. At the 120-packet target $R_0 = 196$ ms, and (7.9) gives that tuning a phase margin of 36.8° with 50 sessions present and −40.3° with 20, because the plant gain varies as N^{-2}. The measurement matches. With the crowd present the queue peaks at 172.5 packets, swings 60.5 packets peak to peak in the first transient and holds within 0.3 packets of target from $t = 16$ s until the crowd leaves. After the crowd leaves the swing is 70.8 packets over the last 25 seconds, with 28.9 % of the run saturated and a marking travel of 3.60. The suggested tuning gives up the peak, letting the queue reach 214.8 packets, and returns a swing of 8.4 packets over the same last 25 seconds with a travel of 0.13 and no saturation at all, although the queue empties briefly when the crowd leaves and the link dips to 93.5 % utilization at $t = 32.3$ s. The footnote of Table 7.4 gives the verdict of the completion rule on both tunings. The loop was validated under load and commissioned onto an empty network, which is the general case whenever the plant gain falls as the disturbance grows.

DID YOU KNOW? The Failing Tuning Has the Smaller Error

From the simulator: in the challenge, the shipped tuning ends with an IAE of 1087 packet-seconds and fails the completion rule, while the suggested tuning ends at 1595 and passes. The shipped tuning moves the marking probability by 3.60 in all, the suggested one by 0.13 (Tables 7.4 and 7.5).

A controller gain scheduled on the session count, in the sense of Section 1.4.13, would cancel the N^{-2} factor of the plant gain. The session count is declared and not measured in this application, so such a schedule would need an estimate of it. Chapter 6 carries a gain schedule through on a plant whose gain moves far more.

Pause. Try Exercise 7.3 before reading on.

7.8.6 Effect of the Queue Setpoint on the Dead Time

Scenario 6 raises the target to 160 packets and then drops it to 40. The queue reaches 181.6 packets and settles at 160.0 with the round trip at 228 ms, then falls to a minimum of 26.6 packets and settles near 40 with the round trip at 132 ms. Sixty extra packets of queue on this link is 48 ms of extra delay on every packet of every session, bought for no throughput at all, because the link was already fully used at 100 packets. Going the other way costs headroom instead of latency. Raising the target to 250 packets, set one step after the scripted move to 160 at $t = 8\,\mathrm{s}$ and held until the scripted move to 40 at $t = 35\,\mathrm{s}$, puts the round trip at 300 ms. Over the second half of that interval the nominal network holds a sustained 40.5-packet swing with the marking probability on its lower limit 19 % of the time, close to the 40.4-packet swing scenario 4 produces at the same round trip. Nothing about the traffic or the tuning was touched. The setpoint of this loop sets the dead time of the loop.

7.8.7 Application Access

The application is used as Section 1.6.4 describes for every framework app. The scorecard reports integral absolute error, marking travel, the fraction of the run spent against a marking limit, the queue error, the mean link utilization, the seconds of buffer overflow and the current round-trip time, and those are the quantities reported in Table 7.5. The applied marking probability is displayed together with $p(t - R)$ and the window together with $W(t - R)$, so the dead time appears as a horizontal displacement between two curves of the same shape, and the 40-column export at 20 samples per simulated second carries both pairs so that a cross-correlation recovers the round-trip time offline.

7.9 Technology in Practice

The loop of Section 7.5 reads the queue q exactly and acts through the marking probability p with no lag of its own. Figure 7.5 draws it with the parts that a congested path carries in place of the schematic's symbols.

What Measures It

The measurement sits in the router or modem that owns the bottleneck queue, and both designs of this section measure a delay rather than a count. One estimates the queueing delay from the bytes waiting and the departure rate at each update, or from timestamps [16]. Another tracks the sojourn time, the time each packet spent in the queue, because it observes the delay every packet experiences and does not depend on the link rate [20]. The model of this chapter reads the instantaneous queue in packets at every execution, which Section 7.3.6 calls an ideal measurement. Its queueing delay is q/C in Equation (7.3).

