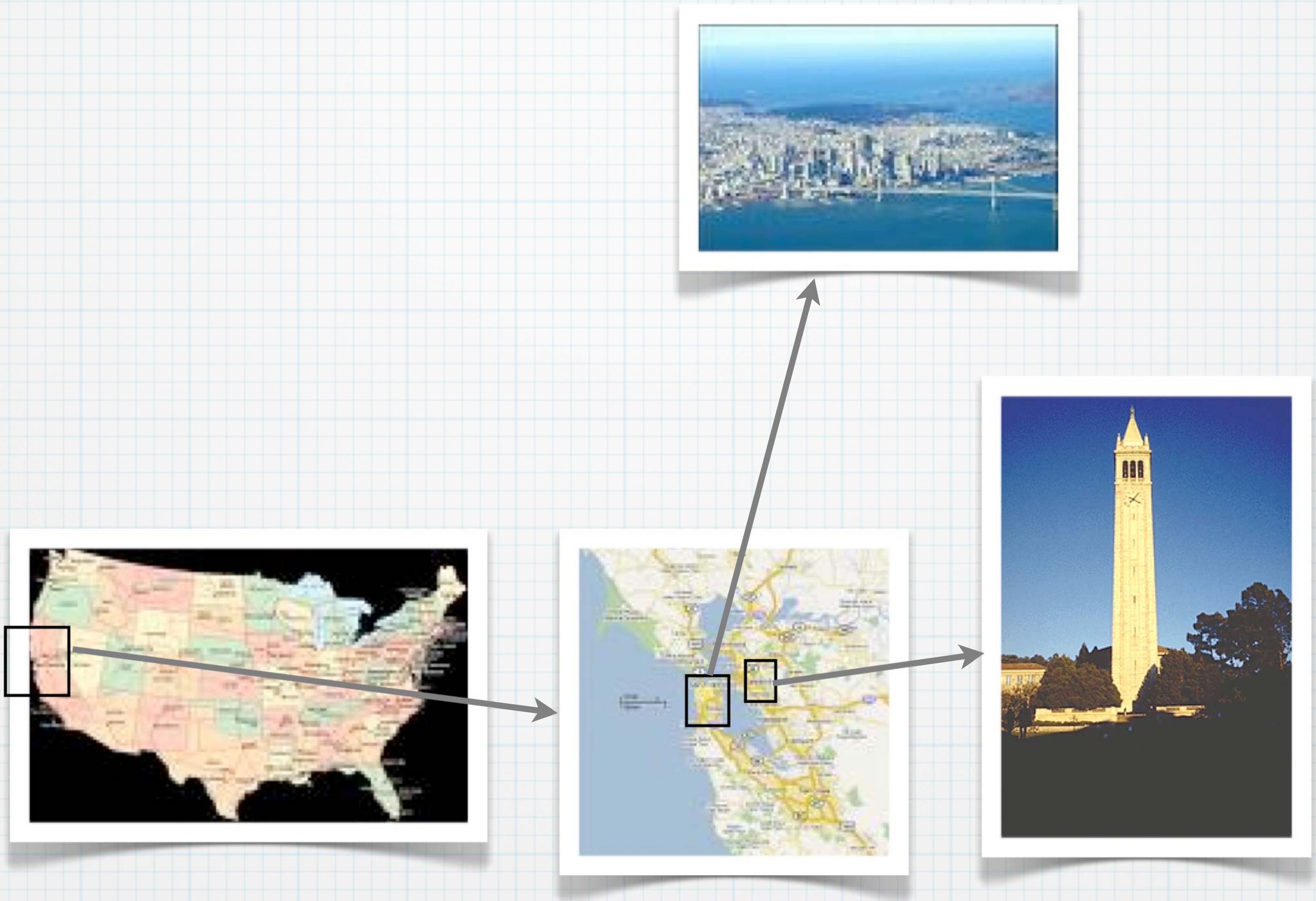
Distributed Scheduling in Communication and Processing Networks

Jean Walrand

Department of Electrical Engineering and Computer Sciences

University of California, Berkeley

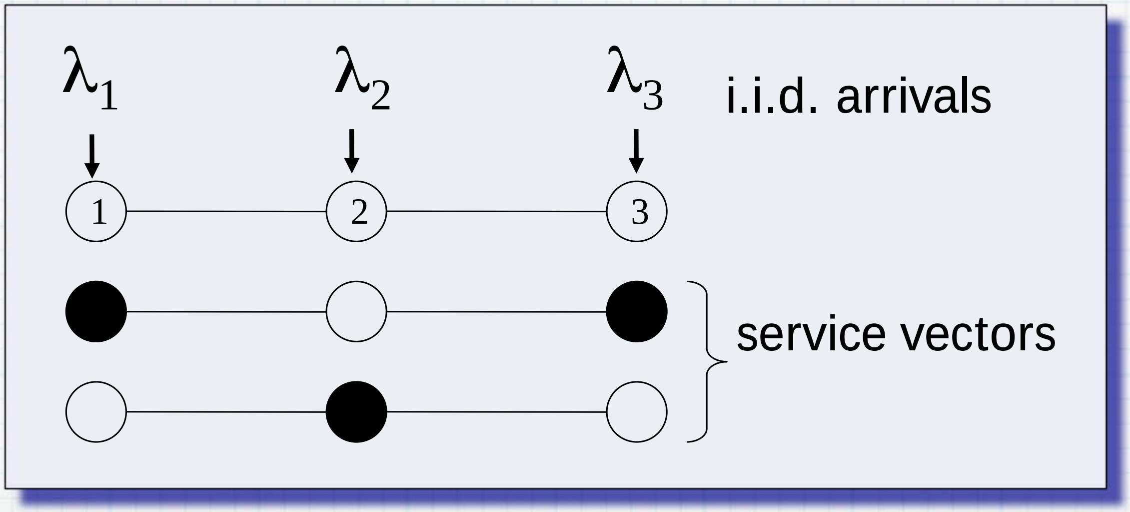
Joint work with **Libin Jiang** (earlier work with **Antonis Dimakis**)



Outline

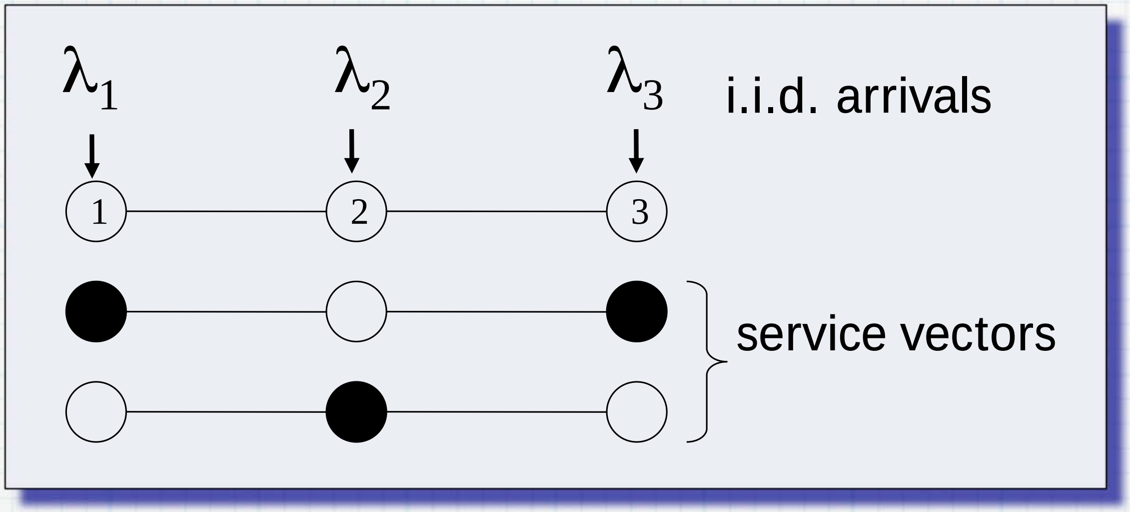
* Examples and MWM
* Longest Queue First
* Wireless Backpressure
* Processing Networks: DMW
* Summary

Examples and MWM



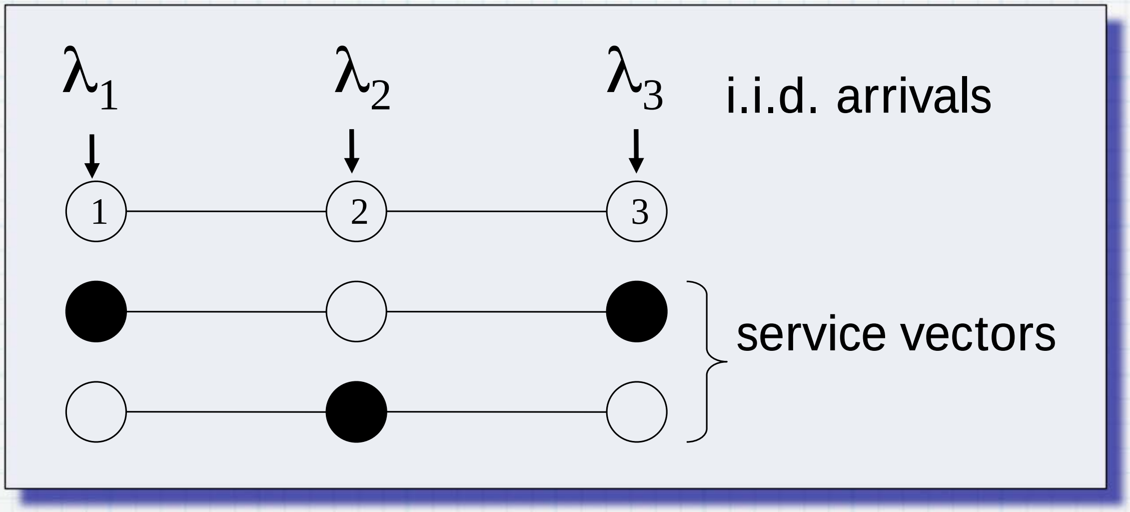
* Three wireless links: 1, 2, 3
* Links (1, 2) interfere and cannot transmit together; same for links (2, 3)
* Links (1, 3) can transmit together

Examples and MWM



- Question: Which links should transmit at any given time?
- Goal: Keep up with arriving packets (rates $\lambda_1, \lambda_2, \lambda_3$).
- Typical approach: Try after random delay; try again if you fail but increase randomization interval.
- Simple but not "maximum throughput".

