Architectures and Algorithms for Wireless Networks

A. Eryilmaz
(*Ohio State University*)

R. Srikant
(*UIUC*)

Softcopy available at: http://www.ece.osu.edu/~eryilmaz/UIUCSummerSchool09.pdf

Principles and Applications of Architecture and Algorithm Design for Wireless Networks

Atilla Eryilmaz

Electrical and Computer Engineering Dept.

THE OHIO STATE UNIVERSITY

Collaborators: *R. Srikant (UIUC), R. Li, N. Shroff, E. Ekici, E. Koksal (OSU), A. Ozdaglar, M. Medard, D. Lun, (MIT), Lei Ying (Iowa State)*

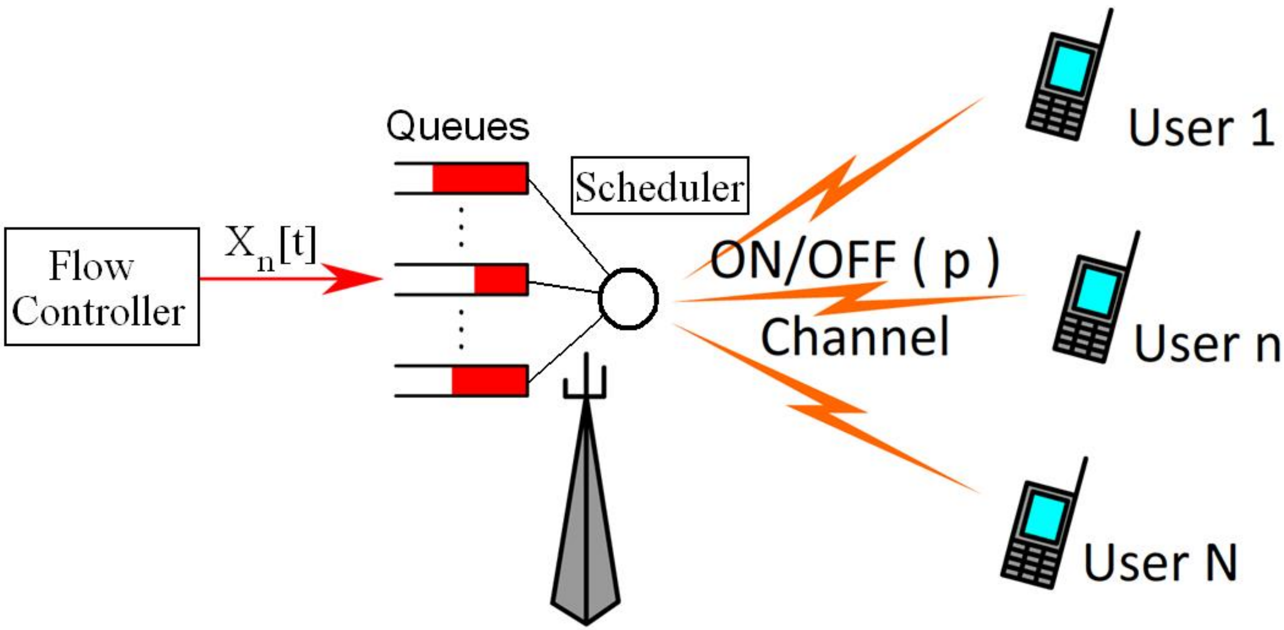
Outline

- Introduction
- Theory
  - Optimization-based modeling of resource allocation problems in general wireless networks
  - Systematic development of architectures and algorithms using dual decomposition techniques
- Applications
  - Modeling and Solution Methods for Resource Allocation in Wireless Networks
  - Efficient Architecture and Algorithm Design for
    - Long-Term Fairness (or Network Utility)
    - Intersession Network Coding amongst flows

Multihop wireless networks

- Wireless Communication is subject to
  - Variations in the channel quality (a.k.a. – Fading)
  - Interference from other transmissions
  - Limited resources – power, time, bandwidth, etc.
- Different types of traffic sharing the wireless network:
  - Unicast (single destination) and multicast (multiple destinations)
  - Short flows and long flows
  - Elastic (controllable rate) and Inelastic (fixed rate)
  - Real-time (with delay & jitter requirements) and non-real-time
- Need: Theory and methodologies for the systematic design of efficient network architectures and resource allocation algorithms to serve these different types of flows.

1) Flow Control & Scheduling under Fading



$C_n[t]$ = 1 if Channel n is ON at time t; 0 otherwise

$X_n[t]$: Number of incoming User n packets

$S_n[t]$ = 1 if Queue n is served at time t; 0 otherwise

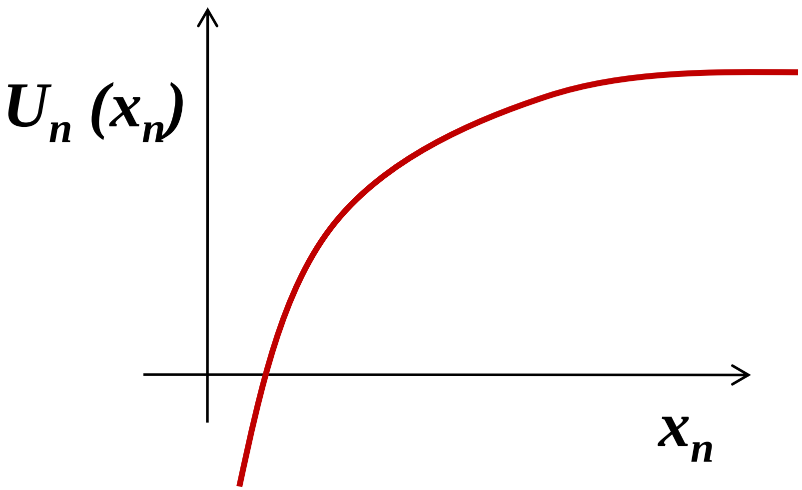
$Q_n[t]$: length of Queue n at the beginning of slot t

$$Q_n[t+1] = (Q_n[t] - C_n[t] \cdot S_n[t])^+ + X_n[t]$$

- Each channel state varies in time between ON and OFF states
- Only one transmission is allowed in every *time slot*
- Goal: to design a joint flow controller and scheduler to maximize the long-term network utility, subject to queue stability

Measuring Long-term Utility (or Fairness)

- Let x_n denote the average throughput provided to User n
- Then, $U_n(x_n)$ is a *utility function* that measures the long-term satisfaction (or preferences) of User n

