POWER & THROUGHPUT ISSUES IN NEXT-GENERATION PACKET SWITCHES

Nick Bambos

Stanford University

The Rising Problem of Power (Density)

Switching: chips ~100W+ … systems ~ KW … clusters/farms/centers ~ 10KW++

Computing: processors ~100W+/- … systems ~200W+

Racks: ~ 512 CPUs+ … … … ~100KW+ … *high power density* … *liquid cooled!*

Data centers: 100,000 CPUs+ … … … ~10MW++ … high power density

High power distribution cost… high power bills… *high cooling costs…*

Electronic infrastructure costs get amortized … power costs don't…

Both matter: power *volume* and power *density* (perhaps more)

High/uneven power density … high thermal stress … fast component aging …

reliability problems…

Throughput/Speed vs. Power

Throughput/speed can *potentially…* be adjusted by

frequency scaling… clocking up/down

voltage scaling…

component shutdown (path/structure scaling)

…

Power scales up ***super-linearly*** with throughput/speed

Where/How to Attack the Problem

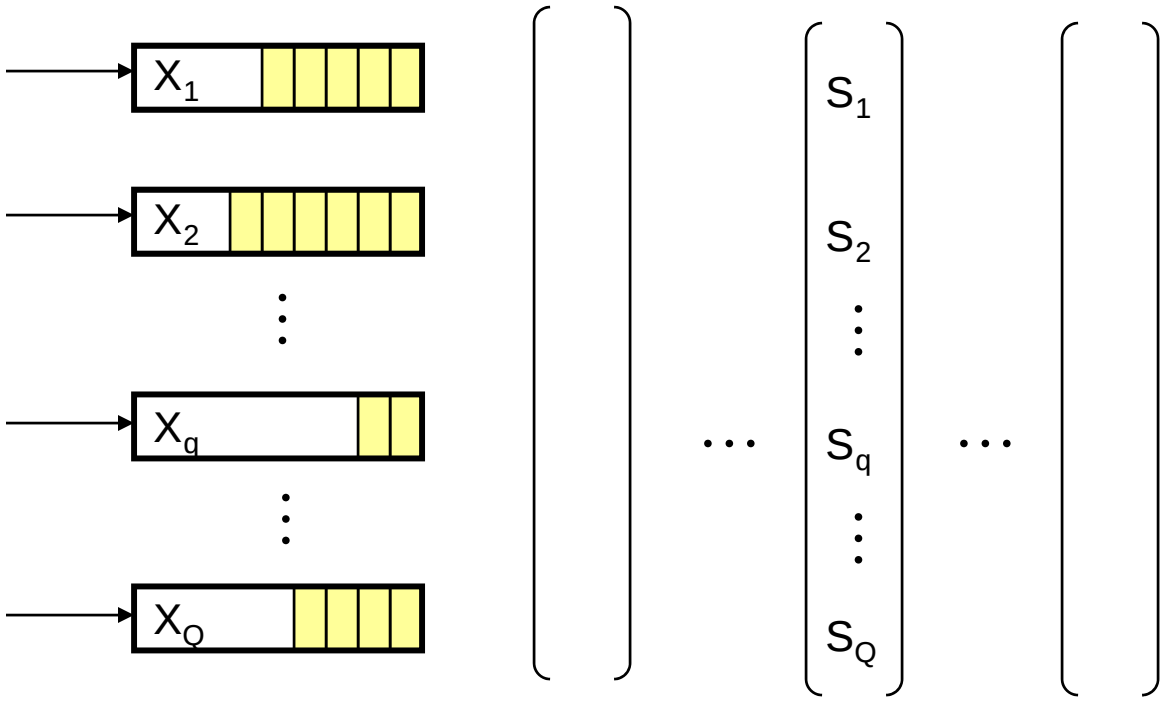
Level 3: algorithms and operations

Level 2: system architecture

Level 1: low-power circuit design

The Basic Model …

A Canonical Model of Interacting Resource Allocation



X = backlog vector

S = service vector/mode/configuration/allocation

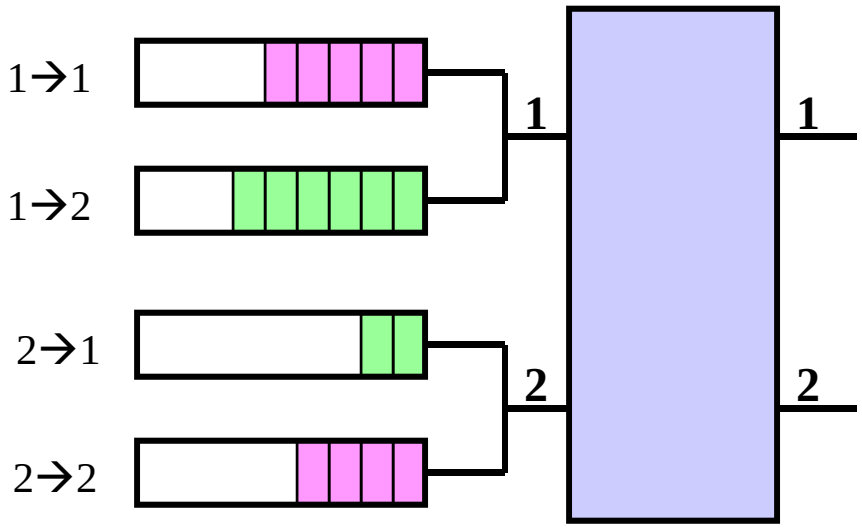
S = set of all possible service vectors

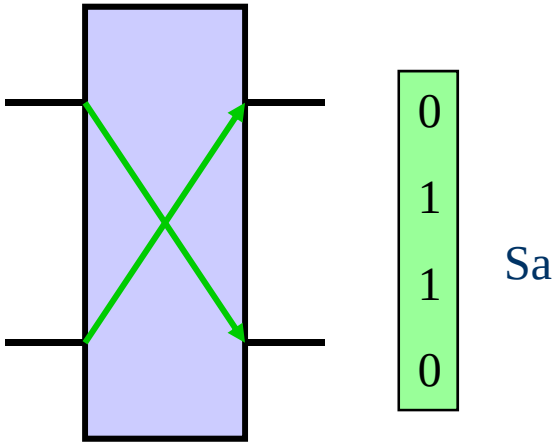
P_S = cost (power…) of service configuration S

Core Issue… dynamically choose S (based on backlog/service history, etc.) to maximize throughput, minimize (power/delay) cost…

