Dynamic Overload Balancing in Server Farms

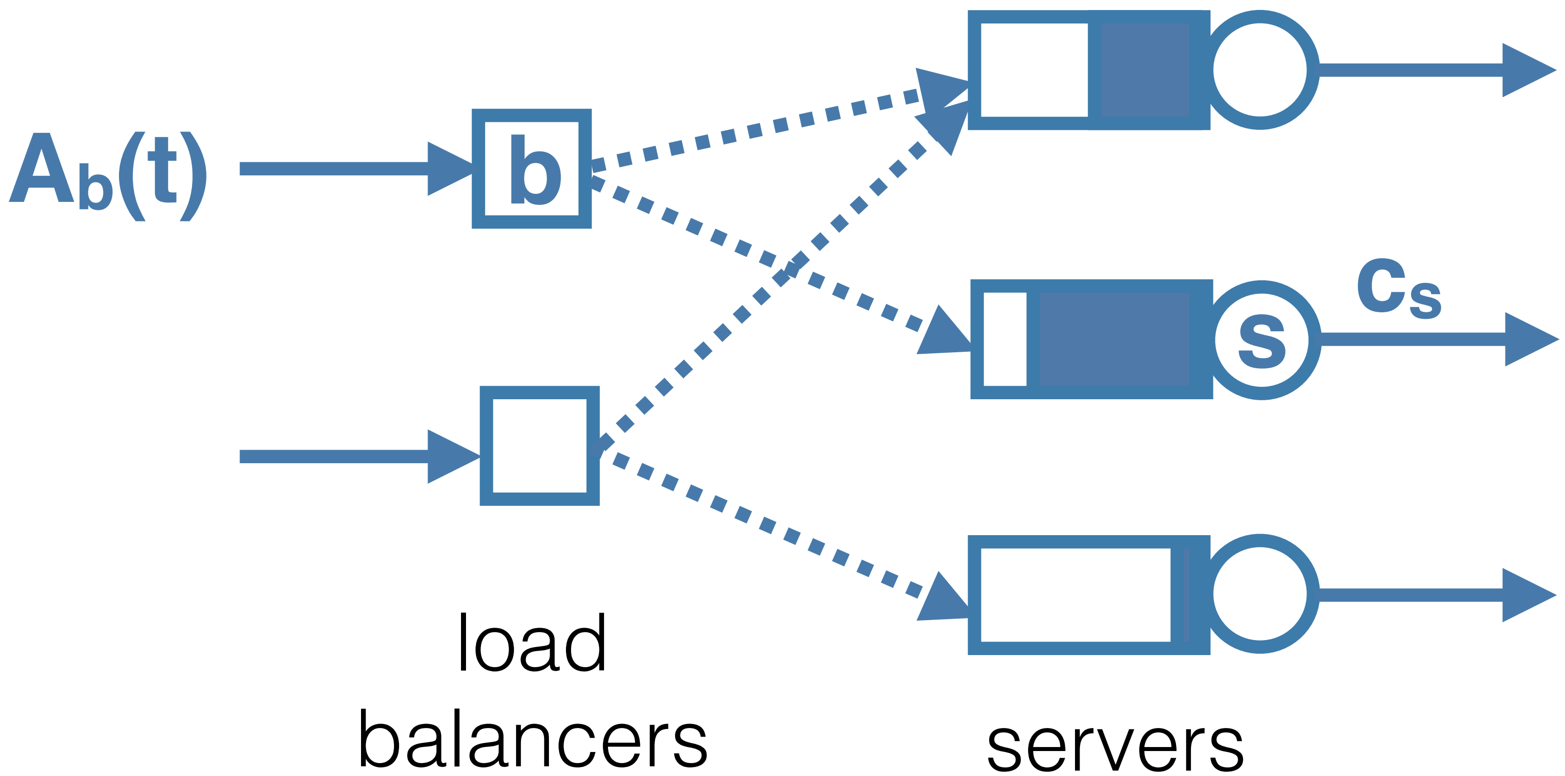
Chih-ping Li (MIT)
Georgios S. Paschos (MIT, CERTH/Greece)
Leandros Tassiulas (University of Thessaly, Greece)
Eytan Modiano (MIT)

IFIP Networking 2014

Overload in network systems

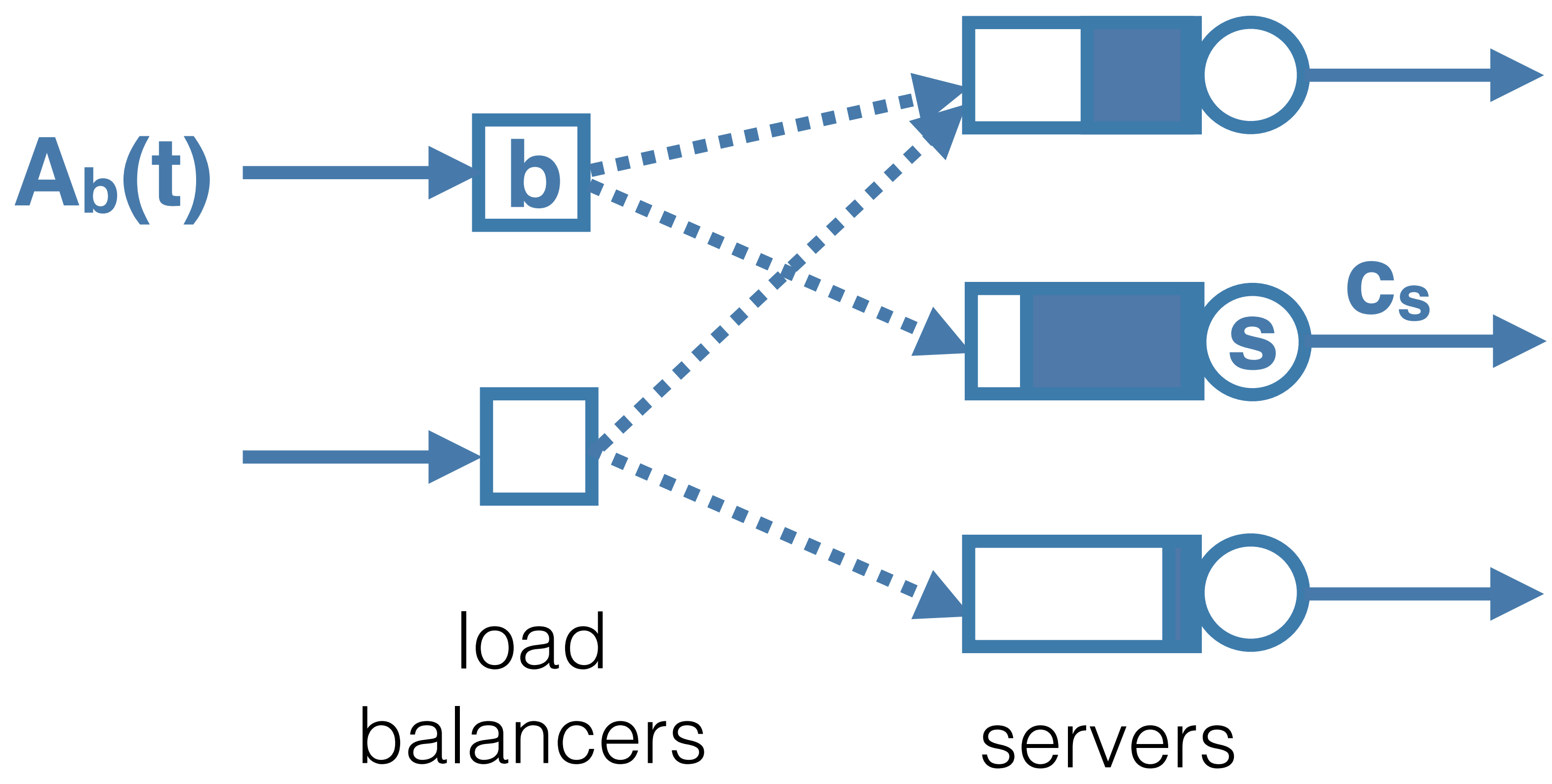
- Overload = traffic demand > system capacity
- Due to unpredictable demand fluctuation, flash crowd, DDoS attacks, node/link failures, natural disaster, power outage, etc
- Throughput loss and increasing delay
- Need to optimize throughput and overload surges

System model



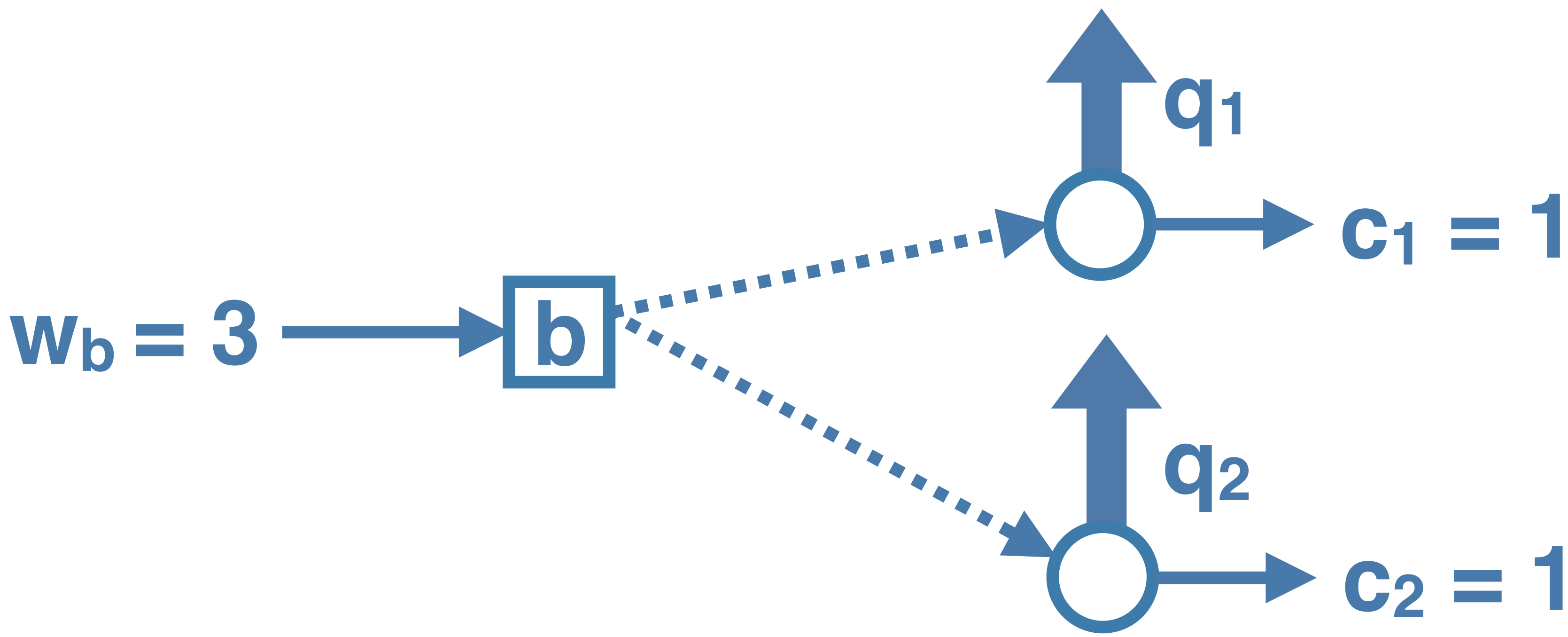
- $A_b(t)$ = # of jobs arriving at load balancer b in slot t
- $A_b(t)$ ~ i.i.d. over slots with mean λ_b
- Each job has an i.i.d. random size X, which is unknown
- c_s = service rate of server s (X/c_s is the job service time)
- $w_b = \lambda_b E[X]$ = workload arrival rate

