The Dynamics and Control of Internet Attacks

James G. Garnett
Liz Bradley

University of Colorado
Department of Computer Science

(JGG now at Secure64)

Internet fundamentals, part I

- Design assumes that users are good citizens and that hosts don’t move around
- No screening, address verification, …
- Source of many current woes

"Malware"

- popups
- spam
- worms, viruses
- botnets
- spoofing
- sniffers
- direct attacks
- denial-of-service (DoS) attacks
- …

Solutions

- popups: good browser design & hygiene
- spam: spam filters
- worms, viruses: anti-virus software
- botnets: anti-virus software
- spoofing: authentication
- sniffers: cryptography, anti-virus software
- direct attacks: firewalls
- ***denial-of-service (DoS) attacks: this talk***

Internet fundamentals, part II:

- Design assumes that data can get lost
- So *retransmission* is built into its protocols
- Which means that it's OK to drop resource requests

- The trick is to drop as few of them as possible to keep the resource unclogged.

Internet fundamentals, part III:

- The “black hats” observe the defenses and adapt
- Rapid *co-evolution*
- So any kind of static response won’t work

- Have to respond adaptively…

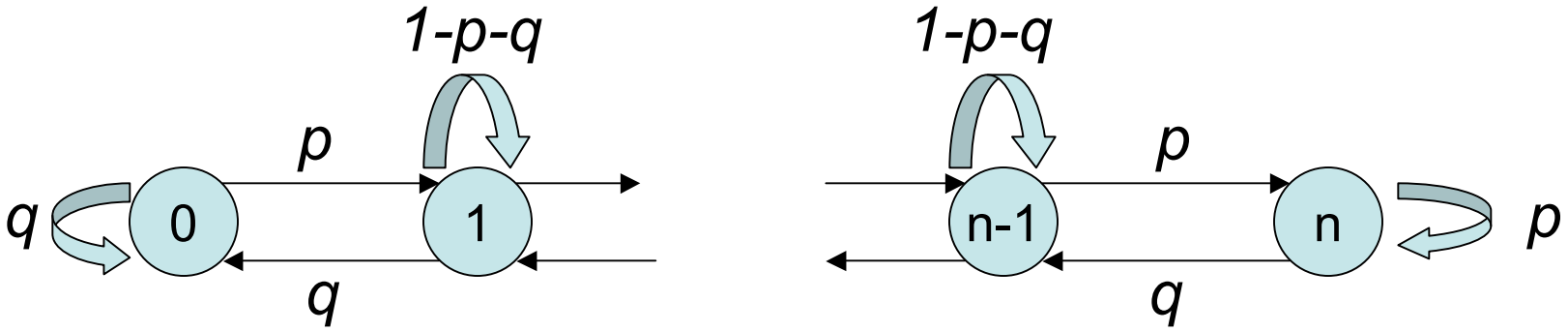
- Build an adaptive stochastic model of resource usage

- Use a nonlinear model-reference PID controller to screen resource requests

What computer systems typically do to handle overload:

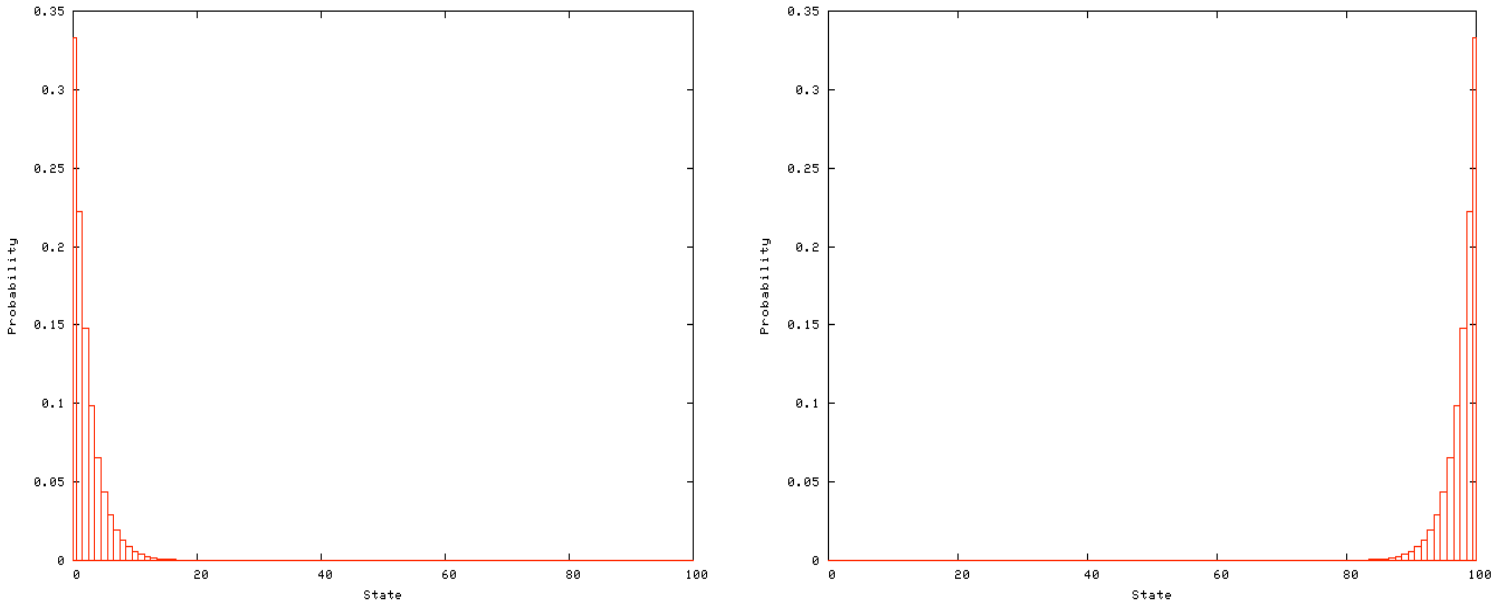
- Set hard limits (e.g., drop-tail queue mgmt)
- Control *average* demand
- Use *ad hoc* linear proportional closed-loop controllers (at best)

The model: Birth/Death Markov chain



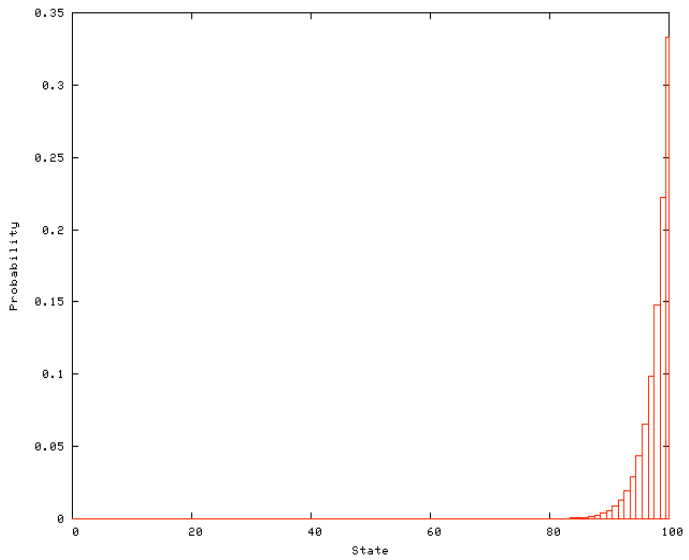
- Well known, widely used, and broadly applicable
- State ranges from *0* to *n*
- Edges denote possible state transitions
- Edges are annotated with transition probabilities

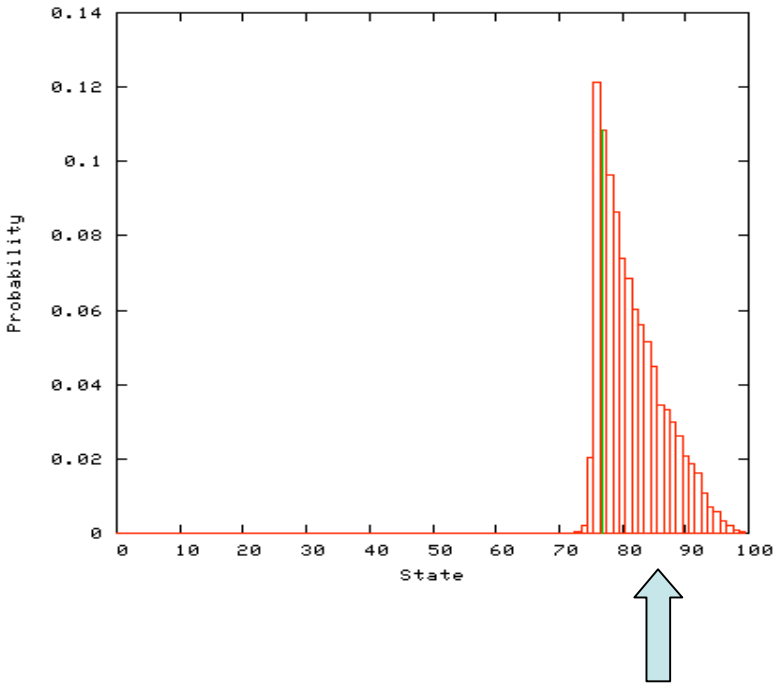
Stationary distributions of the BD chain:



Key point: can calculate the distribution shape from p and q

What if you wanted a different distribution?