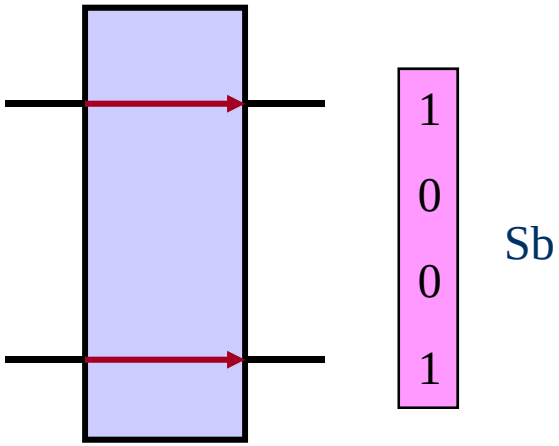
Input Queued Packet Switches …

2x2 switch … simplest model (…not simplistic)… scales to NxN



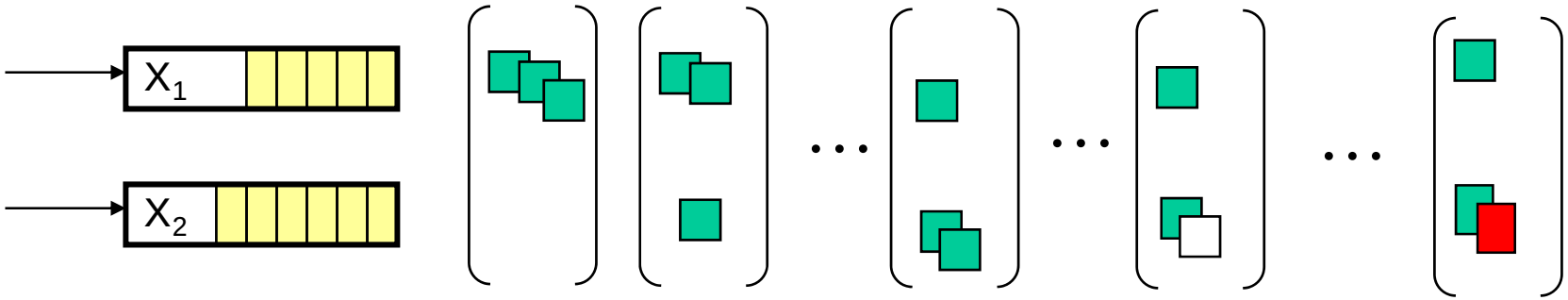


Service vectors Sa and Sb



Multiprocessors ...

Allocating 3 processors to two job queues



fast

normal

slow

... in general ... *any set* of service vectors

Throughput …

Load and Throughput...

Arbitrary traffic traces

Load $\rho = (\rho_1, \rho_2, \ldots, \rho_q, \ldots \rho_Q)$ ... long term avg. packet load

$$\rho_q = \frac{\text{packet load arriving to queue q in (0,t)}}{t} \quad \ldots \text{ as t grows large}$$

Throughput .... job departure rate = job arrival rate

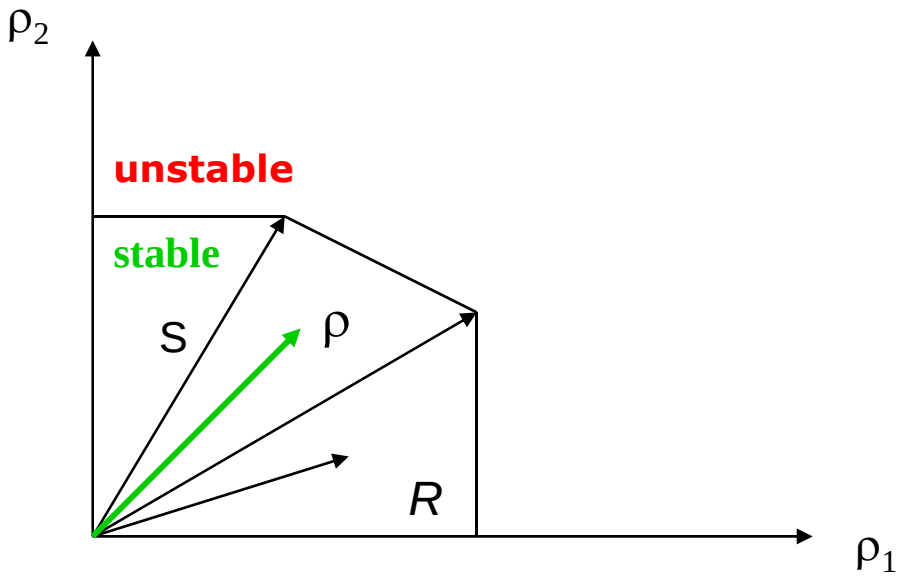
... in/out flow balance

... **flow conservation**

... stability

$$\frac{X(t)}{t} \longrightarrow 0 \quad \ldots \text{ at large times t}$$

Throughput ... Capacity Region



$$R = \{ \rho : \quad \rho \leq \sum_{S \in S} \phi_S S \ldots \quad \text{for some } \phi_S > 0 \text{ with } \sum \phi_S = 1 \}$$

Projective Cone Schedules (PCS) Maximize Throughput

PCS algorithm... when backlog X, choose S to maximize projection on **B**X

$$\max \langle S, \mathbf{B}X \rangle \quad \text{over } S \text{ in } \mathcal{S}$$