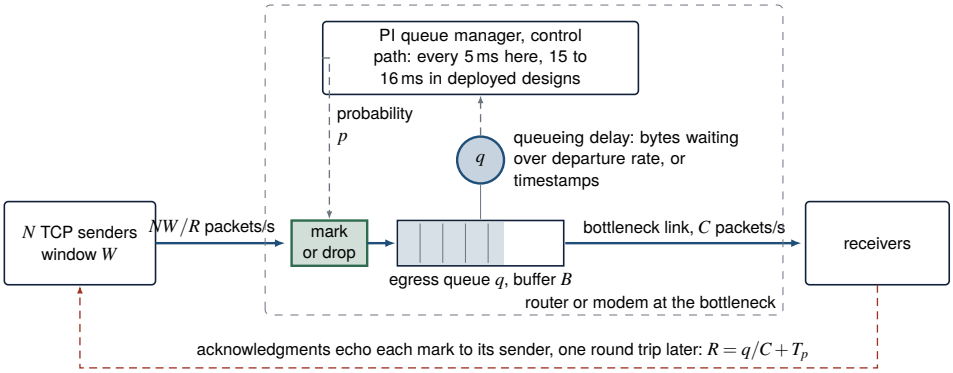


Figure 7.5 The queue loop with the parts of a real path in place of the schematic's symbols. The device that owns the bottleneck estimates its queueing delay, runs the PI law and marks or drops arriving packets, and each mark reaches its sender in an acknowledgment one round trip later. Notice that sensor, controller and final element sit in one device, so the whole dead time lies in the actuation path and none of it in the instrument.

> **TECHNOLOGY** **Sensor: Queueing Delay From a Rate or a Timestamp**
>
> A router measures its queue in one of two ways. It can divide the bytes waiting by an estimate of the departure rate, by Little's law, or it can stamp each packet with the time it joined the queue and read the delay as the packet leaves [16]. A timestamp yields a sample only when a packet reaches the head of the queue. The departure rate varies when other queues share the link or the link rate changes, which is common on wireless links, and short bursts that empty the queue make the rate estimate less accurate [16]. A queue length alone misleads: a drop in link rate shows on its trend as a spike that is not lasting congestion [20].

What Moves It

The final element is a decision taken for each arriving packet: mark it, drop it or let it pass. The marking probability p of Equation (7.1) stands for the share of arrivals that carry either signal, applied at once and with no slew limit, and its hard bounds of 0 and 1 are the actuator limits that scenarios 4 and 8 reach.

> **TECHNOLOGY** **Final Element: The Congestion Mark or the Drop**
>
> The router acts through two bits of the packet header. A sender that supports explicit congestion notification (ECN) sets a capable codepoint, and a congested router sets the congestion experienced codepoint instead of dropping the packet. The receiver echoes the mark in a flag of the TCP header, and the sender must respond as to a single dropped packet, halving its window once for any window of data that held a mark or a drop. A full queue leaves the router no choice but to drop [22]. A mark can be lost on the way: a broken receiver, or a device on the path, can hide it, so a router must still drop excessive traffic even from senders that claim to support marks [23].

Where the Controller Runs

The queue manager runs inside the network device. The cable modem variant of the PI design splits the algorithm into a control path, which updates the drop probability every 16 ms, and a data path, which makes the drop decision as each packet is queued. The data path may be built in hardware, and the control path is best kept in software so that it can be updated [17]. The general design updates every 15 ms by default against a delay target of 15 ms [16], and the controller of this chapter executes every 5 ms. That design also writes its law in increments: each update adds a term on the delay error and a term on the change of the delay since the last update to the drop probability, which is the

velocity form of Equation (7.14). It divides that increment by up to 2048 when the drop probability is small, a gain schedule on the level of congestion [16].

> **IN PRACTICE A Queue Manager Should Need No Tuning**
>
> The best current practice for queue management, RFC 7567 (BCP 197), asks that a queue manager need no tuning of its initial or configuration parameters in common use. Its default must auto-tune to reasonable performance, any manual parameters should be visible to the operator, and its response may have to be set for the largest expected path round trip, which cannot be known at initialization. Logging and alarms may compare input, output and mark or drop rates over time [23]. Scenario 7 of this chapter is the case that the round-trip clause warns about.

What Is Modernizing It

Three developments touch this loop. The first is marking in place of dropping: the Internet Engineering Task Force recommends that deployed queue managers support ECN as well as loss [23]. The second is the move from a target in packets to a target in time, because a queue length controls latency only indirectly [16]. The third is a dual queue that keeps traffic from scalable congestion controls apart from classic traffic and runs a PI controller whose output, squared, sets the drop probability of the classic traffic. In residential tests it held the average queueing delay of the low-latency queue below 1 ms [24].