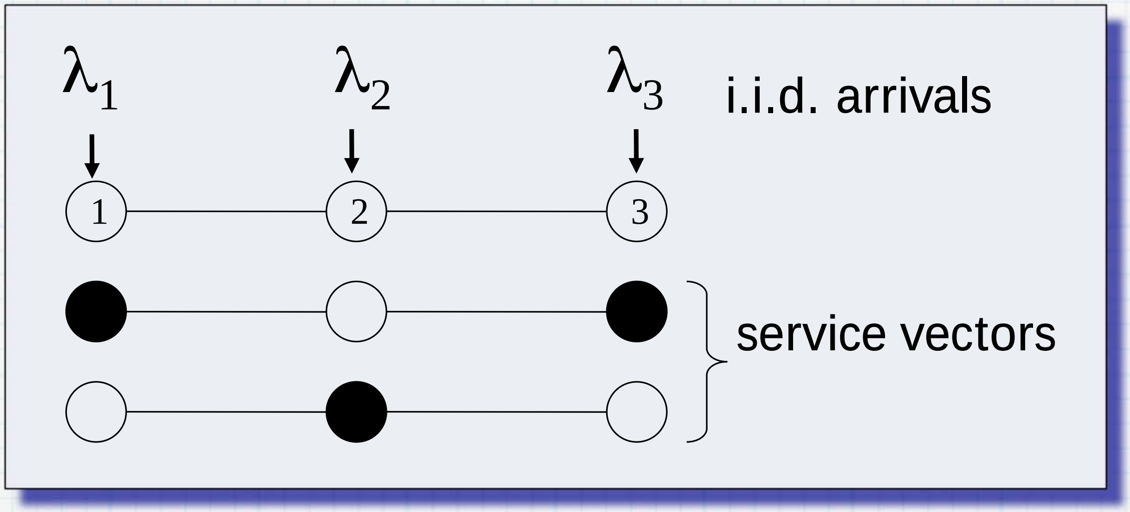
Examples and MWM



Backlogs

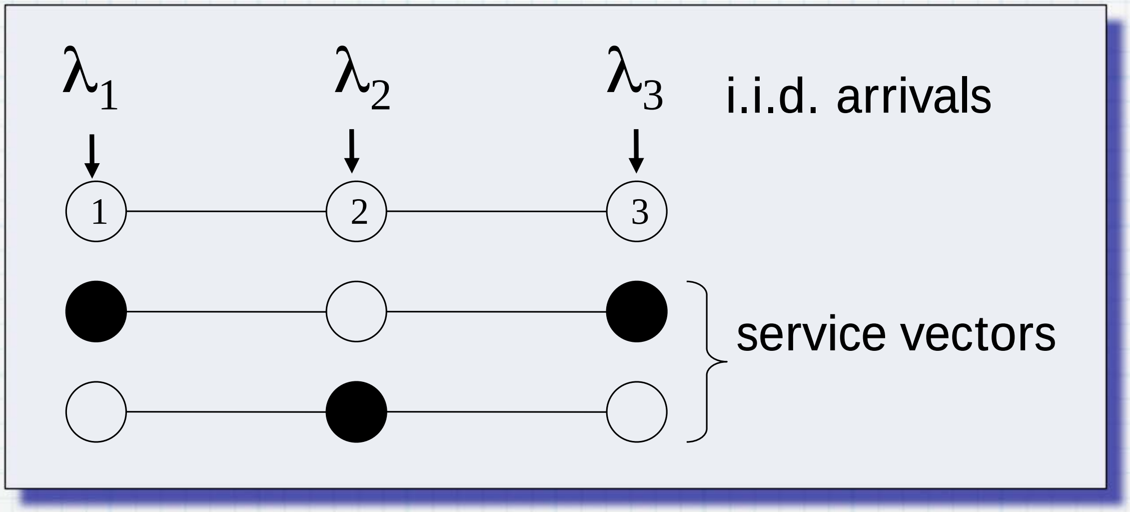
- Maximum Weighted Match:
  - Links (1, 3) should transmit if $X_1 + X_3 > X_2$
  - Link 2 should transmit if $X_2 > X_1 + X_3$
  - If $X_1 + X_3 = X_2$, flip a coin

Examples and MWM



- MWM Examples:
  - $(X_1, X_2, X_3) = (3, 6, 2) \Rightarrow$ Link 2 should transmit
  - $(X_1, X_2, X_3) = (3, 4, 2) \Rightarrow$ Links 1 and 3 should transmit

Examples and MWM

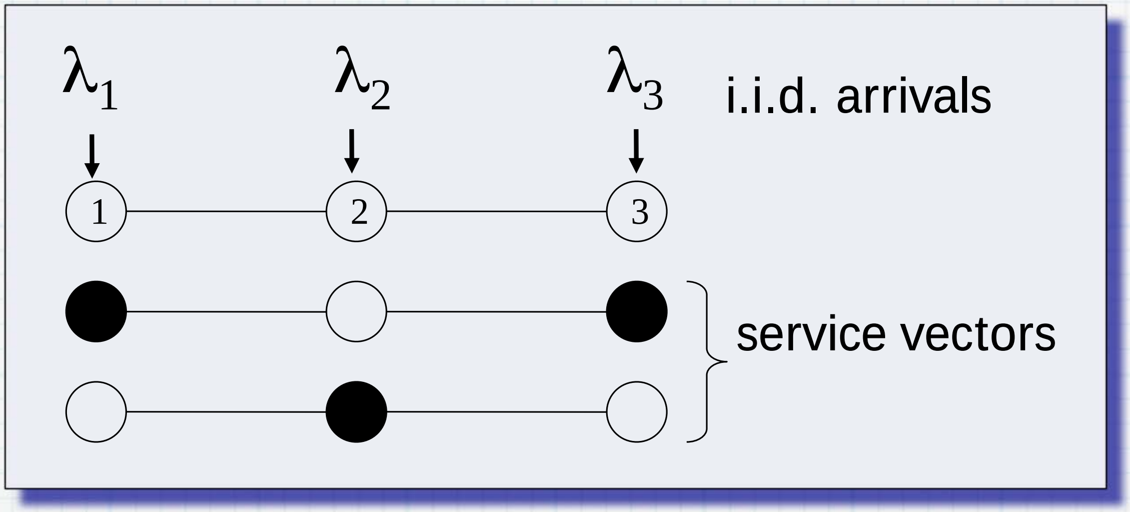


- THEOREM: MWM achieves the maximum throughput!
- That is, queues are stable as long as

$$\lambda_1 + \lambda_2 < 1 \text{ and } \lambda_2 + \lambda_3 < 1$$

- Key Idea:
  MWM makes $X_1^2 + X_2^2 + X_3^2$ decrease, on average

Examples and MWM

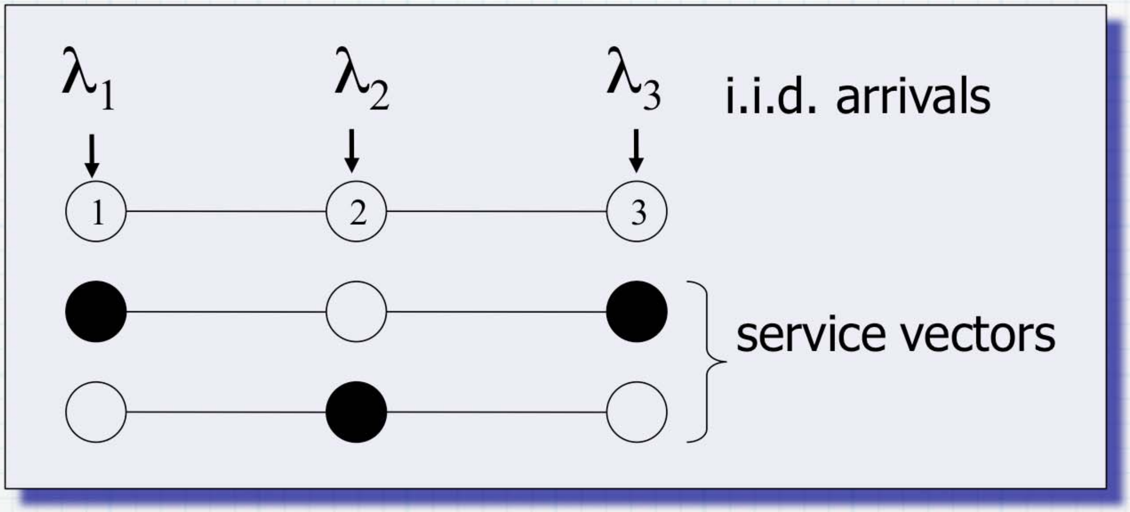


- MWM makes $X_1^2 + X_2^2 + X_3^2$ decrease, on average

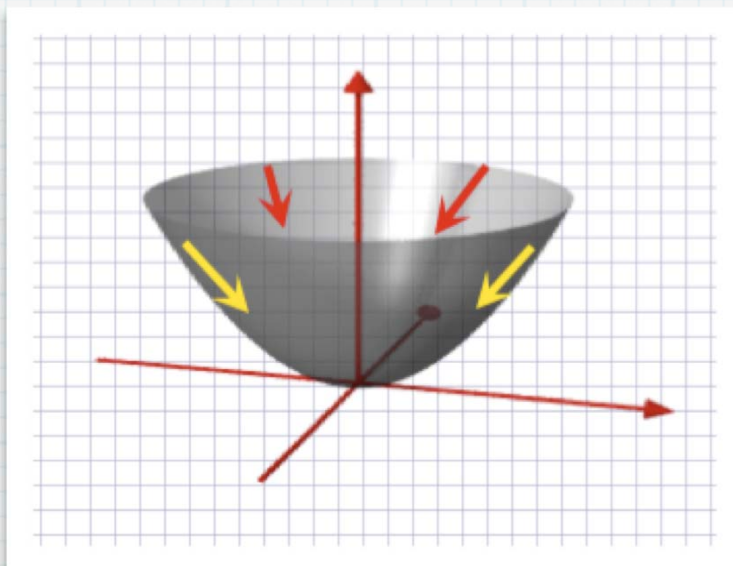
$$(X_i + A_i - S_i)^2 - X_i^2$$
$$= 2X_iA_i - 2X_iS_i - 2A_iS_i + A_i^2 + S_i^2$$
$$E[\cdot|X] \leq K + 2\lambda_i X_i - 2X_iS_i.$$
$$\sum_i E[\cdot|X] \leq 3K + 2\sum_i \lambda_i X_i - 2\sum_i X_iS_i.$$