maximizes throughput

for ***any*** fixed matrix **B** that is

positive-definite, **symmetric** and has

negative/zero off-diagonal elements

MWM algorithm ... PCS with B=I

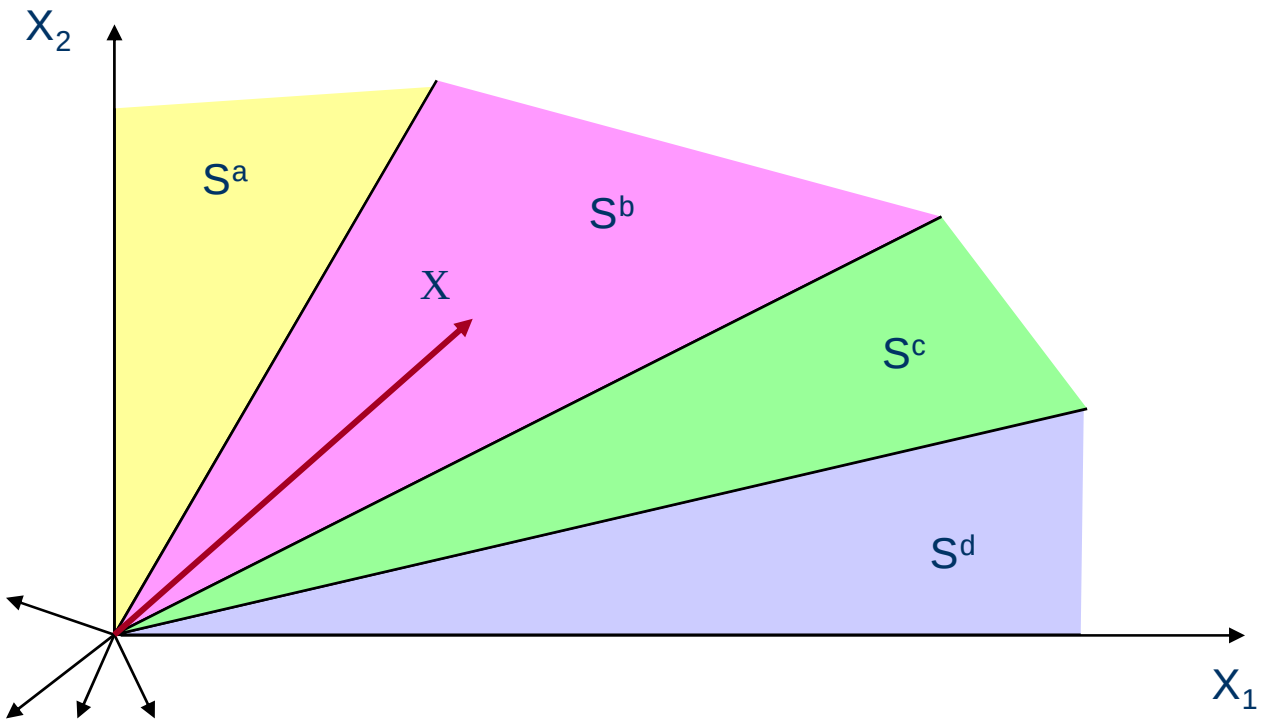
Rich family of schedules... (~ Q^2 matrix parameters to tweak and tune)

Extremely robust schedules

Interesting geometric intuition

Geometry of Projective Cone Schedules (PCS)

Given X, choose S to maximize <S,**B**X> over all S in *S*



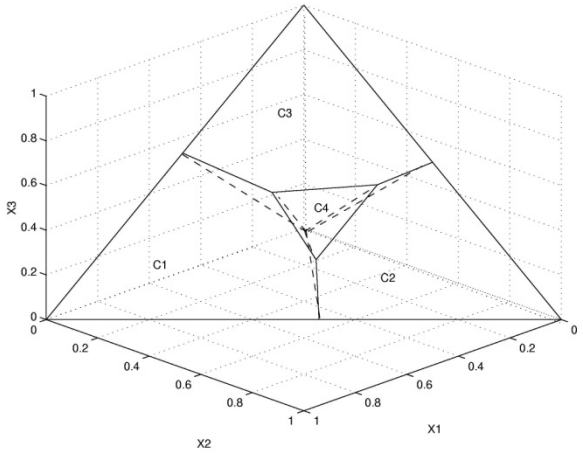
When X in cone C,

choose S = S(C) corresponding to that cone

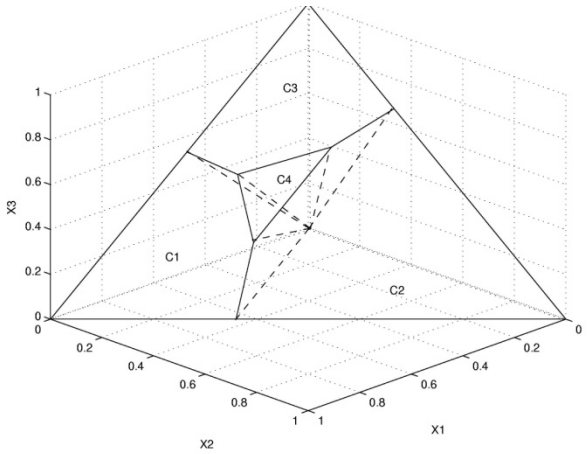
A 3-Queue Example

S1=(9,0,0) / S2=(0,8,0) / S3=(0,0,8) / S4=(3,4,3)

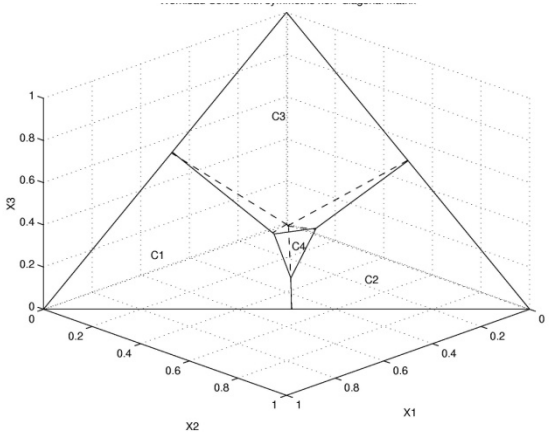
B=[1,0,0; 0,1,0; 0,0,1]



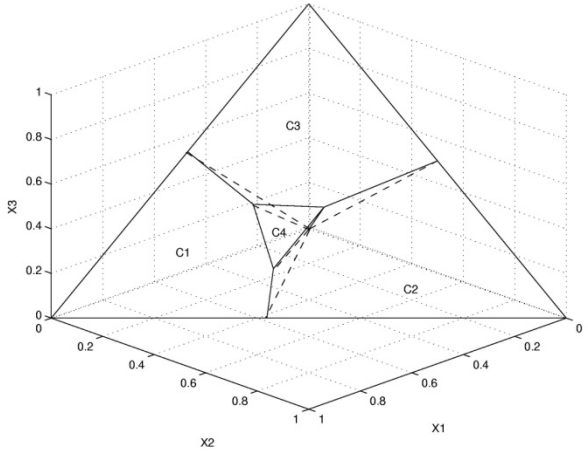
B=[1,0,0; 0,2,0; 0,0,1]



B=[1,-0.5,0; -0.5,1,0; 0,0,1]

