Delay and Rate-Optimal Control in a Multi-Class Priority Queue

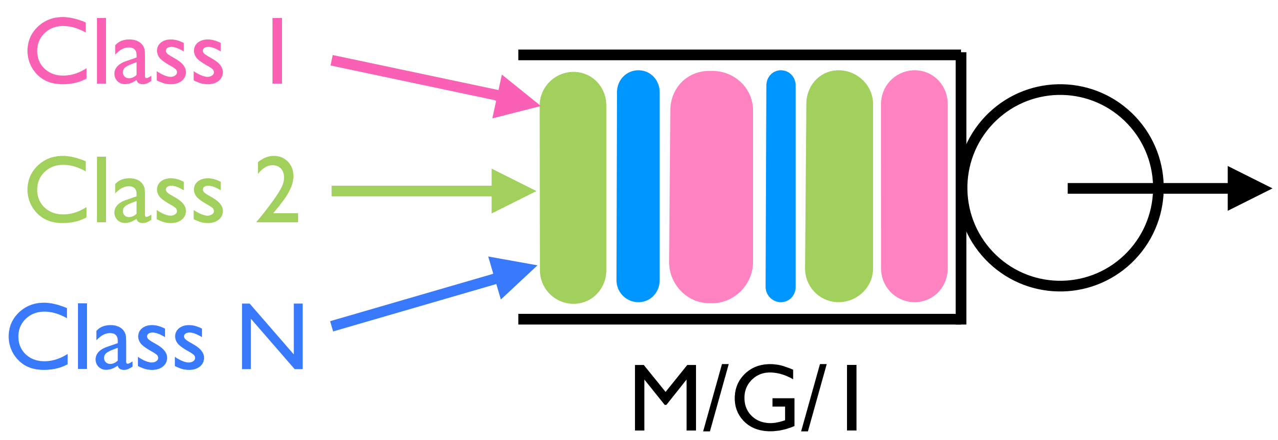
Chih-ping Li

Massachusetts Institute of Technology

joint work with Michael Neely (USC)

INFOCOM 2012

Revisit a classical queue control problem



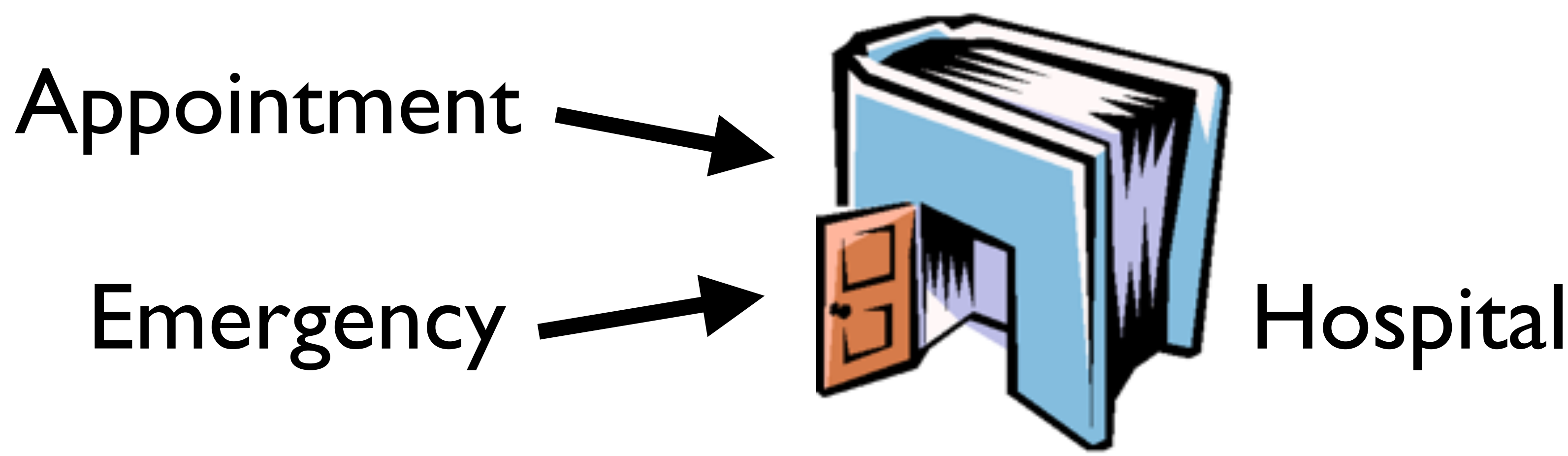
An M/G/1 queue serves N traffic classes

Pick a class, serve a job (nonpreemptively) one at a time

The service rate may be controllable (with costs)