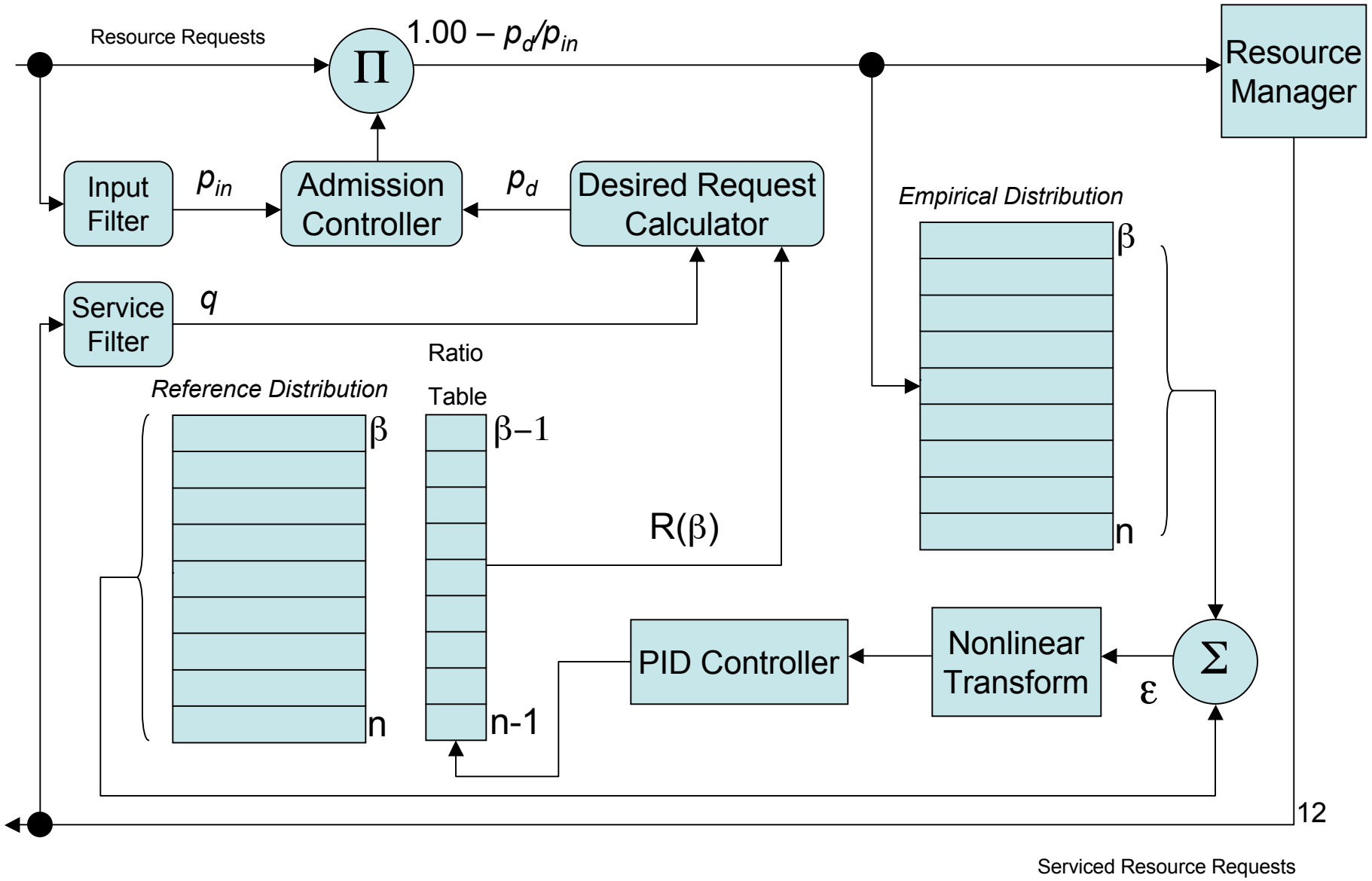


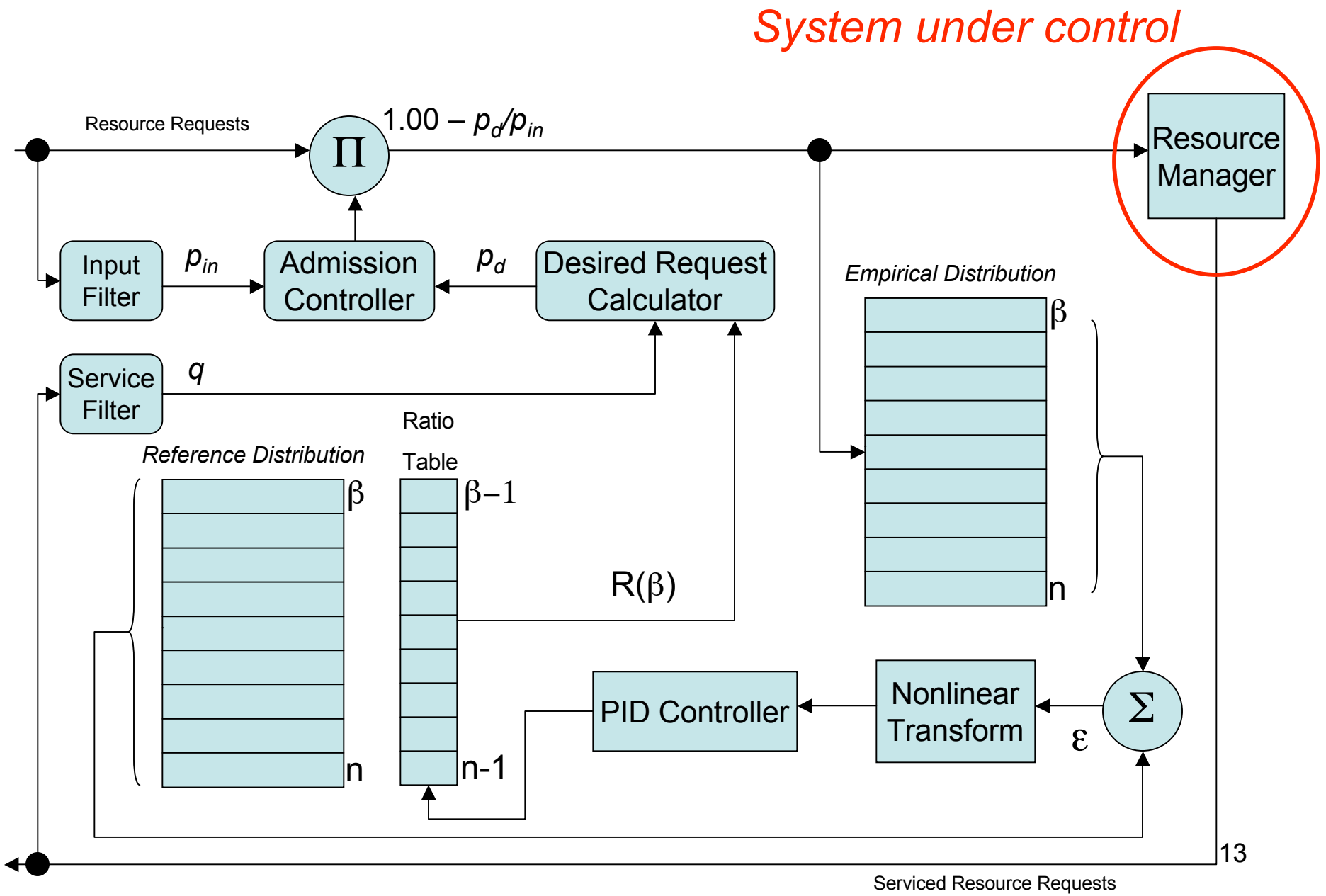
Key point: can calculate what p and q would give rise to this shape

Control strategy:

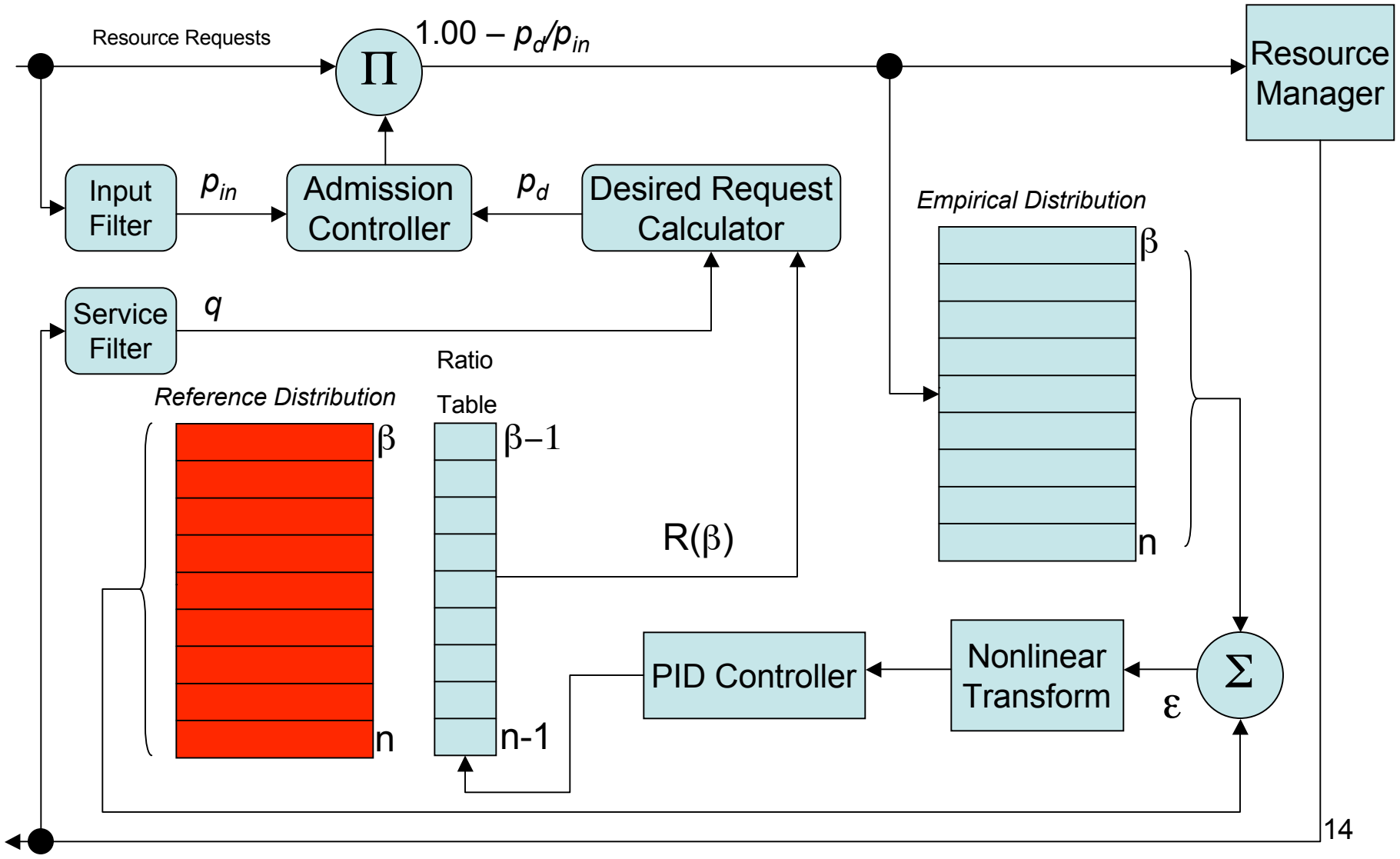
- Calculate desired p, q
- Estimate *actual* p, q
- Gatekeep on the difference

Controller architecture:

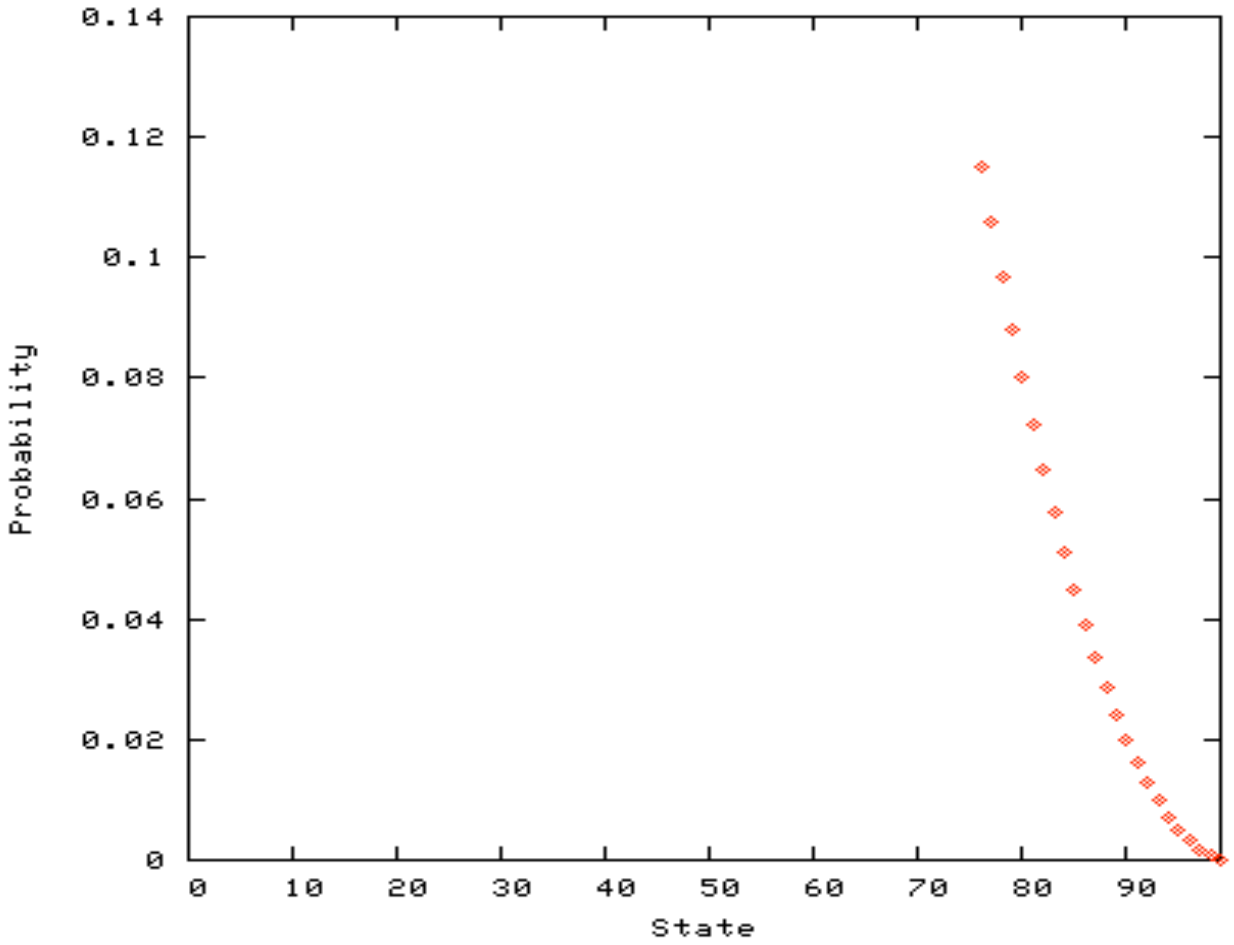



System under control
Resource Requests
Π
1.00 – p_d/p_{in}
Resource Manager
Input Filter
p_{in}
Admission Controller
p_d
Desired Request Calculator
Empirical Distribution
β
n
Service Filter
q
Ratio Table
Reference Distribution
β
β–1
n
n-1
R(β)
PID Controller
Nonlinear Transform
Σ
ε
13
Serviced Resource Requests

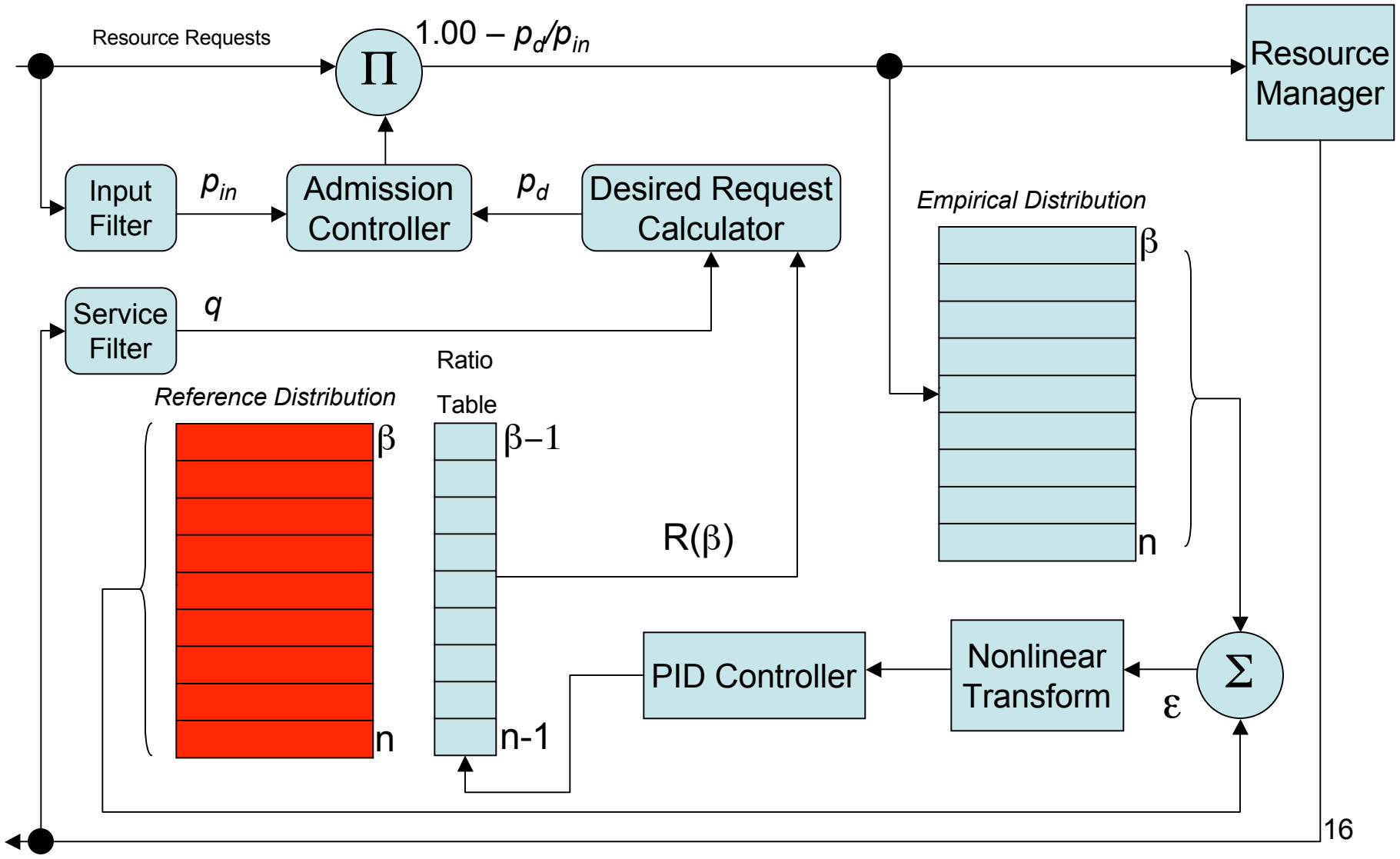
Reference distribution: Q(i)