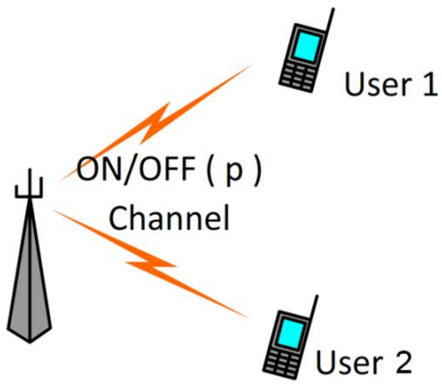


$$x_n := \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} E[X_n[t]]$$

$$\max_{\{\mathbf{X}[t],\mathbf{S}[t]\}} \quad \sum_{n=1}^{N} U_n(x_n)$$
$$s.t. \quad \text{Queue Stability}$$

- $U_n(x_n)$ is assumed to be concave and non-decreasing (law of diminishing returns)
- Can measure various forms of fairness (Mo and Walrand ['99])
- Examples: x_n , $log(x_n)$, $-1/x_n$, *etc.*

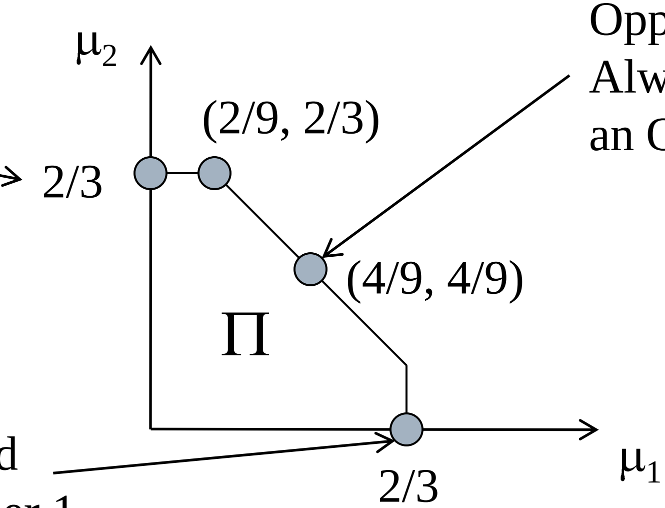
Understanding the Stability Region



- Suppose N=2 and p=2/3 in the previous setup
- What is the region of achievable service rates, Π ?
- μ_n : Mean service rate of User n, $E[S_n]$

Achieved when User 2 is always served.

Opportunistic Scheduling: Always serve a user with an ON Channel



Achieved when User 1 is always served.

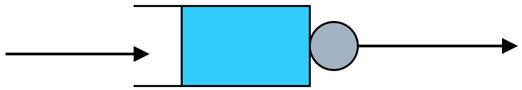
- Π is always convex, and usually difficult to compute.

Static Optimization Formulation of the Problem

- Static formulation of the queue stability condition:

Queue n

Mean injection rate: $\boldsymbol{x_n}$



Mean service rate: $\boldsymbol{\mu_n}$

$$x_n := \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} E[X_n[t]]$$

$$\mu_n := \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} E[S_n[t]]$$

Queue Stability ➔ $\boldsymbol{x_n \leq \mu_n}$

- We propose to solve:

$$\begin{aligned} \max_{x\geq 0} \quad & \sum_{n=1}^{N} U_n(x_n) \\ s.t. \quad & x_n \leq \mu_n, \qquad n = 1, \cdots, N, \\ & \mu \in \Pi \end{aligned}$$

- This is a static convex optimization problem with a *separable* objective function

Solution through Dual Decomposition

- Let q_n be the *Lagrange multiplier* associated with: $x_n \le \mu_n$
- Then, the *Lagrangian function* becomes

$$
\begin{aligned}
L(\mathbf{x}, \mathbf{q}) &= \sum_n U_n(x_n) - q_n(x_n - \mu_n) \\
&= \sum_n (U_n(x_n) - q_n x_n) + \sum_n q_n \mu_n
\end{aligned}
$$

- Hence, the *Dual Function* is

$$
\begin{aligned}
D(\mathbf{q}) &= \max_{\{\mathbf{x} \ge 0, \mu \in \Pi\}} \left[\sum_n (U_n(x_n) - q_n x_n) + \sum_n q_n \mu_n \right] \\
&= \sum_n \boxed{\max_{x_n \ge 0} (U_n(x_n) - q_n x_n)} + \boxed{\max_{\mu \in \Pi} \sum_n q_n \mu_n}
\end{aligned}
$$

Flow Controller: Given **q[t] = (**q_n**[t])**, Choose $X_n[t]$ to maximize this.

Scheduler: Given **q[t] = (**q_n**[t])**, Choose $S_n[t]$ to maximize this.

- Then, update **q[t]** in the direction of decreasing D(**q[t]**).

Flow Controller and Scheduling Algorithm

- It turns out that $q_n[t] \approx \varepsilon\, Q_n[t]$ for a design parameter $\varepsilon > 0$
- This suggests the following Flow Control and Scheduling Algorithm for the original stochastic system:

Flow Controller: The mean number of packets User *n* injects at slot *t* are:

$$E[X_n[t]] = \arg\max_{x_n \geq 0} (U_n(x_n) - \epsilon Q_n[t] x_n)$$

Scheduler: At slot *t*, assign services $S_n[t] \in \{0,1\}$ such that

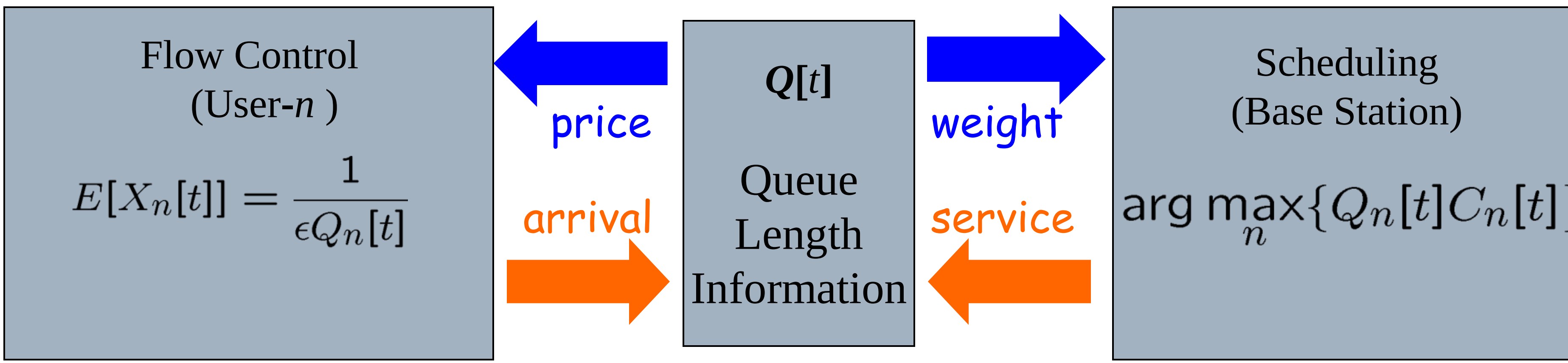
$$\mathbf{S}[t] \in \arg\max_{\{\sum_n S_n \leq 1\}} \sum_n S_n Q_n[t] C_n[t]$$