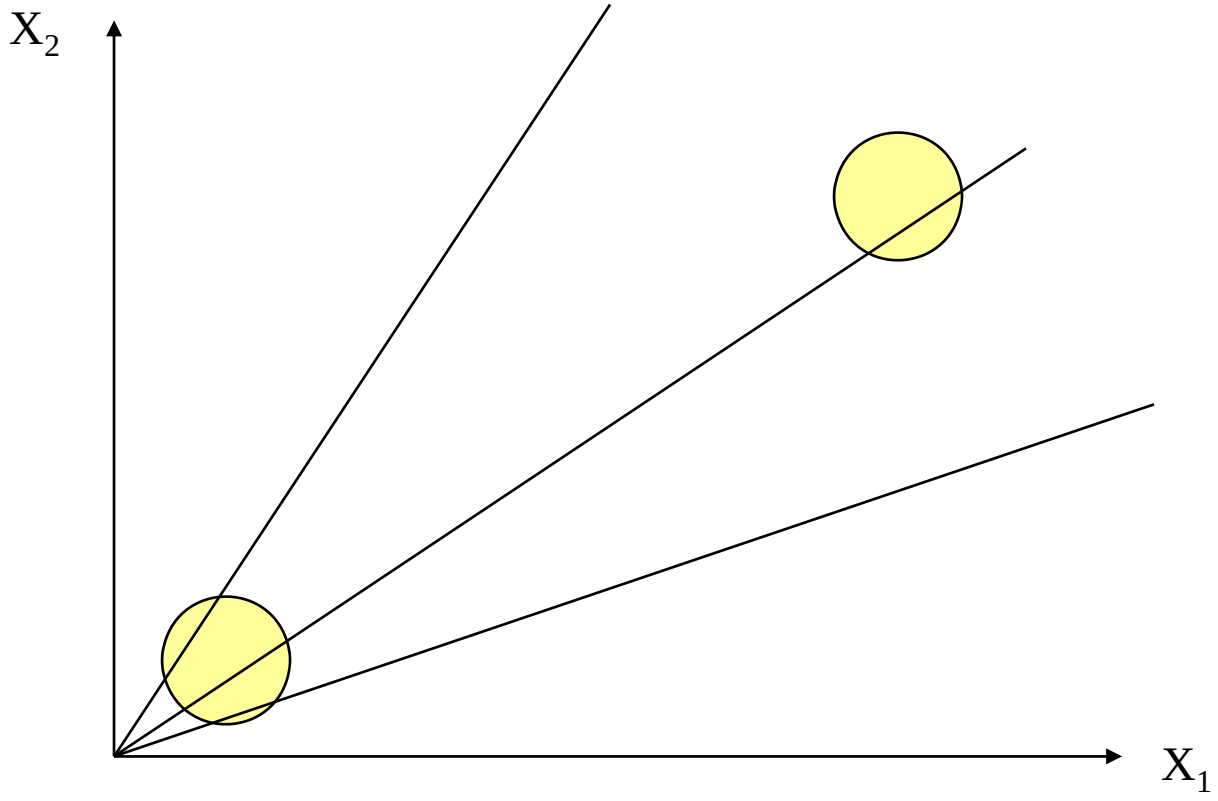


B=[1,-0.5,0; 0,1,0; 0,0,1]



Scalable Projective Schedules – Local PCS

Assume bound on “workload displacement/jump”