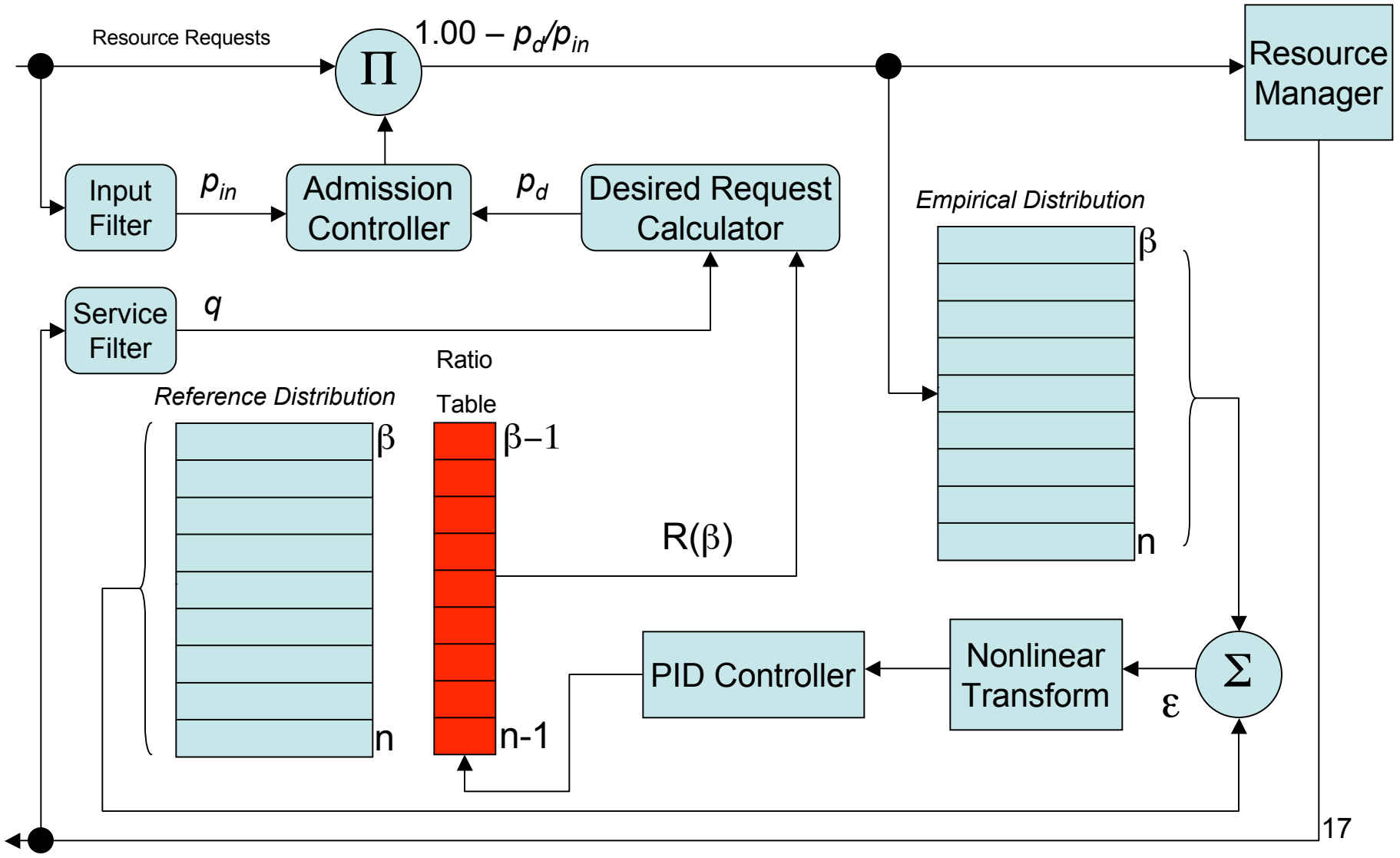
Q(i): The control goal specification



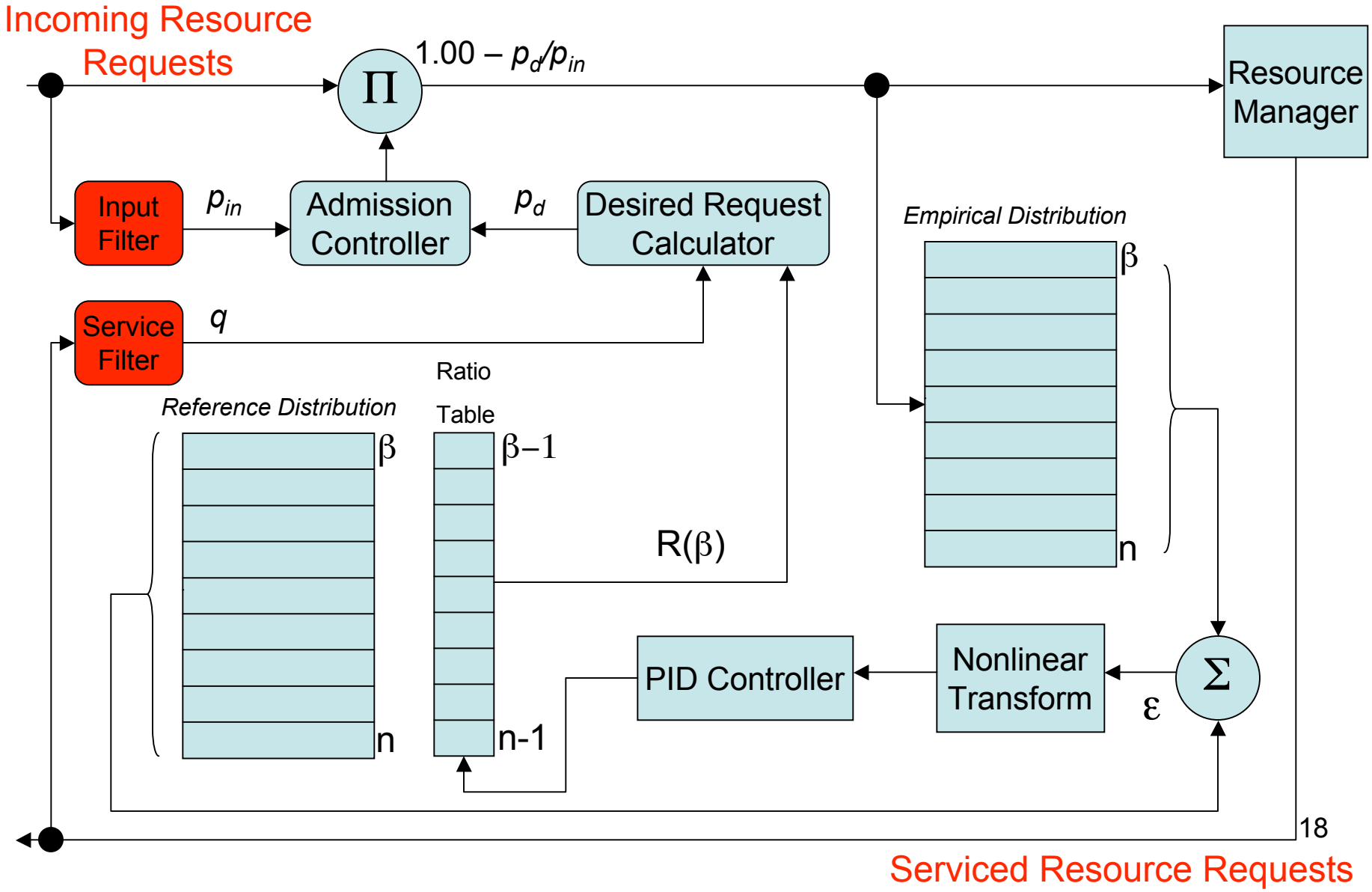
Reference distribution: Q(i)