System model

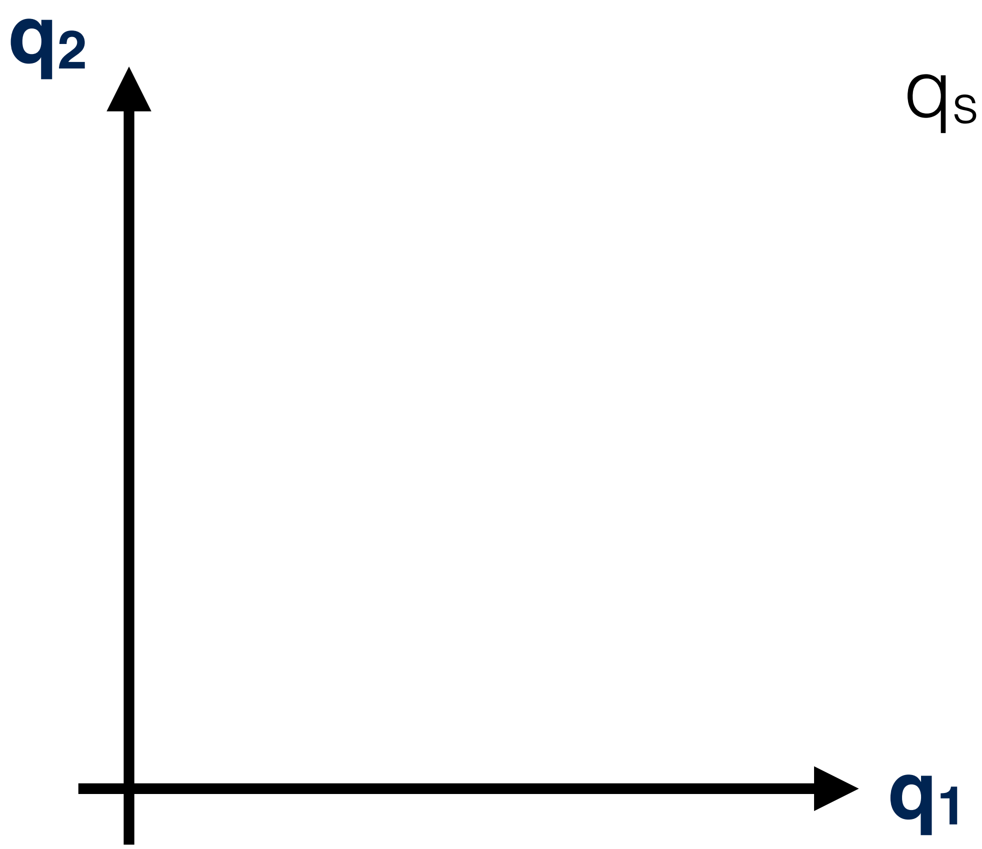


- Workload arrival rate > total system capacity
- Achievable performance?
- Control policy to maximize throughput and enforce queue overflow rates to have “good” properties?