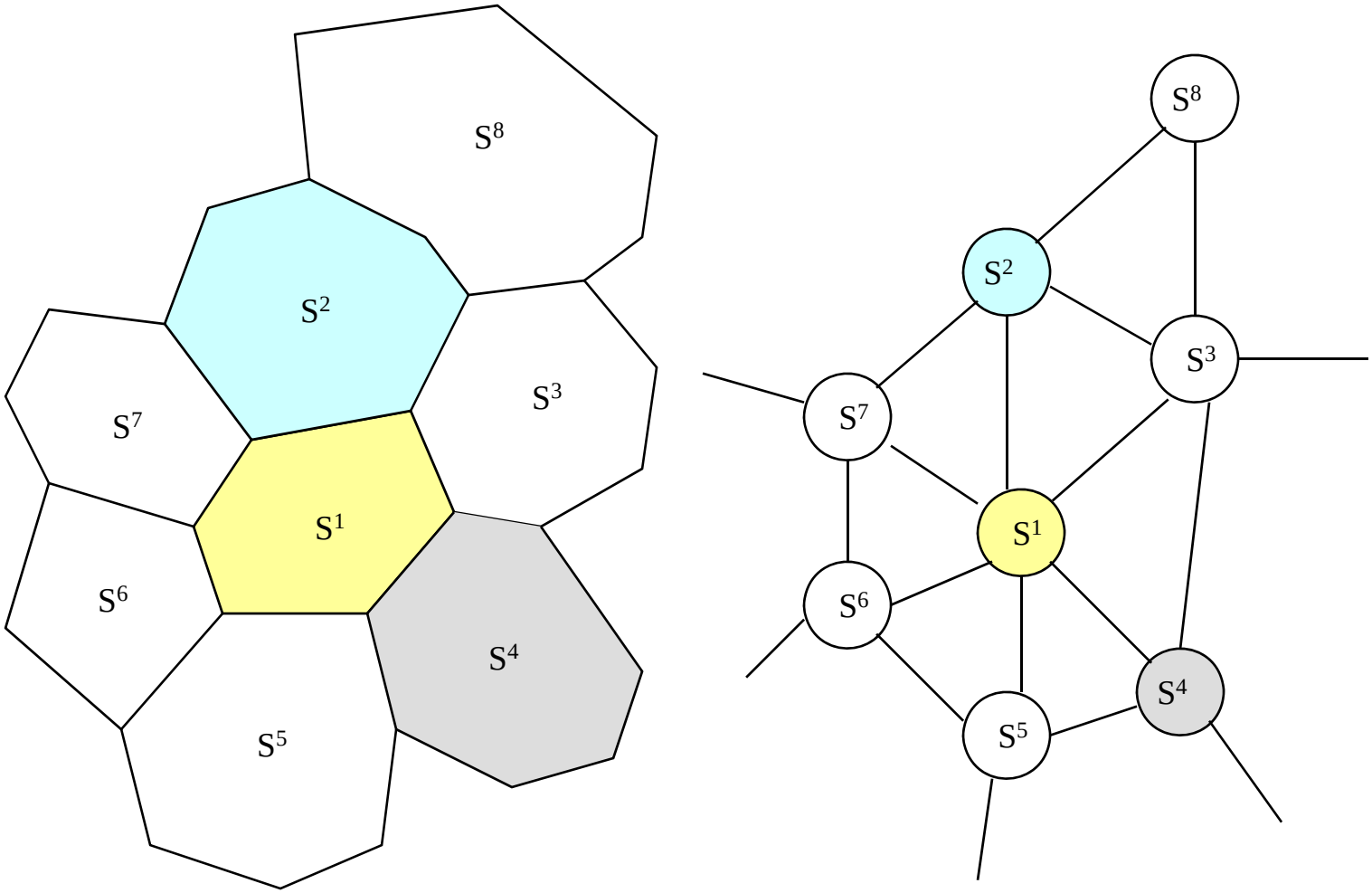


Have to search only neighbor cones … fewer as workload grows! … **Local Search**

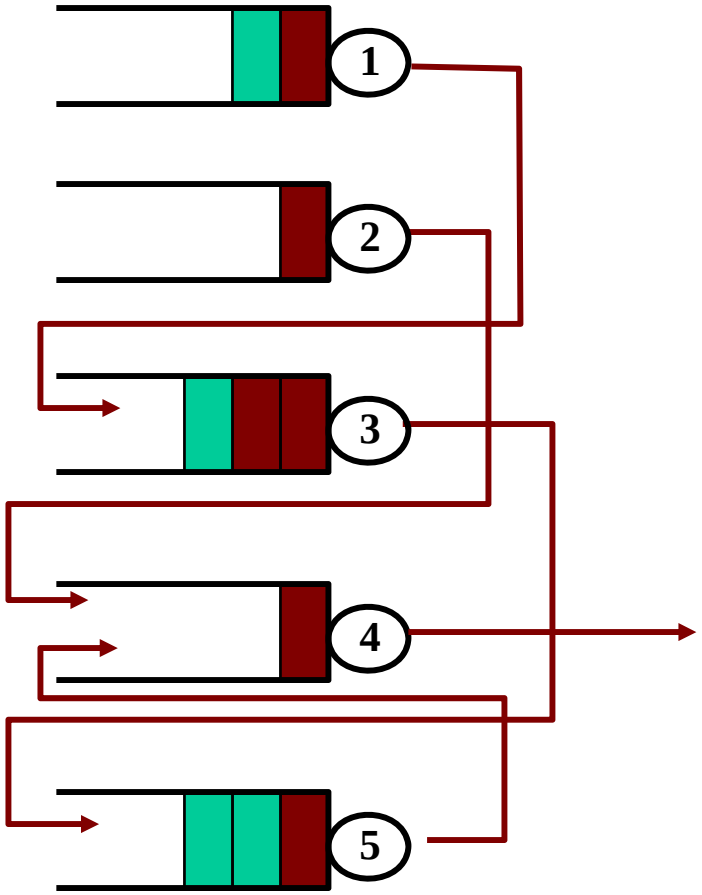
Graph Representation of PCS ... Local Search Idea ...

When at S, check out <S,**B**X> on **neighbor nodes** (only) ...

... chose S' with maximal <S,**B**X> ... **steepest ascent**



Networks ... and PCS

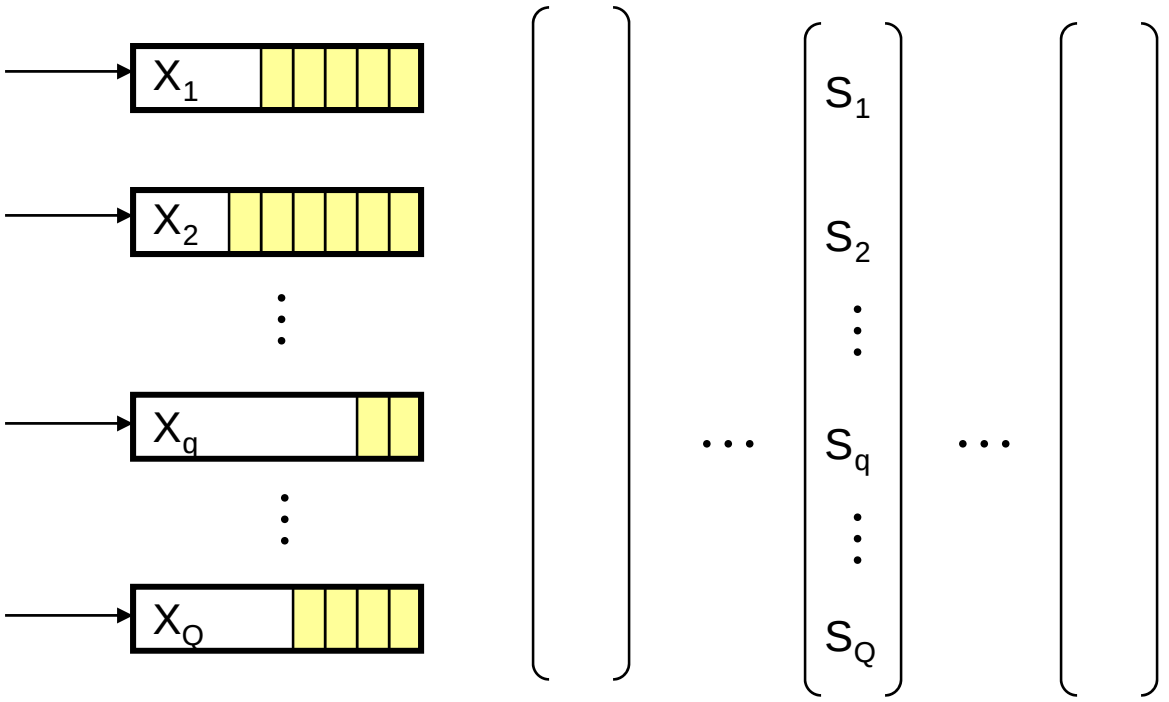


$$\begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 1 \\ 2 \\ 2 \end{pmatrix}$$

Service = Departures - Arrivals

Power ...