7.10 Limitations

The simulator is not a network planning tool. It represents one bottleneck, one congestion control algorithm and one round-trip time shared by every session, so it carries no information about multiple congested hops, about the interaction of flows with different delays or about short flows, which are most of the flows on a real link. It has no slow start and no retransmission timeout, so it cannot reproduce the behavior of a path under heavy loss. The fluid approximation removes packets, and with them burstiness, jitter and every question about queue occupancy on a scale shorter than a round trip.

The AQM measurement is idealized in a way that favors the controller. Reading the instantaneous queue exactly, with no averaging and no sampling interval, gives the loop a measurement no deployed queue manager has, and the lag that a real averaging filter adds falls on a loop that has no margin to spare. The delay itself is resolved only to one integration step, which is 2.8 % of the round trip at the default and up to 25 % in the shortest-delay corner. The linear analysis of Section 7.4 holds the delay fixed at R_0 while the simulator lets it move with the queue, so the predicted critical round trip of 297 ms is an approximation to a state-dependent problem and sits 0.7 % above the measured upper bracket of 295 ms.

The controller is a fixed-gain PI loop with no gain scheduling, no delay compensation and no adaptation, which is deliberate but is also the reason scenario 7 fails. A Smith predictor removes the delay from the characteristic equation only to the accuracy of the delay it carries internally [3, 25]. Here that delay is $q/C + T_p$, so a predictor built around a constant loses accuracy exactly when the queue grows and the loop has least margin left. Its internal delay would have to be recomputed from the measured queue. Volume 5, Chapter 5 has a gauge loop that a process model turns into a Smith predictor around a transport delay that is known. A predictive controller of the kind Section 1.6.3 describes would carry the delay inside its model, with the $[0, 1]$ bound on the marking probability and the buffer limit as constraints, and this application does not offer one.

Four complications are retained rather than simplified away, the state-dependent round trip, the hard bounds on the marking probability, the window floor and the buffer limit, and each is retained because the controller has to be designed around it.

OPEN CHALLENGE **A Queue Manager That Tunes Itself**

The recommendations of 2015 call for research on ECN marking that needs no operational tuning, on the right use of delay-based methods, and on flash crowds of short flows that the network signals as if they were long ones [23]. In this chapter the plant gain moves with the session count and the dead time with the path. Progress would be a marking law with no site parameters that keeps its margin across both.

7.11 Discussion

Congestion control decides how a shared link is divided, and it decides the delay every user of that link experiences. The second consequence is the one that took longest to be recognized. A router with a large buffer and no queue manager will fill that buffer, and a full buffer converts unused capacity into latency at no cost in throughput, so a link can be simultaneously 100 % utilized, loss-free by the usual measurement and unusable for anything interactive [18]. Scenario 10 reduces that condition to two numbers, a round trip that goes from 180 to 420 ms while utilization stays at 100 %.

The remedy has been a control problem rather than a hardware problem for twenty-five years. The recommendation to deploy active queue management was made in 1998 and the obstacle named at the time was parameter selection [14]. The fluid model and its linearization turned that obstacle into a design calculation [6, 9], and the algorithms that followed target delay rather than occupancy precisely because a queue expressed in packets means different things on different links [19]. The same argument governs any loop closed over a network, including supervisory control hosted away from the plant and demand-response schemes that report to a central scheduler. Volume 5, Chapter 8 follows a force-reflecting loop closed through such a channel. What matters in those cases is not the average latency of the link but the phase it costs the loop at crossover.

7.12 Conclusion

This chapter studied a congested internet path as a plant whose dead time is set by one of its own two states. The simulator advances the fluid-flow window and queue equations with every marking term evaluated at $t - R$, where $R = q/C + T_p$ is computed from the state being regulated, using a 2048-entry interpolated history seeded with a constant initial function and a fourth-order step made legitimate by a 20 ms floor on the propagation delay.