Maximized by MWM

Examples and MWM

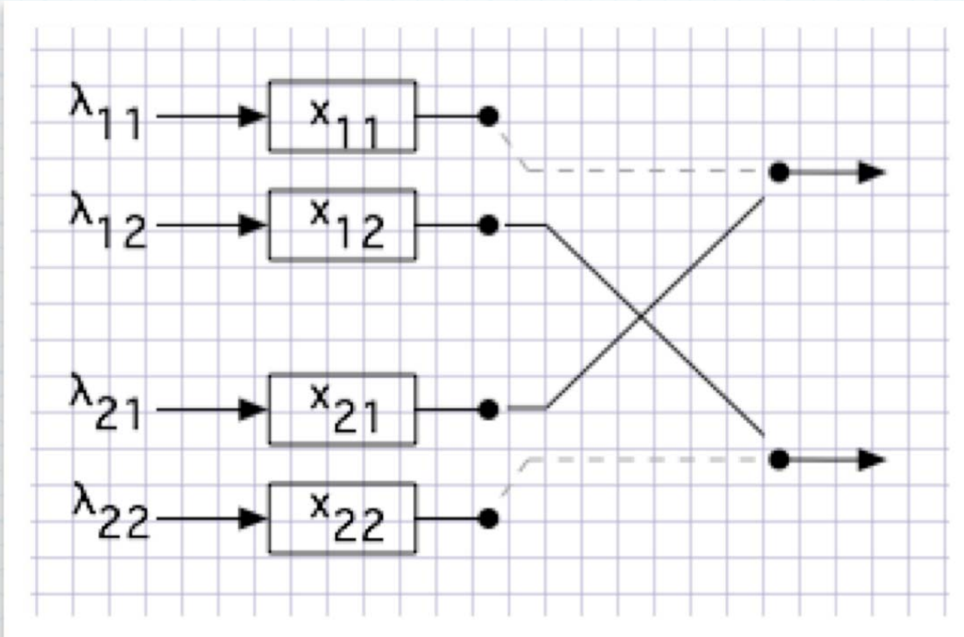


- MWM makes $X_1^2 + X_2^2 + X_3^2$ decrease, on average



Tassiulas & Ephremides, 92

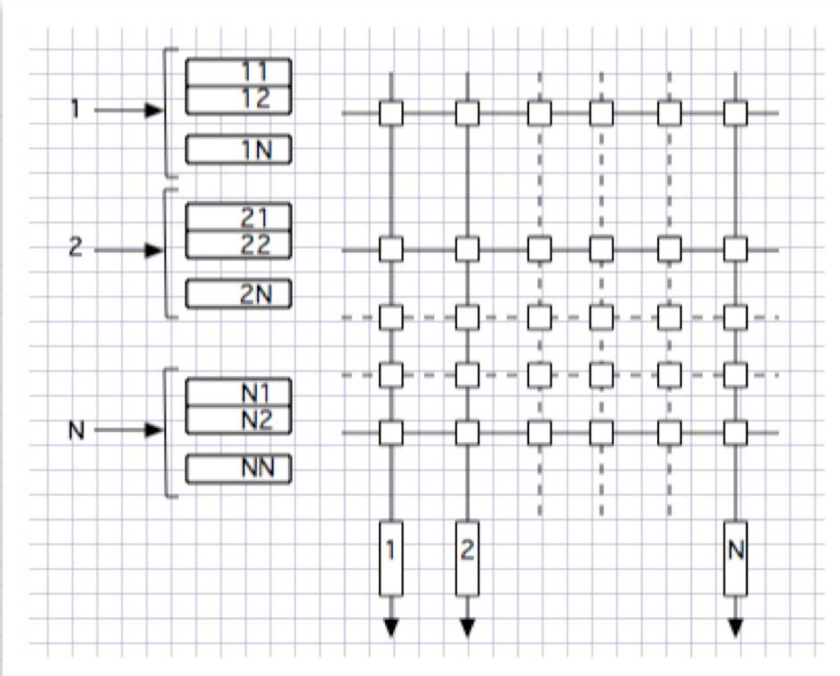
Examples and MWM



- VOB SWITCH
  Can serve (1 1 and 22) or (1 2 and 21)
- MWM: Serve (1 1 and 22) if $X_{11} + X_{22} > X_{12} + X_{21}$
- THEOREM: MWM achieves maximum throughput

N. McKeown, A. Mekkittikul, V. Anantharam, J. Walrand, 99

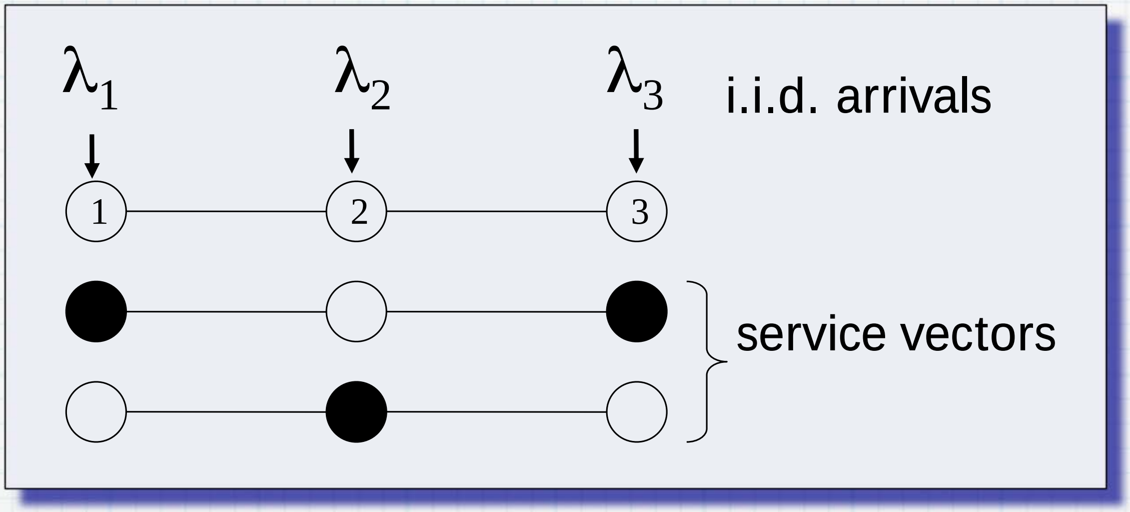
Examples and MWM



- BUFFERED CROSSBAR SWITCH
  Each crosspoint can hold one packet
- Each input: send to any free crosspoint
  Each output: read from any nonempty crosspoint
- THEOREM: Achieves maximum throughput

Shang-Tse Chuang, Sundar Iyer, Nick McKeown, '05

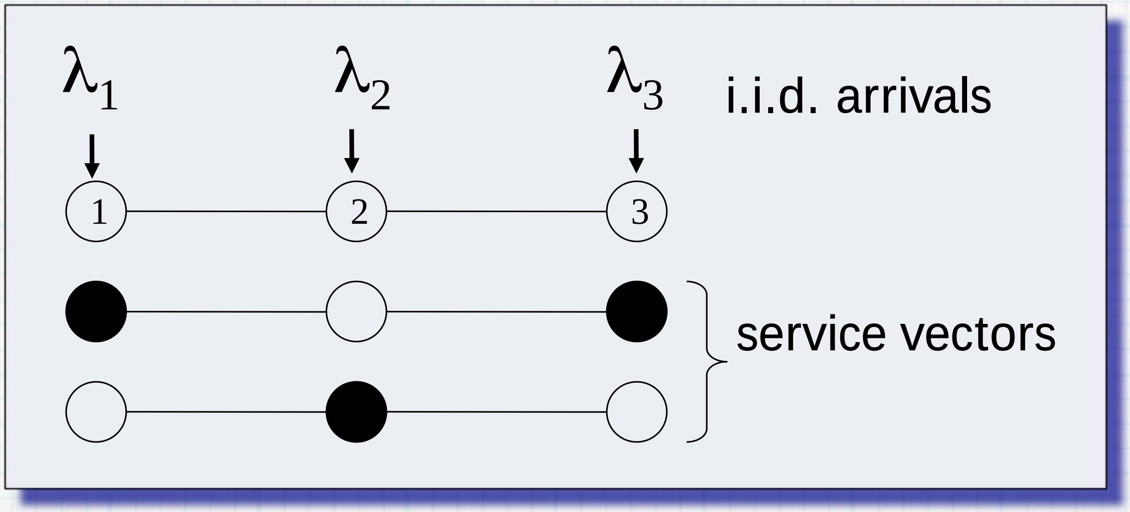
Longest Queue First



- LQF:
  - First, pick longest queue
  - Next, pick longest among other compatible queues
- Examples:
  - (3, 4, 2) $\Rightarrow$ Serve queue 2 [Note: MWM: 1 & 3]
  - (5, 4, 1) $\Rightarrow$ Serve queues 1 and 3

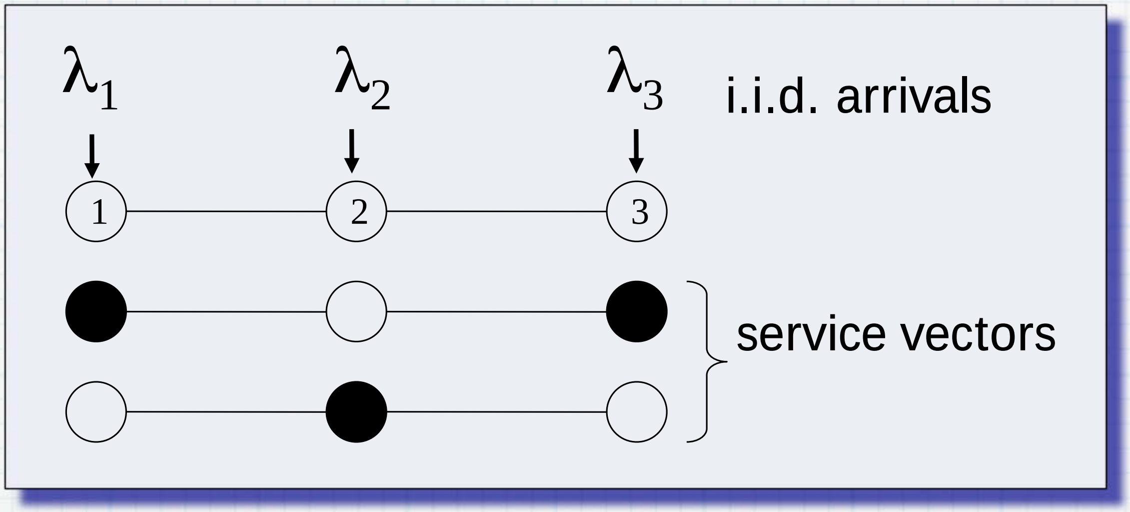
Antonis Dimakis and Jean Walrand, '05

Longest Queue First



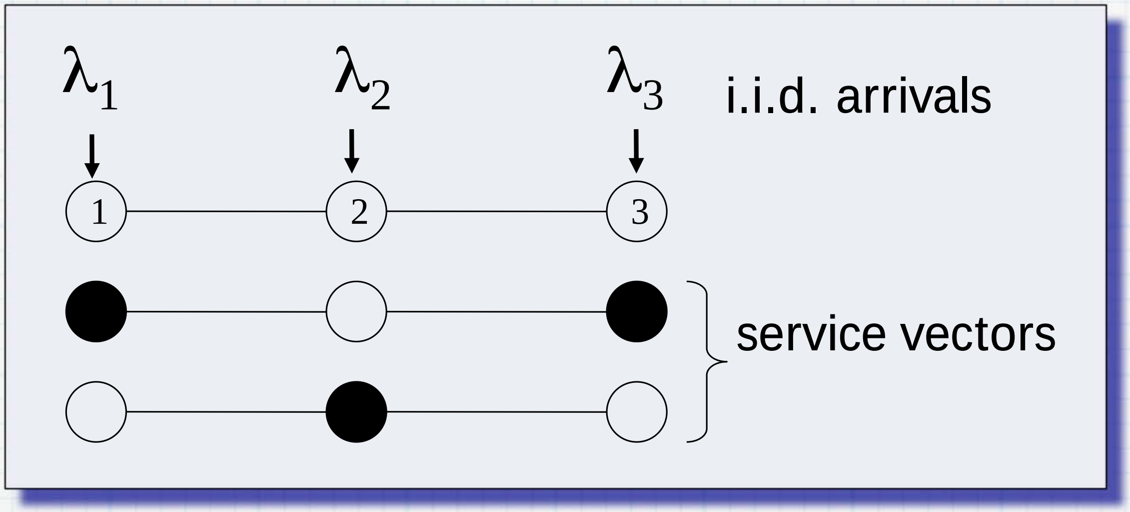
- THEOREM: LQF achieves the maximum throughput (in this network)
- Key Idea: Longest queue decreases, on average
    - Say queue 2 is longest ⇒ Decreases under LQF [LQF serves it at rate 1 and $\lambda_2 < 1$]