The Canonical Model Again… Power vs. Delay



X = backlog vector

S = service vector/mode/configuration/allocation

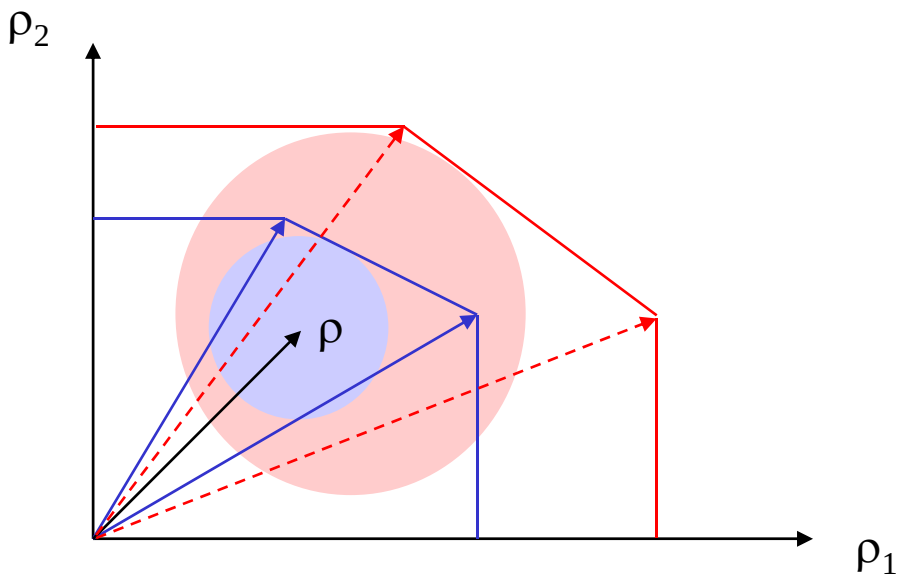
S = set of all possible service vectors

P_S = cost (power…) of service configuration S

Core Issue… dynamically choose S to **minimize power + delay cost**

dynamic programming formulation … intractable…

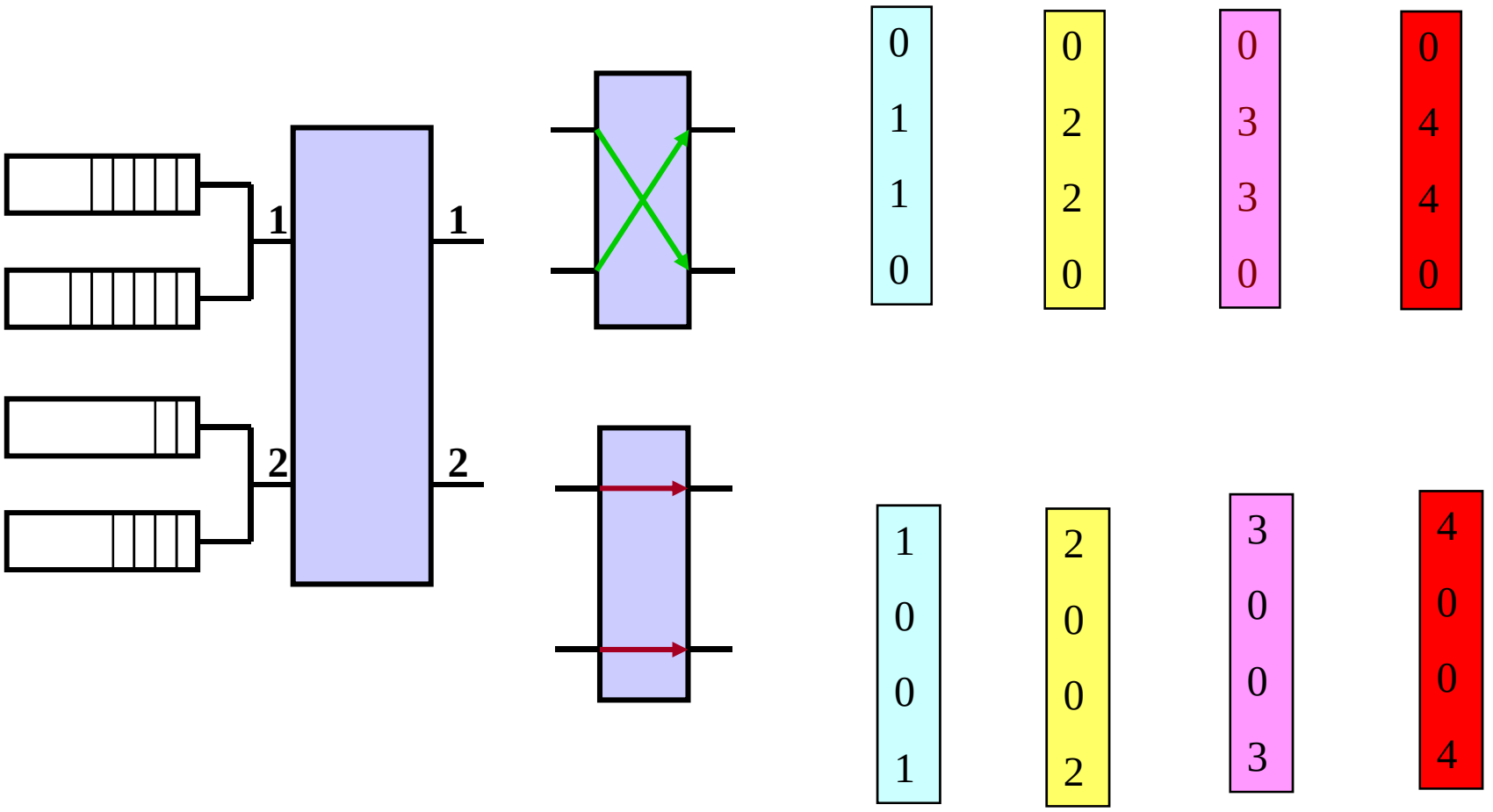
Control and Optimization ... Preview



Activating faster/slower, higher/lower-power, more/less expensive service vectors

... adjusting the capacity space

Packet Switches ... with Deceleration Modes



Speed Mode m: slower/cooler ... faster/hotter

The Control Problem – Ricatti Recursion

Service Vector **S** = Speed Mode **m** x Port-Matching Configuration **C**

In each slot, have to choose **m** & **C** … **S=mC**

Backlog Cost… X'**E**X …per slot … **E** pos-def & sym

Power Cost … S'**F**S …per slot … **F** pos-def & sym

No arrivals…

$J(X) = \min[\, X'\mathbf{E}X + S'\mathbf{F}S - J(X - S/\text{depts}) \,]$ min over S

Continuous Relaxation … **Linear Quadratic Regulator (LQR)**… **Ricatti Equation**