Linearizing about the default operating point gives two real poles at 0.988 and 5.556/s behind one round trip of dead time with a DC gain of 7119 packets per unit marking probability, and the resulting characteristic equation places the phase margin at 67.6° at a 180 ms round trip and at zero at 297 ms. The simulator settles at 290 ms and sustains an oscillation of 37.4 packets peak to peak at 295 ms, with a measured period of 2.07 s against a predicted 2.09 s.

Around that comparison the application arranges ten scenarios and a graded challenge, including an operating point that does not exist, a correct tuning commissioned onto the wrong path and a drop-tail router that reaches 420 ms of round trip at full utilization. A fuller model would add a delay-compensating controller whose internal delay is recomputed from the measured queue, a spread of round-trip times across the session population and a second bottleneck. The exercises that follow measure what a lengthening delay costs a feedback loop.

KEY POINTS

- A mark made at the router reaches the senders only a full round trip later, so the queue loop carries that trip as dead time.
- At the default PI tuning the phase margin falls from 67.6° at a 180 ms round trip to zero at 297 ms, and the simulator stops settling between 290 and 295 ms.
- The setpoint of this loop sets the dead time of the loop.

Where else this applies. Volume 5, Chapter 8 closes a loop over a delayed channel.

References

[1] V. Jacobson, "Congestion avoidance and control," in *Symposium Proceedings on Communications Architectures and Protocols (SIGCOMM '88)*. ACM, 1988, pp. 314–329.

[2] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, no. 1, pp. 1–14, 1989.

[3] D. E. Seborg, T. F. Edgar, D. A. Mellichamp, and F. J. Doyle, III, *Process Dynamics and Control*, 4th ed. Hoboken, NJ: John Wiley & Sons, 2016.

[4] B. W. Bequette, *Process Control: Modeling, Design, and Simulation*. Upper Saddle River, NJ: Prentice Hall, 2003.

[5] K. J. Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*, 2nd ed. Princeton, NJ: Princeton University Press, 2021.

[6] V. Misra, W.-B. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '00)*. ACM, 2000, pp. 151–160.

[7] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong, "A control theoretic analysis of RED," in *Proceedings IEEE INFOCOM 2001*, vol. 3. IEEE, 2001, pp. 1510–1519.

[8] ——, "On designing improved controllers for AQM routers supporting TCP flows," in *Proceedings IEEE INFOCOM 2001*, vol. 3. IEEE, 2001, pp. 1726–1734.

[9] C. V. Hollot, V. Misra, D. Towsley, and W. Gong, "Analysis and design of controllers for AQM routers supporting TCP flows," *IEEE Transactions on Automatic Control*, vol. 47, no. 6, pp. 945–959, 2002.

[10] S. H. Low, F. Paganini, and J. C. Doyle, "Internet congestion control," *IEEE Control Systems Magazine*, vol. 22, no. 1, pp. 28–43, 2002.

[11] J.-P. Richard, "Time-delay systems: an overview of some recent advances and open problems," *Automatica*, vol. 39, no. 10, pp. 1667–1694, 2003.

[12] J. Nagle, "Congestion control in IP/TCP internetworks," RFC Editor, RFC 896, 1984.

[13] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[14] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on queue management and congestion avoidance in the internet," RFC Editor, RFC 2309, 1998.

[15] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, 1998.

[16] R. Pan, P. Natarajan, F. Baker, and G. White, "Proportional integral controller enhanced (PIE): A lightweight control scheme to address the bufferbloat problem," RFC Editor, RFC 8033, 2017.

[17] G. White and R. Pan, "Active queue management (AQM) based on proportional integral controller enhanced (PIE) for data-over-cable service interface specifications (DOCSIS) cable modems," RFC Editor, RFC 8034, 2017.

[18] J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the internet," *Queue*, vol. 9, no. 11, pp. 40–54, 2011.

[19] K. Nichols and V. Jacobson, "Controlling queue delay," *Queue*, vol. 10, no. 5, pp. 20–34, 2012.

[20] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled delay active queue management," RFC Editor, RFC 8289, 2018.

[21] K. J. Åström and T. Hägglund, *Advanced PID Control*. Research Triangle Park, NC: ISA – The Instrumentation, Systems, and Automation Society, 2006.

[22] K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to IP," RFC Editor, RFC 3168, 2001.

[23] F. Baker and G. Fairhurst, "IETF recommendations regarding active queue management," RFC Editor, RFC 7567 (BCP 197), 2015.