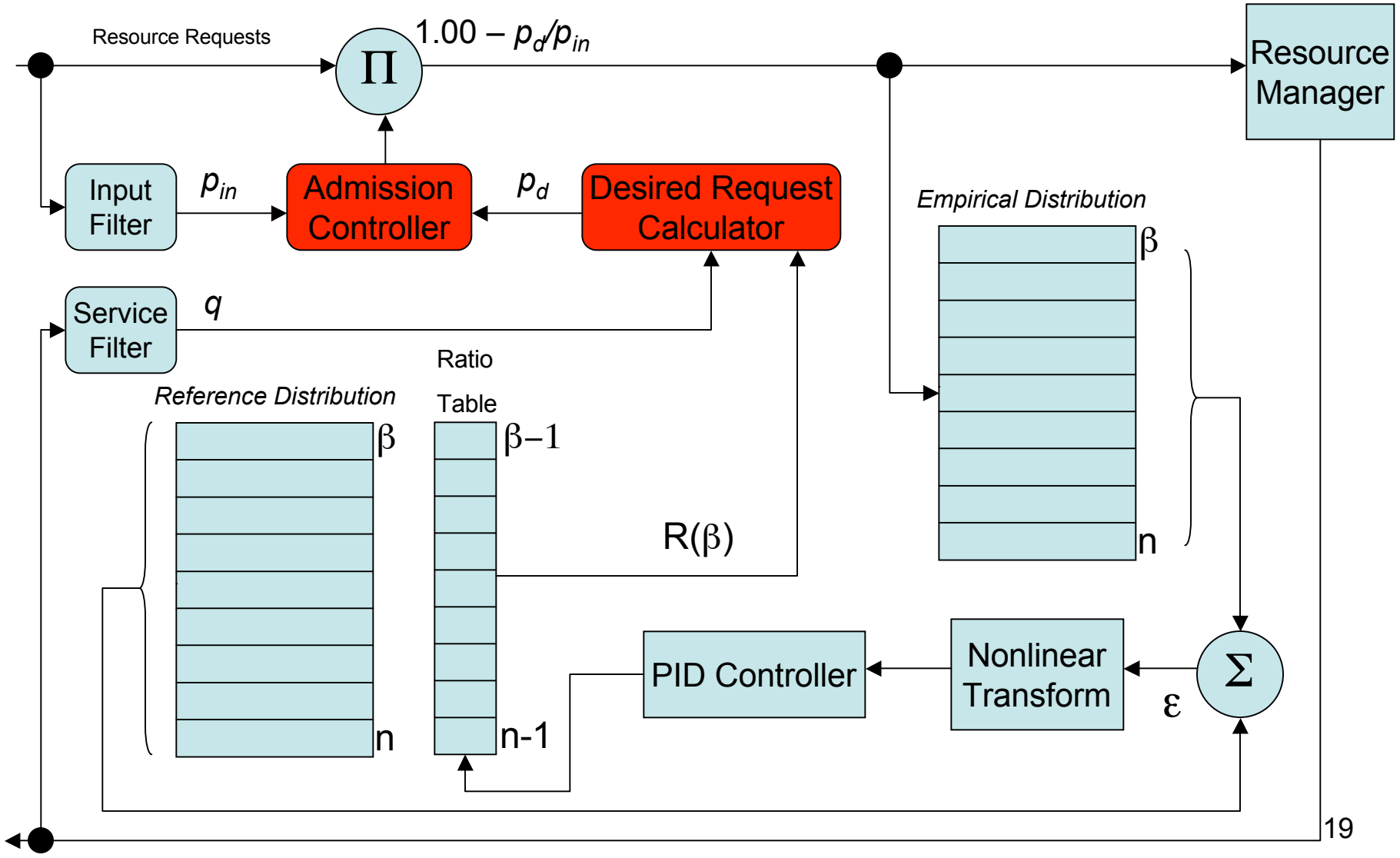
Calculate transition ratios: Q(i+1)/Q(i)



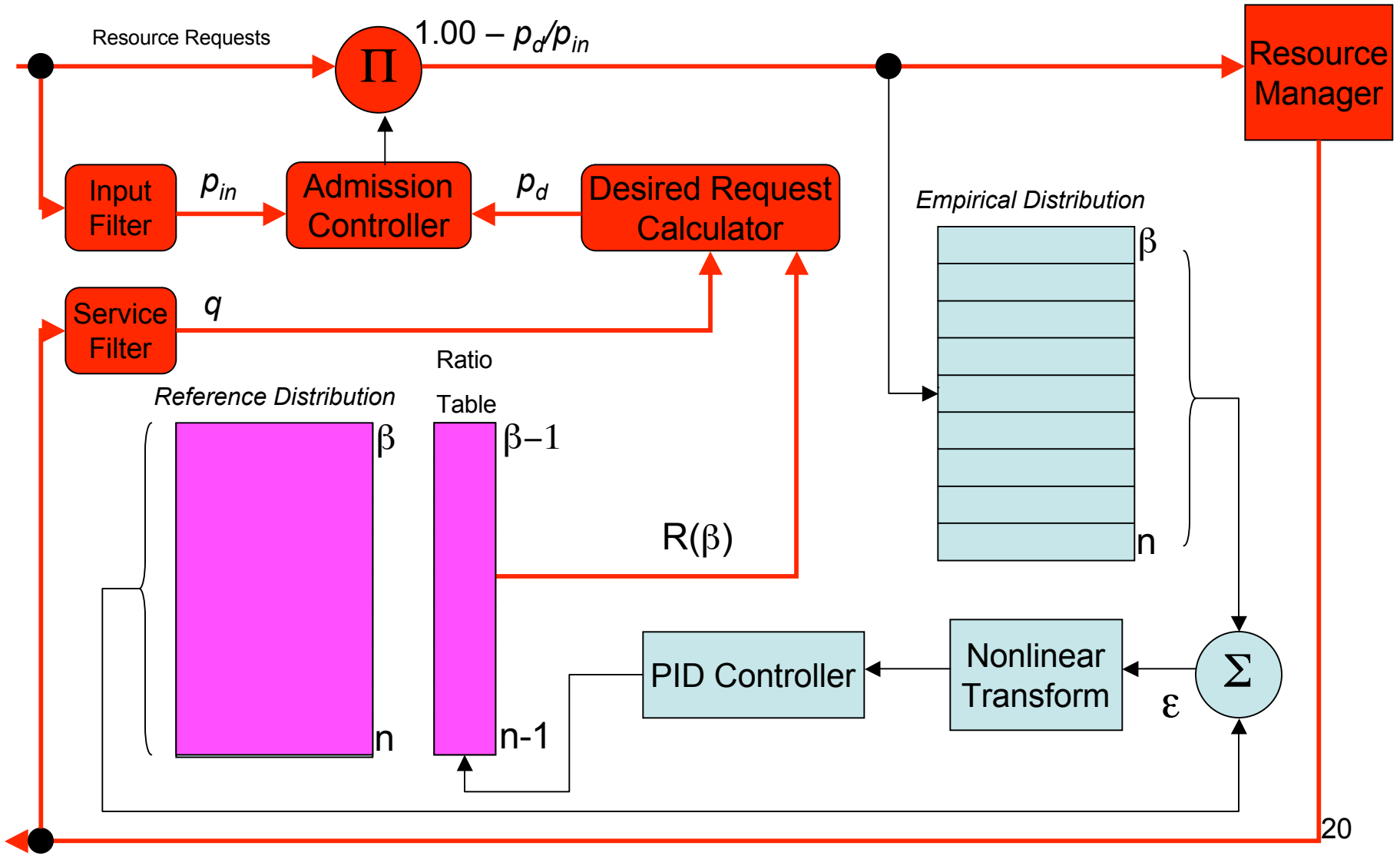
Estimate transition probabilities:



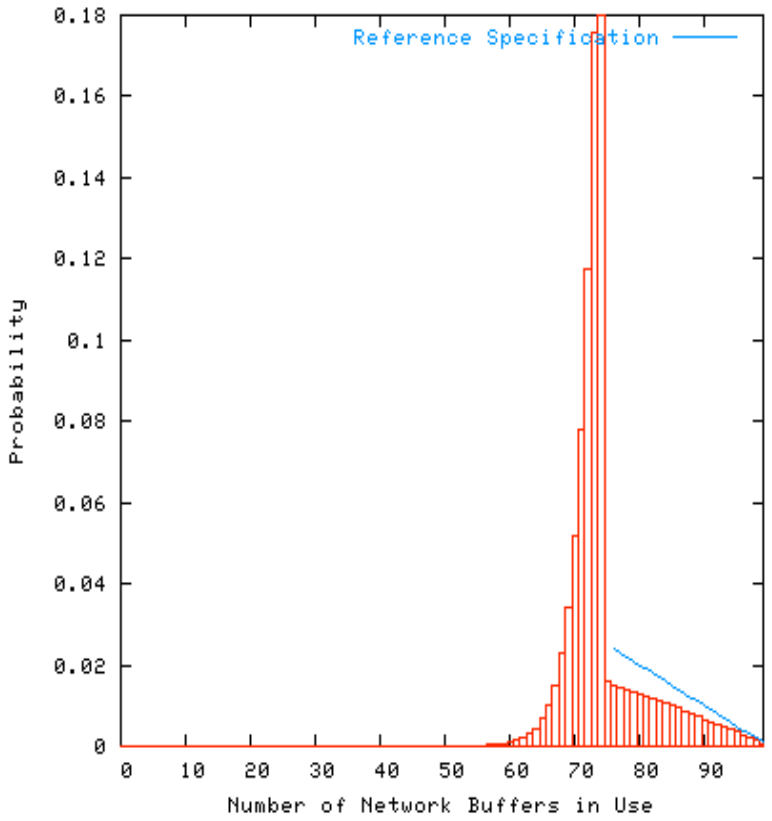
Calculate desired p_d and drop resource requests accordingly:

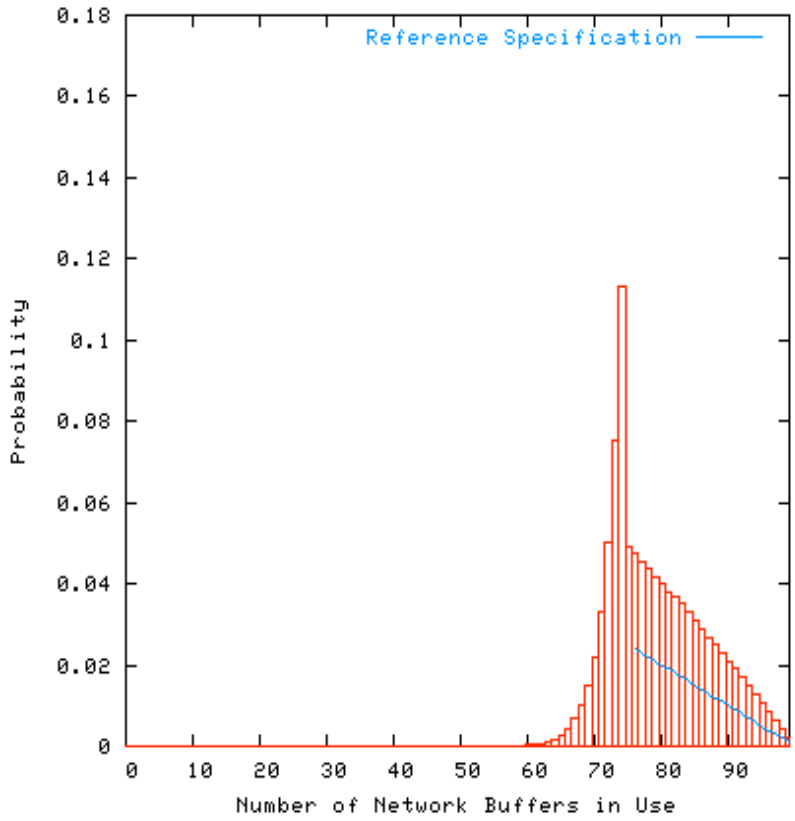


Model-reference feedback control loop:



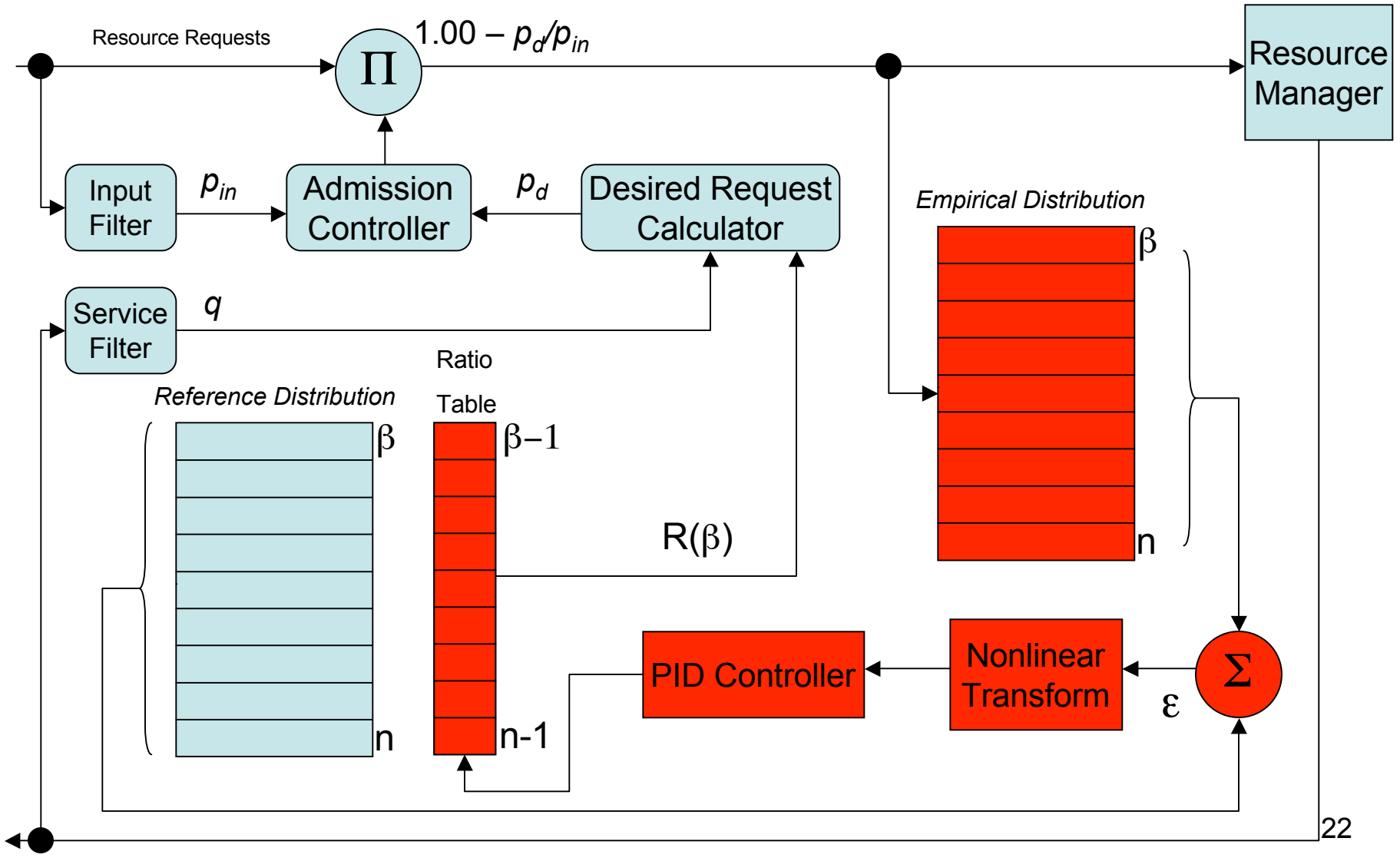
What if R(β-1) is incorrect?