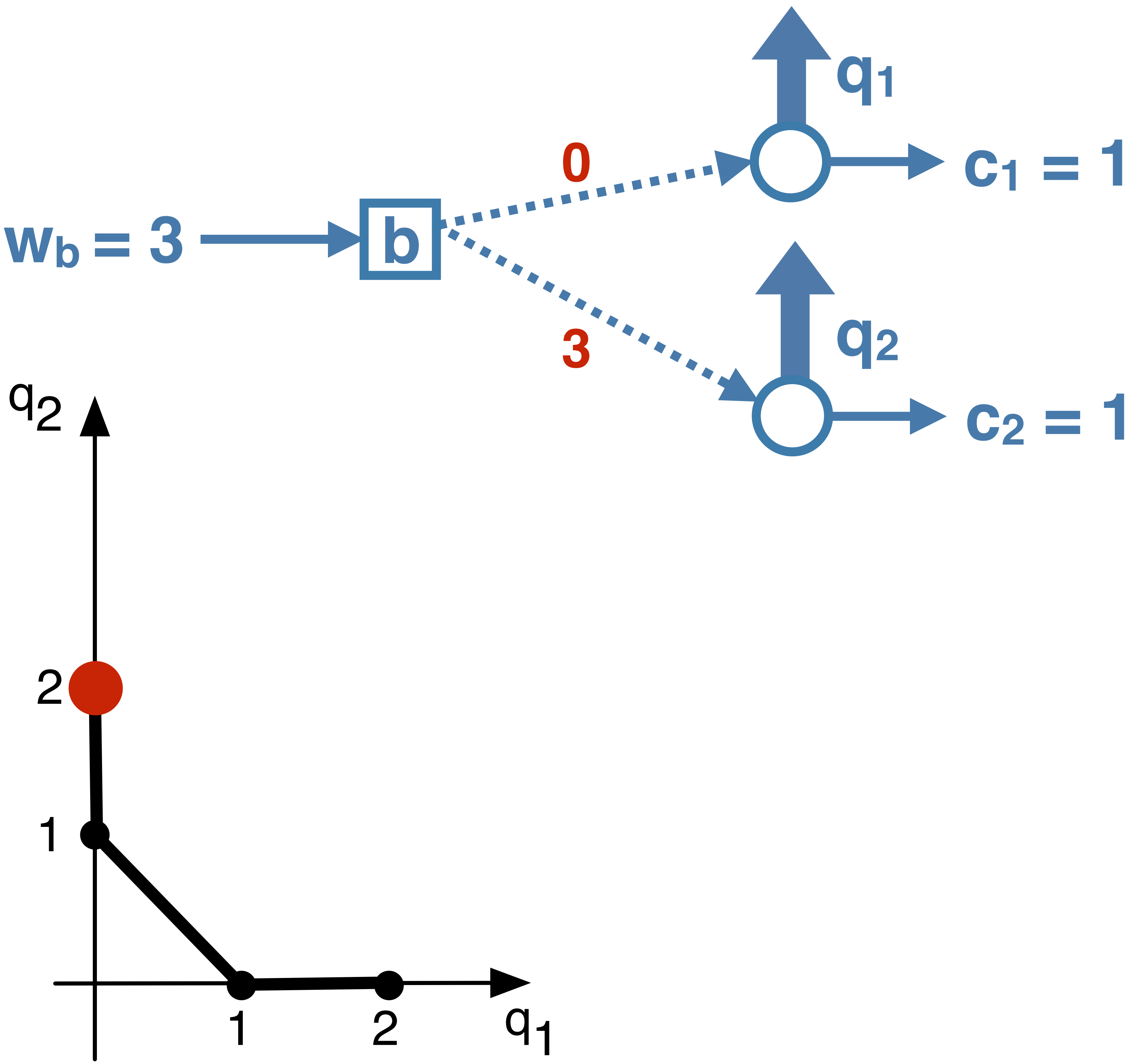
Fluid-level achievable performance



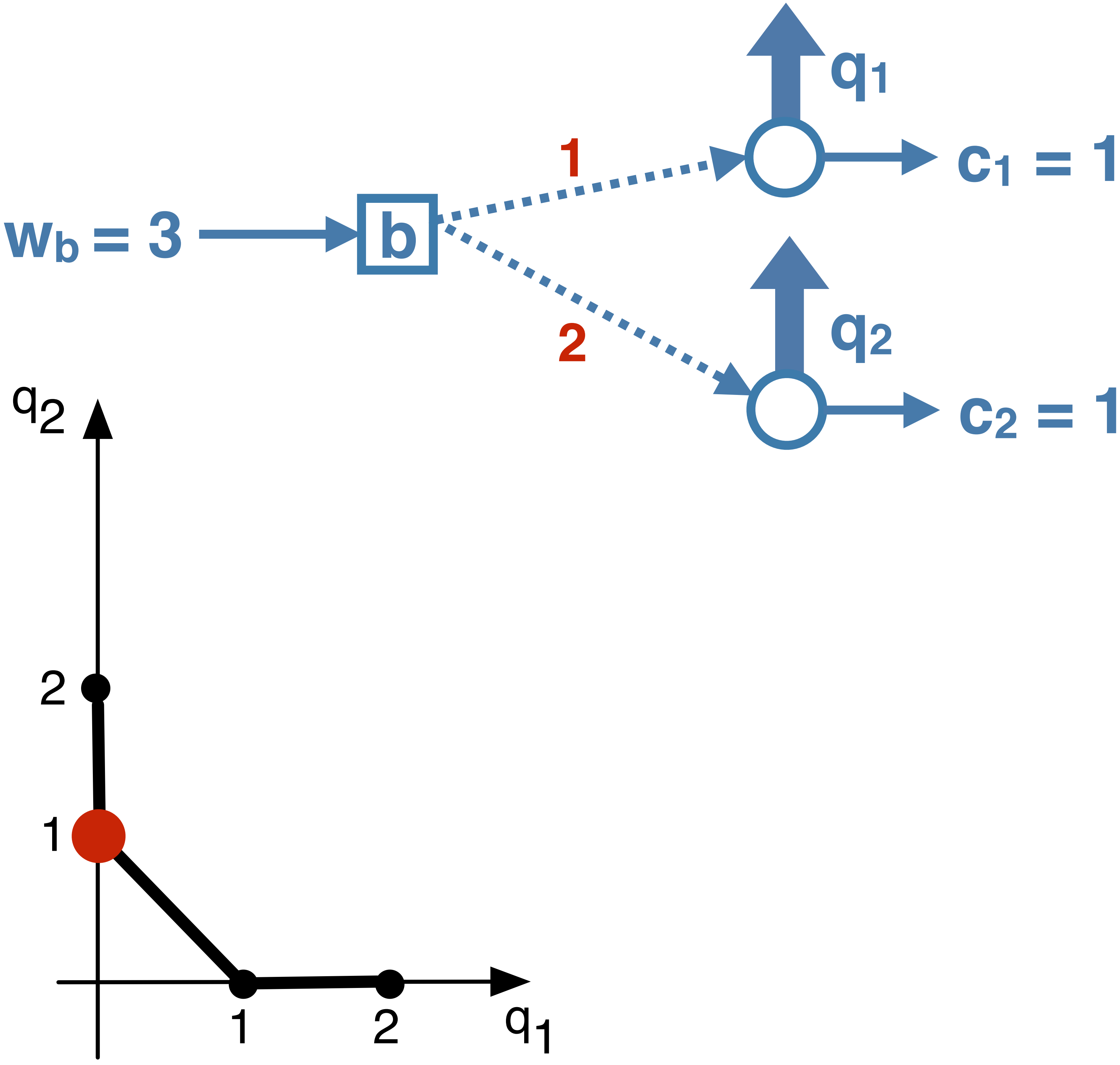
q_s = queue overflow rate at server s



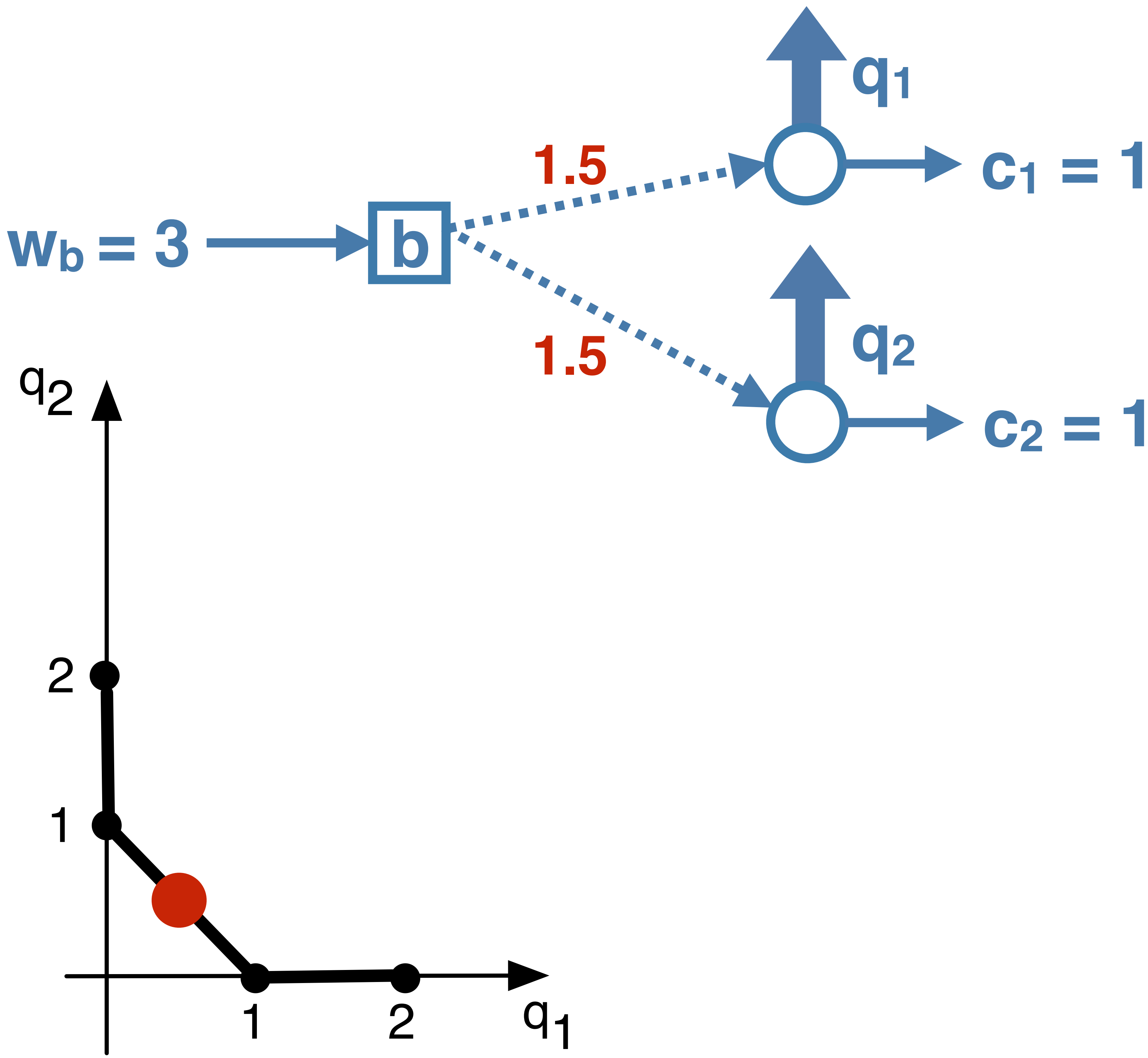
Fluid-level achievable performance



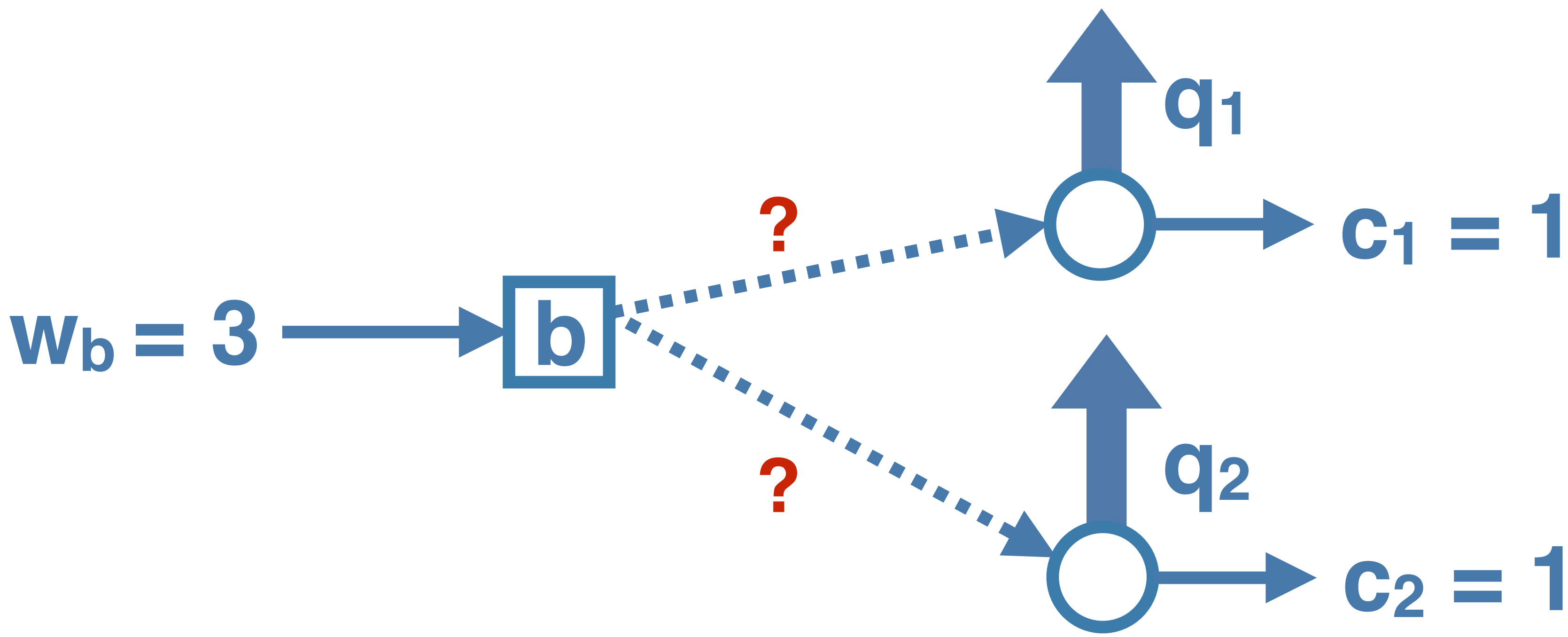
Fluid-level achievable performance

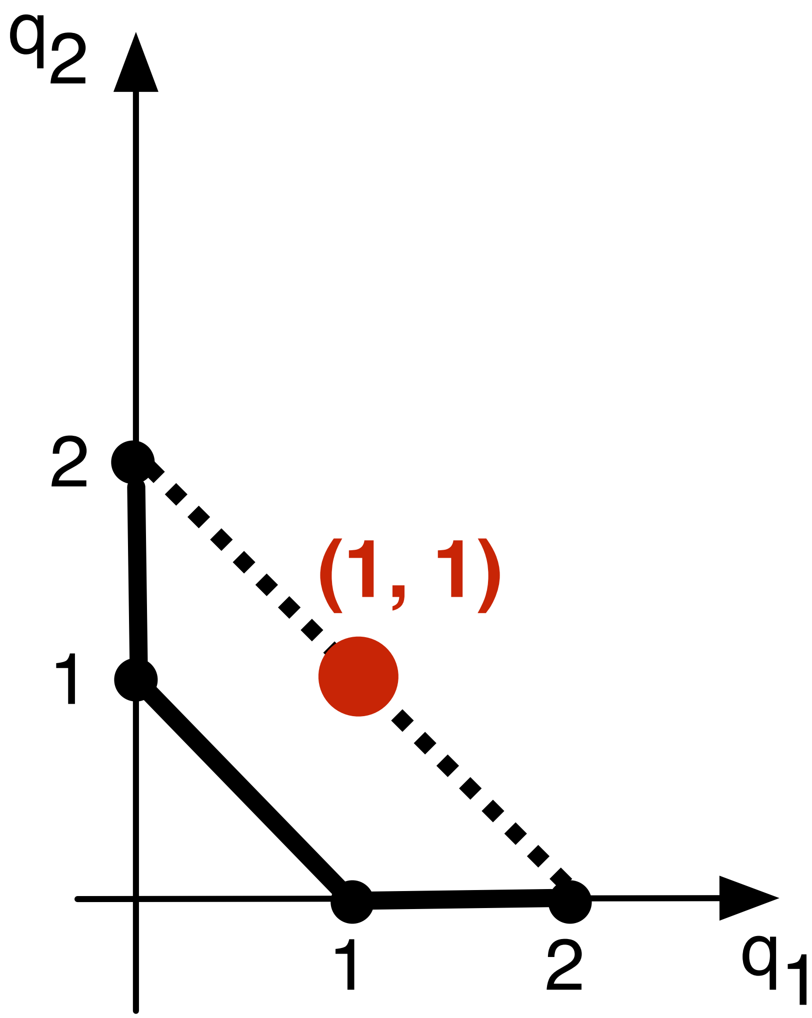


Fluid-level achievable performance



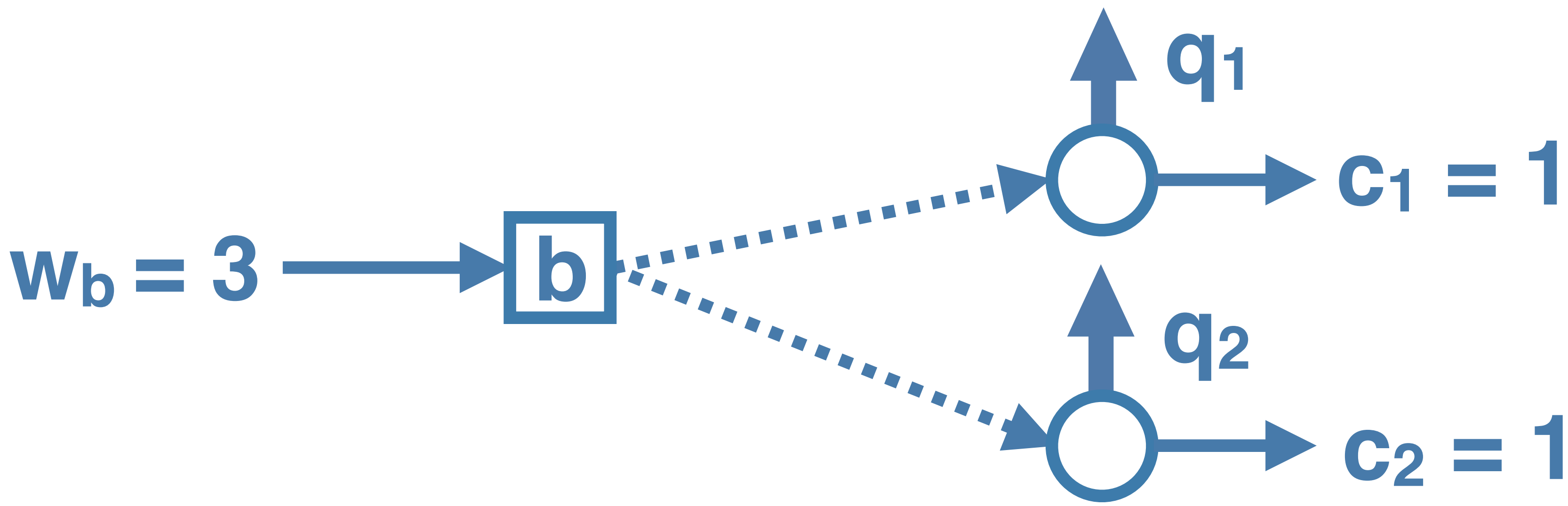