[24] K. De Schepper, B. Briscoe, and G. White, "Dual-queue coupled active queue management (AQM) for low latency, low loss, and scalable throughput (L4S)," RFC Editor, RFC 9332, 2023.

[25] O. J. M. Smith, "Closer control of loops with dead time," *Chemical Engineering Progress*, vol. 53, no. 5, pp. 217–219, 1957.

7.13 Exercises

7.1 Concept check · 2 min

The propagation delay of the default network lengthens until the round trip at the 100-packet target goes from 180 to 300 ms. The tuning, the traffic and the capacity stay as they were. Why does the phase margin fall from 67.6° to −1.3°?

(a) The dead time alone takes the phase, at a crossover frequency that stays at 2.09 rad/s

(b) The process gain alone rises, as the cube of the round trip, and the dead time takes the same phase as before

(c) The process gain falls with the longer round trip, so the integral term adds its lag nearer to crossover

(d) The dead time takes more phase, and the gain rising as the cube of the round trip moves crossover up, where it takes more still

7.2 Concept check · 1 min

The gain from the marking probability to the queue length is negative. The queue rises above its setpoint. What must the controller do, and what is that action called in the convention of Section 1.4.3?

(a) Raise the marking probability, which is direct action because the output rises with the measurement

(b) Raise the marking probability, which is reverse action because the process gain is negative

(c) Lower the marking probability, which is reverse action because the output falls as the measurement rises

(d) Lower the marking probability, which is direct action because the error $q_{SP} - q$ is negative

7.3 Concept check · 2 min

The tuning shipped with the challenge, $K_c = 15\,\%$ per 100 packets and $\tau_I = 1.2\,\text{s}$, has a phase margin of 36.8° with 50 sessions on the link and −40.3° with 20. What makes the lighter load the harder case?

(a) Fewer sessions lengthen the round trip at the same queue target, so the dead time takes more phase

(b) The process gain varies as N^{-2}, so it is 6.25 times higher with 20 sessions than with 50

(c) The bias p_0 is computed for 50 sessions, so the integral term winds up when the crowd leaves

(d) Each window is larger with 20 sessions, so the one-packet window floor limits the marking probability

7.4 Theory and calculation · 5 min

In scenario 3 the capacity of the link falls from 1250 to 600 packets per second. The 20 sessions, the propagation delay of 100 ms and the 100-packet target stay.

(a) Find the round trip, the window and the marking probability of the new operating point. *Checkpoint:* 267 ms, 8.00 packets and 3.03 %.

(b) Find the queueing delay of the 100-packet target before and after the cut. *Checkpoint:* 80 and 167 ms.

(c) Find the largest session count for which an operating point exists on the slower link at this target. *Checkpoint:* 160 sessions.

7.5 Theory and calculation · 8 min

The third step of scenario 4 puts the round trip at $R_0 = 300\,\text{ms}$, with $C = 1250$ packets per second, $N = 20$ and the default tuning, $K_c = 3.5 \times 10^{-4}$ per packet and $\tau_I = 5\,\text{s}$.

(a) Find the two poles and the steady gain $|P(0)|$ of the linearized plant, and the factor by which the gain has risen from the 7119 packets of the 180 ms round trip. *Checkpoint:* 0.356 and 3.33/s, 32 959 packets, a factor of 4.63.

(b) The loop gain crosses unity near 3.02 rad/s. Find the phase lag of the dead time at that frequency. *Checkpoint:* 51.9°.

(c) Add the phase of the PI controller and of the two poles at the same frequency and find the phase margin. *Checkpoint:* −1.2°, within 0.2°.

7.6 Theory and calculation · 5 min

With the marking probability at 100 % every window rests on its one-packet floor, so N sessions offer N/R packets per second.

(a) For the 500 sessions of scenario 8 on the default link, find the queue and the round trip at which the offered load equals the capacity, and the error that remains from the 100-packet target. *Checkpoint:* 375 packets, 400 ms and 275 packets.

(b) Back-calculation with $\tau_t = \tau_I$ holds the demand where $c_k - p_k = K_c e_k$. Find the pre-clamp command for $K_c = 3.5 \times 10^{-4}$ per packet. *Checkpoint:* 1.096.