Queue Evolution: Update the queues

$$Q_n[t+1] = (Q_n[t] - C_n[t] \cdot S_n[t])^+ + X_n[t]$$

Example Scenario

- Suppose $U_n(x_n) = log(x_n)$
- Then
  - $E[X_n[t]] = 1/(\varepsilon\, Q_n[t])$, i.e., discouraged arrivals
  - Serve the user the Longest Connected Queue (LCQ) [Tassiulas, Ephremides`93]
- A useful functional visualization is



- This structure of critical information sharing through a pricing/ queueing mechanism appears repeatedly in allocation problems.

Optimality of the Stochastic Algorithm

Proof (Outline) [E., Srikant '05]: $\{(\mathbf{Q}[t])\}_t$ forms an irreducible, aperiodic Markov Chain

Foster – Lyapunov criterion : Suppose Markov Chain $Q[t]$ is irreducible and aperiodic. Let $V(Q)$ be a function such that $V(Q) \geq 0 \ \forall \ q$ and

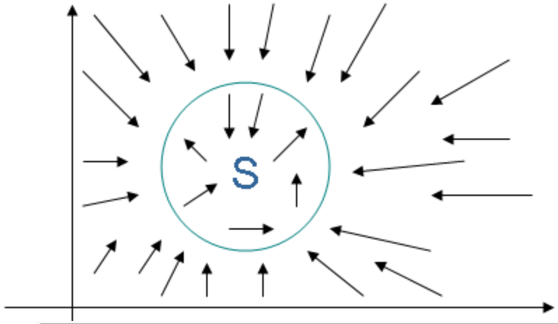
There exists a finite set S such that,

for $\Delta V(Q[t]) := V(Q[t+1])-V(Q[t])$

(i) $E[\Delta V(Q[t]) \mid Q[t]] \leq -\delta$ for $Q[t] \in S^c$

(ii) $E[\Delta V(Q[t]) \mid Q[t]] \leq M$ for $Q[t] \in S$

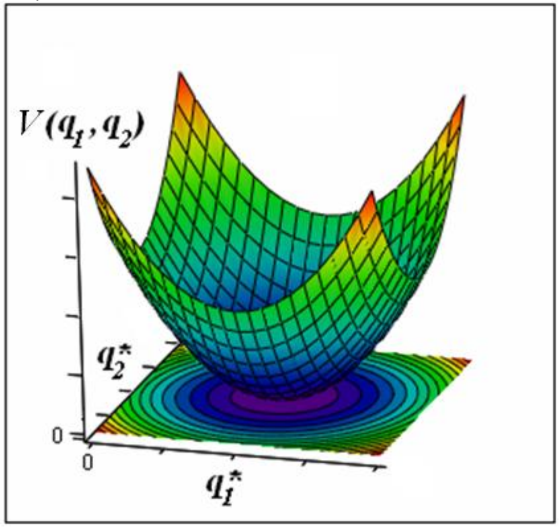
Then, the Markov Chain is positive recurrent (stable).



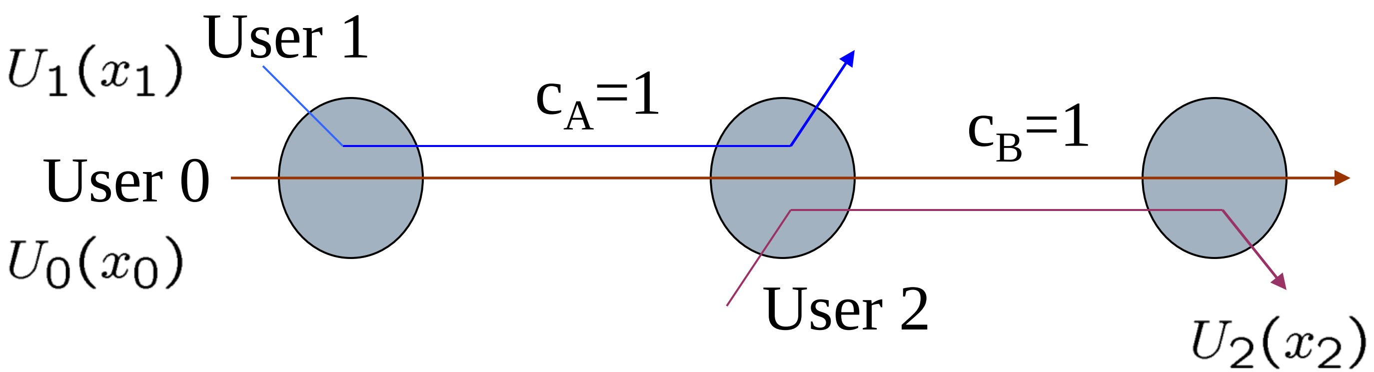
In our case, the typical Lyapunov function is

$$V(\mathbf{Q}) = \frac{1}{2}\sum_{\mathbf{n}}(\epsilon \mathbf{Q}_{\mathbf{n}} - \mathbf{q}^\star_{\mathbf{n}})^2,$$

where $\mathbf{q}^*$ is an optimal Lagrange multiplier of the static optimization problem .



2) Flow Control & Scheduling in Multi-hop Wireless Networks



Either link A or link B can be active, but not both.