Fluid-level achievable performance

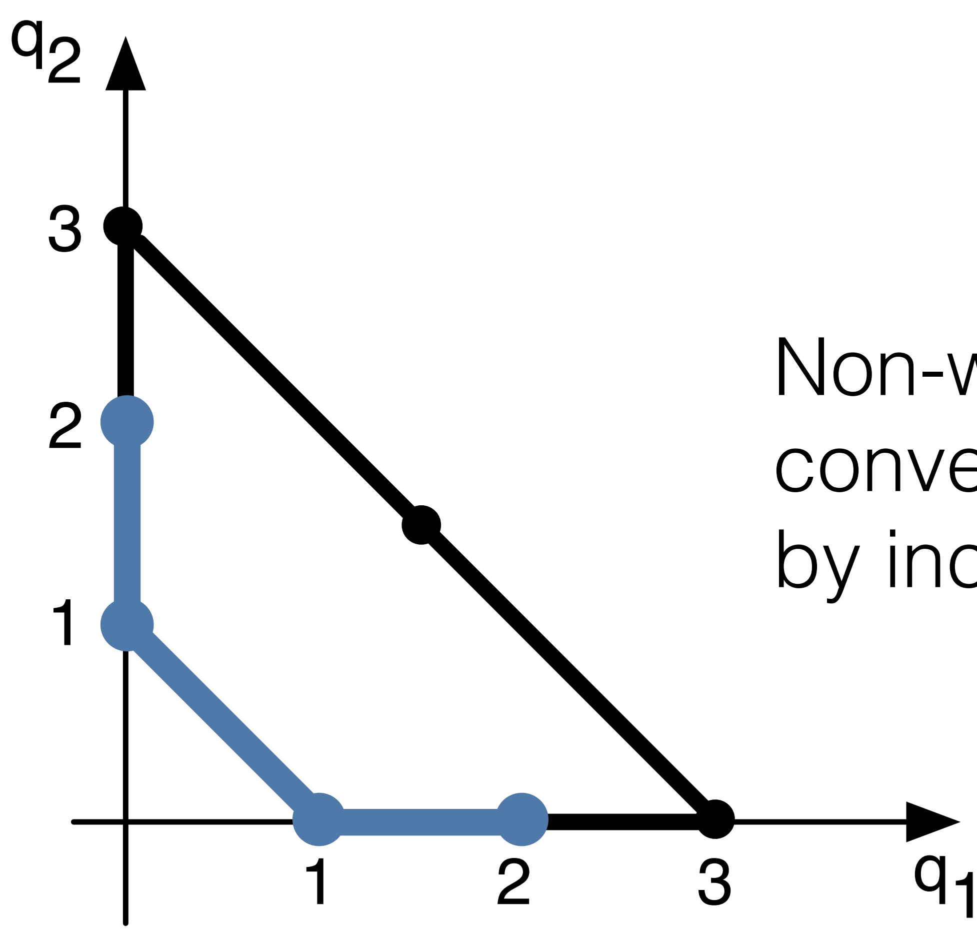




- Is the achievable region convex?
- Can (1, 1) be achieved by equal time-sharing between (2, 0) and (0, 2)?
- No! it induces queue overflow = (0.5, 0.5)

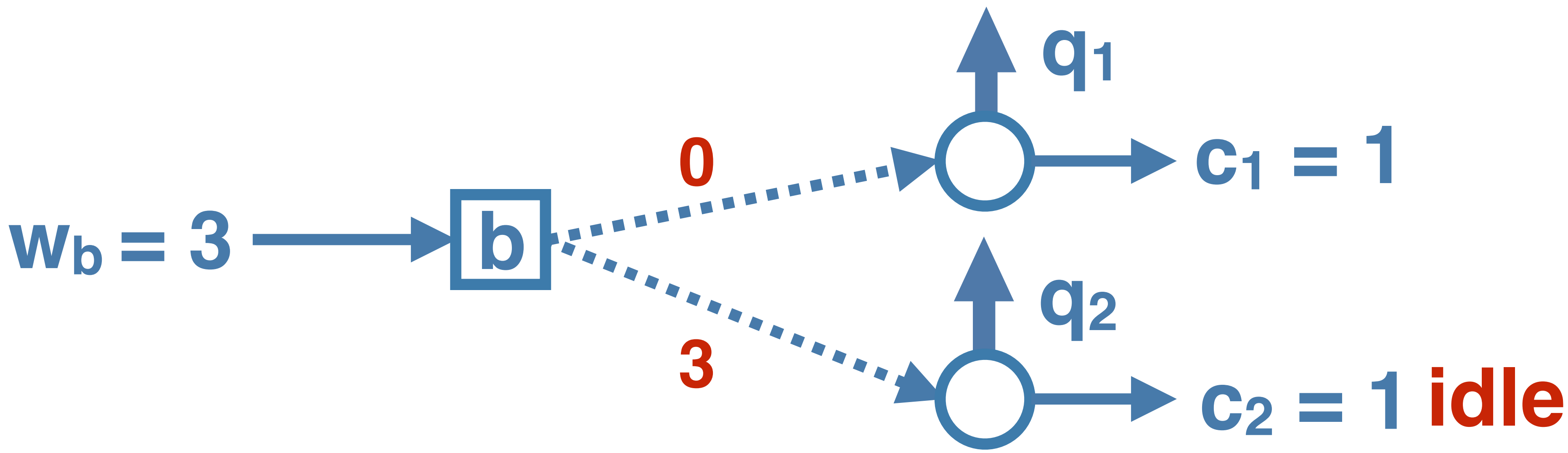
Fluid-level achievable performance

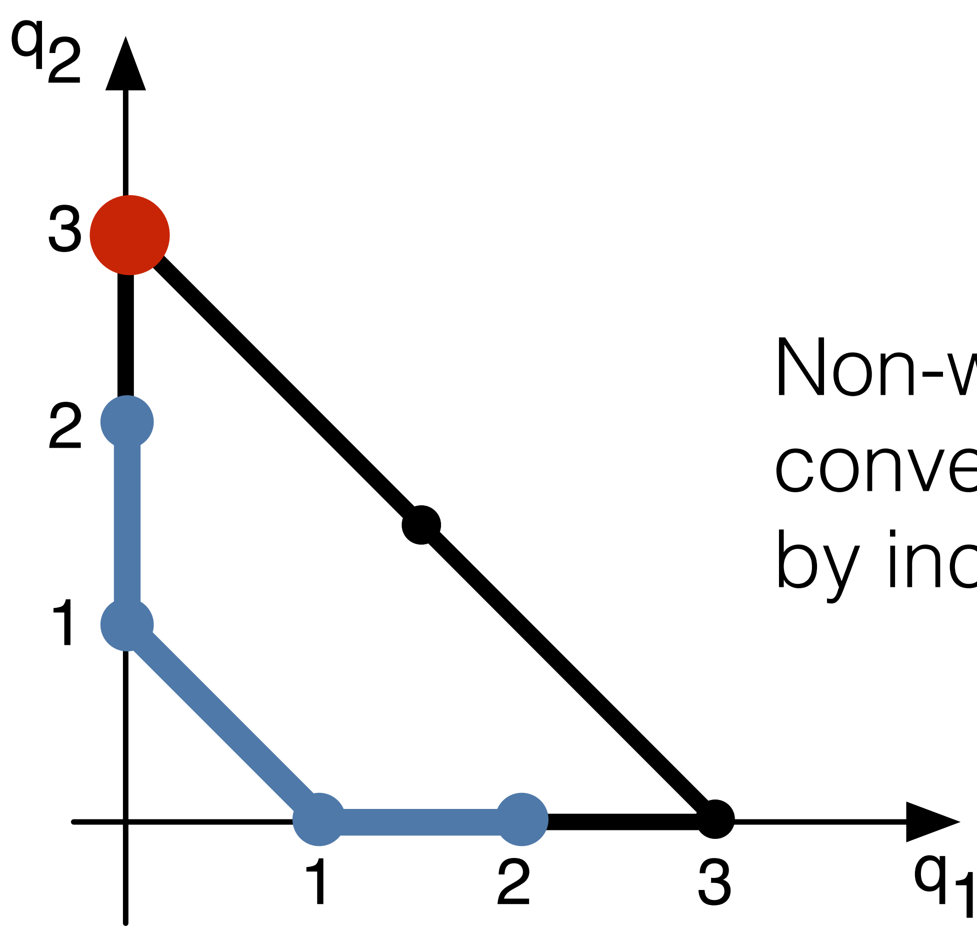




Non-work-conserving policies convexify the queue overflow region by including suboptimal points