$\mathbf{S^*(X) = [\, (F+P)^{-1} P \,] X}$

Where **P** solves $\mathbf{E = P\,(F+P)^{-1}\,P}$

Power-Aware MWM... PA-MWM

Reduced Backlog Cost... $X'\mathbf{E}X$ $\sim b \, \mathrm{Sum}[X_q^2]$

Reduced Power Cost ... $S'\mathbf{F}S$ $\sim f \, [\mathrm{Sum} \, S_q]^2$

PA-MWM algorithm

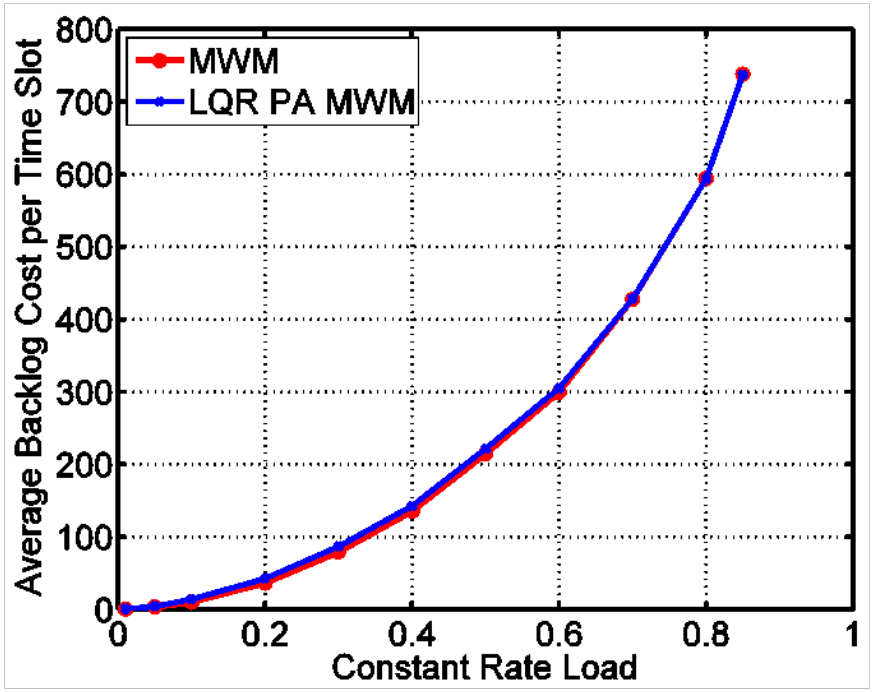
alignment

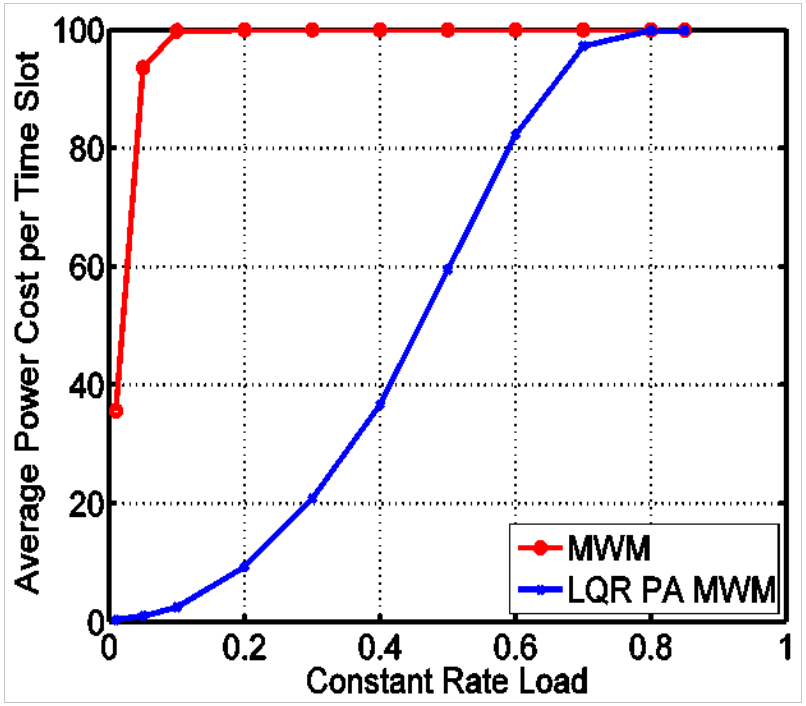
Choose Configuration **$C^*(X) = \mathrm{argmax} \langle C,X \rangle$**

Choose Speed Mode **$m^*(X) = \mathrm{argmin} \, |m - a \, \mathrm{TotalBacklog}|$**