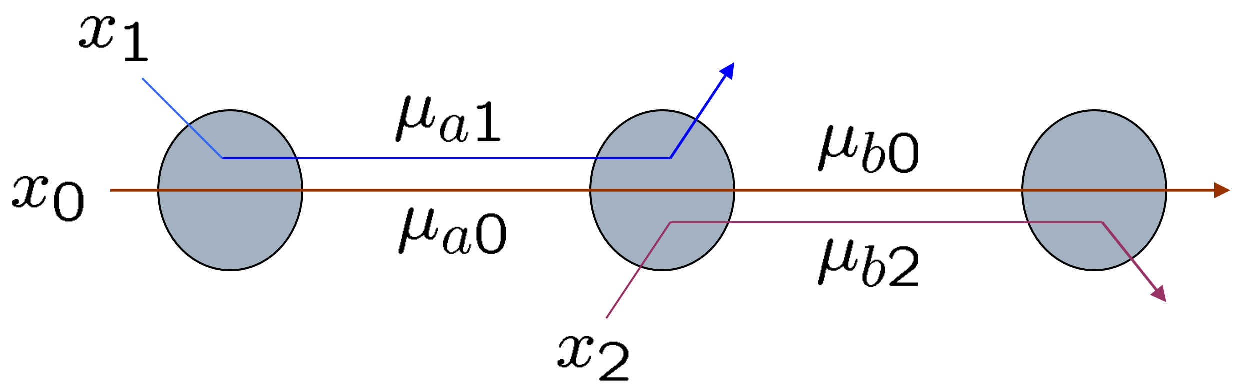
$$\max_{x,\mu\geq 0} \sum_i U_i(x_i)$$

subject to

$$\begin{aligned} x_0 &\leq \mu_{a0} \\ x_1 &\leq \mu_{a1} \\ \mu_{a0} &\leq \mu_{b0} \\ x_2 &\leq \mu_{b2} \\ \mu_{a0}+\mu_{a1}+\mu_{b0}+\mu_{b1} &\leq 1 \end{aligned}$$

μ_{a0} is the fraction of time link A is used for user 0

Lagrange Multipliers



$$\max_{x,\mu} \sum_i U_i(x_i) - q_{a0}(x_0 - \mu_{a0}) - q_{a1}(x_1 - \mu_{a1})$$
$$-q_{b0}(\mu_{a0} - \mu_{b0}) - q_{b2}(x_2 - \mu_{b2})$$

subject to
$$\begin{aligned} \mu_{a0} + \mu_{a1} + \mu_{b0} + \mu_{b2} &\leq 1 \\ x, \mu &\geq 0 \end{aligned}$$

Lagrangian Decomposition

Congestion control:

$$\max_{x \geq 0} \sum_i U_i(x_i) - q_{a0}x_0 - q_{a1}x_1 - q_{b2}x_2$$

$\Rightarrow$ User 0:

$$\max_{x_0 \geq 0} U_0(x_0) - q_{a0}x_0$$

User 1:

$$\max_{x_1 \geq 0} U_1(x_1) - q_{a1}x_1$$

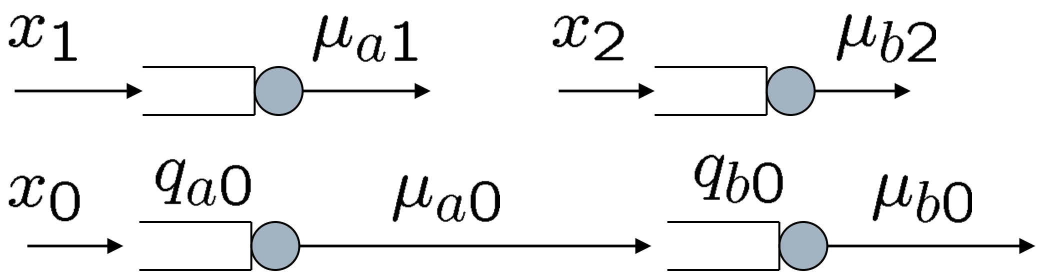
…

MAC or Scheduling:

$$\max_{\sum \mu_i \leq 1} \mu_{a0}(q_{a0} - q_{b0}) + \mu_{b0}q_{b0} + \mu_{a1}q_{a1} + \mu_{b2}q_{b2}$$

Solution is an extreme point!

Resource Constraints and Queue Dynamics



$$\max_{x,\mu \geq 0} \sum_i U_i(x_i)$$

subject to

$$\begin{aligned} x_0 &\leq \mu_{a0} \\ \dot{q}_{a0} &= x_0 - \mu_{a0} \\ \mu_{a0} &\leq \mu_{b0} \\ \dot{q}_{b0} &= \mu_{a0} - \mu_{b0} \\ &\vdots \end{aligned}$$

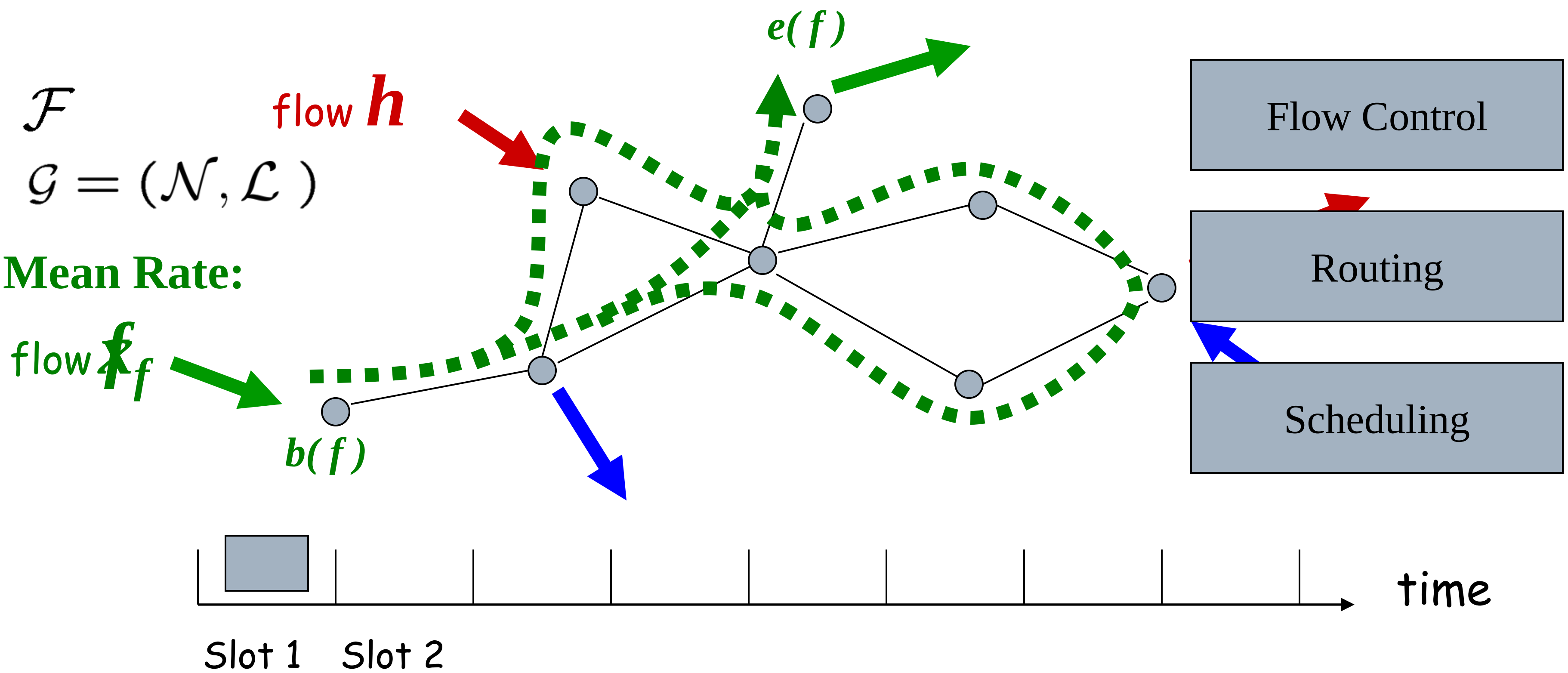
- Lagrange multipliers ≈ Queue lengths
- Arrival rate into a queue is departure rate from previous queue