Longest Queue First



- THEOREM: LQF achieves the maximum throughput
- Key Idea: Longest queue decreases, on average
  - Say queues 1 and 2 are both longest ⇒ decrease
    [Set (1, 2) served at rate 1 under LQF and $\lambda_1 + \lambda_2 < 1$]
  - Similar for (2, 3), (1, 3), and (1, 2, 3)

Longest Queue First



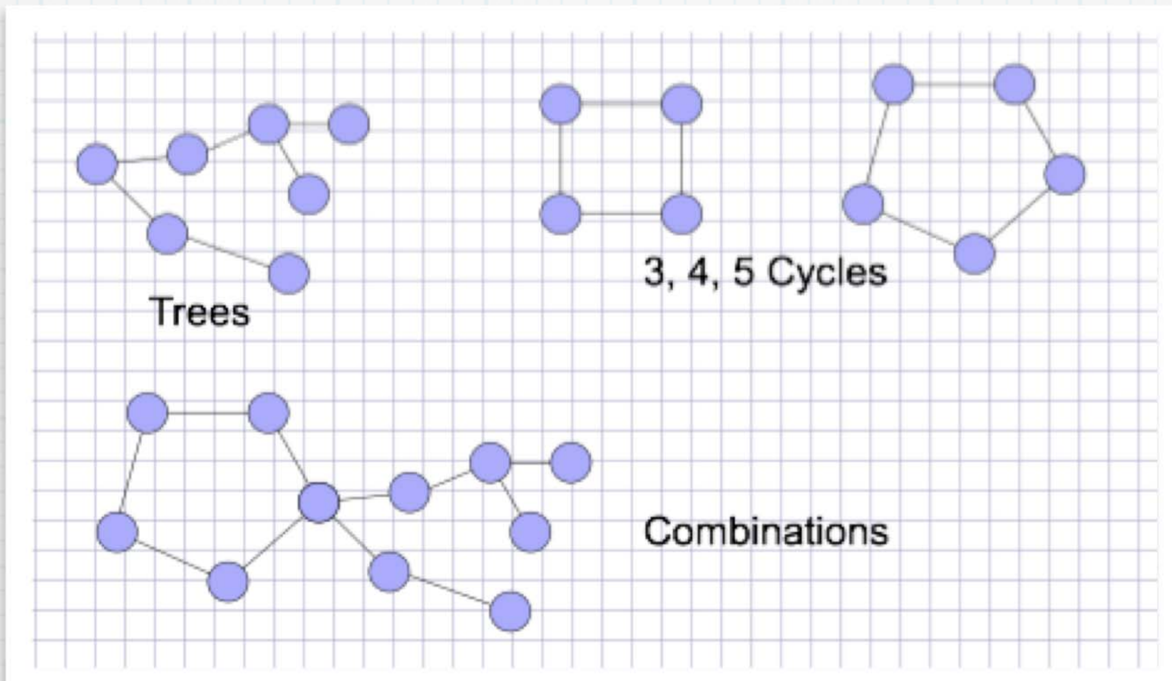
- Note that for any set L of longest queues,
LQF serves a subset S of those queues at constant rate; that rate is larger than the arrival rate in S
- $L = \{1, 2, 3\} \Rightarrow S = \{1, 2\}$; otherwise, $S = L$

Longest Queue First

- LOCAL POOLING PROPERTY of GRAPH:

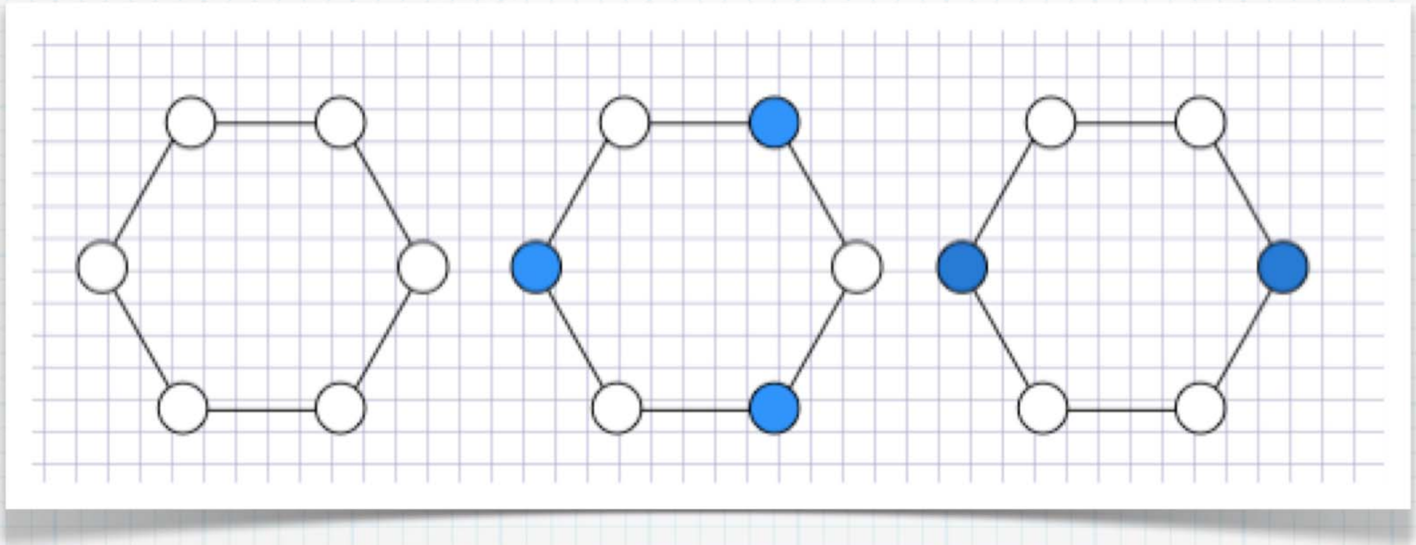
  For any set L of longest queues,
  LQF serves a subset S of those queues at constant rate; that rate is larger than the arrival rate in S

- EXAMPLES of graphs with Local Pooling Property:



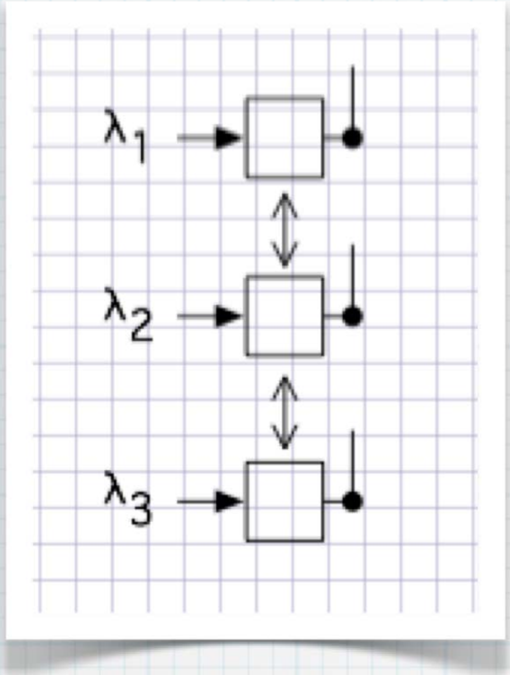
Longest Queue First

Note: 6 Cycle does not satisfy local pooling



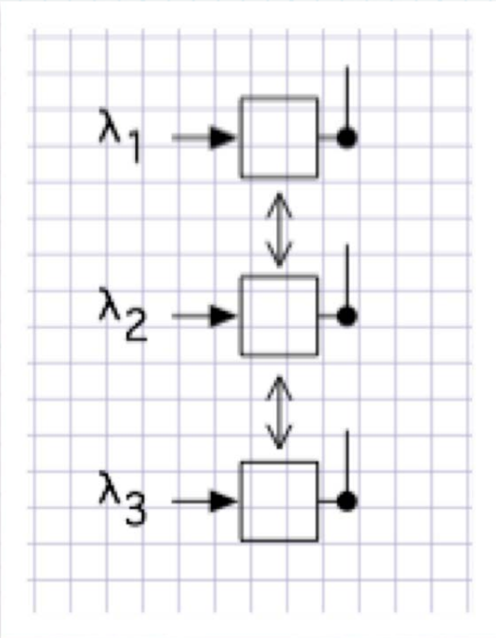
If L = {1, 2, 3, 4, 5, 6}, there is no subset S of L that LQF serves at a constant rate larger than the arrival rate in S.

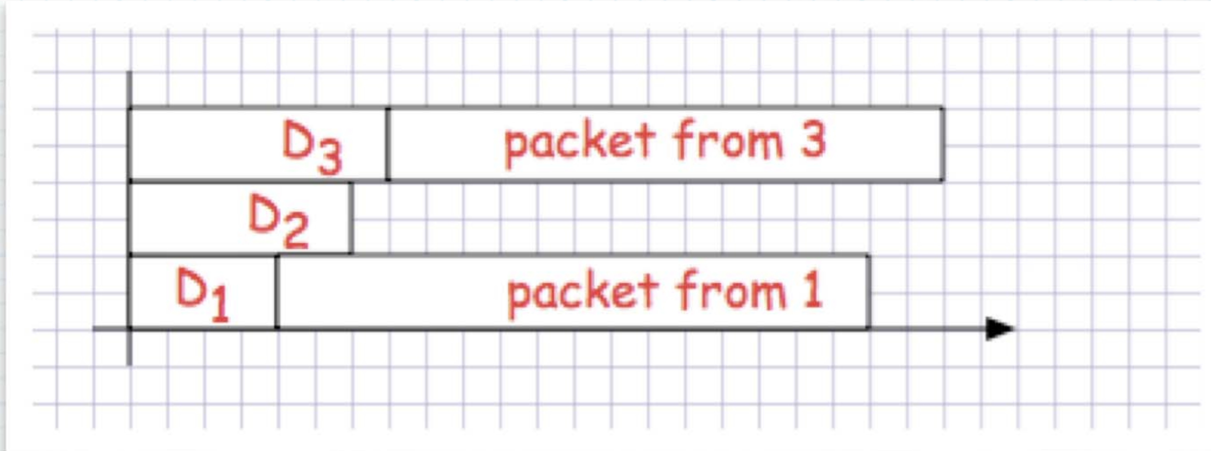
Wireless Backpressure



- As before, links (1, 2) conflict and so do (2, 3)
- There is no central coordination
- Links want to keep up with arriving packets