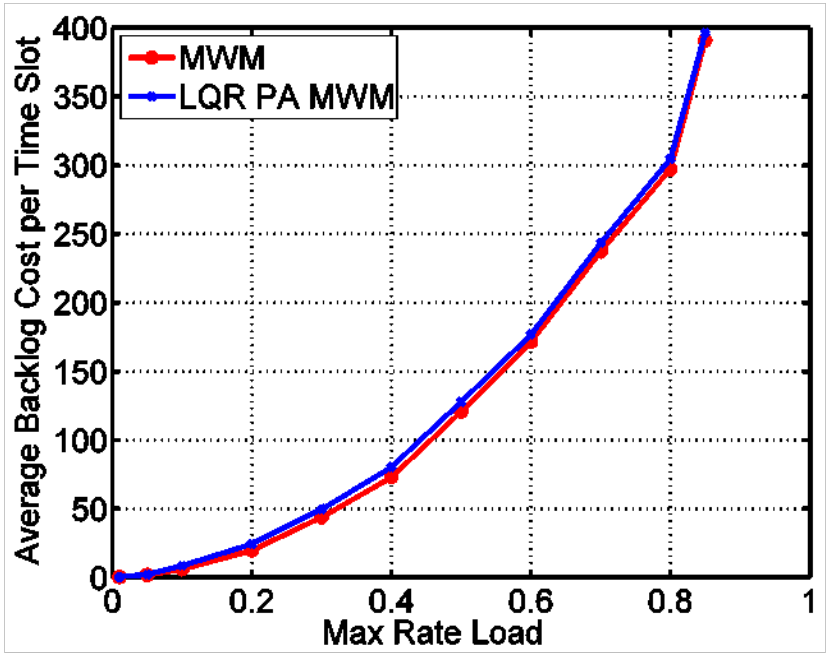
congestion

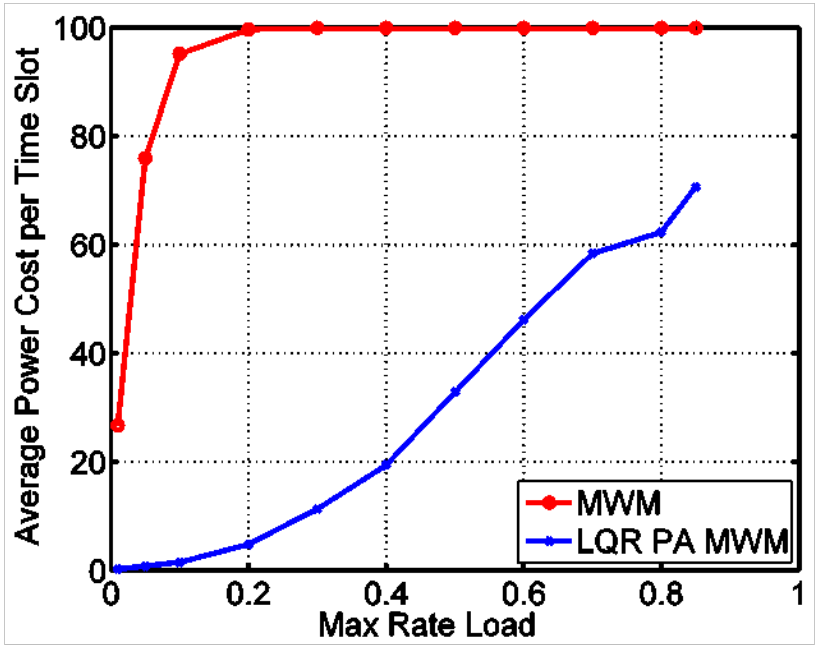
PA-MWM I


MWM
LQR PA MWM
Average Backlog Cost per Time Slot
Constant Rate Load
0
100
200
300
400
500
600
700
800
0.2
0.4
0.6
0.8
1

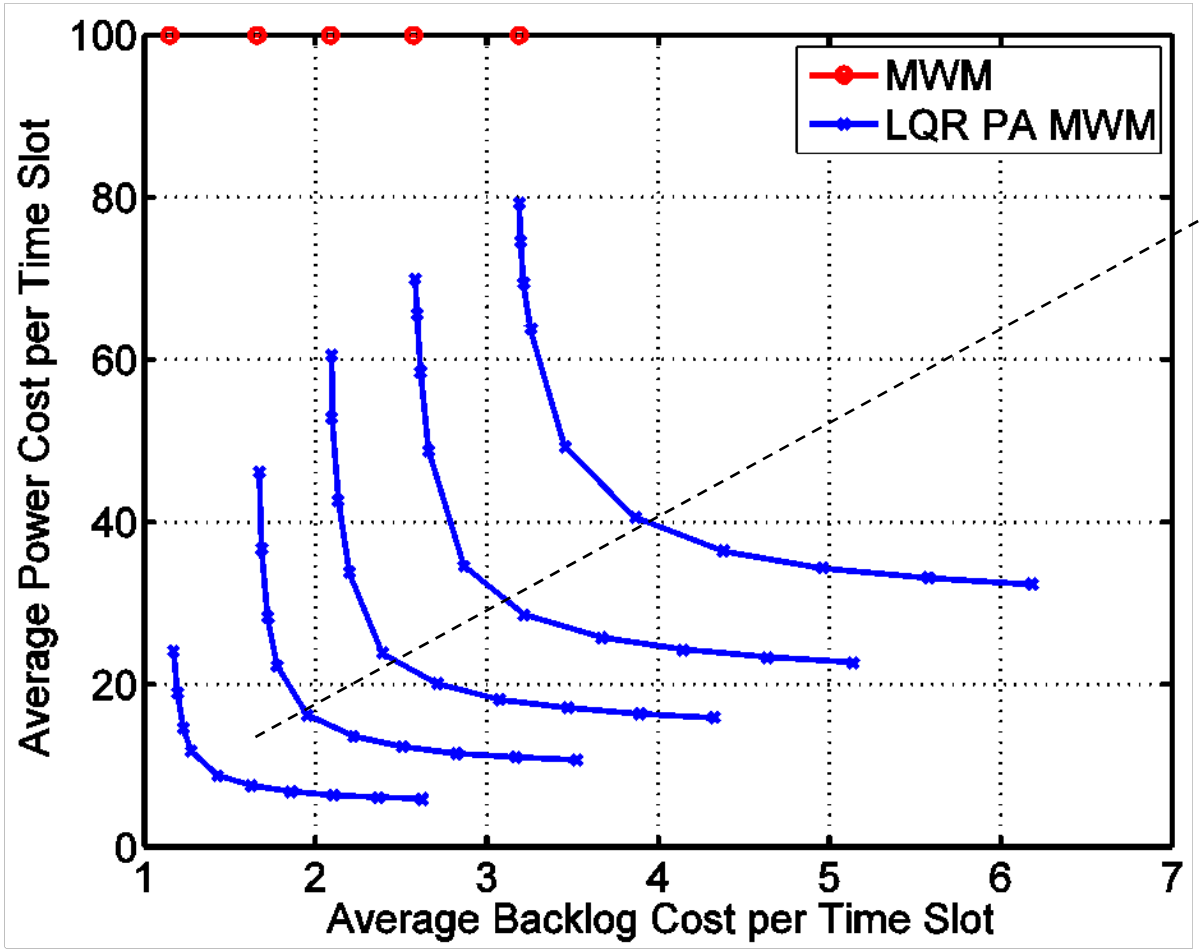

MWM
LQR PA MWM
Average Power Cost per Time Slot
Constant Rate Load
0
20
40
60
80
100
0.2
0.4
0.6
0.8
1

PA-MWM II


MWM
LQR PA MWM
Average Backlog Cost per Time Slot
Max Rate Load
0
50
100
150
200
250
300
350
400
0
0.2
0.4
0.6
0.8
1


MWM
LQR PA MWM
Average Power Cost per Time Slot
Max Rate Load
0
20
40
60
80
100
0
0.2
0.4
0.6
0.8
1

PA-MWM II



Higher Load (Packet Arrival Rate)

Total Cost = BacklogCost + a PowerCost … vary ‘a’

Thank You!