3) Joint Flow Control, Scheduling and Routing

$\mathcal{F}$

$\mathcal{G} = (\mathcal{N}, \mathcal{L})$

Mean Rate:

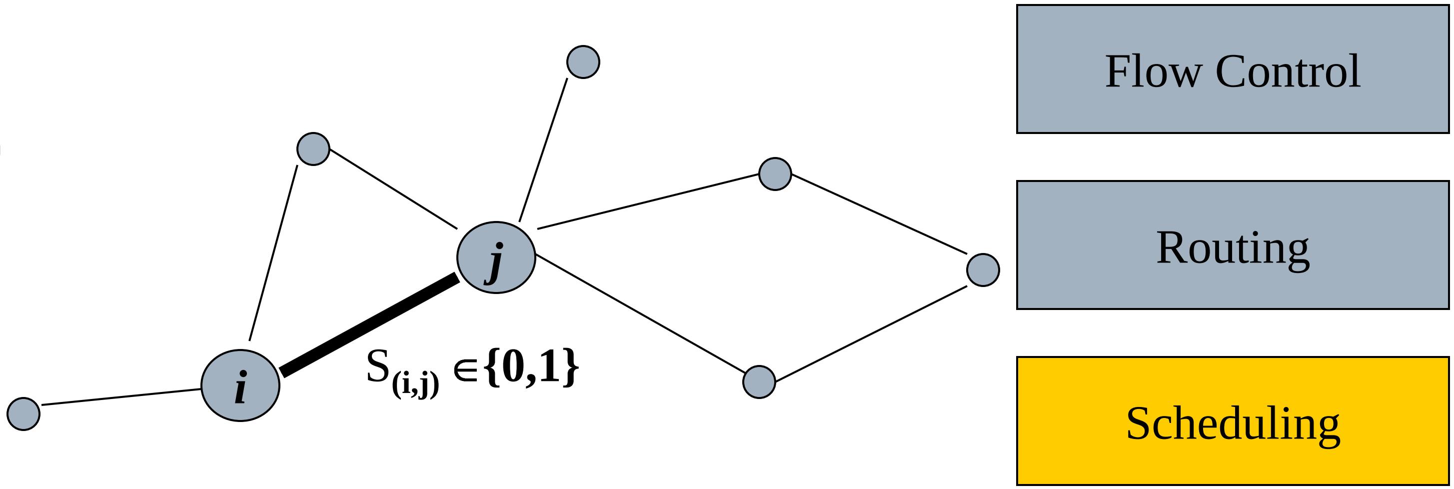


- $U_f(\cdot)$ is a concave, non-decreasing function that measures the utility of Flow-*f* as a function of its mean rate

Joint Flow Control, Scheduling and Routing

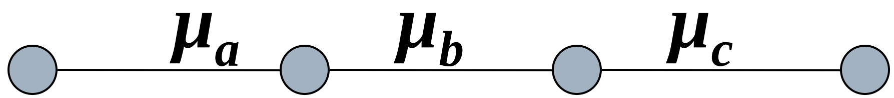
$\mathcal{F}$

$\mathcal{G} = (\mathcal{N}, \mathcal{L})$

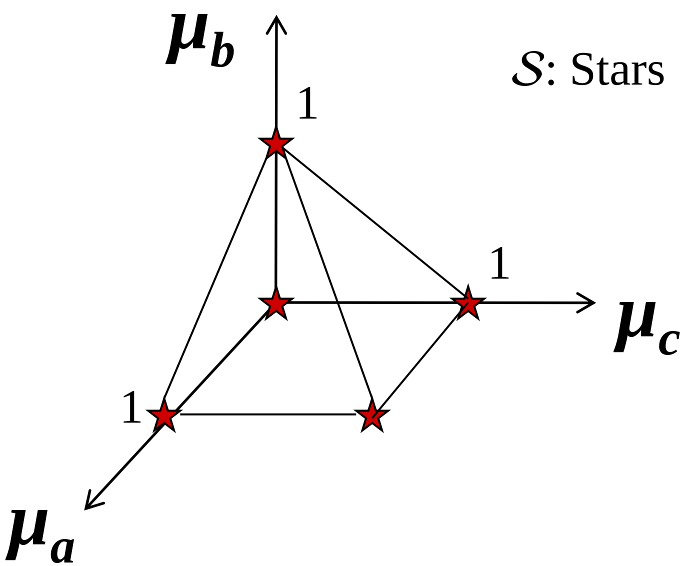


- $\mathcal{S}$: Set of *feasible link activation vectors* (or *feasible schedules*)
- Schedule of slot t, denoted $\mathbf{S}[t] = (S_{(i,j)}[t])_{(i,j)\in\mathcal{L}}$, must be in $\mathcal{S}$, $\forall t$
- Π = Convex Hull$\{\mathcal{S}\}$: *Achievable mean link rates*
- A *scheduling policy P* is a mapping from the current "state" of the system to feasible schedules
- Let $\mathcal{P}$ denote the *set of all scheduling policies*