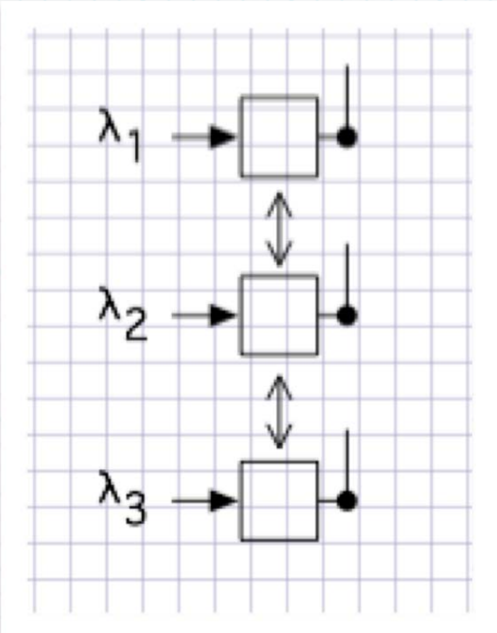
Wireless Backpressure

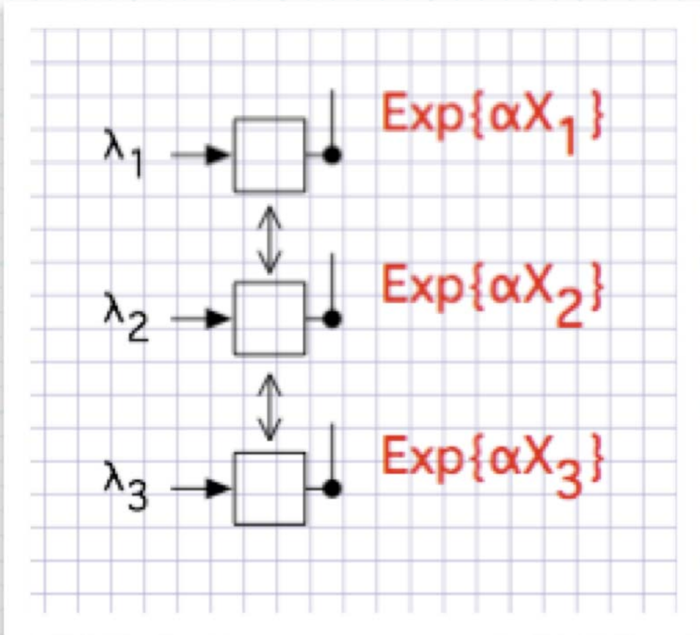




- DISTRIBUTED SCHEME: CSMA
- Nodes pick independent random "backoff" delays
- Node with smallest delay starts transmitting
- If next node does not hear anything, it transmits

Wireless Backpressure

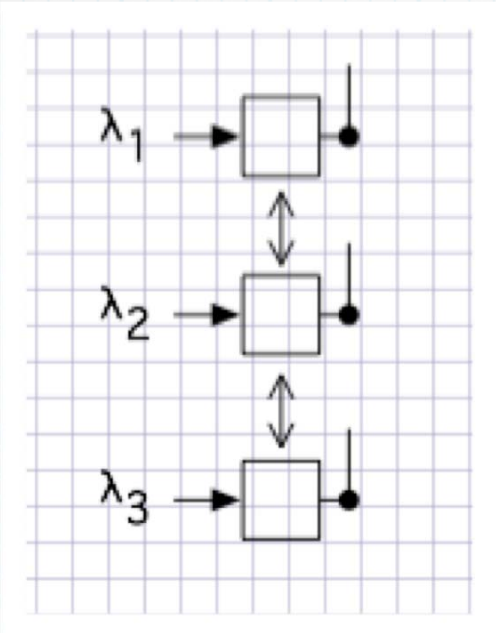


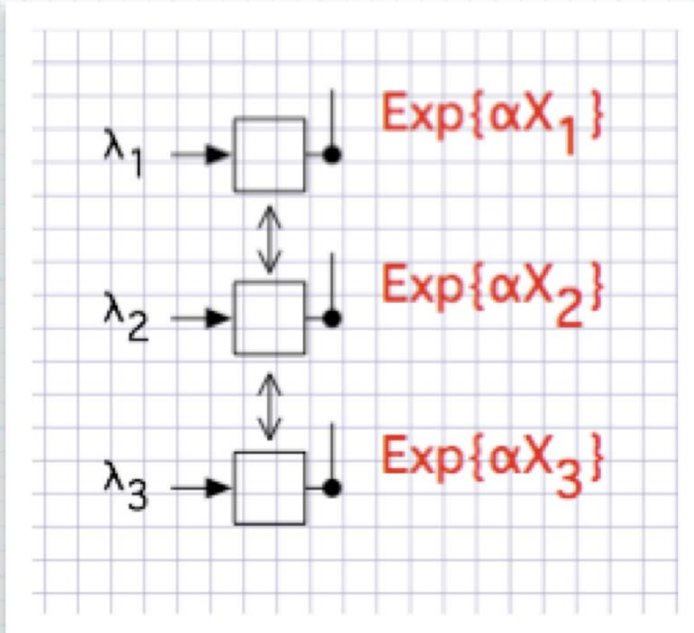


- BACKPRESSURE-BASED BACKOFF
- D_i is exponentially distributed with rate Exp{ αX_i}
- Thus, the mean backoff delay decreases fast with X_i
- The longest queue tends to transmit first, then ...

Libin Jiang and Jean Walrand, Allerton 08

Wireless Backpressure





d(r) = KL-distance between $\pi(r)$ and p, where
$\pi(r)$ = inv. dist. of independent sets a_j under r
[when backoff of i is exp. with rate $\exp(r_i)$]
p = dist. of ind. sets a_j s.t. $\lambda = \Sigma_j\, p_j a_j$

- THEOREM: WB achieves the maximum throughput
- Key Ideas:

Libin Jiang and Jean Walrand, 12/08

  - Backoffs with $r_i \rightarrow$ Service rates $s_i(r)$ of queues
  - Minimize d(r) over r: Minimizer r^* such that $s(r^*) \geq \lambda$
  - Gradient algorithm yields $r_i = \alpha X_i$

Wireless Backpressure

Some details:

- $d(r) = \sum p_i \log(p_i/\pi_i(r))$
- $\nabla d(r) = - [\lambda - s(r)]$
- $r(n+1) = r(n) + \alpha [\lambda - s(r(n))]^+$
  $= r(n) + \alpha$ [expected arrival rate - expected service rate]$^+$
  $\approx r(n) + \alpha$ [actual arrival rate - actual service rate]$^+$

Justified by stochastic approximation argument

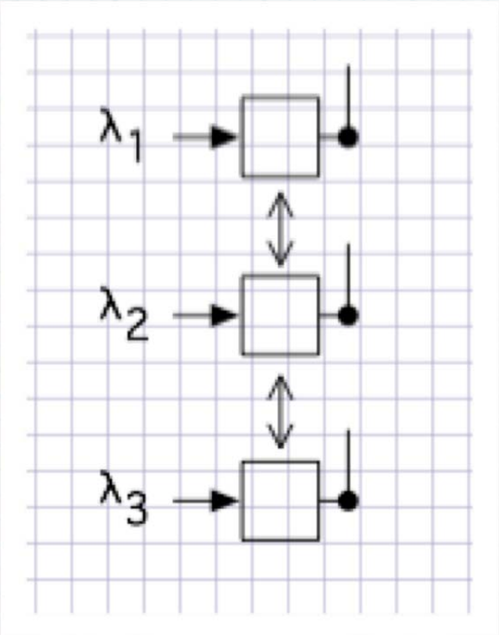
Also

- $q(n+1) = q(n) + \alpha$ [actual arrival rate - actual service rate]$^+$

Hence

- $r(n) = \beta q(n)$: distributed

Wireless Backpressure



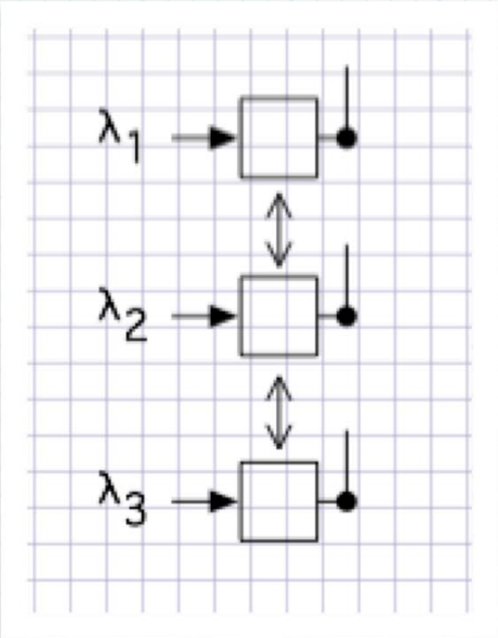
Utility: Concave, increasing

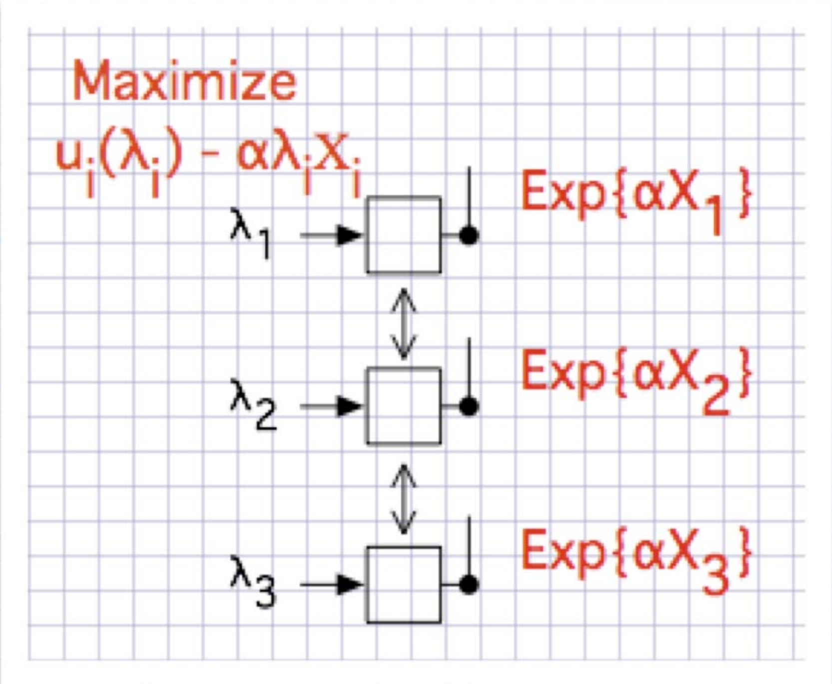
- Links want to maximize the "total utility"