Goal: Solve convex delay optimization problems

Q1: Providing delay guarantees



Feasibility

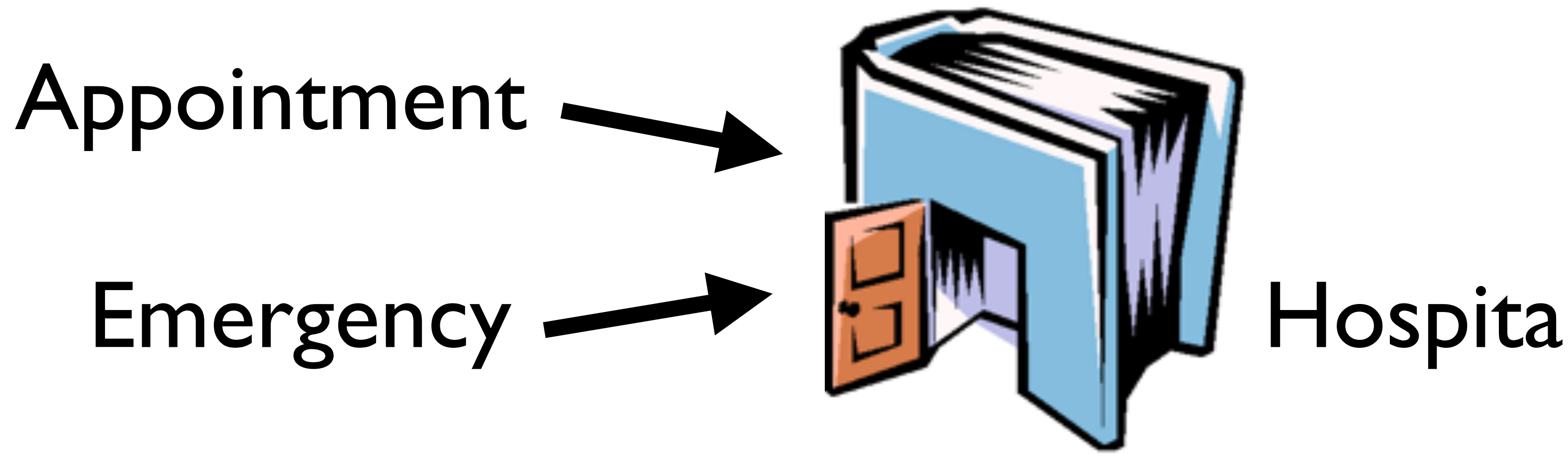
$$\overline{W_1} \leq 2$$

$$\overline{W_2} \leq 5$$

$$\cdots\cdots$$

$$\overline{W_N} \leq 3$$

Q2: Providing delay fairness



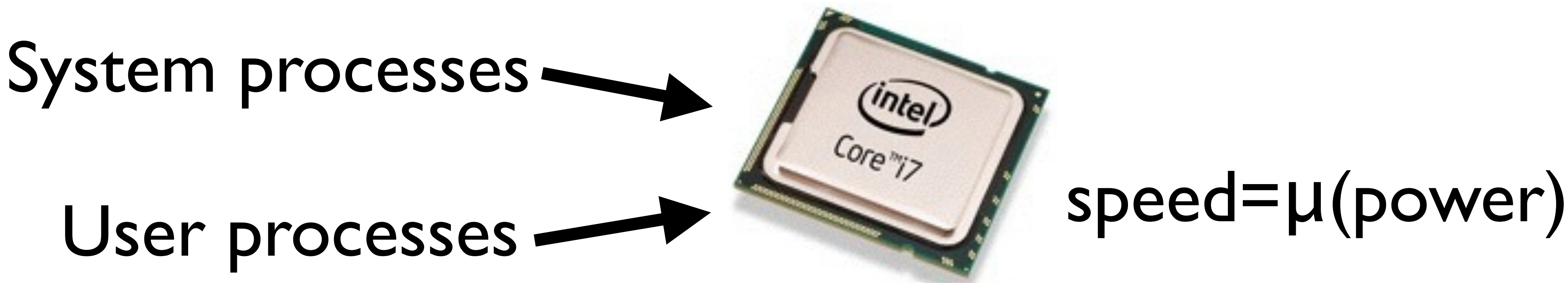
Fairly treat two types of patients

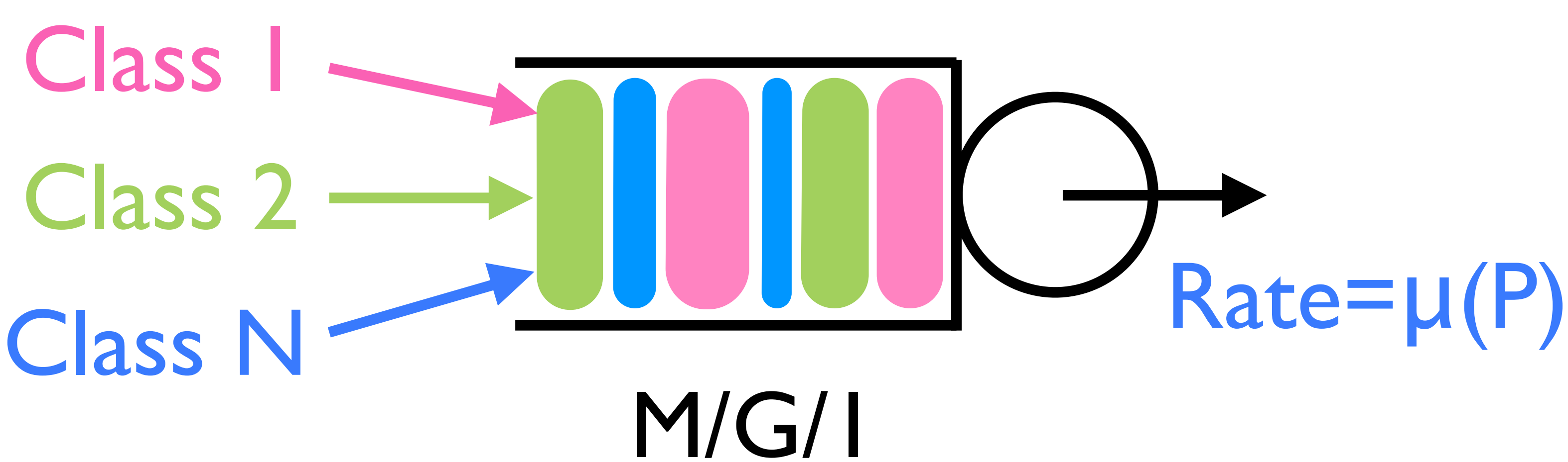
Convex delay penalty

Minimize: $\sum_{n=1}^{N} f_n(\overline{W_n})$

subject to: $\overline{W_n} \leq d_n, \ \forall n$

Q3: Minimizing service cost





(update service rates across busy periods)

Rate control

Minimize: P_{av}

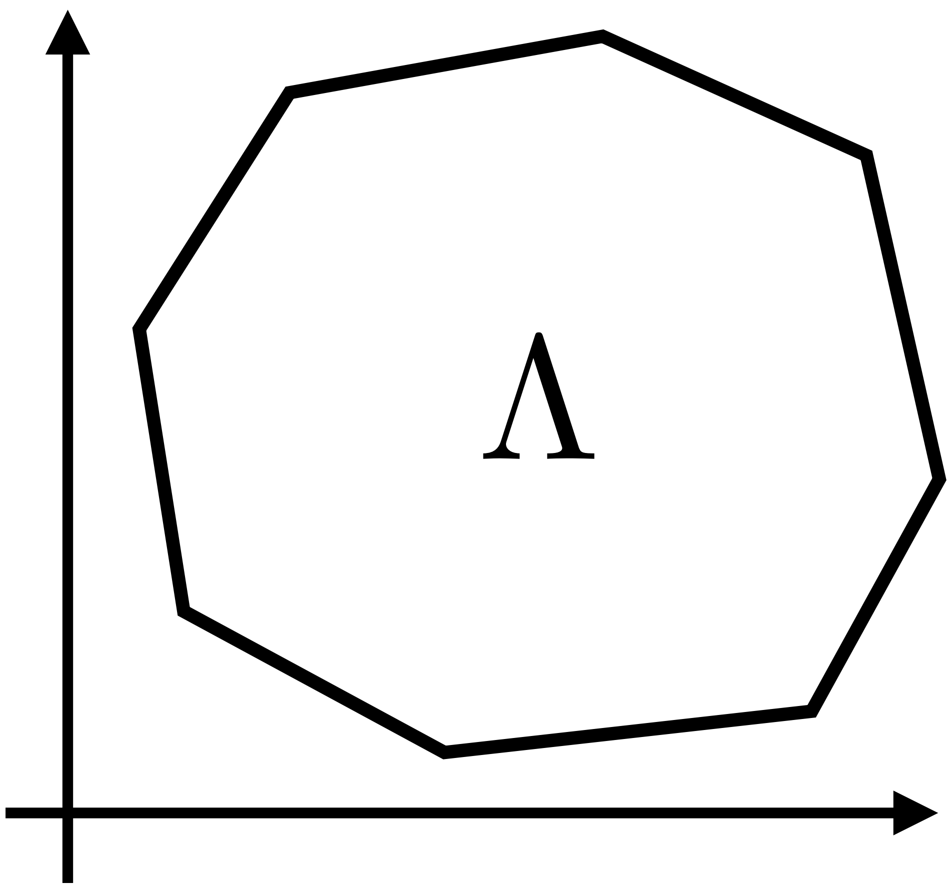
subject to: $\overline{W_1} \leq 2$

$\cdots\cdots$

$\overline{W_N} \leq 3$

Optimization approach

(i) Compute performance region



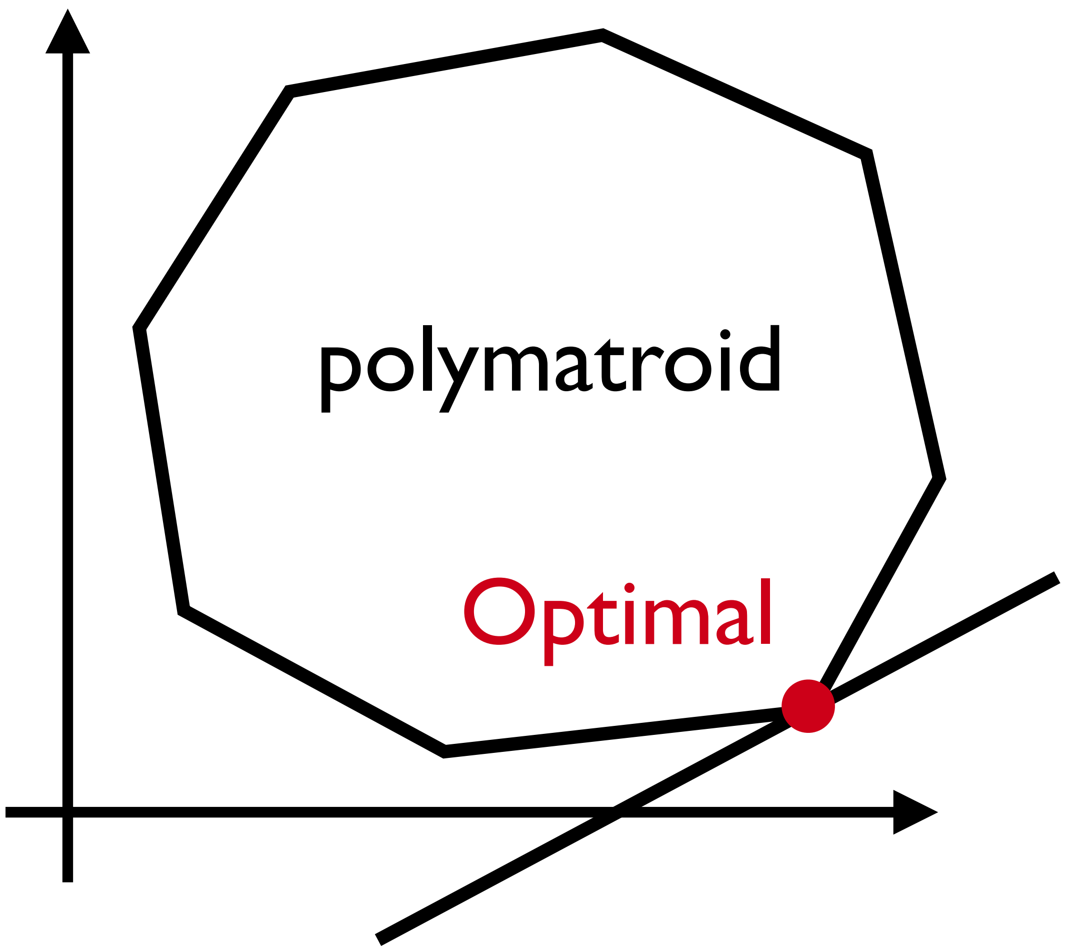
(ii) Solve optimization problem

$$\text{Minimize: } \sum_{n=1}^{N} c_n \overline{W_n}$$

$$\text{subject to: } (\overline{W_n})_{n=1}^{N} \in \Lambda$$

Gelenbe, Mitrani, Federgruen, Groenevelt, Shanthikumar, Tsoucas, Bertsimas, Dacre, Glazebrook, Garbe, Nino-mora, Yao, Coffman, ...