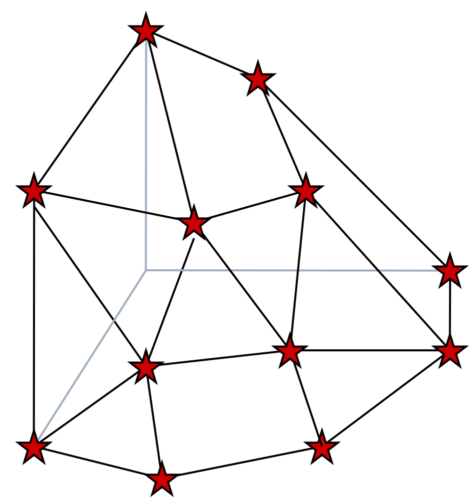
Example on Π - Primary Interference



<u>Primary interference model</u>:
Any two active links must be separated by ≥1 link



<u>Kth order interference model</u>:
Any two active links must be separated by ≥ K links



- In general Π is a complex set of rate allocations that depends on the topology and interference model

Definitions

- A queue, say q_i^d, is *stable* if

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} q_i^d[t] < \infty$$

- A *queue-length based flow control policy* $X : \mathbf{q} \rightarrow [0, M]^{|\mathcal{F}|}$ is a mapping from queue-lengths to feasible rates
- Let $\mathcal{X}$ denote the set of all queue-length-based flow control policies
- Then, the queue-length evolution for a given scheduling policy P, can be written as

$$\mathbf{q}[t+1] = h(\mathbf{q}[t], P, X(\mathbf{q}[t])),$$

for some function h

Translation of Π to Λ

Achievable mean link rates $\boxed{\Pi} \xrightarrow{\text{Set of Flows}} \boxed{\Lambda}$ ***Network Stability Region***

- $\boldsymbol{x} = (x_{b(f)}^{e(f)})_f \geq \mathbf{0} \in \Lambda$ iff
  - there exists a $\mu \in \Pi$ for which we have:

$$x_i^d + \mu_{into(i)}^d \leq \mu_{out(i)}^d, \qquad d, i \in \mathcal{N}$$

(Flow Conservation Constraints)

- Our goal can then be posed as

$$\begin{aligned} x^\star \in \arg\max_{x,\mu} \quad & \sum_{f \in \mathcal{F}} U_f(x_f) \\ s.t. \quad & x_i^d + \mu_{into(i)}^d \leq \mu_{out(i)}^d, \qquad d, i \in \mathcal{N} \\ & \mu \in \mathbf{\Pi} \end{aligned}$$

Dual Decomposition

- Implied architecture: $Q_i^d[t]$ for packets at node *i*, destined to *d*
- A ***Dual function*** associated with the previous problem is

$$D(\mathbf{q}) = \sum_{f \in \mathcal{F}} \boxed{\max_{x_f \geq 0} \{U_f(x_f) - x_f q_{b(f)}^{e(f)}\}}$$

Distributed Flow Control

$$+ \boxed{\max_{\mu \in \mathbf{\Pi}} \sum_{(i,j) \in \mathcal{L}} \mu_{(i,j)} \max_{d \in \mathcal{N}} \left\{ q_i^d - q_j^d \right\}}$$

Backpressure

Scheduler/Router

where we λ_i^d can be interpreted as the price associated with sending a unit rate of flow from node *i* to node *d*.

- Then the ***Dual Problem*** is given as: $\min_{\lambda \geq \mathbf{0}} D(\lambda)$
- **Fact**: There is no duality gap and there exists a nonempty set Ψ^* such that:

$$\sum_{f \in \mathcal{F}} U_f(x_f^\star) = D(\lambda^\star), \qquad \forall \lambda^\star \in \mathbf{\Psi}^\star$$

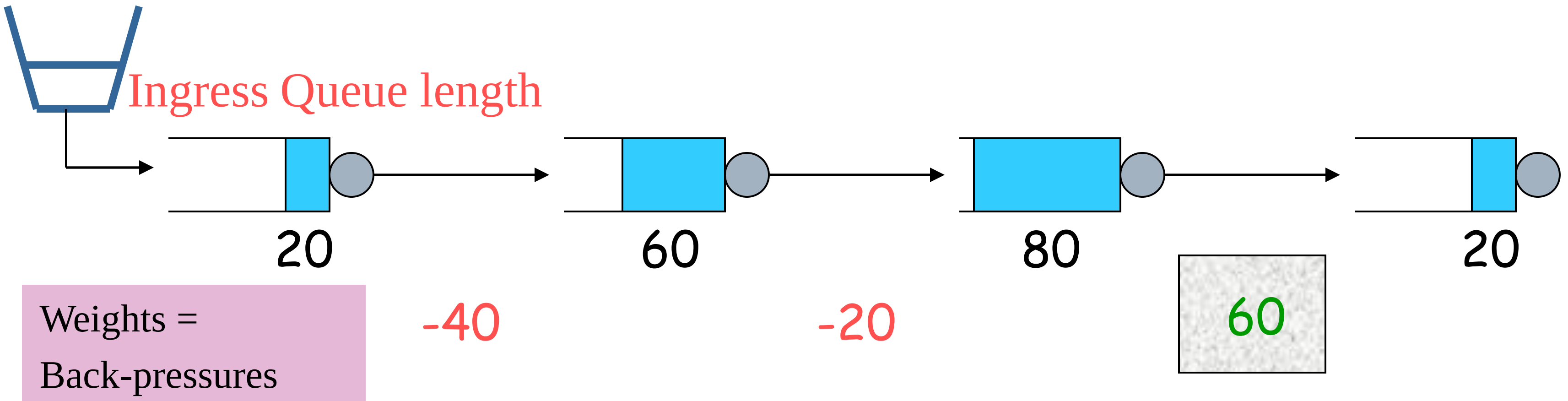
Sub-gradient Methods to Solve NUM

- Employ Dual (or Primal-Dual) Methods:

$$q_i^d[t+1] = \left[q_i^d[t] + \epsilon\left(x^{(d)}_{into(i)}[t] + \mu^{(d)}_{into(i)}[t] - \mu^{(d)}_{out(i)}[t]\right)\right]^+$$