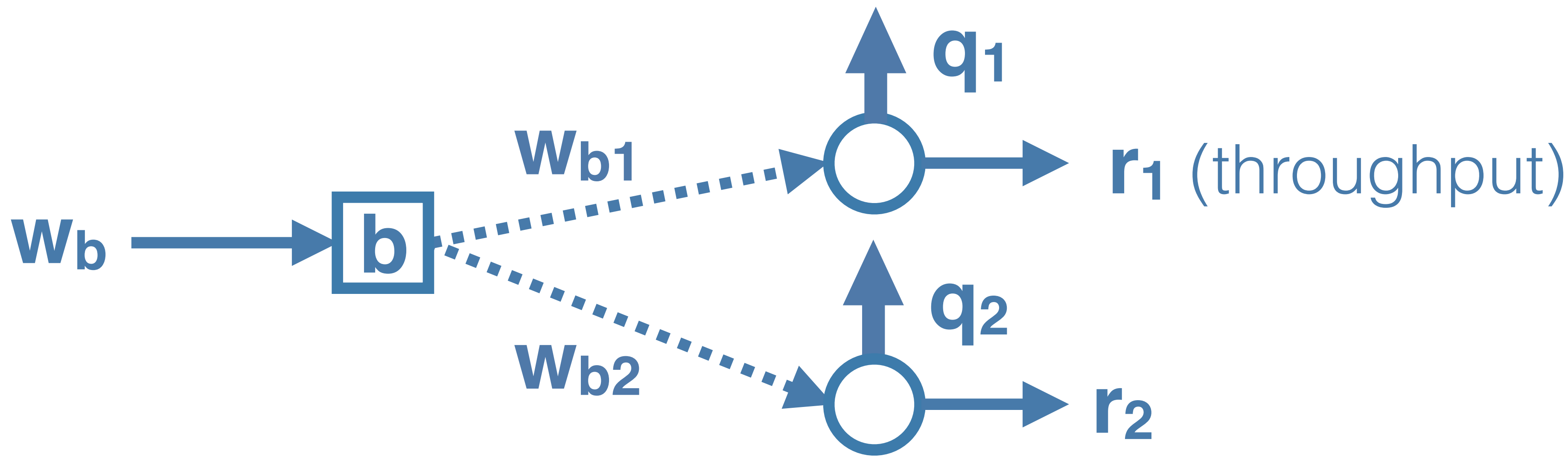
Fluid-level achievable performance

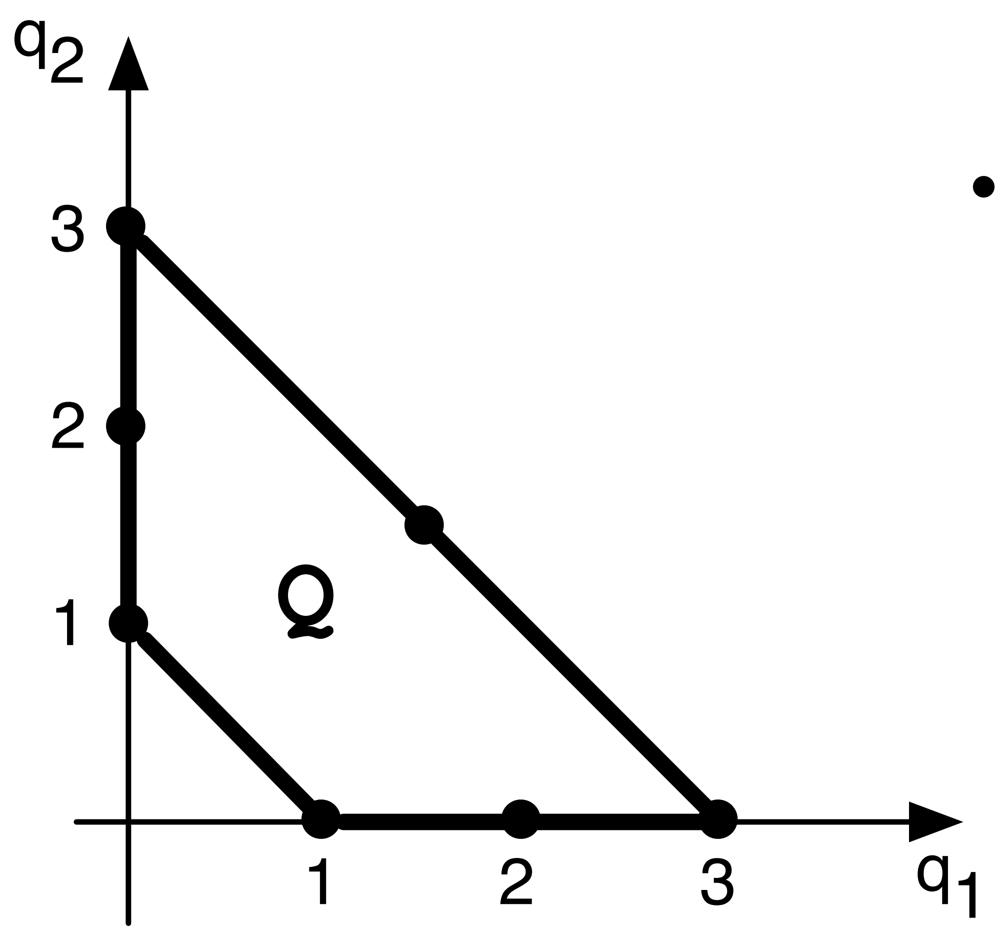




Non-work-conserving policies convexify the queue overflow region by including suboptimal points

Convex queue overflow region





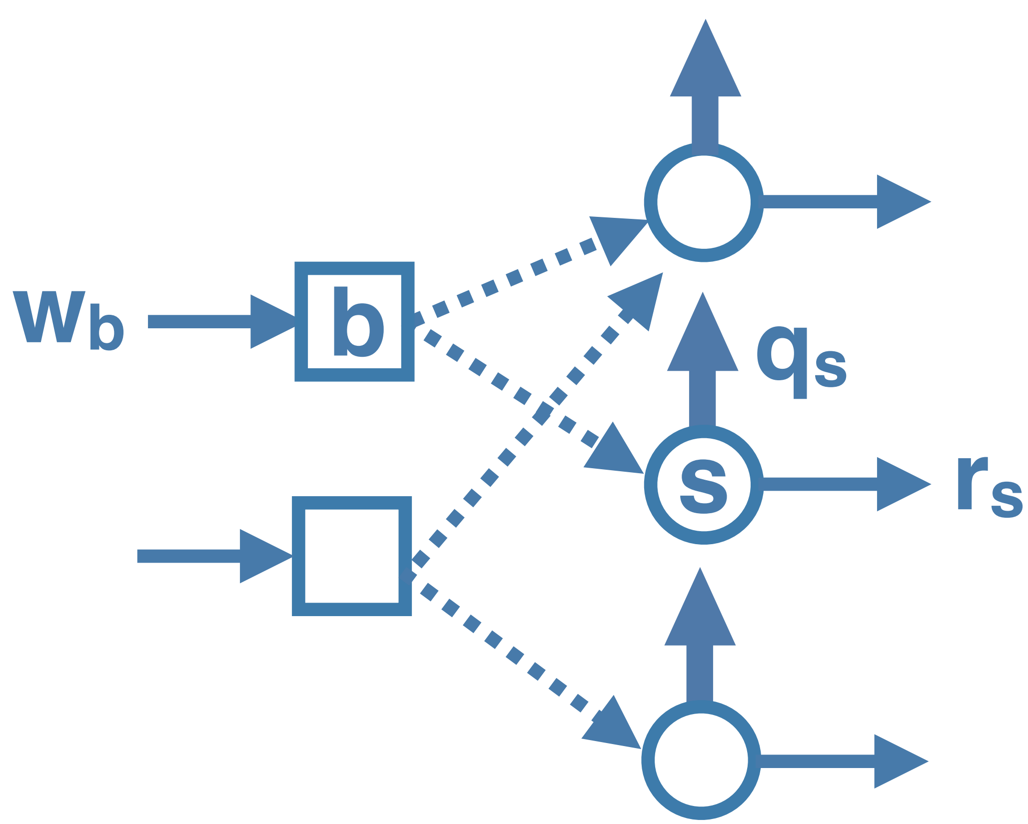
- Overflow vector **q** is feasible if there exist flow variables that satisfy
  - $w_b = w_{b1} + w_{b2}$
  - $w_{bs} = q_s + r_s$
  - $r_s \leq c_s$

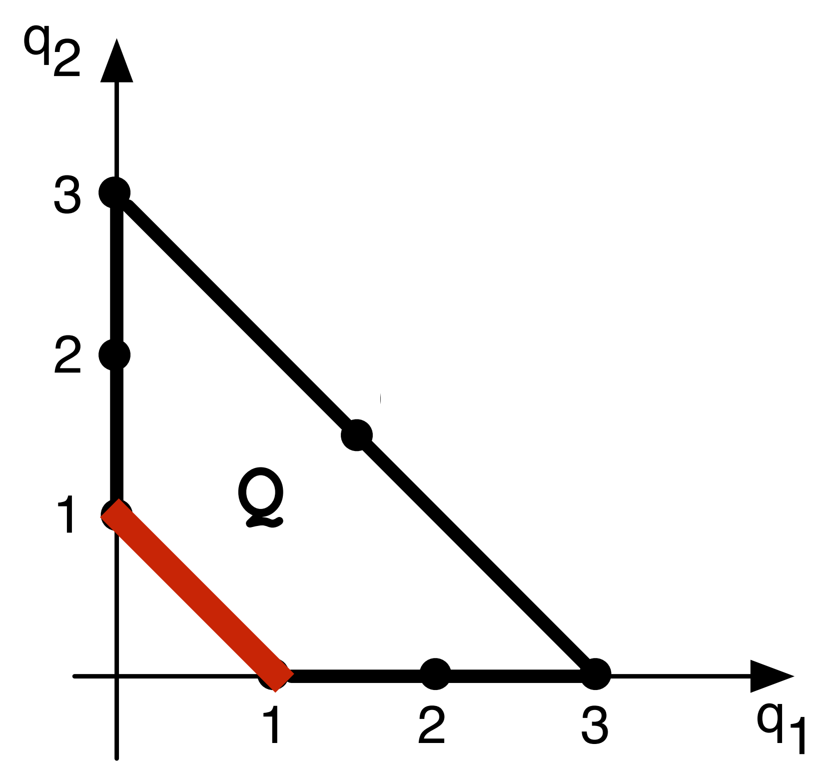
Desired operating points

- Maximum sum throughput

$$\sum_s r_s = \sum_b w_b - \sum_s q_s$$

  - Total throughput = total workload - total queue overflow
  - Equivalent to minimizing total queue overflow