Linear optimization is elegantly solved



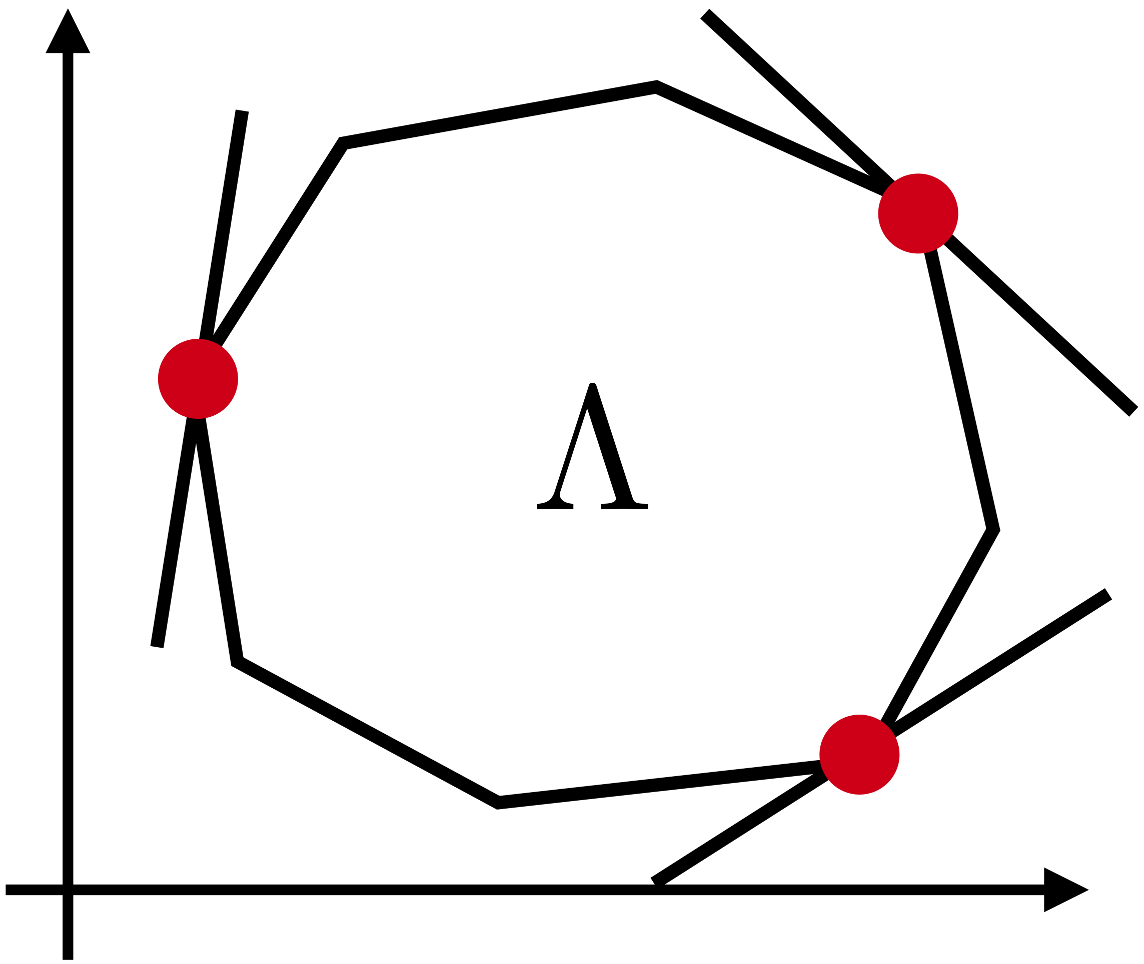
Minimize: $\sum_{n=1}^{N} c_n \lambda_n \overline{W_n}$

subject to: $(\overline{W_n})_{n=1}^{N} \in \Lambda$

Delay region (with fixed rate) is the base of a polymatroid [Federgruen-Groenevelt '88]

The $c\mu$ rule:
Assign priorities to job classes in the decreasing order of $c_n \mu_n$

More good news



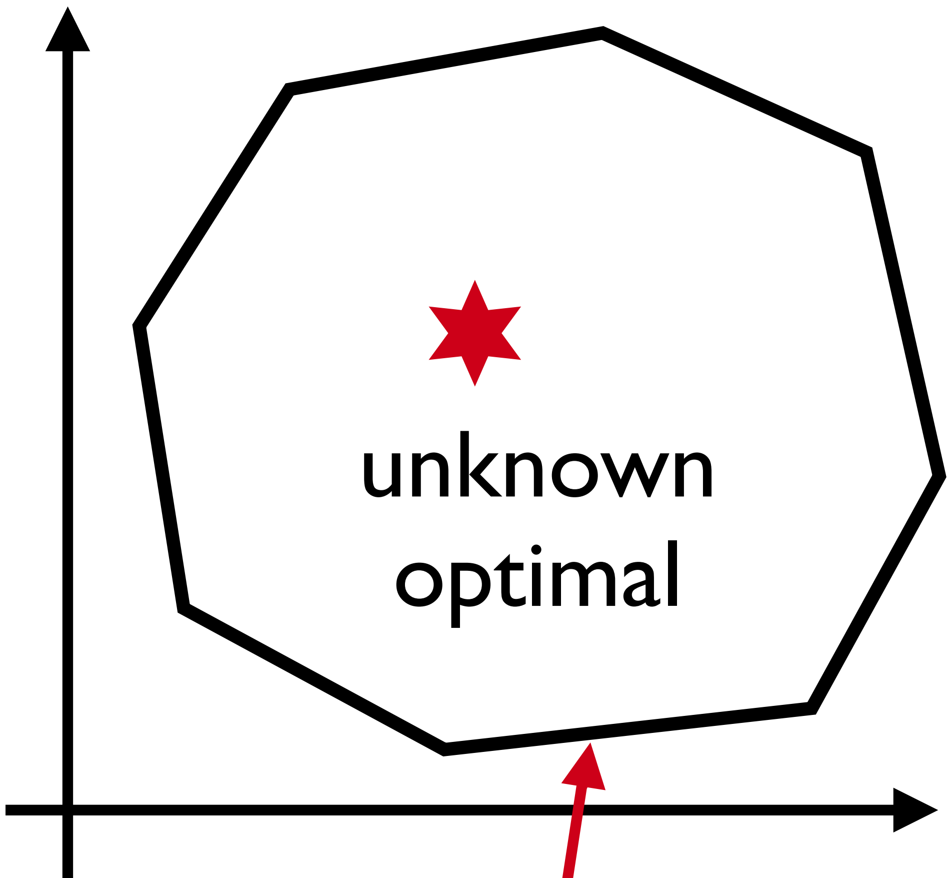
Linear program

Minimize: $\sum_{n=1}^{N} c_n \overline{W_n}$

subject to: $(\overline{W_n})_{n=1}^{N} \in \Lambda$

Every vertex = a priority policy

However, in general cases.....