$$x_f[t] = \left[U_f'^{-1}\left(q^{e(f)}_{b(f)}[t]\right)\right]^+$$

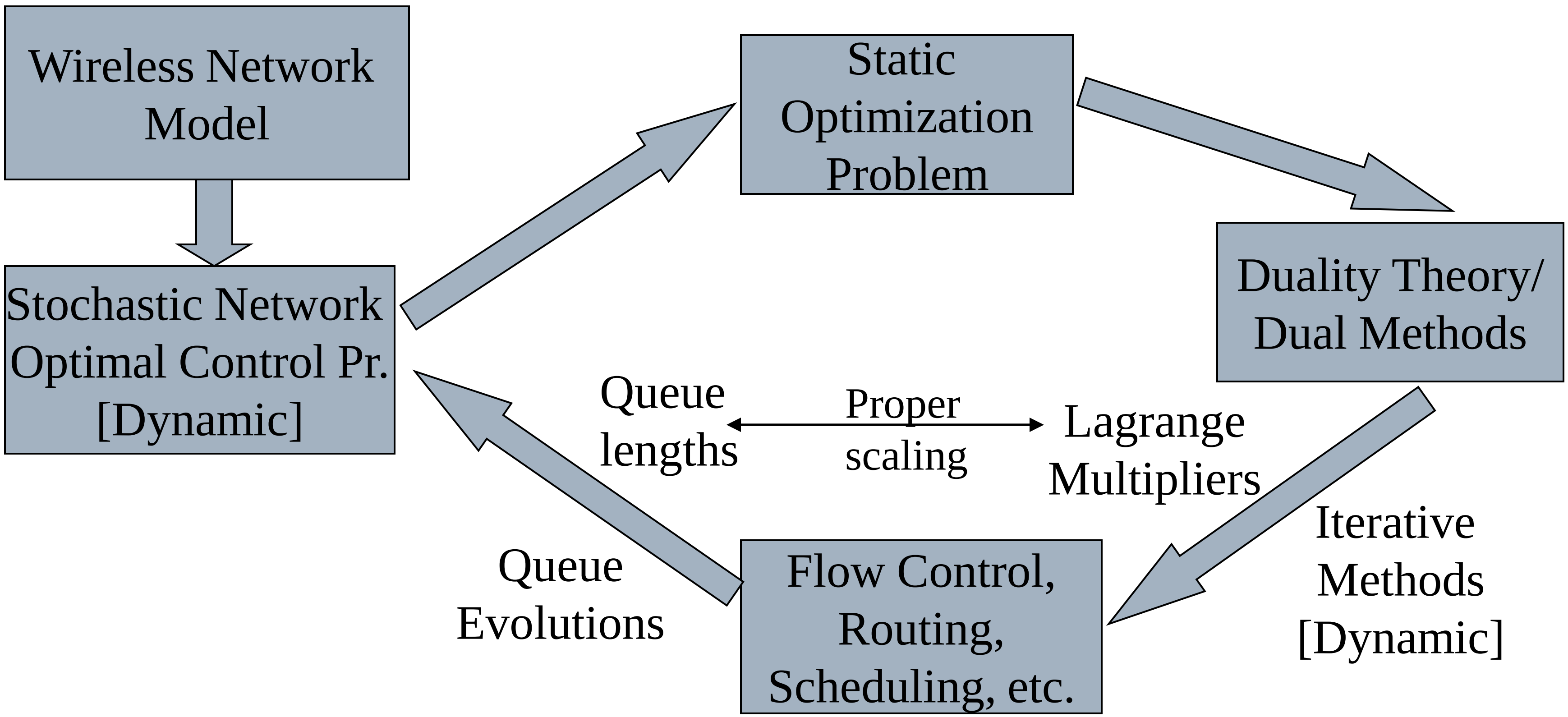
where $\varepsilon > 0$ is the step-size.



- Then, using results from optimization theory, we have, under appropriate step-size rules,

$$q[t] \rightarrow \Psi^*, \text{ and } x[t] \rightarrow x^*$$

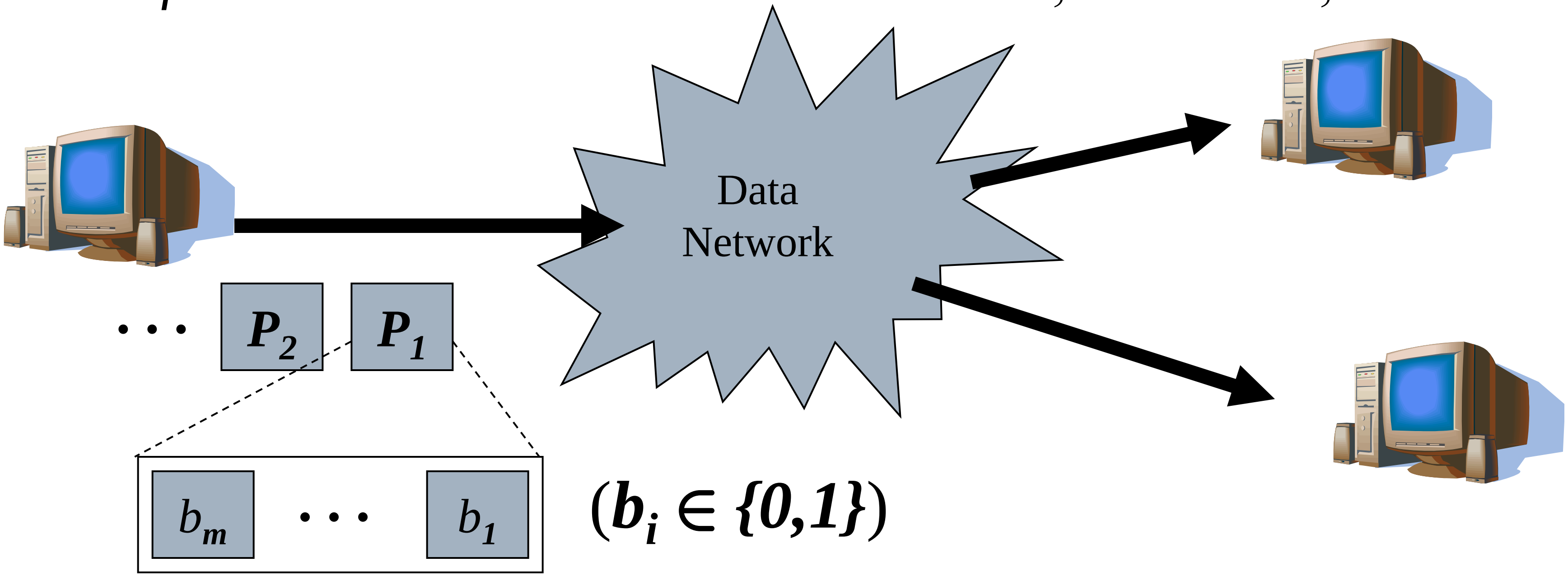
A Summary of the Design Methodology



- This gives a systematic approach to developing architectures and algorithms for stochastic wireless networks

4) Architecture and Algorithm for Intersession Network Coding

- Data Communications convey *information*
- *information* = *bits* and bits can be added, subtracted,…