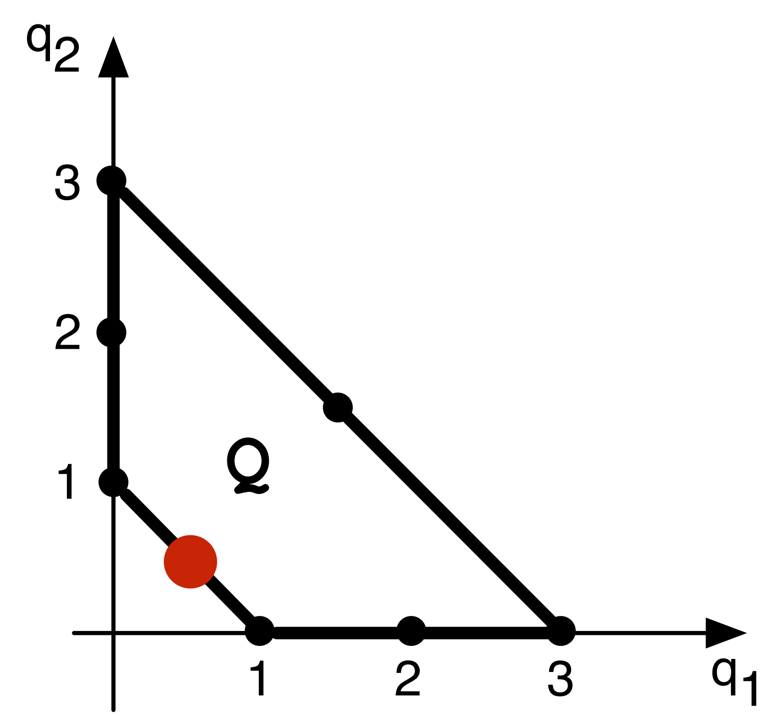




Desired operating points

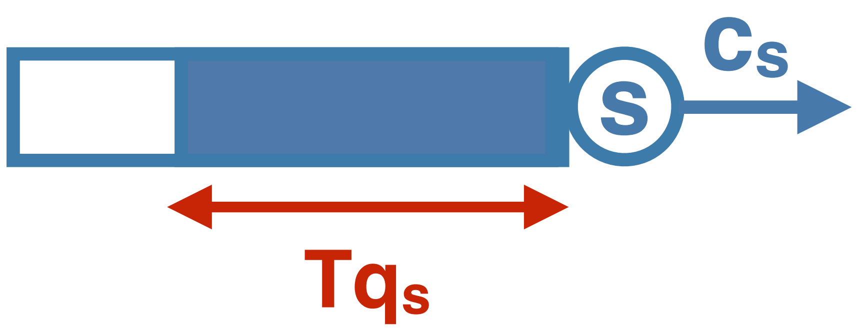
- Min-max-fair queue overflow vector
  - Maximizing the time to first buffer overflow

  - B/q_s = time to buffer overflow at server s
  - min_s $\{B/q_s\}$ = time to first system buffer overflow
  - Solve: max_**q** min_s $\{B/q_s\}$
    - Equivalent to min_**q** max_s $\{q_s\}$

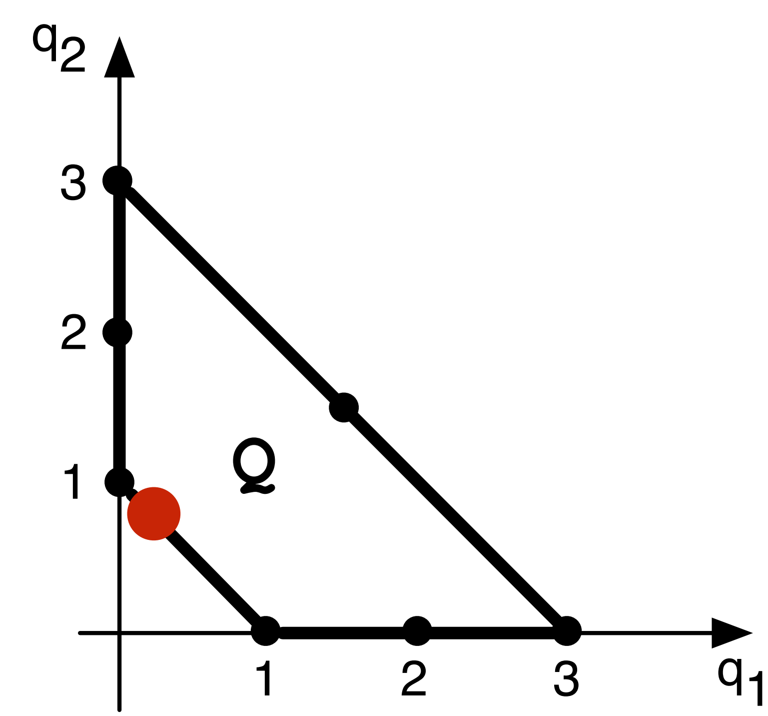


Desired operating points

- Weighted min-max-fair queue overflow vector
  - Minimizing recovery time from temporary overload

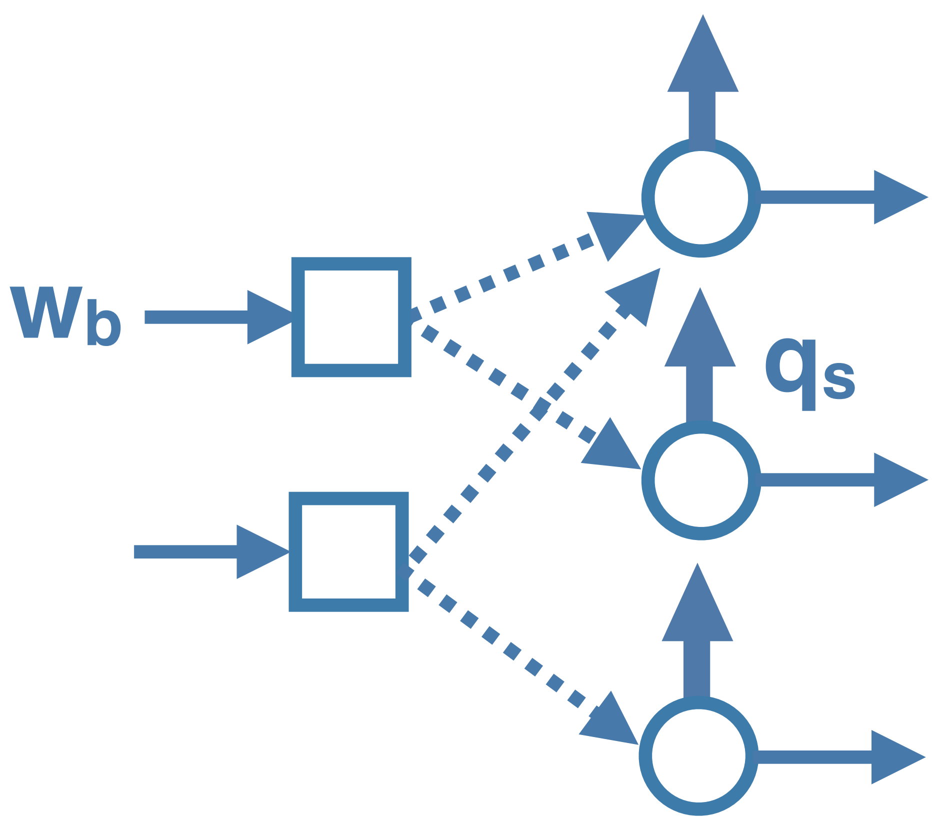


  - Overload happens in [0, T]
  - Tq_s = accumulated backlog at server s
  - Tq_s/c_s = backlog clearance time at s
  - max_s $\{Tq_s/c_s\}$ = system recovery time
  - Solve: min_**q** max_s $\{Tq_s/c_s\}$



Convex penalty minimization

- Achieving desired operating points by solving convex minimization problems



minimize: $\sum_s h_s(q_s)$

subject to: $(q_1, \ldots, q_S) \in \mathcal{Q}$

h_s: convex and increasing

α-fair penalty function

$$h(q) = \frac{q^{1+\alpha}}{1+\alpha}, \quad \alpha \geq 0$$

- Generalization of α-fair utility function by allowing $\alpha < 0$

$$g(q) = \frac{q^{1-\alpha}}{1-\alpha} \quad \text{(concave function)}$$

- α=0, h(q) = q, minimizing total queue overflow

$$\text{minimize: } \sum_s q_s$$

$$\text{subject to: } (q_1, \ldots, q_S) \in \mathcal{Q}$$

Penalty proportional fairness (α=1)

$$h(q) = q^2/2$$

- The optimal queue overflow vector **q*** satisfies

$$\sum_{s=1}^{S} (q_s - q_s^*)\, q_s^* \geq 0,\ \ q \neq q^*$$

  - To improve one's penalty by 1%, the others' penalty must worsen by at least 1%

- Rate proportional fairness **r*** (with g(r)=log(r)) satisfies

$$\sum_n \frac{r_n - r_n^*}{r_n^*} \leq 0,\ \ r \neq r^*$$

  - To improve one's reward by 1%, the others' reward must worsen by at least 1%

- Product form vs ratio form

Min-max fairness (α=∞)

- Let **q***(α) solve

$$\text{minimize: } \sum_{s=1}^{S} \frac{(q_s)^{1+\alpha}}{1+\alpha}, \text{ subject to: } (q_1, \ldots, q_S) \in \mathcal{Q}$$