$(2^N - 1)$ inequalities

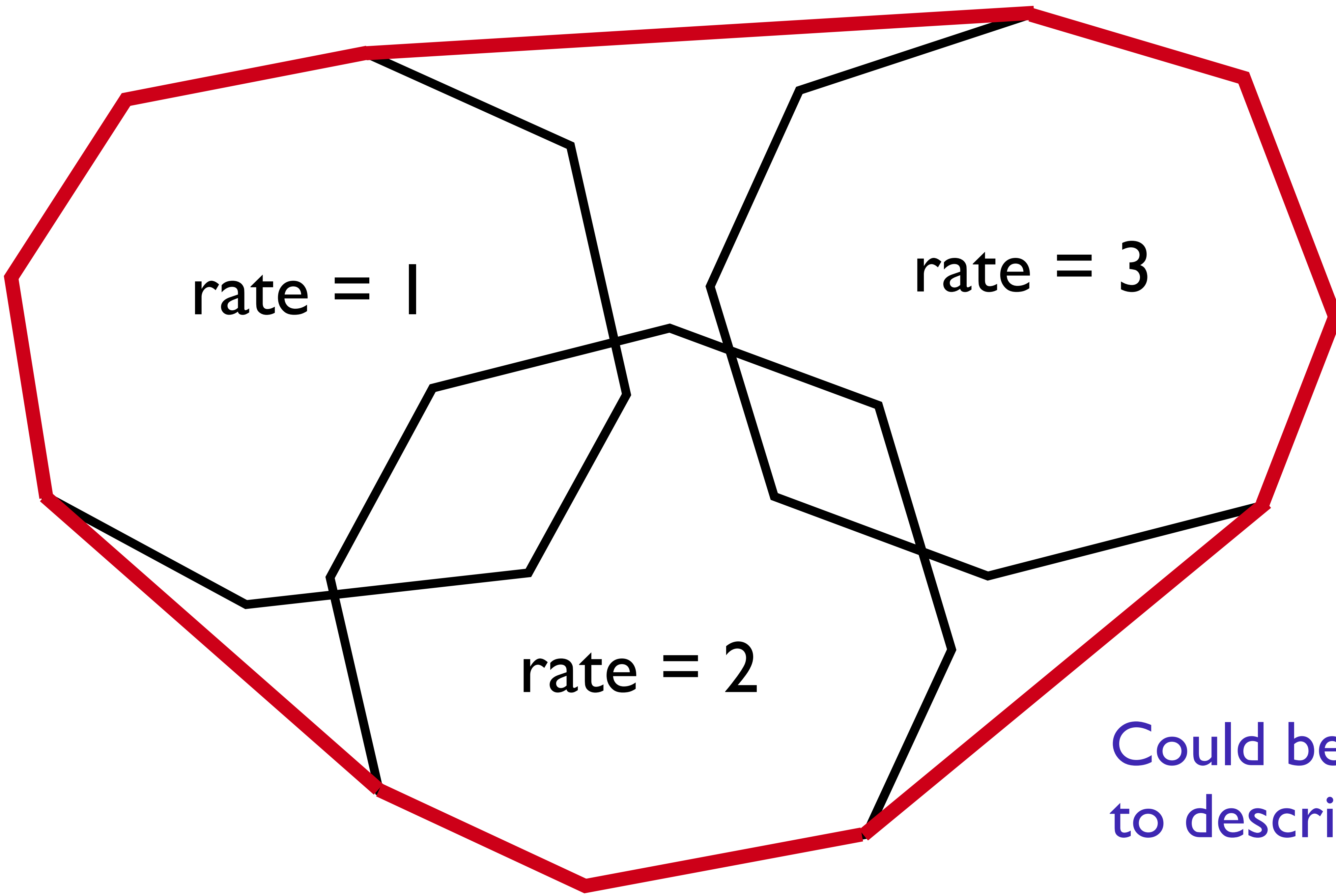
Minimize: $\sum_{n=1}^{N} f_n(\overline{W_n})$