  $u_1(\lambda_1) + u_2(\lambda_2) + u_3(\lambda_3)$

- Approach: CSMA + input rate control

Wireless Backpressure



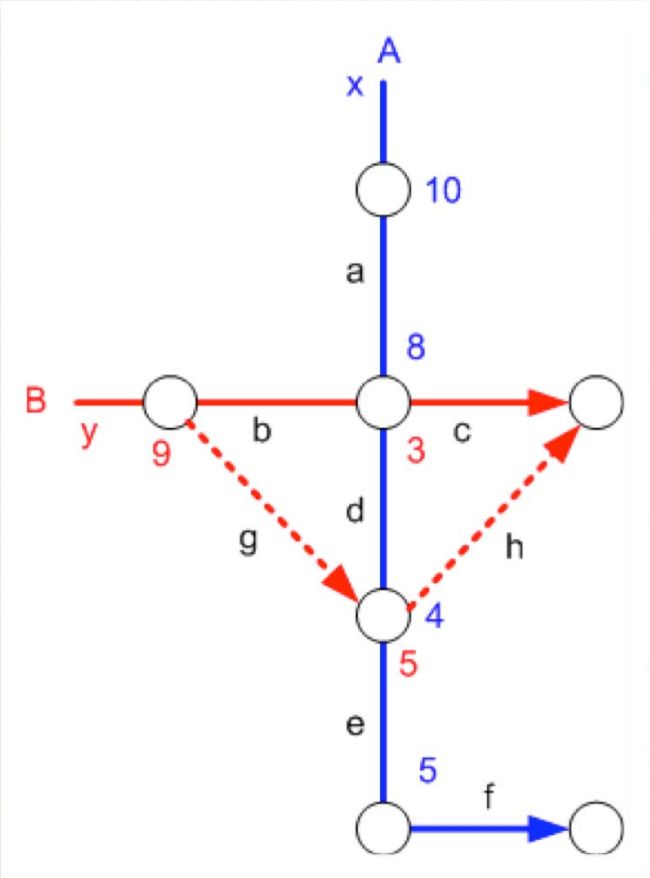


- Links want to maximize the "total utility"

$$u_1(\lambda_1) + u_2(\lambda_2) + u_3(\lambda_3)$$

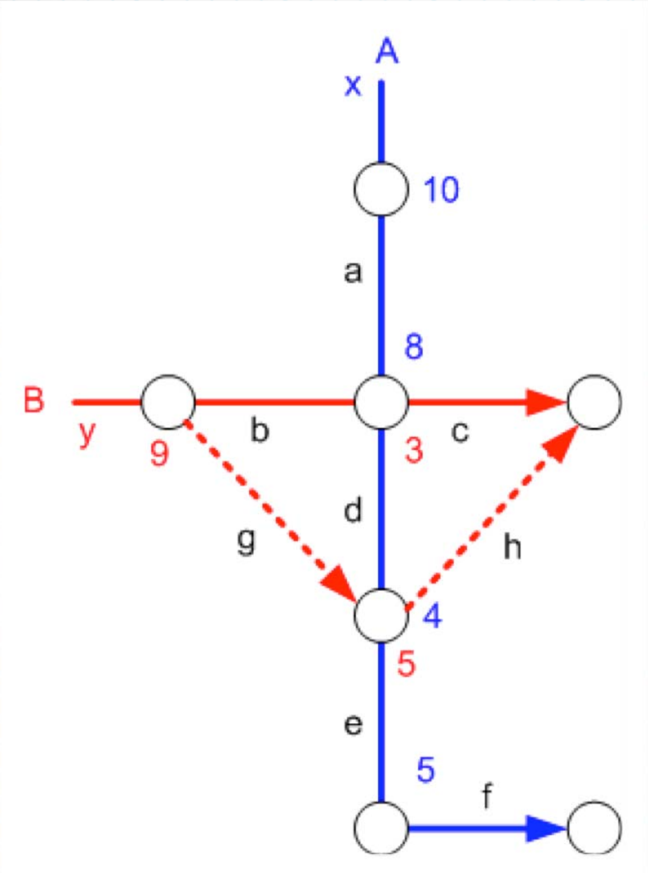
- Approach: CSMA + input rate control
- THEOREM: Achieves Maximum Utility

Wireless Backpressure



- Wireless links, with interference
- Goal: maximize total utility of flows
- Note: Adjust input rates, scheduling, and routing

Wireless Backpressure



Protocol

Link b chooses $T = \text{Exp}(e^{\alpha(9-3)r(b)})$

Link g chooses $S = \text{Exp}(e^{\alpha(9-5)r(g)})$

If T < S, then b transmits ….

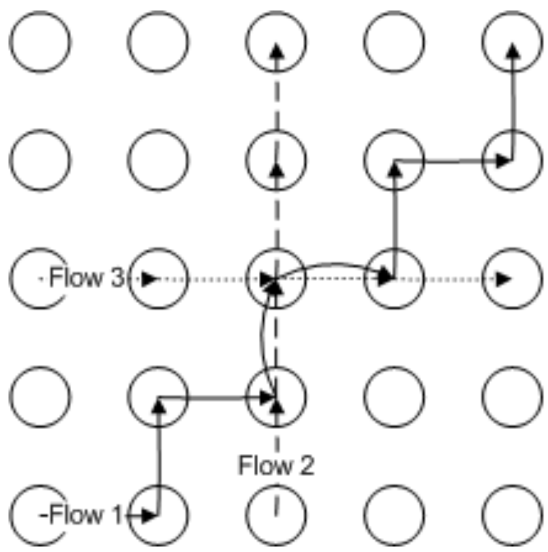
A chooses $x = \text{argmax } U_1(x) - \alpha 10x$

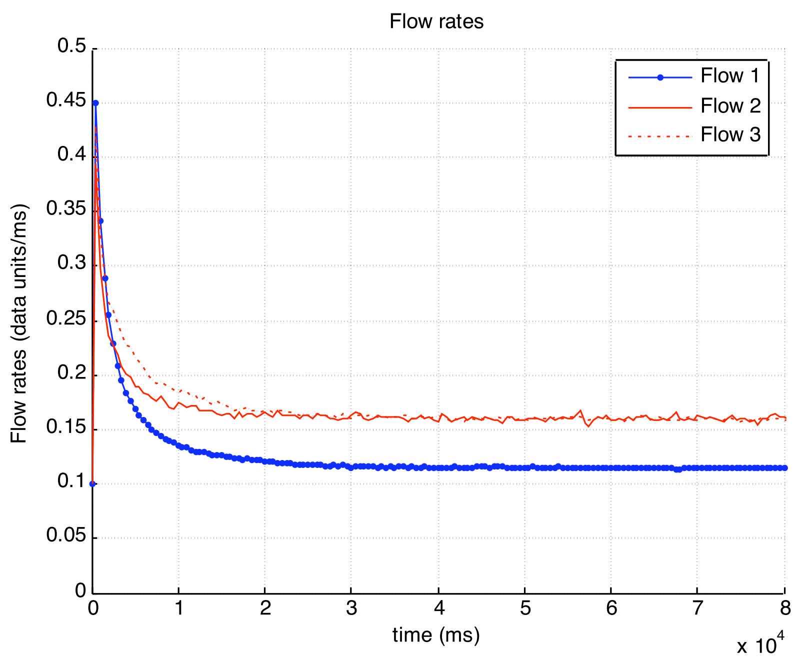
B chooses $y = \text{argmax } U_2(y) - \alpha 9y$