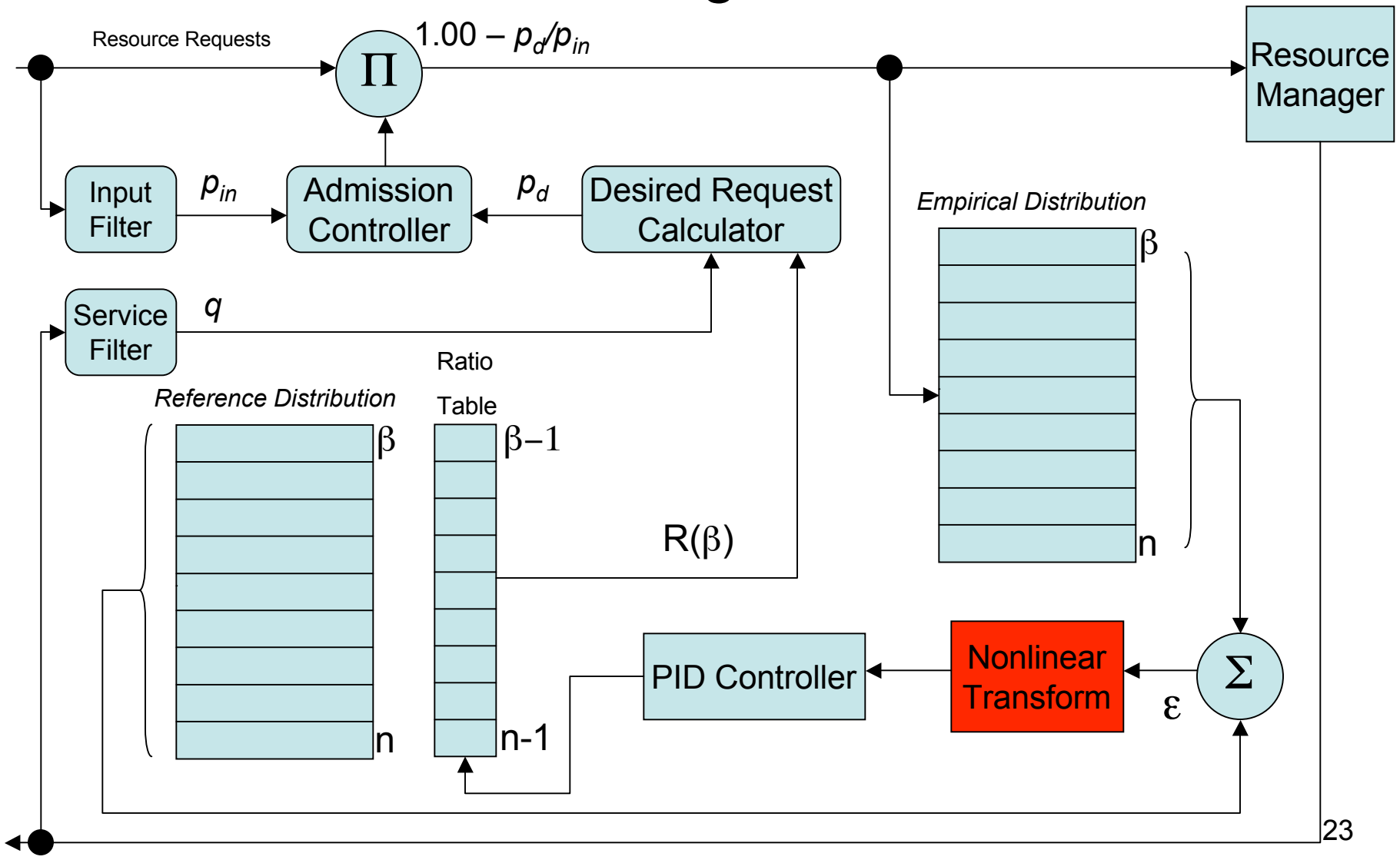


QoS spec

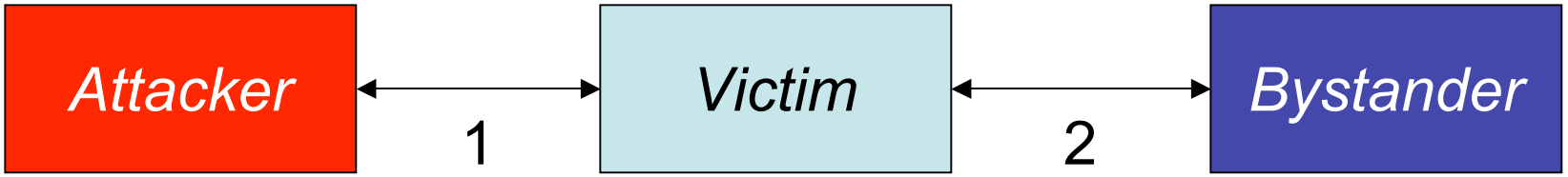
That second feedback loop adjusts it:



Nonlinear transform accelerates convergence:

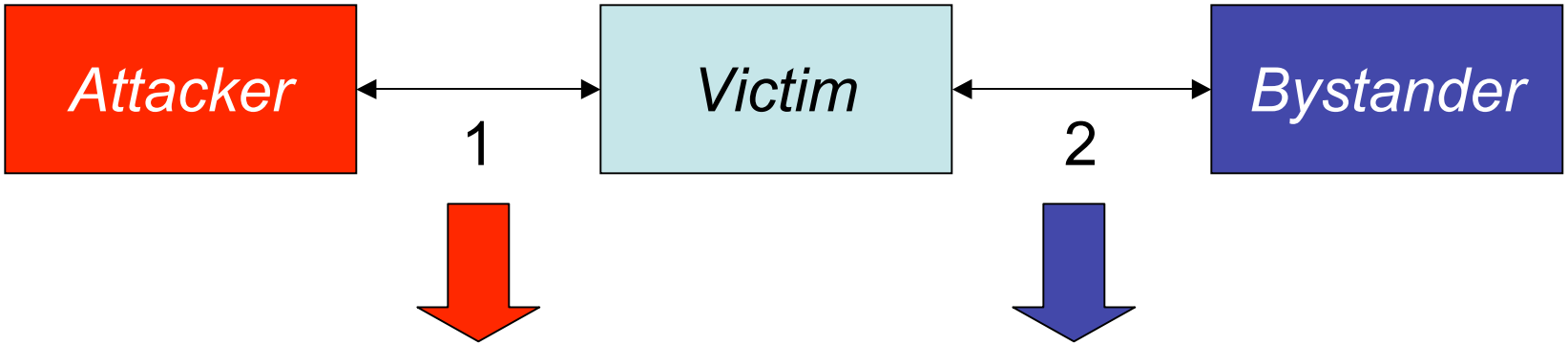


Denial of Service (DoS) example:



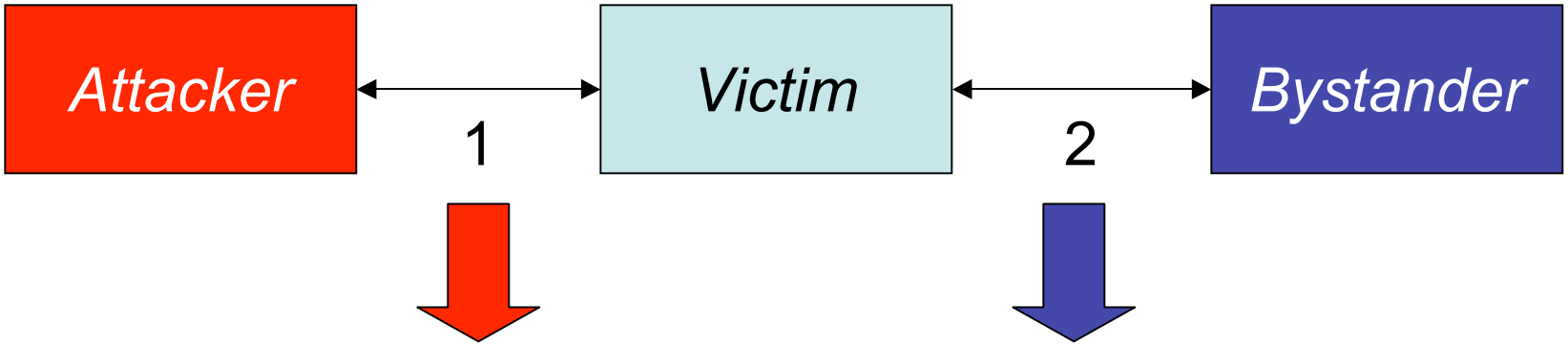
- identical unix machines
- 10 Mb/sec networks
- NB: single s/w manager in victim handles *all* incoming traffic

Without control:



96.9% packet loss *97.0% packet loss*

With control:



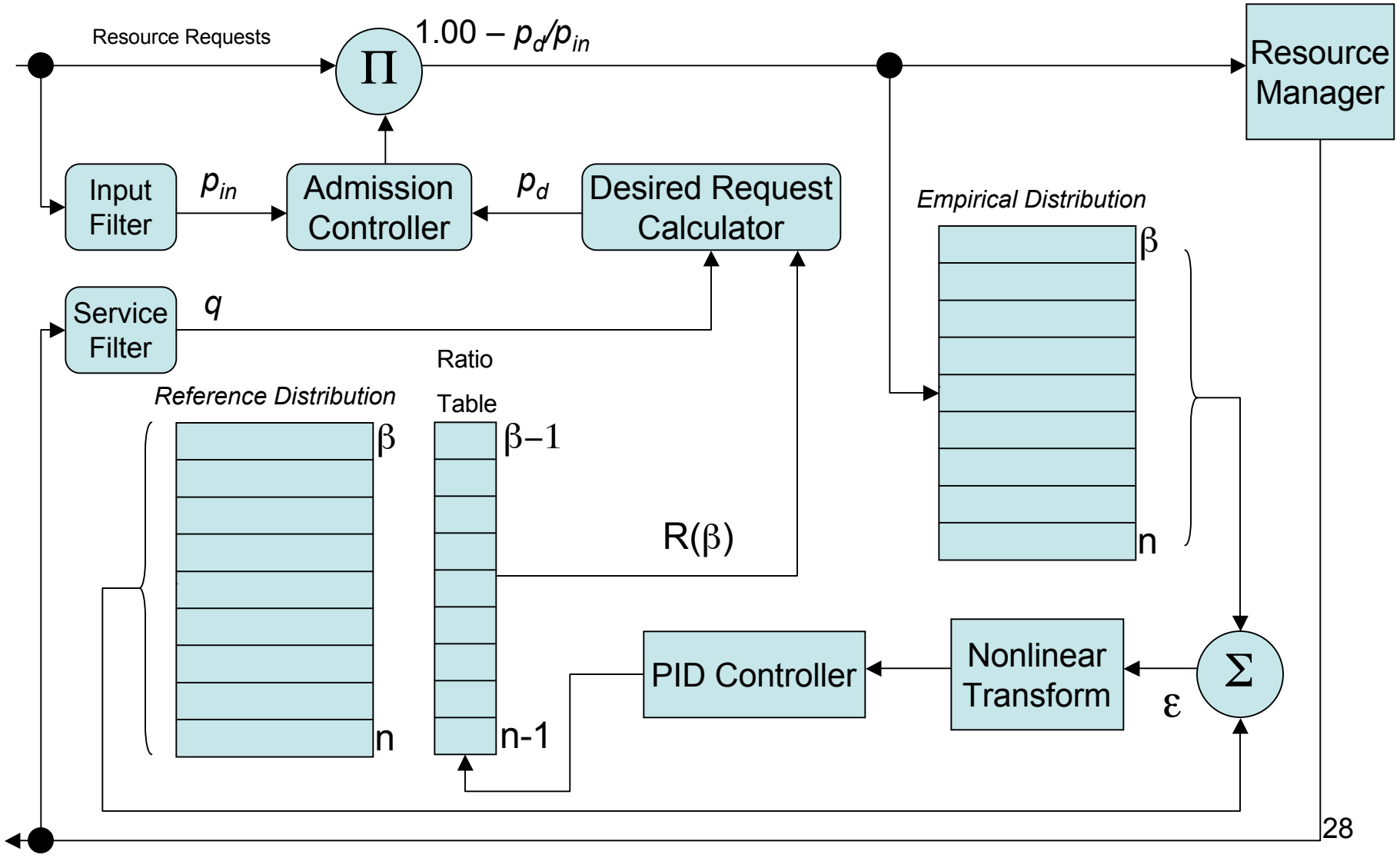
93.4% loss

0.0% loss

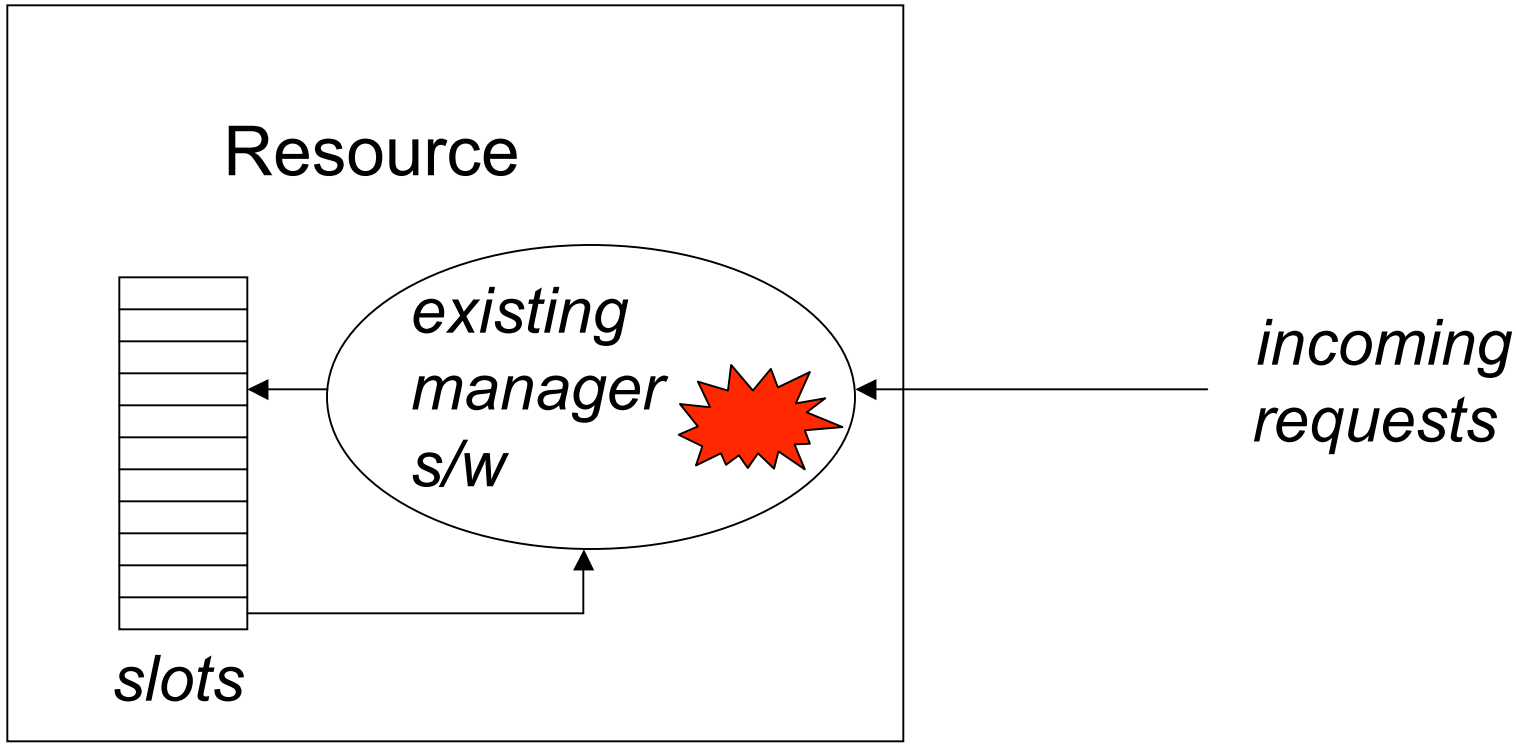
Results:

- It works.
- It converges fairly quickly (1-3 sec in our tests).
- It's lightweight:
  - Small amount of code (~100 lines of C)
  - Low computational and memory overhead
    - |Q| subtracts are primary computational load; runs in μsec
    - 128 bytes per controller for state information
  - Advantages of RED, without RED's disadvantages (this is the IETF's standard for congestion control)

Half a dozen equations, really…



How you implement this:



Conclusions:

- It works.
- It converges fairly quickly (1-3 sec in our tests).
- It's lightweight:
  - Small amount of code (~100 lines of C)
  - Low computational and memory overhead
    - |Q| subtracts are primary computational load; runs in μsec
    - 128 bytes per controller for state information
  - Advantages of RED, without RED's disadvantages
- It's broadly applicable (any system that can be modeled by a G/G/1 queue)
- And it has been already been deployed in practice…

Commercialization…

- Patent filing (6/26/2004)
- Secure64 Wildfire/CE^2 (12/1/2004)
- And then shot down.

JGG's thesis proposal was circulated to other students by a committee member, which constituted "prior disclosure" and kills a patent. (You have one year from the first disclosure to file it.)

Moral: be careful with your ideas if you're thinking of patenting them — keep dated, initialed notebooks, don't share ideas until you're ready to patent, etc.

www.cs.colorado.edu/~lizb/papers/dos.html

On the stove:

Nonlinear dynamics

- Modeling & control of internet attacks
- Nonlinear time-series analysis of computer systems
- MEMS-based flow control in jets
- Recurrence plots
- Computational topology & topology-based filters

Artificial intelligence

- Nonlinear system identification
- Radioisotope dating
- Movement patterns
- Clear-air turbulence forecasting

www.cs.colorado.edu/~lizb

Collaborators

•*graduate students:*

Jenny Abernethy, Matt Easley, James Garnett, John Giardino, Kenny Gruchalla, Joe Iwanski, Zhichun Ma, Ricardo Mantilla, Todd Mytkowicz, Laura Rassbach, Vanessa Robins, Natalie Ross, Reinhard Stolle

•*postdocs:*

Tom Peacock (now at MIT)

•*undergrads:*

Ellenor Brown, Nate Farrell, Jesse Negretti, John Nord, Alex Renger, Roscoe Schenk, Stephen Schroeder, Evan Sheehan, Josh Stuart (now at UCSC)

•*faculty:*

— Jessica Hodgins, Computer Science, CMU

— David Capps, Theater & Dance, Hunter College

— Jean Hertzberg & YC Lee, Mechanical Engineering, CU

— Amer Diwan, Computer Science, CU

Related work in computer systems lit:

- Software Control
  - Floyd et al. (RED [2001])
  - Hellerstein et al. (servers [1999 – 2003])
  - Stankovic (realtime scheduling [1999])
- Markov Chain Monte Carlo
  - Sinclair & Jerrum (Conductance [1989])
  - Morris & Peres (Evolving Sets [2003])
- DoS Mitigation
  - Mirkovic (D-WARD [2002])

None uses adaptive nonlinear closed-loop control, though Karmanolis (HotOS 2005) moves in that direction

What's different here, from the standpoint of that community:

Control (shape) the *distribution* of resource states, rather than just the *average* of that distribution or the instantaneous state

Do this with *adaptive nonlinear PID* control

- adaptive: using Markov-chain model and parameter estimation
- nonlinear: to overcome quasistability effects and improve performance
- PID: to allow wider range of modern controls techniques