subject to: $(\overline{W_n})_{n=1}^{N} \in \Lambda$

$\overline{W_n} \leq d_n, \ \forall n$

Offline solution doesn't reveal the optimal scheduling policy

Complex performance region



Could be difficult to describe

A new methodology to solve these problems

Feasibility

$$\overline{W_1} \leq 2$$

$$\overline{W_2} \leq 5$$

$$\cdots\cdots$$

$$\overline{W_N} \leq 3$$

Convex delay penalty

Minimize: $\sum_{n=1}^{N} f_n(\overline{W_n})$

subject to: $\overline{W_n} \leq d_n, \ \forall n$

Rate control

Minimize: P_{av}

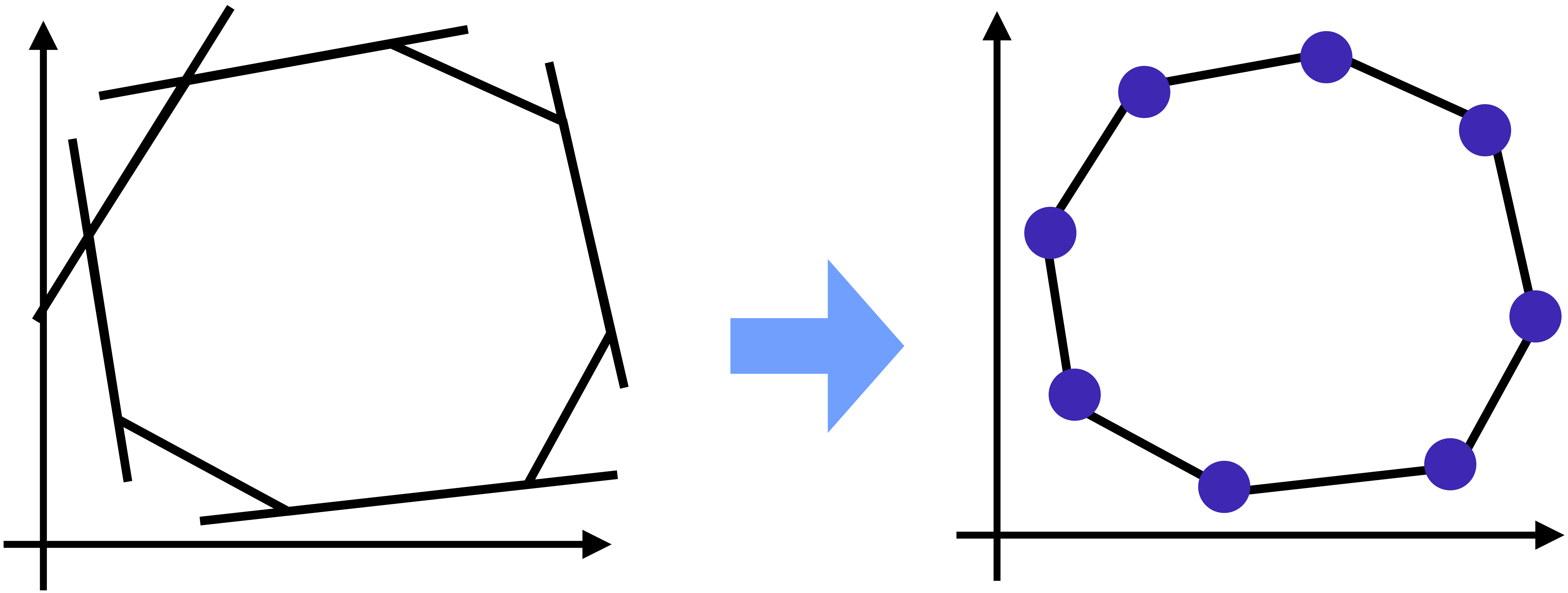
subject to: $\overline{W_1} \leq 2$

$$\cdots\cdots$$

$$\overline{W_N} \leq 3$$

Optimal policies are as simple as the $c\mu$ rule

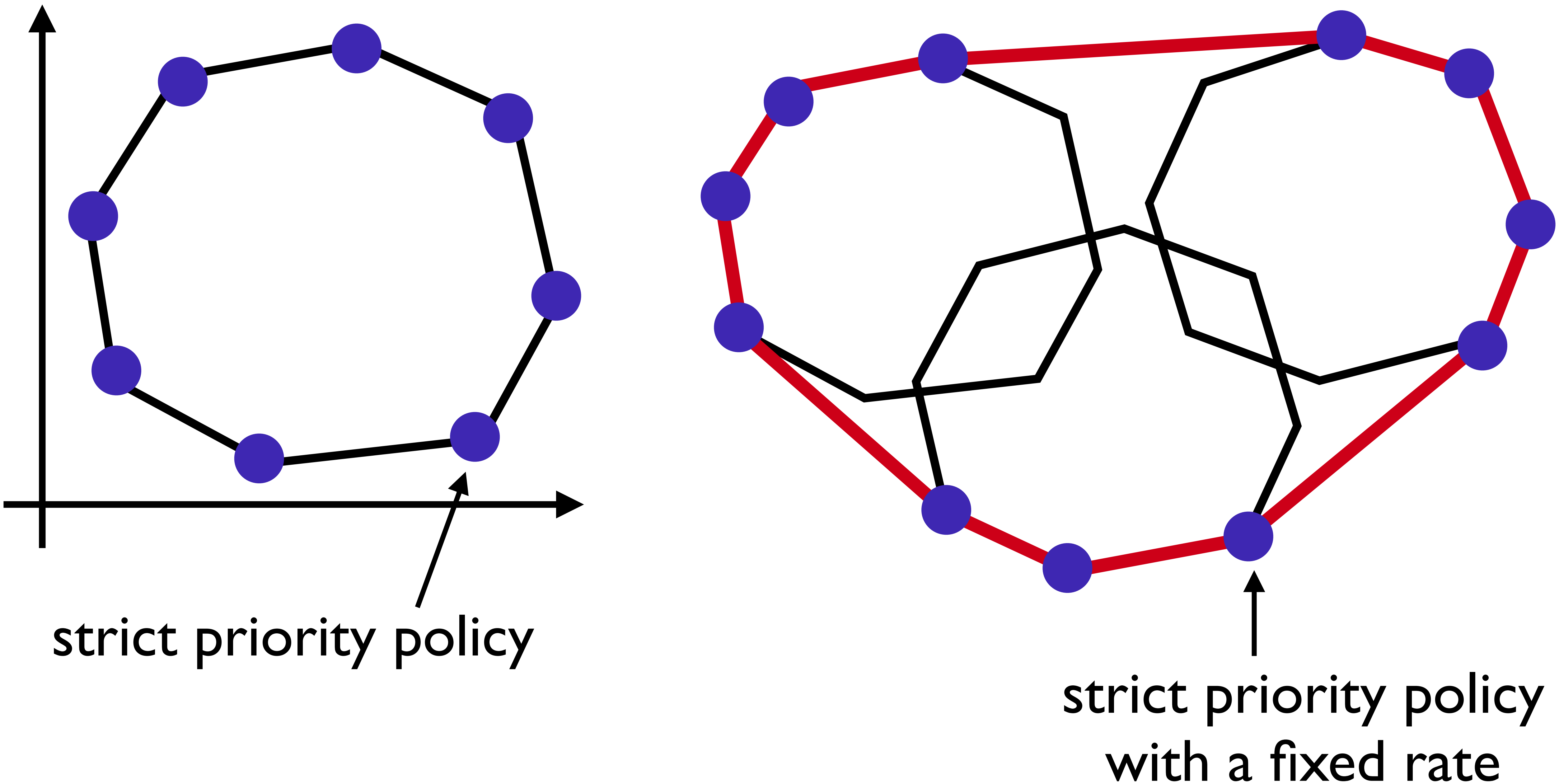
A new methodology



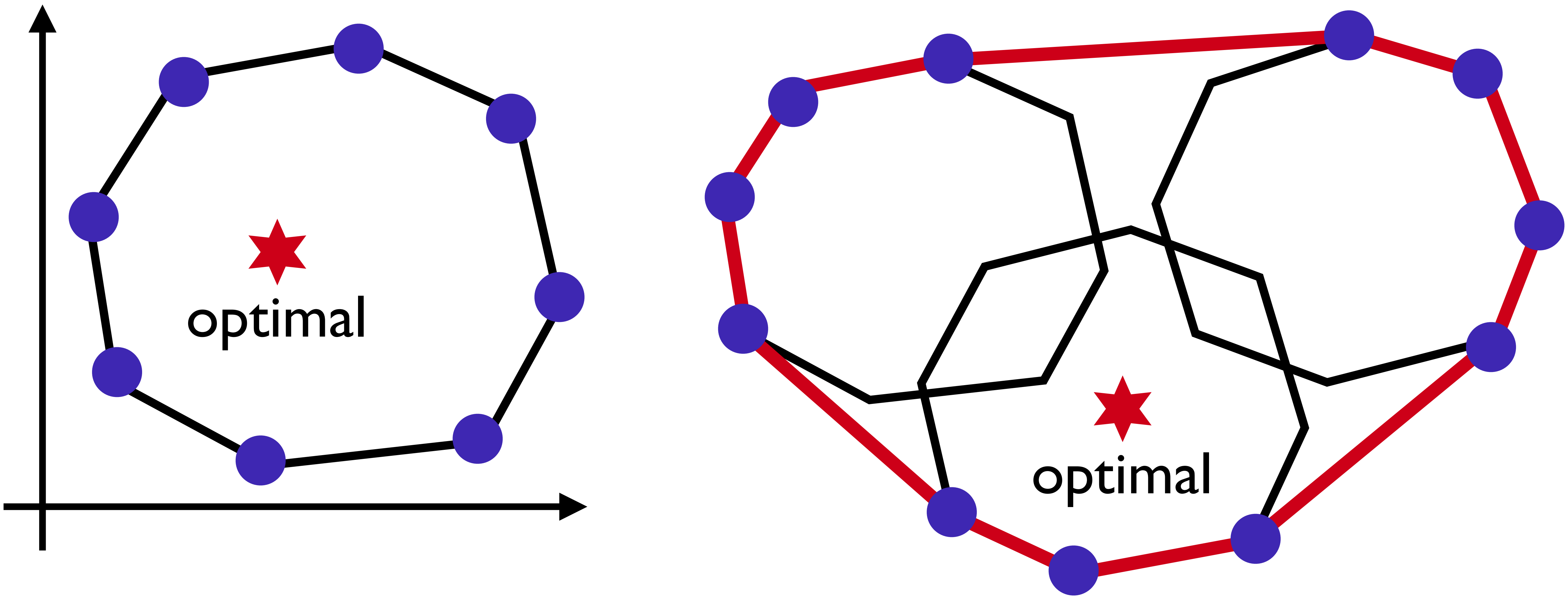
Performance region =
a set of inequalities

Performance region =
all convex combinations
of vertices

A new methodology



A new methodology



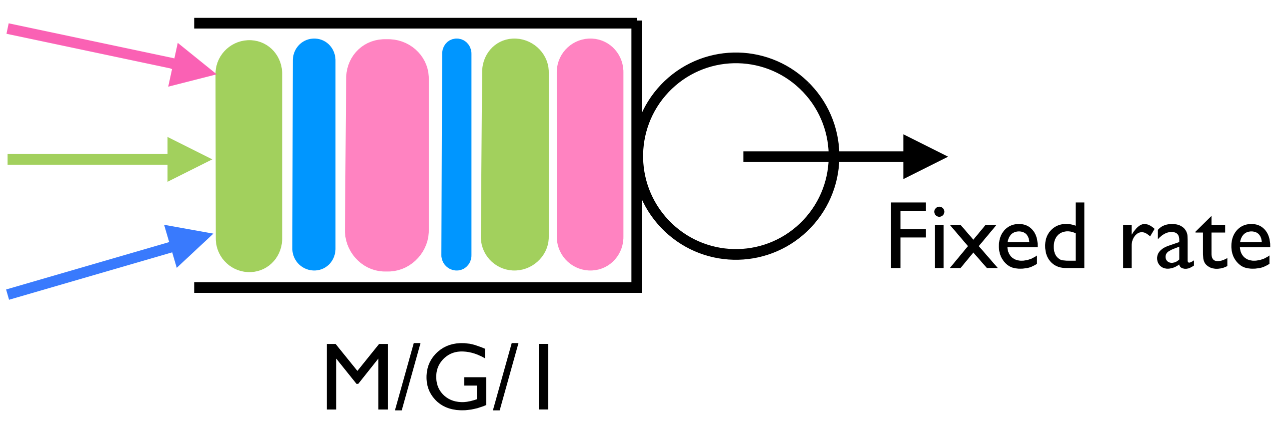
The optimal solution
= an optimal time-sharing of vertex-achieving policies

Providing delay guarantees

Minimize: 1

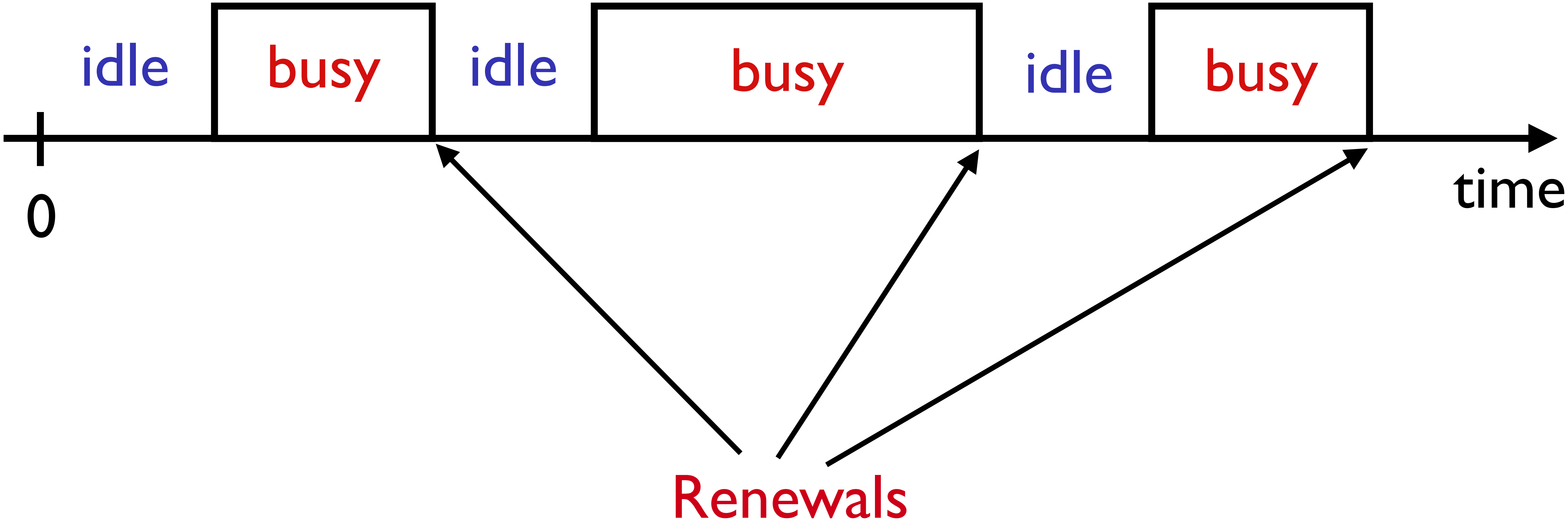
subject to: $\overline{W_n} \leq d_n, \; n \in \{1, \ldots, N\}$