(c) Find the session count at which this resting queue reaches the 400-packet buffer. *Checkpoint:* 525 sessions.

7.7 App challenge · 8 min

Load scenario 5, Sudden traffic arrival (flash crowd). Forty sessions join the 20 at 6 s. Run the scenario as shipped for 45 s at the default seed and export the file. The probability columns hold fractions, so 0.0299 is 2.99 %.

(a) Read `mark_probability` in the row at 6.20 s. Then find the first row after 6.00 s in which `mark_probability_delayed` has left 1.556 %. *Checkpoint:* 2.99 %, and the row at 6.25 s.

(b) Find the largest value of `queue_packets` and its time. *Checkpoint:* 226.8 packets at 7.20 s, within 0.5 packets.

(c) Read `mark_probability` and `window_packets` in the row at 45 s, and compare them with the operating point for 60 sessions. *Checkpoint:* 11.8 % and 3.88 packets, against 12.45 % and 3.75 packets.

Explain the delay found in part (a) from the queue in the same rows.

7.8 App challenge · 9 min

Load scenario 4, Propagation delay, step by step. Run it as shipped for 100 s at the default seed and export the file.

(a) Find the peak-to-peak swing of `queue_packets` over 41 to 52 s, 63 to 74 s and 87 to 100 s. *Checkpoint:* 2.0, 40.4 and 54.4 packets, each within 0.5 packets.

(b) Find the share of the rows in which `mark_saturated` is 1 over the last two of those windows. *Checkpoint:* 22 and 42 %, each within one point.

(c) Measure the period between 63 and 74 s from successive peaks of the queue. *Checkpoint:* 2.10 s, within 0.05 s.

Name the step at which the loop stops settling, and compare it with the margins of Table 7.6 and the critical round trip.

7.9 App challenge · 8 min

Load the Challenge, ride out a flash crowd. Thirty sessions join at 8 s and leave at 32 s, and the target is 120 packets. Run each case for 75 s at the default seed and export the file.

(a) Run the case as shipped, with the Proportional gain at 15 and the Integral time at 1.2 s. Find the highest queue between 8 and 32 s, and the lowest and the highest over the last 25 s. *Checkpoint:* 172.5 packets, then 74.1 and 144.9 packets.

(b) Set the Proportional gain to 3.5 and the Integral time to 5.0 s, press Reset and run again. Find the same three values. *Checkpoint:* 214.7 packets, then 111.5 and 119.9 packets.

(c) In the file of part (a), read `loop_dc_gain_packets` in the rows at 5 and 30 s. *Checkpoint:* 9191 and 1471 packets, a ratio of 6.25.

Accept each queue value within 0.5 packets. Say which run holds the ±25-packet band over the last third of the run, and why the shipped tuning fails only after the crowd has left.

7.10 Application area · 3 min

A queue target is a latency budget written in packets.

(a) Find the queueing delay of a 100-packet target on the default link of 1250 packets per second and on a link of 600 packets per second. *Checkpoint:* 80 and 167 ms.

(b) An operator specifies 5 ms of queueing delay instead. Find that target in packets on each link, and say what it does to the dead time of the loop when the capacity changes. *Checkpoint:* 6.25 and 3 packets.

7.11 Application area · 4 min

The model hands the controller an exact and instantaneous queue. Suppose a deployed queue manager averages the queue with a first-order filter of time constant 0.1 s and updates the marking probability every 0.1 s, under the default tuning.

(a) Estimate the phase that the filter and the update period take at the default crossover of 2.09 rad/s. Treat the update period as half a period of added dead time (Section 1.4.11). Find what is left of the 67.6° margin. *Checkpoint:* 11.8° and 6.0°, which leave about 50°.

(b) Decide whether the same filter and period can be used at the 260 ms round trip, where the margin is 16.9°. *Checkpoint:* No. They take at least 17.8° there.

7.12 Review the agent · 5 min

An engineer asks an AI assistant how to give the default router more room for a flash crowd. The assistant answers: “Raise the queue setpoint from 100 to 250 packets. The buffer holds 400 packets, so 150 packets of headroom remain. The setpoint is the target of the loop and not part of the plant, so the default tuning keeps its phase margin of 67.6°, and the only cost is a fuller buffer.” Find the flaw. Say how a calculation and one run of the app would each expose it.