- $q^*(\infty) = \lim_{\alpha \to \infty} q^*(\alpha)$ exists and is the min-max fair point in the feasible queue overflow region
- Example: α=9, $\mathbf{q_1}$ = (3, 2.1, 1), $\mathbf{q_2}$ = (3, 2, 1.9)
    - $\mathbf{q_2}$ is fairer than $\mathbf{q_1}$
    - Double check: $3^{10} + (2.1)^{10} + 1^{10} > 3^{10} + 2^{10} + (1.9)^{10}$
    - α-fair penalty function singles out the largest components that are different
- Weighted min-max fairness: $h(q_s) = (q_s/c_s)^{1+\alpha}/(1+\alpha)$

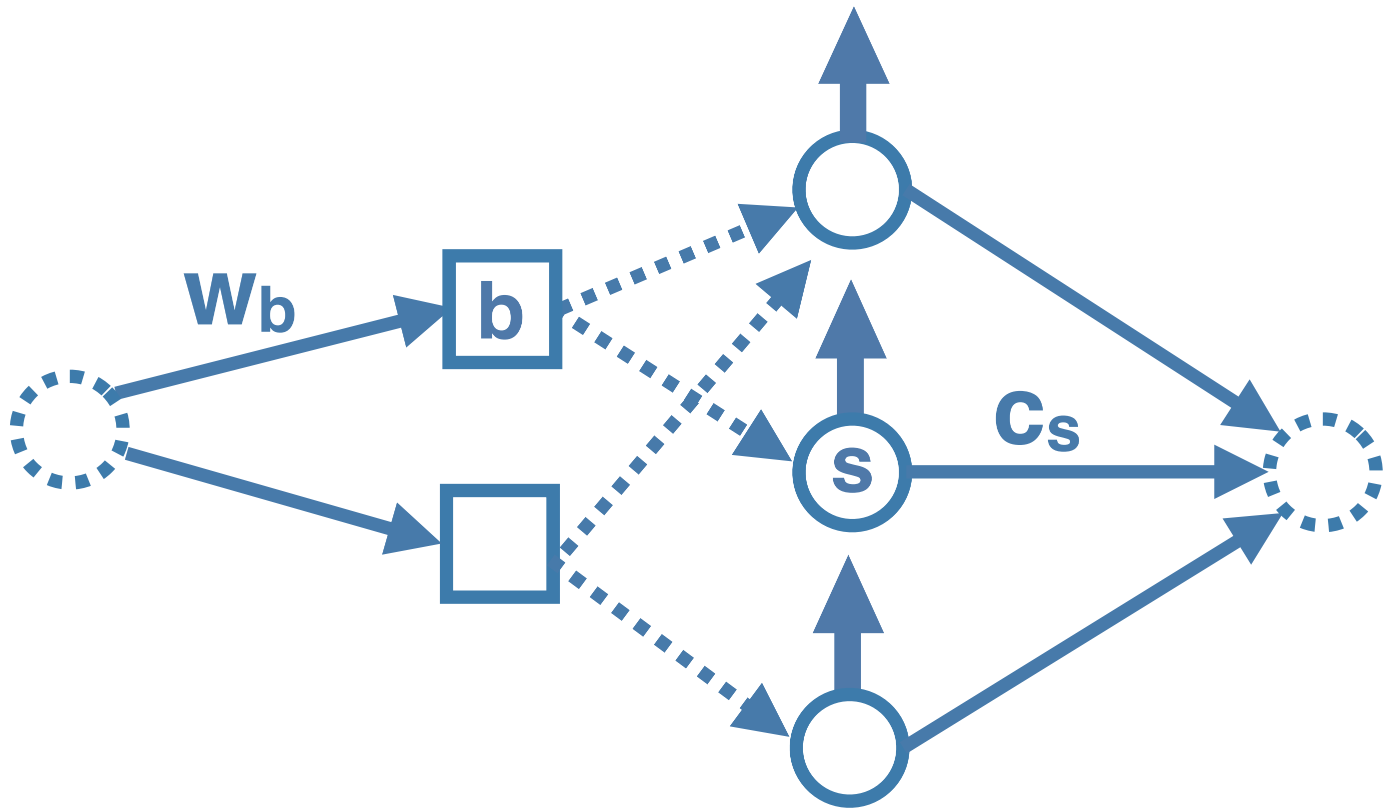
Throughput optimality

- Any control policy that solves

$$\text{minimize: } \sum_s h_s(q_s), \text{ subject to: } (q_1, \ldots, q_S) \in \mathcal{Q}$$

must achieve maximum throughput

- Proof: Construct a single-commodity network with queue overflows and use max-flow min-cut theorem



Dynamic control policy

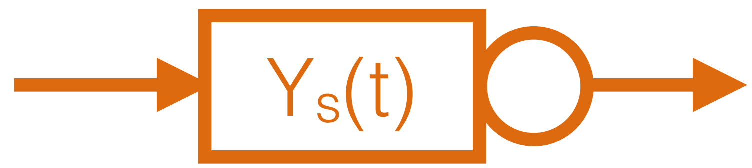
- Control queue overflow rates by job tagging

- Untagged jobs have strict priority over tagged jobs
- Keep # of untagged jobs $N_s(t)$ finite at server s
  - $N_s(t)$ = # of untagged jobs waiting at server s in slot t
  - job untagging rate = throughput at s
  - job tagging rate = queue overflow rate at s
- Use virtual queue to optimize job tagging rate

Optimizing job-tagging rate

- Set up a virtual queue (i.e., counter) $Y_s(t)$ at server s

# of jobs tagged
by server s in slot t

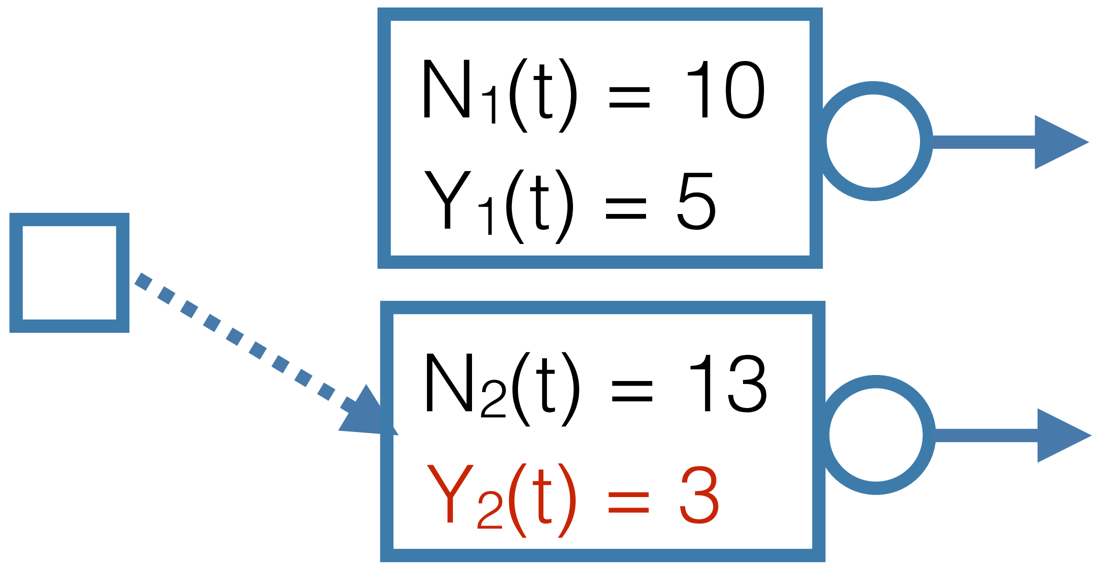


$y_s(t)$

- Minimizing h_s(job tagging rate at server s) by:
  - Stabilize queue $Y_s(t)$
  - Minimize h_s(average service rate of queue $Y_s(t)$)
  - Lyapunov drift-plus-penalty algorithm

Optimal overload balancing policy

- Each balancer b routes jobs $A_b(t)$ in a slot to the accessible server with the shortest queue over $N_s(t)$ and $Y_s(t)$



- Server s tags all incoming jobs if and only if $N_s(t) > Y_s(t)$
- Update $N_s(t)$ and $Y_s(t)$ in every slot, where the service rate $y_s(t)$ of $Y_s(t)$ is the solution to

$$\text{minimize: } \boldsymbol{V h_s}(\boldsymbol{y}\mathbb{E}[\boldsymbol{X}]) - \boldsymbol{Y_s}(\boldsymbol{t})\,\boldsymbol{y}, \text{ subject to: } \boldsymbol{y} \geq 0$$

Optimal overload balancing policy

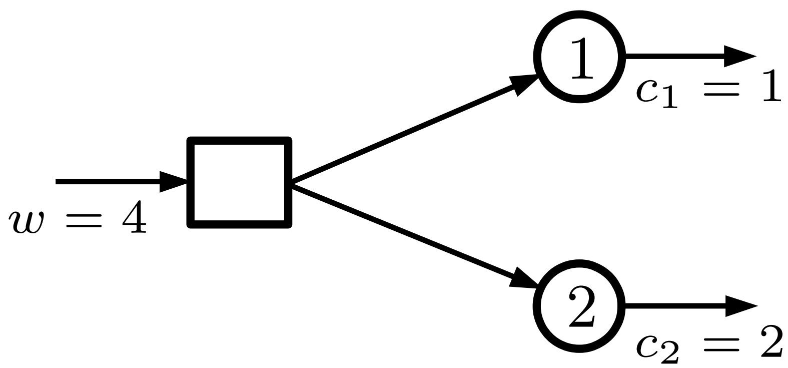
- Theorem: The control policy yields queue overflow rates that satisfy

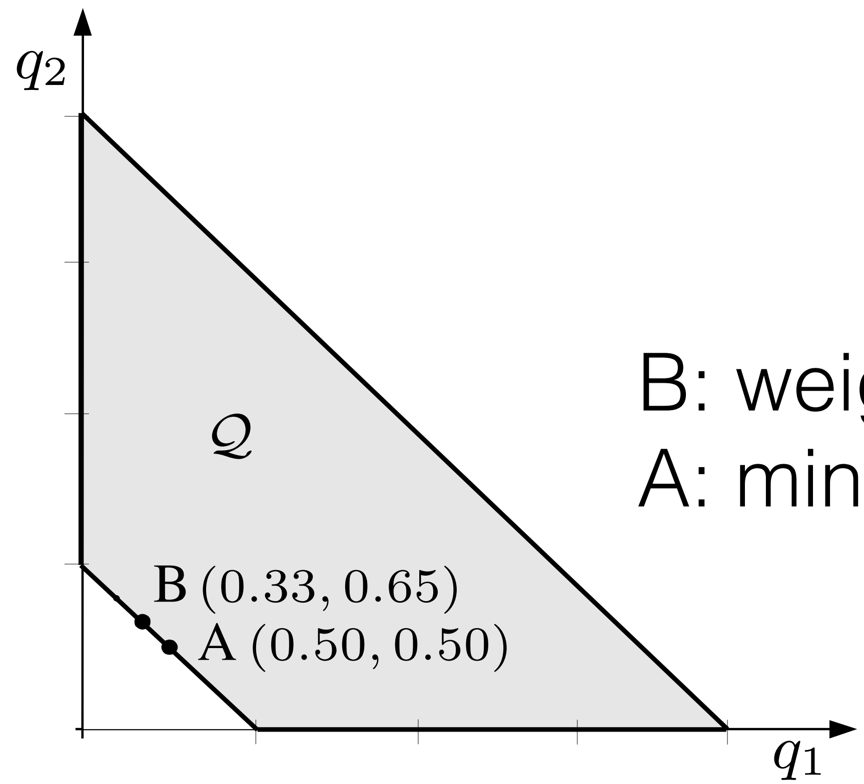
$$\sum_{s=1}^{S} h_s(q_s) \leq h^* + F/V$$