$$(\overline{W_n})_{n=1}^{N} \in \Lambda$$



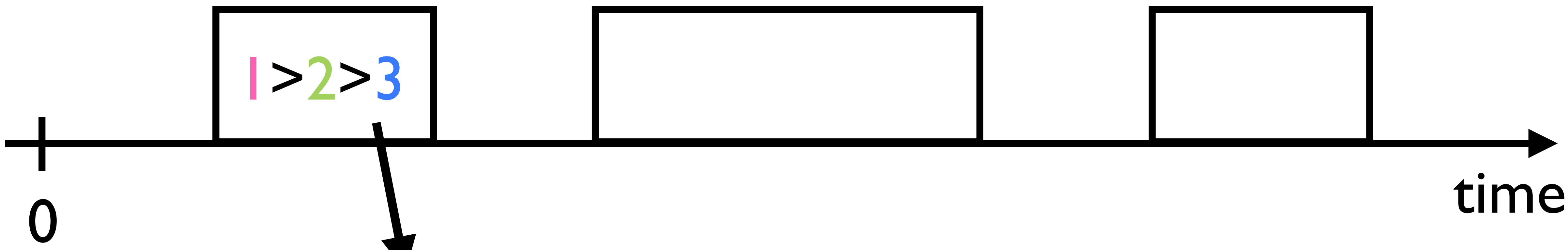
Providing delay guarantees

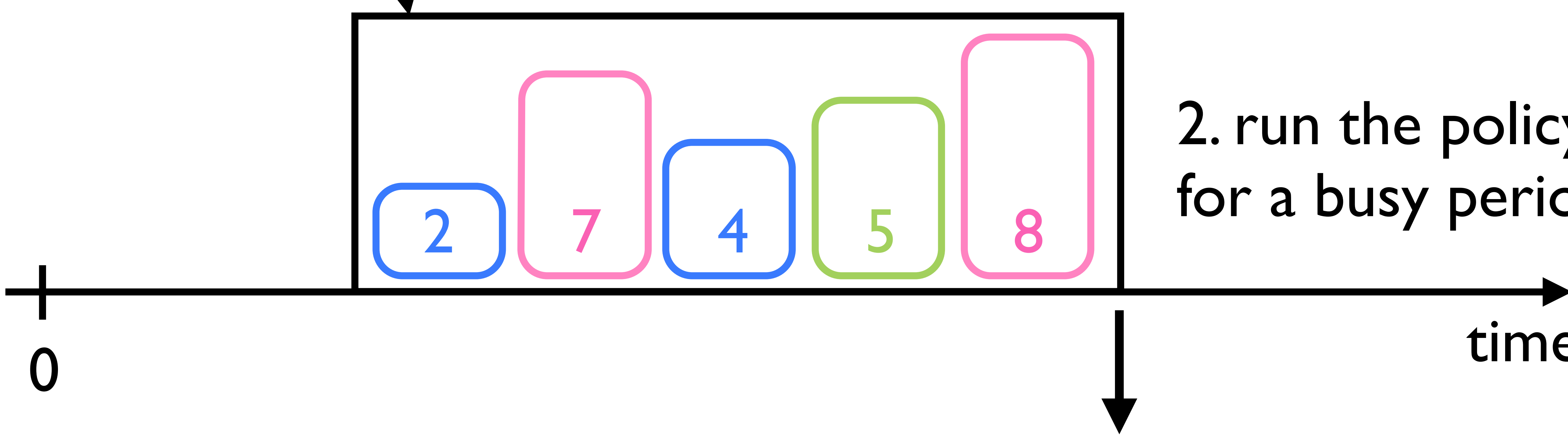
One priority policy per busy (renewal) period



Providing delay guarantees

1. initial priority ordering





2. run the policy for a busy period

3. compute the delay debt owed to each class and re-prioritize

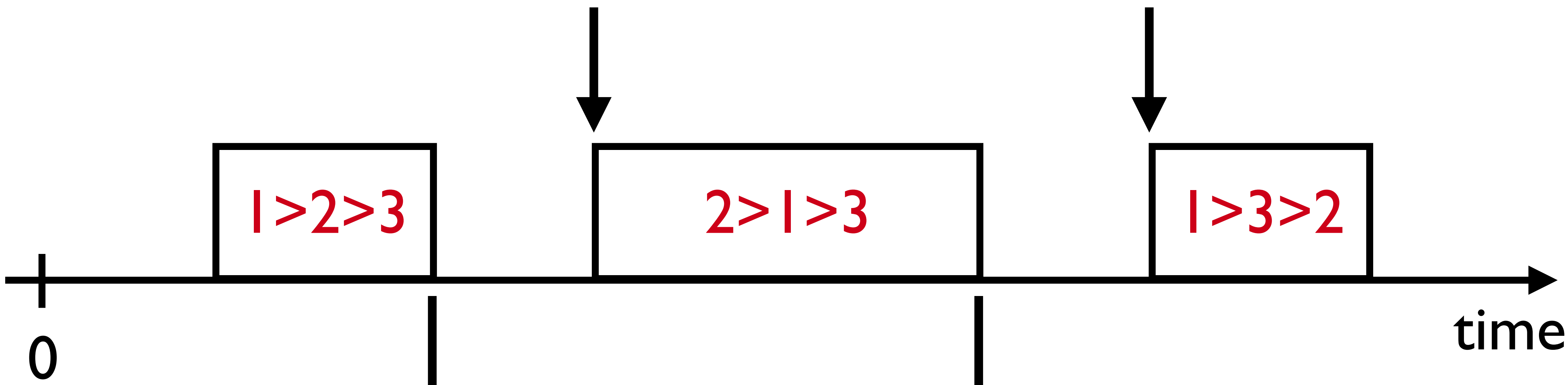
$$Z_1 \leftarrow \max\left[Z_1 + (8 - d_1) + (7 - d_1), 0\right]$$

$$Z_2 \leftarrow \max\left[Z_2 + (5 - d_2), 0\right]$$

$$Z_3 \leftarrow \max\left[Z_3 + (4 - d_3) + (2 - d_3), 0\right]$$

Providing delay guarantees

Re-prioritize job classes in the decreasing order of Z_n



$$Z_n \leftarrow \max\left[Z_n + \sum_{i \in \text{class n}} \left(w_n^{(i)} - d_n\right),\ 0\right]$$

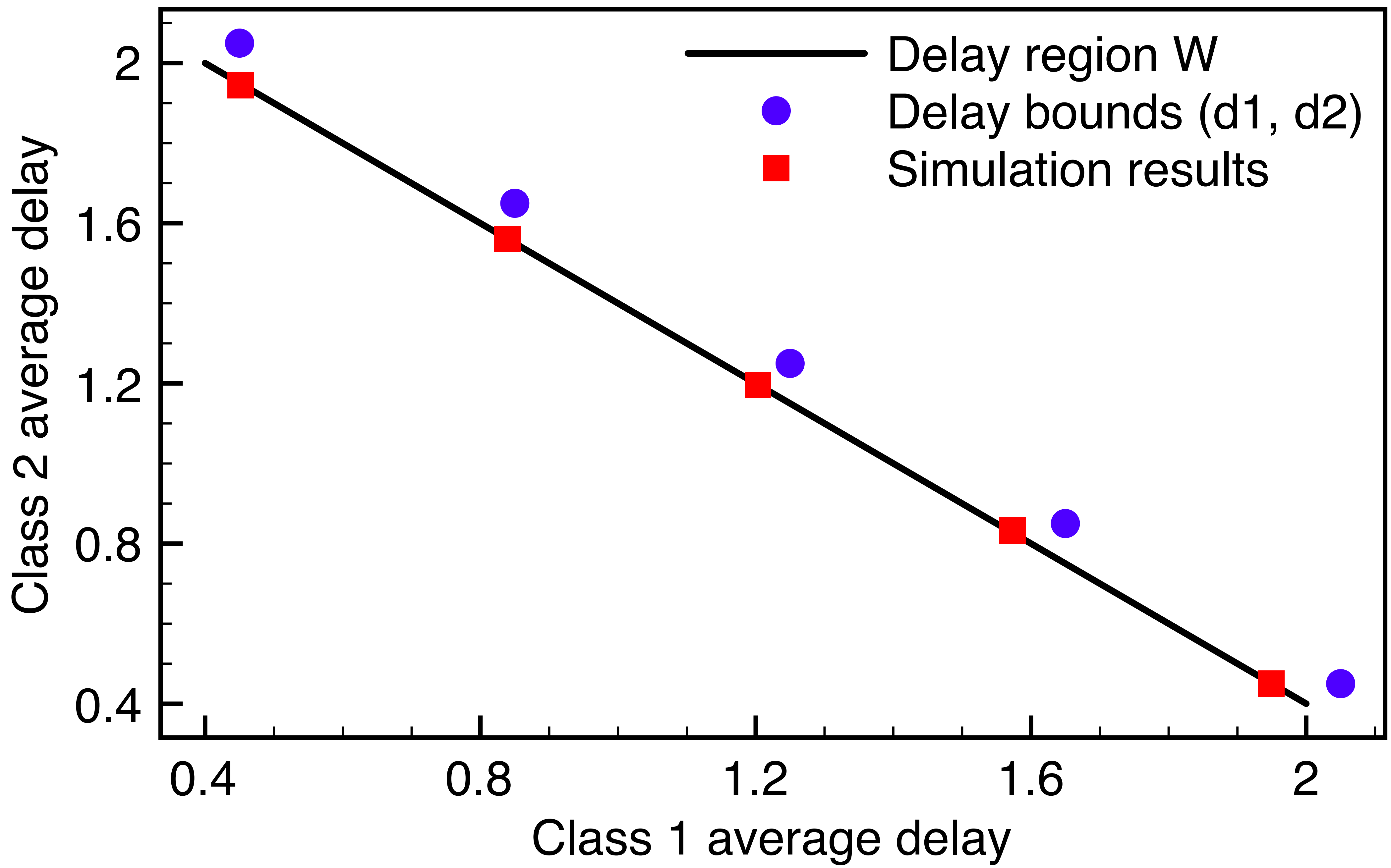
amount of delay exceeding the desired bound