r(b) = rate of link b

Note: per-flow queues

Wireless Backpressure

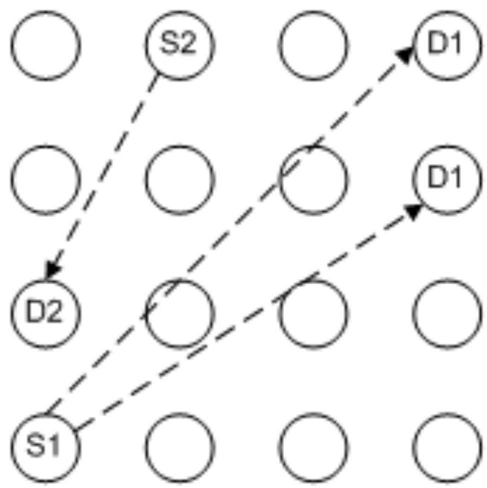


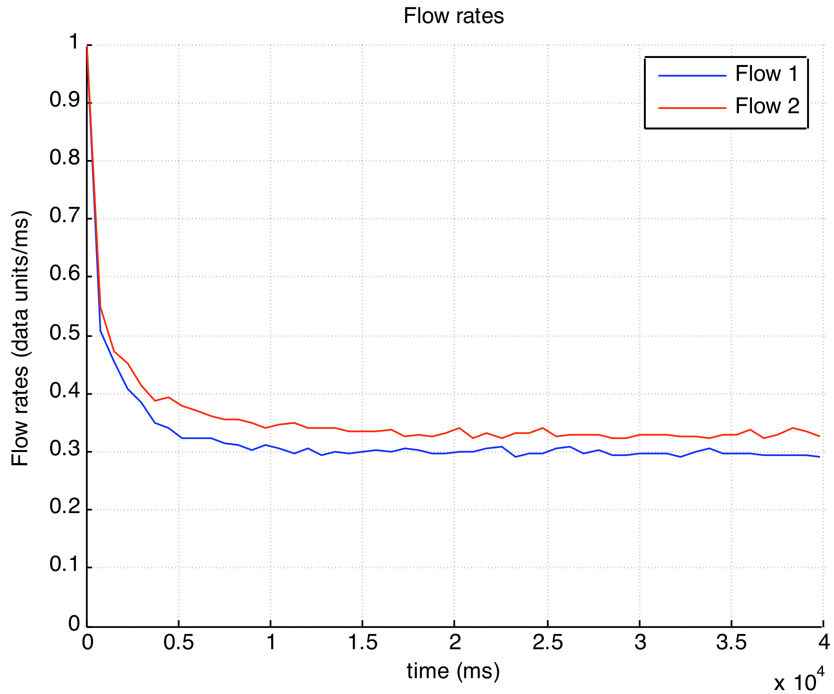


One-way interference

Theoretical optimal flow rates:
0.1111, 0.1667 and 0.1667

Wireless Backpressure

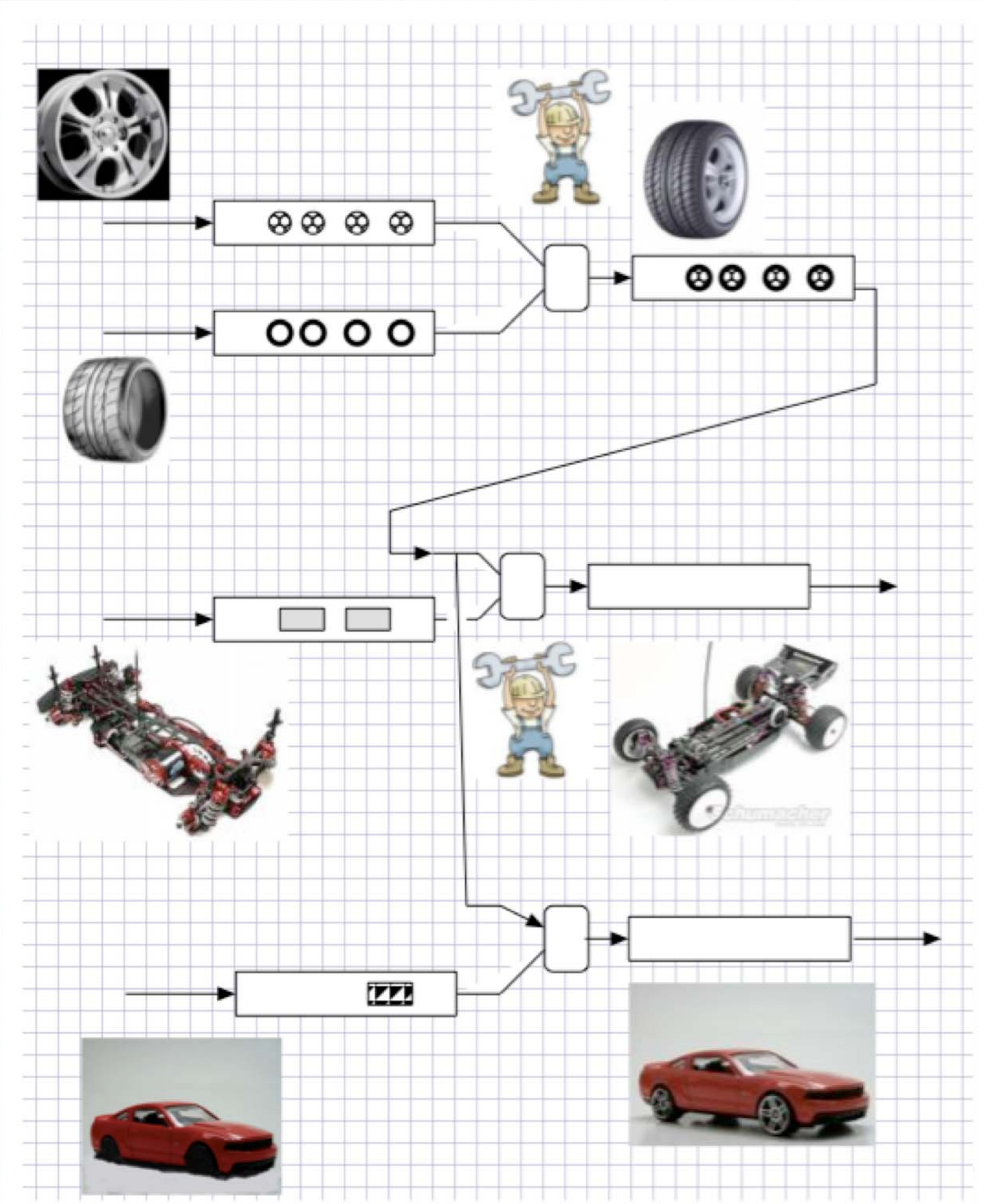




Multipath routing allowed

Unicast S2 -> D2
Anycast S1 to any D1

Processing Networks

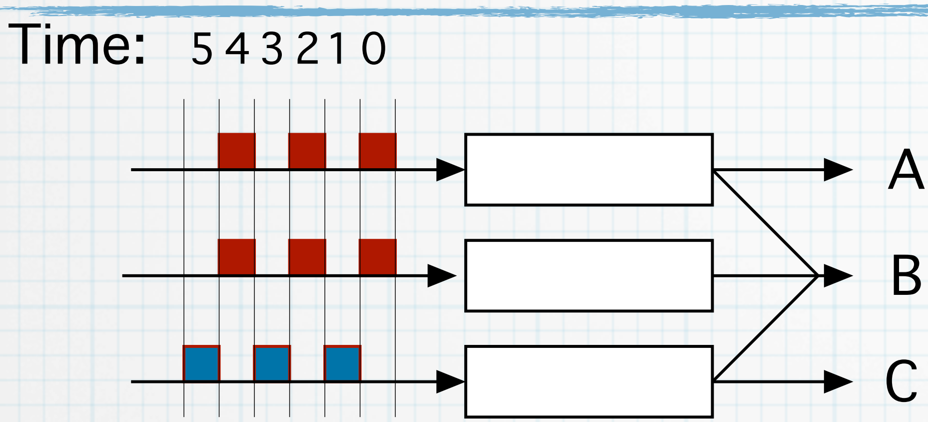


- Tasks need parts and resources
- Goal: maximize utility
- Approach:

  Deficit Maximum Weight
- Note

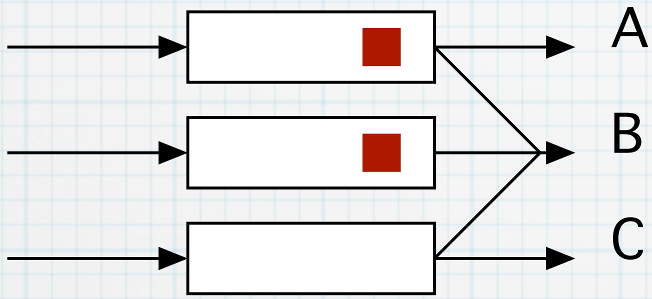
  MWM Unstable

PN: Basic Problem

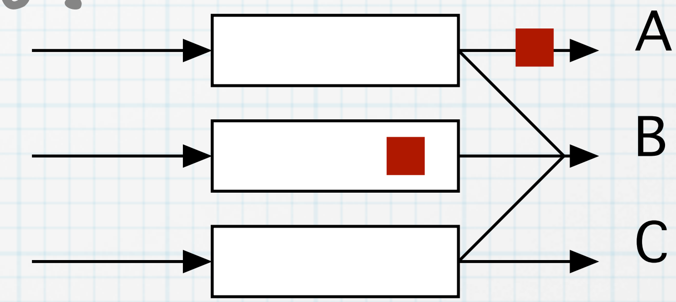


Task A requires a part from queue 1
Task B requires a part from all queues
Task C requires a part from queue 3

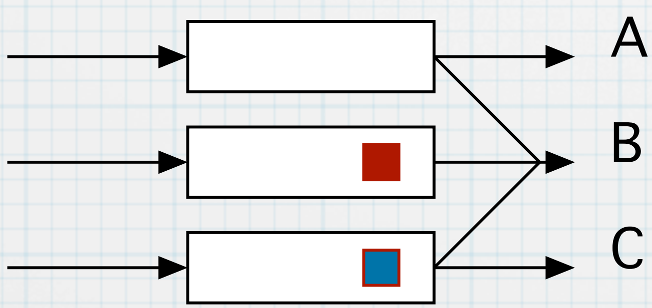
Time 0



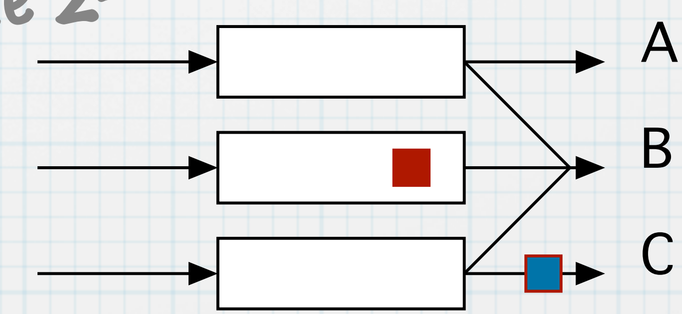
Time 1-



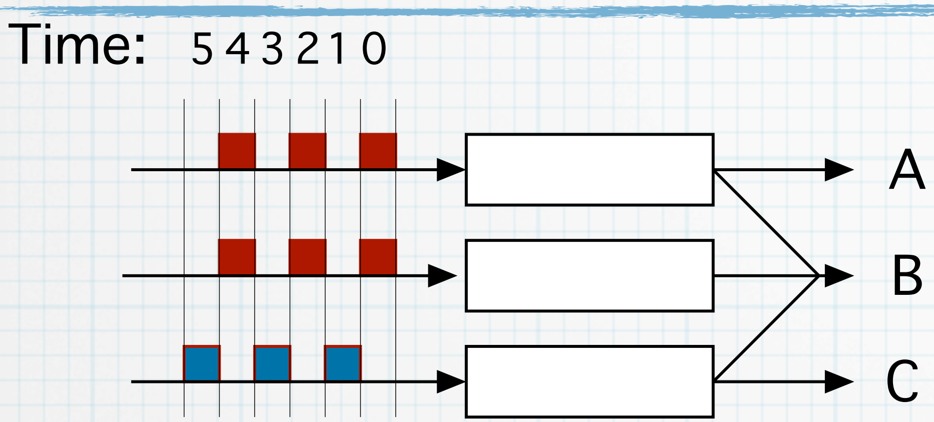
Time 1



Time 2-

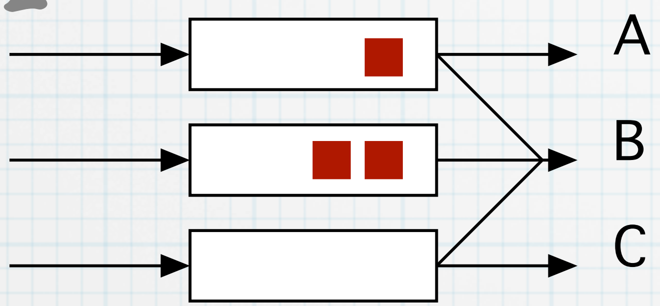


PN: Basic Problem

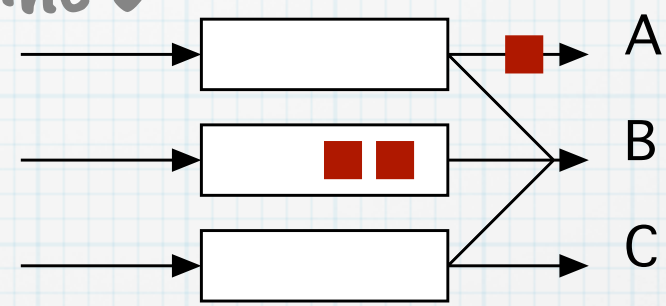


Task A requires a part from queue 1
Task B requires a part from all queues
Task C requires a part from queue 3

Time 2

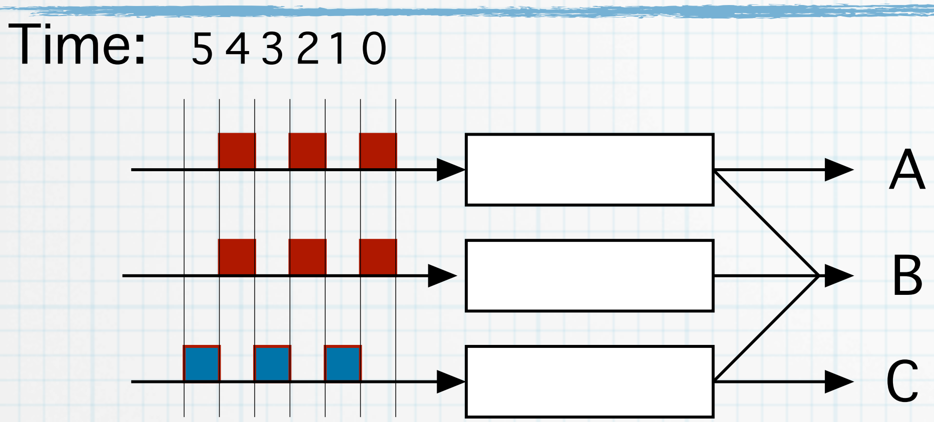


Time 3-



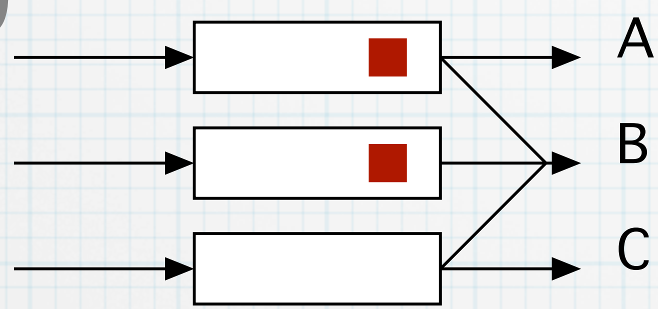
Maximum Weighted Matching is not stable.