F: finite constant
V: control parameter
h*: optimal penalty

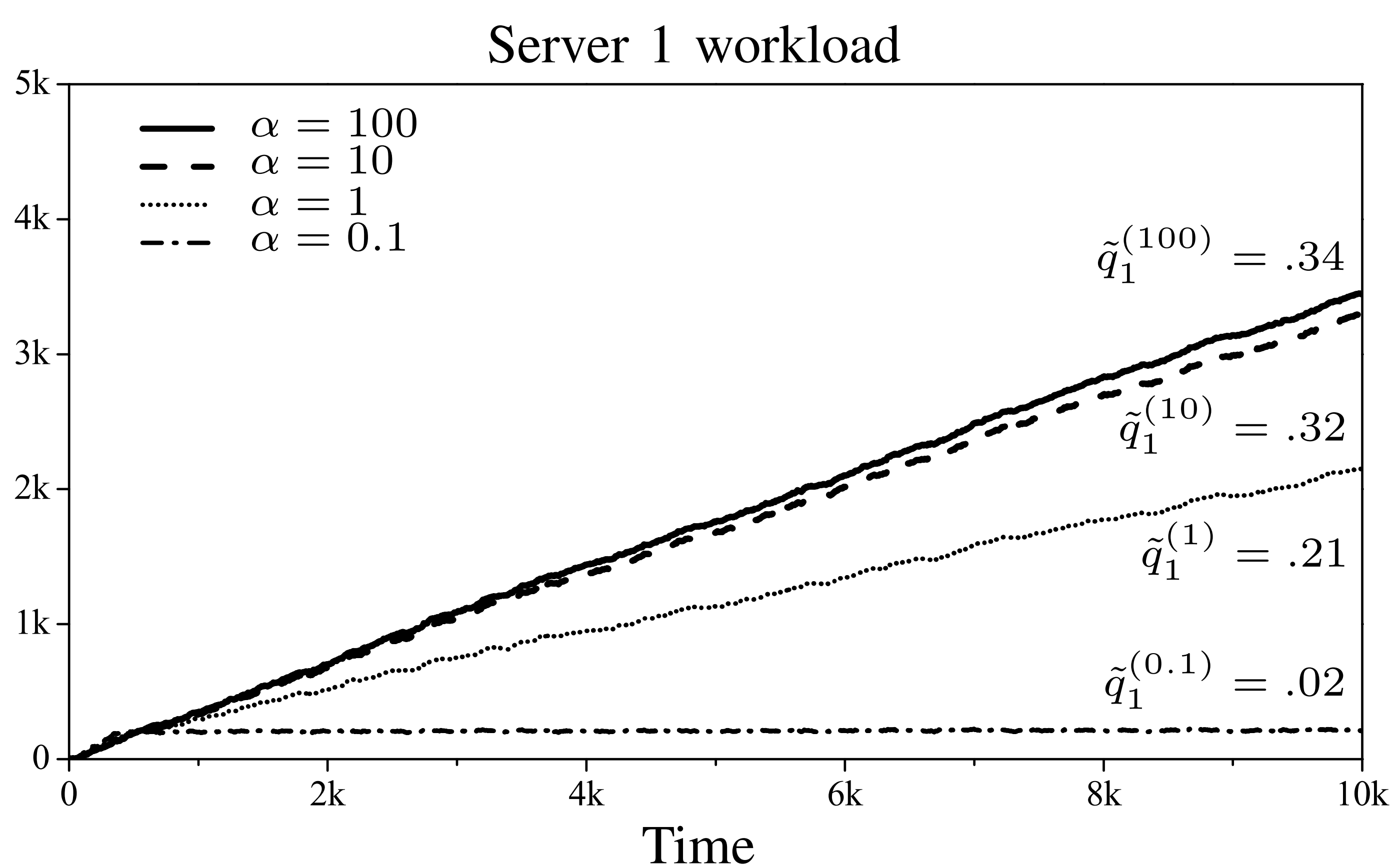
- The policy doesn't need arrival statistics and is thus robust

Simulations



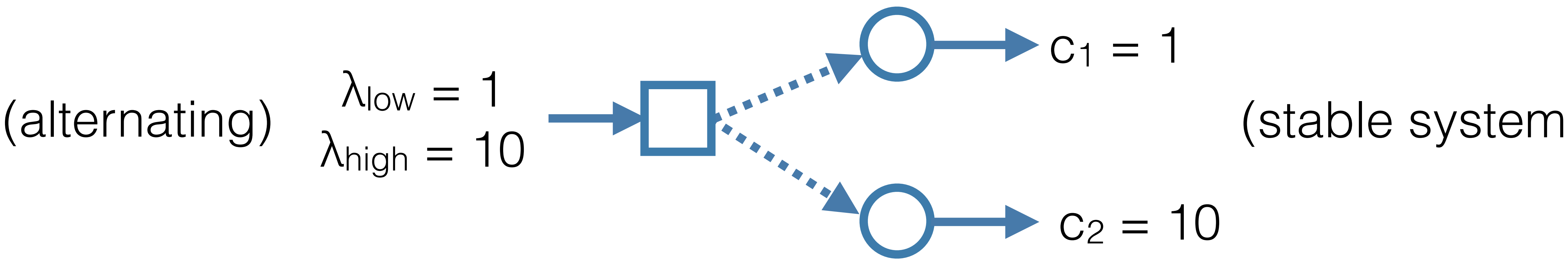


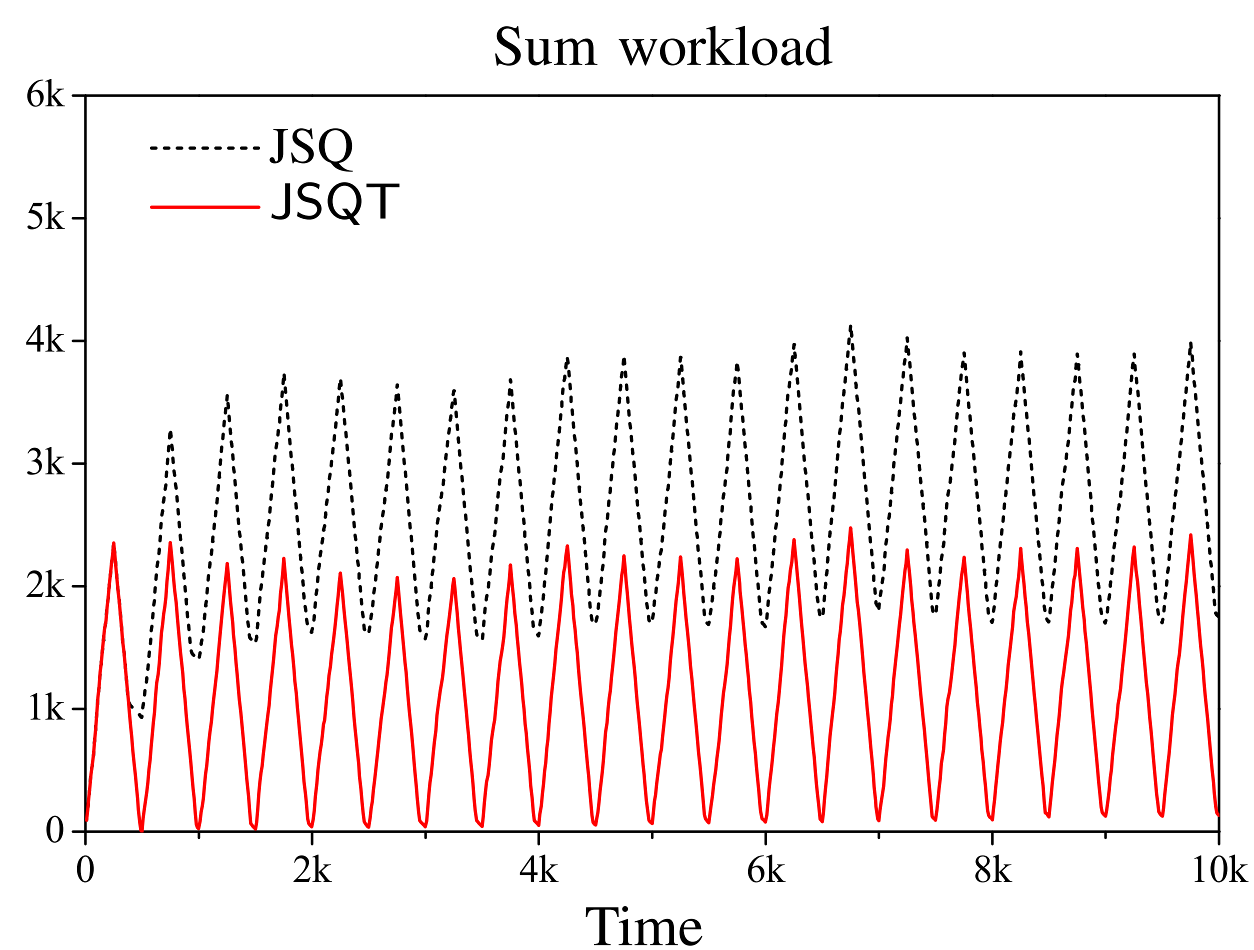
B: weighted min-max fairness
A: min-max fairness



Server 1 workload
in case B

Simulations





Summary

- Overload is increasingly common in network systems
- Overload control as a convex optimization problem in a stochastic queueing network
- α-fair penalty function
- A job-level dynamic control policy
  - Throughput optimal
  - Turn the optimal control of an unstable queueing system into one that stabilizes a virtualized queueing system
- Future research: multi-class traffic, integration with traditional load balancing, applications (traffic offloading, data load shedding, etc)