No statistical knowledge, low complexity

Providing delay guarantees

Theorem:

For every feasible $\{d_1, d_2, \ldots, d_n\}$, we have $\overline{W_n} \leq d_n$ for all n.



Intuition

$$Z_n \leftarrow \max\left[Z_n + \sum_{i \in \text{class n}} (w_n^{(i)} - d_n),\ 0\right]$$

$$\text{class-n delay } w_n^{(i)} \longrightarrow \boxed{Z_n} \longrightarrow d_n$$

virtual queue

If queue Z_n is stable, then $\overline{W_n} \leq d_n$

The dynamic priority policy stabilizes all virtual queues

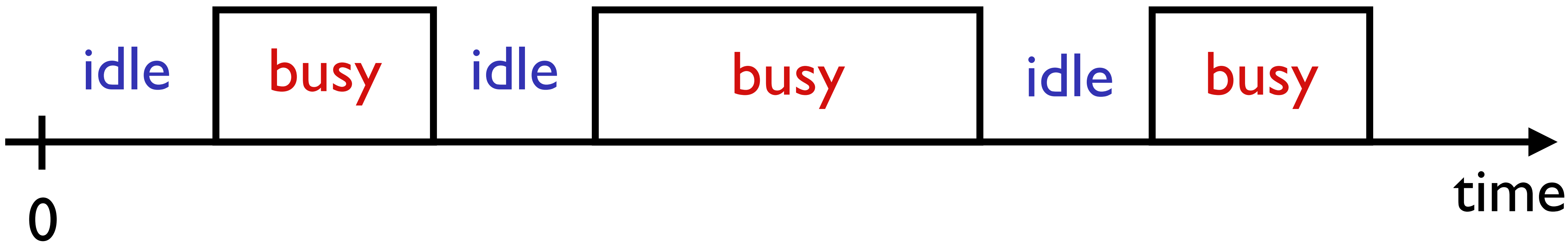
(a max-weight policy in this context)

Providing delay fairness

$$\text{Minimize: } \sum_{n=1}^{N} f_n(\overline{W_n})$$

$$\text{subject to: } \overline{W_n} \leq d_n, \ \forall n$$

Providing delay fairness



In every busy period,
prioritize job classes by sorting $\dfrac{Z_n + Y_n}{\overline{S}_n}$

Two virtual queues:

$$Z_n \leftarrow \max\left[Z_n + \sum_{i \in \text{class } n} (w_n^{(i)} - d_n),\ 0\right]$$

$$Y_n \leftarrow \max\left[Y_n + \sum_{i \in \text{class } n} (w_n^{(i)} - \gamma_n),\ 0\right]$$

γ_n is the solution to:

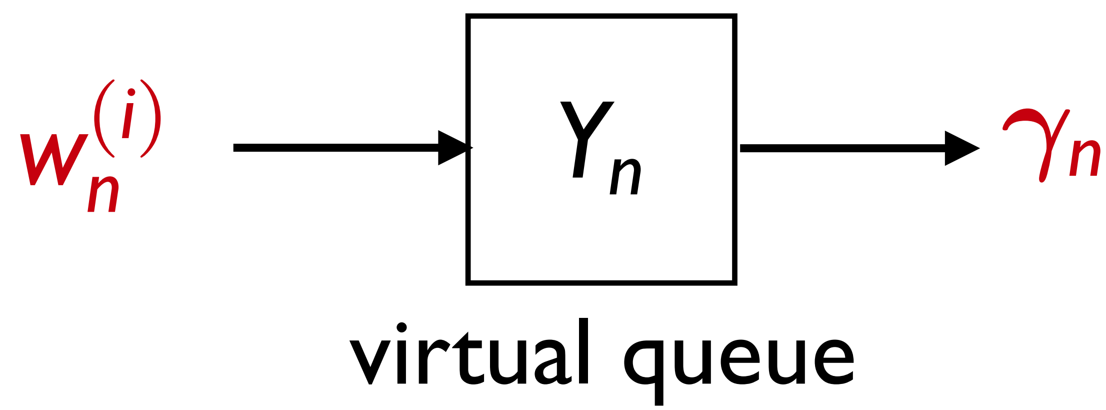
Minimize: $V f_n(\gamma_n) - Y_n \lambda_n \gamma_n$

subject to: $0 \leq \gamma_n \leq d_n$

Intuition

$$Z_n \leftarrow \max\left[Z_n + \sum_{i \in \text{class } n} (w_n^{(i)} - d_n),\ 0\right]$$

$$Y_n \leftarrow \max\left[Y_n + \sum_{i \in \text{class } n} (w_n^{(i)} - \gamma_n),\ 0\right]$$



If queue Z_n is stable, then $\overline{W_n} \leq d_n$

If queue Y_n is stable, then $\overline{W_n} \leq \overline{\gamma_n}$

$$\sum_{n=1}^{N} f_n(\overline{W_n}) \leq \sum_{n=1}^{N} f_n(\overline{\gamma_n})$$

Keep queues Z_n and Y_n stable, and minimize $\sum_{n=1}^{N} f_n(\overline{\gamma_n})$

Providing delay fairness

Theorem:

For every feasible $\{d_1, d_2, \ldots, d_n\}$, we have $\overline{W_n} \leq d_n$ for all n.

$$\text{optimal} \leq \sum_{n=1}^{N} f_n(\overline{W_n}) \leq \text{optimal} + \frac{B}{V}$$

V : a positive control parameter

Simulation

2-class M/M/1 queue:

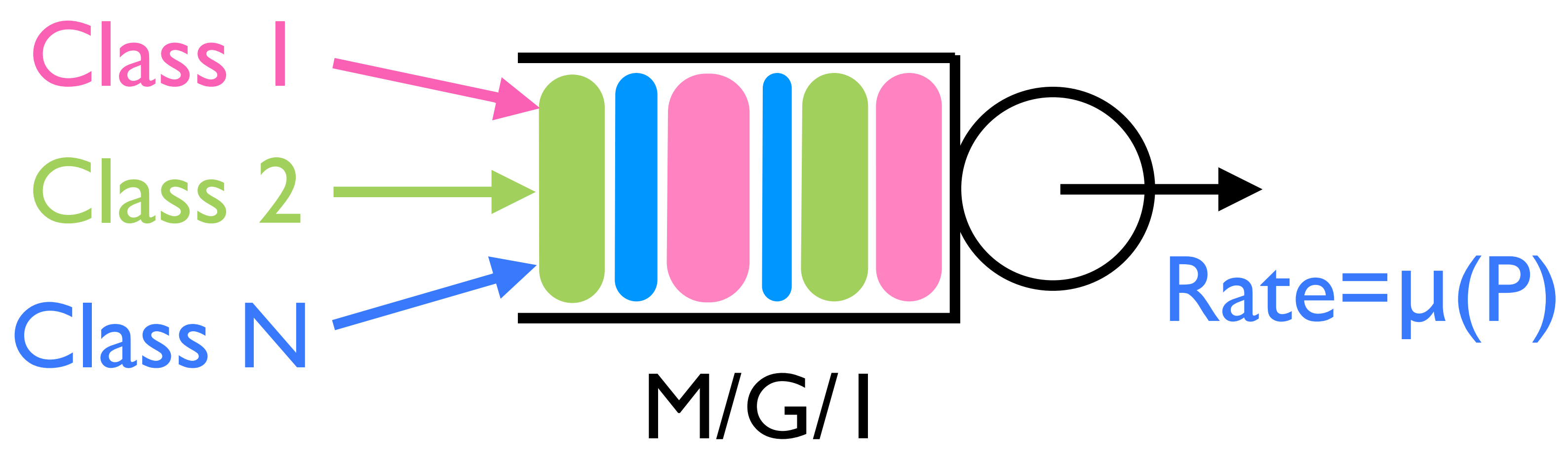
$$\text{Minimize: } \frac{1}{2}\left(\overline{W_1}\right)^2 + 2\left(\overline{W_2}\right)^2$$