PN: Basic Problem

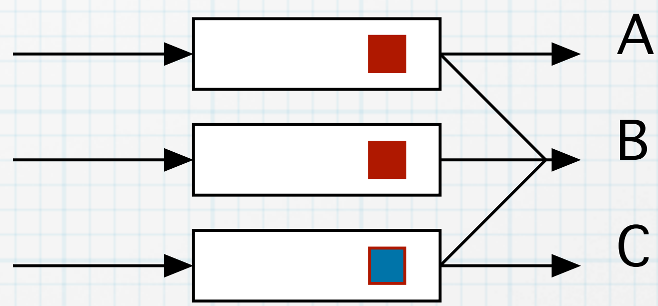


Task A requires a part from queue 1
Task B requires a part from all queues
Task C requires a part from queue 3

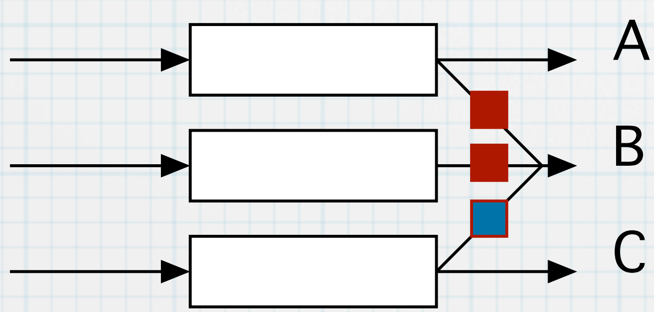
Time 0



Time 1: Do not serve

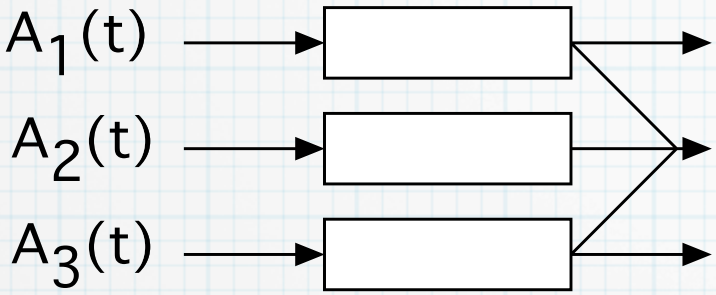


Time 2-



Modified scheduling is stable.

PN: Basic Problem



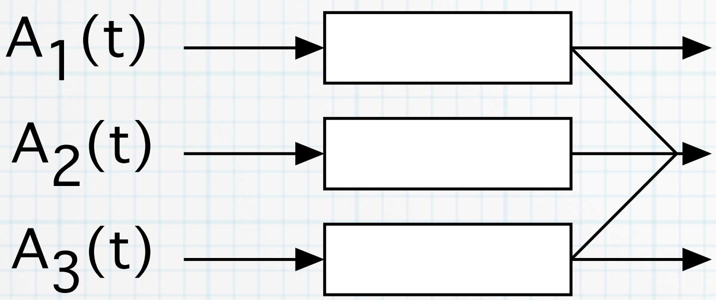
Under a reasonable assumption on the arrival processes, one should be able to stabilize the network.

For instance, assume that the arrival rates are in the convex hull of the service vectors. Moreover, assume that the distance between the arrivals $A(t)$ and their averages λt in $[0, t]$ is bounded*. Then some scheme should stabilize the system.

The goal is to find a scheme that automatically adjusts the schedule.

*This condition is called "bounded burstiness." A weaker condition is presented in the paper.

PN: DMW

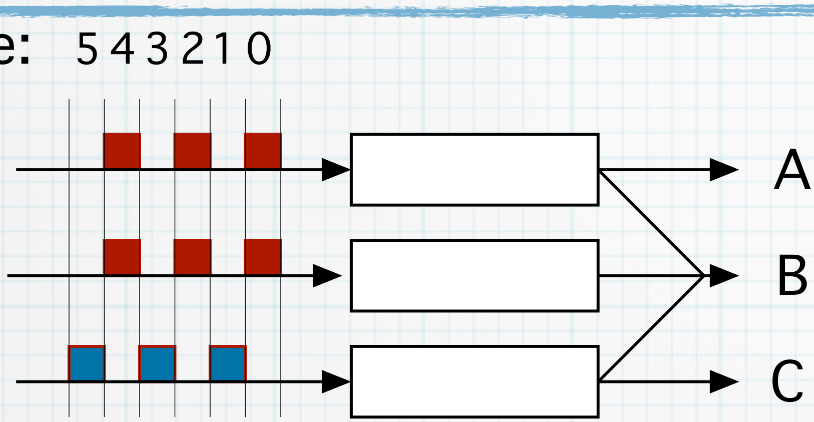


Scheme: Deficit Maximum Weight (DMW).

1) "Augment State" with virtual backlog.
2) Schedule according to virtual backlog which may be negative, thus scheduling a "null activity". Schedule with maximum weight.
3) Prove that the difference between actual and virtual is bounded. Thus, waste a finite amount of time.

Extends to utility maximization.

PN: DMW



q_i = virtual backlog at queue i.

Q_i = actual backlog at queue i.

time	0	1-	1	2-	2	3-
q_1, Q_1	1, 1	0, 1	0, 1	0, 1	1, 2	0, 1
q_2, Q_2	1, 1	0, 1	0, 1	0, 1	1, 2	0, 1
q_3, Q_3	0, 0	-1, 0	0, 1	0, 1	0, 1	-1, 0
Activity	Arrival	B	Arrival	None	Arrival	B
Note:		Virtual				Actual

Libin Jiang and Jean Walrand, Allerton 09

Repeats forever

PN: DMW

* Actual queues Q(t), virtual queues q(t)
  * Allow q(t) to be negative
* Queue dynamics

Activation of SA's decided by MW

$$q_k(t+1) = q_k(t) - \mu_{out,k}(t) + \mu_{in,k}(t), \forall k$$

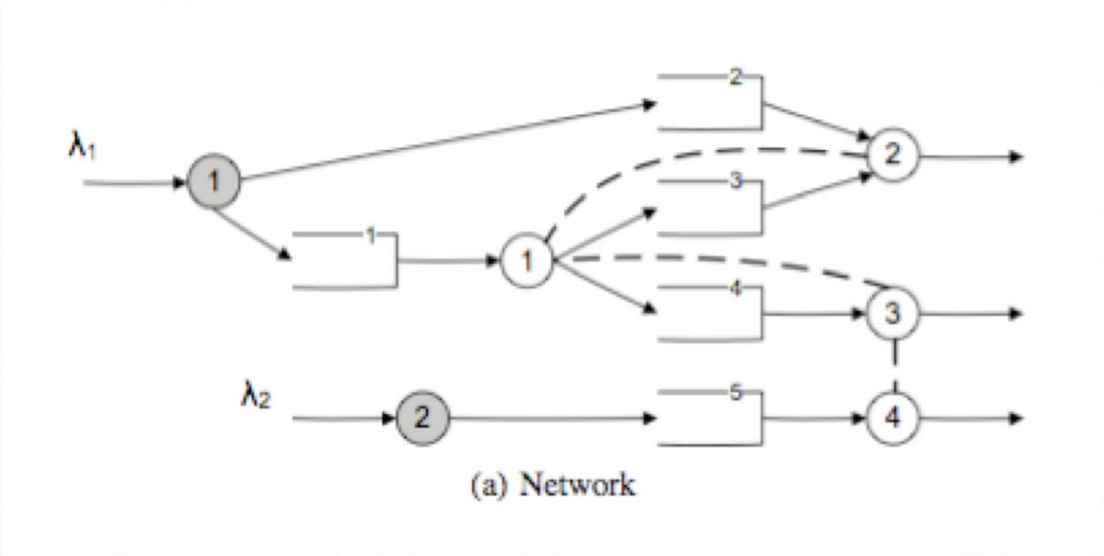
$$Q_k(t+1) = [Q_k(t) - \mu_{out,k}(t)]_+ + \mu_{in,k}(t)$$

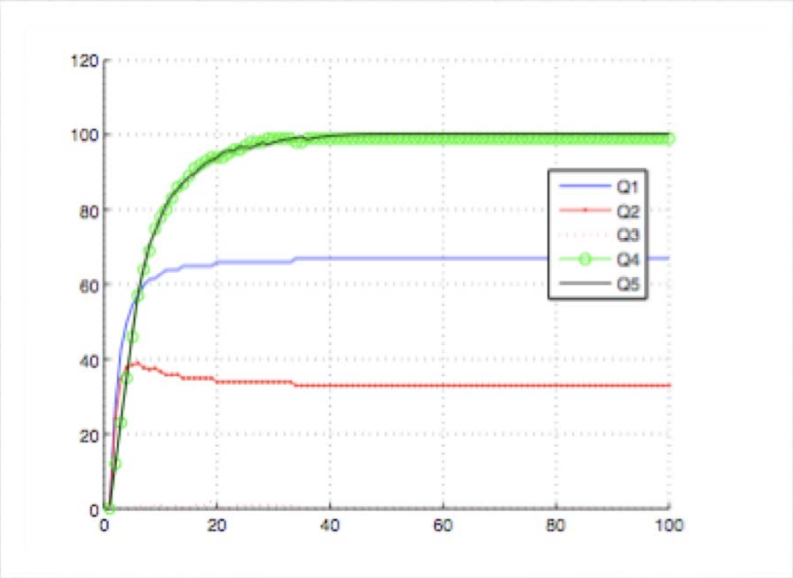
  * If Q_k "underflows", then activate a "null SA" and use "fictitious parts"
* "Deficit" $D_k(t+1) = Q_k(t+1) - q_k(t+1)$

PN: DMW

- Prop. 1: If q(t) is bounded, then both Q(t) and D(t) are bounded. Only a finite number of null SA's occur → long-term throughput not affected.
- Prop. 2: If the arrival process is smooth enough, then q(t) is bounded
  - For example, there exists T>0 so that $\sum_{\tau=t}^{t+T-1} \mathrm{a}(\tau)/T$ is in the interior of the capacity region (uniformly) for t = 0, T, 2T, ...
  - Mild condition
    - More random arrivals:
      - System is still "rate-stable", although Q(t) may slowly drift to infinity
        - Tradeoff between queue lengths and throughput

Processing Networks





- Parts arrive at 1 & 2 with rate λ_1 and at 5 with rate λ_2
- Task 2 consumes one part from 2 and one from 3; ...
- Tasks 1-2, 1-3, 3-4 conflict
- Algorithm stabilizes the queues and achieves the max. utility

Summary

* Problem: Scheduling of conflicting tasks to
  * keep up with arriving jobs, or
  * maximize the total utility of the tasks
* Approach:
  * Longest queue first, if local pooling: LQF
  * Backpressure-based requests for resources
  * DMW for PNs
  * References:

    Talk abstract

    Web: Jean Walrand, EECS, Berkeley