$$\text{subject to: } \overline{W_1} + \overline{W_2} = 2.4$$

$$0.4 \leq \overline{W_1} \leq 1.95, \; 0.4 \leq \overline{W_2} \leq 1$$

V	$\overline{W}_1$	$\overline{W}_2$	Mean delay penalty
100	1.6607 (0.0055)	0.7424 (0.0052)	2.4814 (0.0239)
1000	1.7977 (0.0057)	0.5984 (0.0043)	2.3321 (0.0199)
2000	1.8339 (0.0056)	0.5639 (0.0053)	2.3176 (0.0217)
5000	1.8679 (0.0073)	0.5276 (0.0050)	2.3014 (0.0222)
Optimal value:	1.92	0.48	2.304

Min-cost dynamic rate control

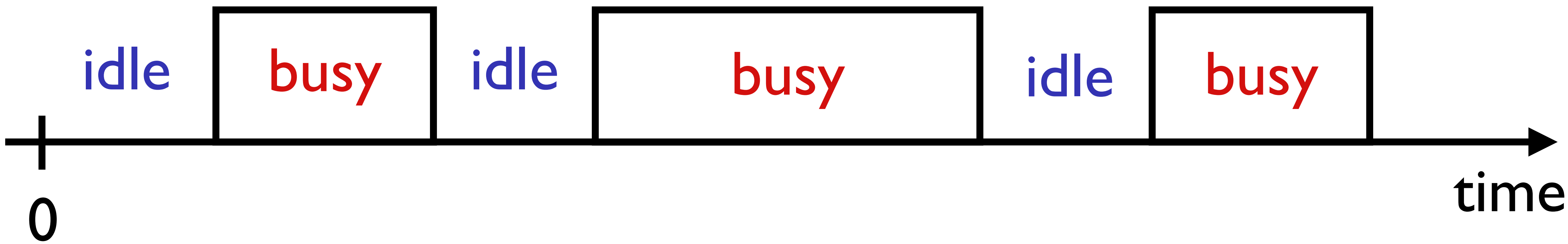


Minimize: P_{av}

subject to: $\overline{W_n} \leq d_n, \ \forall n$

$P(t) \in [P_{\min}, P_{\max}]$

Min-cost dynamic rate control



In every busy period,

Assign priorities by sorting $\frac{Z_n}{\overline{S_n}}$

$$Z_n \leftarrow \max\left[Z_n + \sum_{i \in \text{class } n} (w_n^{(i)} - d_n),\ 0\right]$$

Spend cost P minimizing $\displaystyle c\,V\,\frac{P}{\mu(P)} + \sum_{n=1}^{N} Z_n\,\lambda_n\,\overline{W_n}(P)$

Min-cost dynamic rate control

Theorem:

For every feasible $\{d_1, d_2, \ldots, d_n\}$, we have $\overline{W_n} \leq d_n$ for all n.

$$\text{optimal} \leq P_{\text{av}} \leq \text{optimal} + \frac{B}{V}$$

V : a positive control parameter

Min-cost dynamic rate control

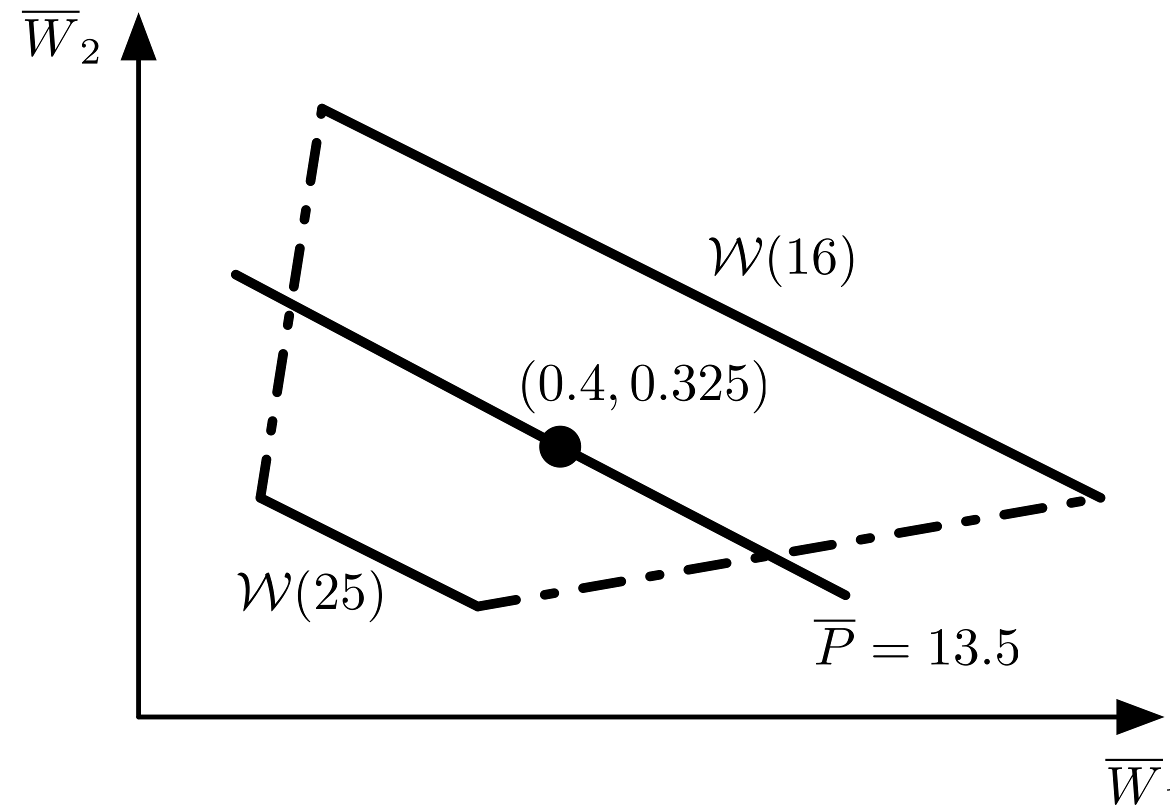
2-class M/G/1 queue

$\mu(P) = \sqrt{P}$, $P(t) \in \{16, 25\}$

Minimize: $\overline{P}$

subject to: $\overline{W_1} \leq 0.4$

$\overline{W_2} \leq 0.325$



V	$\overline{W}_1^{\text{DynRate}}$	$\overline{W}_2^{\text{DynRate}}$	Average cost
1	0.3562 (0.00078)	0.3029 (0.00032)	13.8018 (0.01806)
10	0.3984 (0.00022)	0.3247 (0.00005)	13.5101 (0.02626)
100	0.4003 (0.00013)	0.3252 (0.00010)	13.5044 (0.02197)
Optimal value:	0.4	0.325	13.5

Dynamic index policies

	Delay guarantees	Delay Fairness	Rate control
Index	Z_n	$(Z_n + Y_n)/\overline{S_n}$	$Z_n/\overline{S_n}$
Statistical knowledge	none	$\lambda_n,\ \overline{S_n}$	$\lambda_n,\ \overline{S_n},\ \overline{W_n}(P)$

$$Z_n \leftarrow \max\left[Z_n + \sum_{i \in \text{class } n} (w_n^{(i)} - d_n),\ 0\right]$$

$$Y_n \leftarrow \max\left[Y_n + \sum_{i \in \text{class } n} (w_n^{(i)} - \gamma_n),\ 0\right]$$

Summary

Convex programs over a polymatroid can be solved by online adaptive policies as simple as the $c\mu$ rule

These policies require limited or no statistics by learning from the past

Caveat: large delay variance when queue is heavily loaded (controls are updated over busy periods)

Applications to other queueing systems with polymatroid-type performance regions?

Job-level scheduling using imperfect virtual-queue length information to reduce delay variance?