- **IDEA**: exploit the algebraic nature of information to increase utilization of network resources

Foundation of Network Coding – Single Session

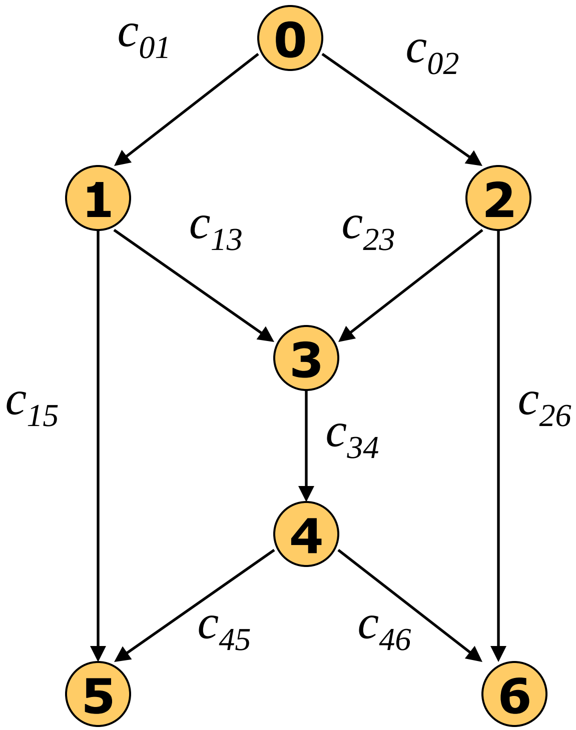
R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow", *IEEE Transactions on Information Theory*, IT-46, pp. 1204-1216, 2000

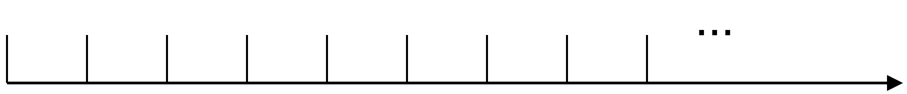
- Network with a single multicast:

$$(0) \Rightarrow (5, 6)$$

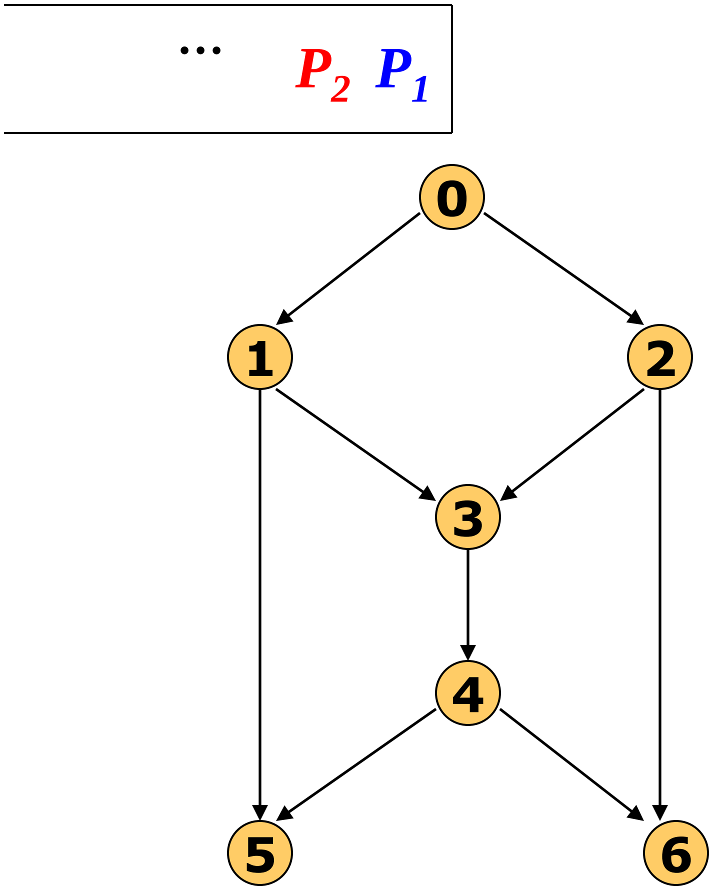
- Constant link rates:

$$c_{i,j} = 1 \text{ packet/slot}, \ \forall i,j.$$





Foundation of Network Coding – Single Session



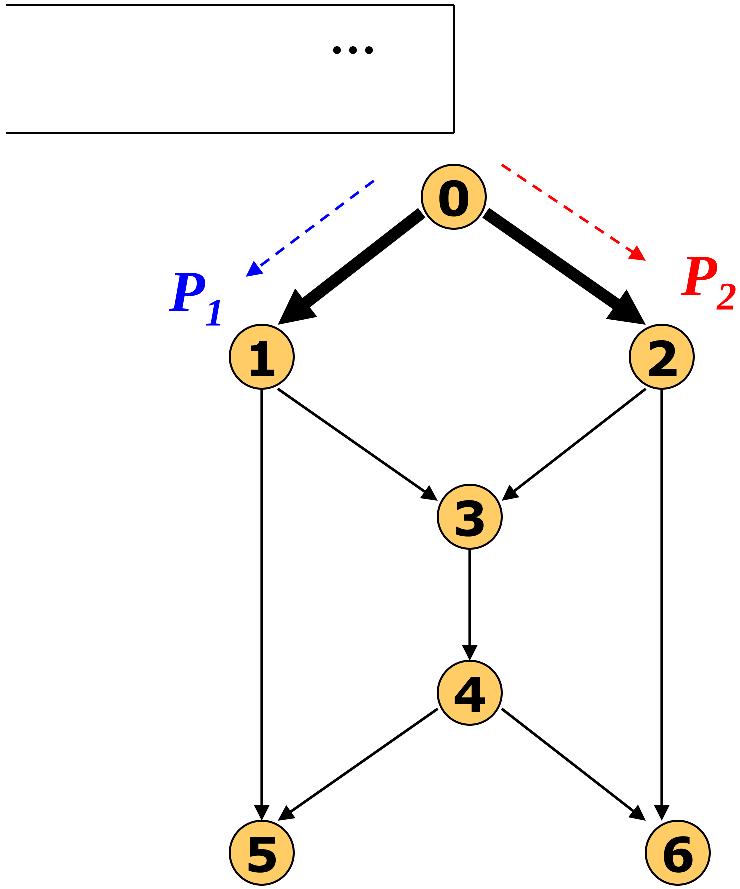
- Network with a single multicast:

$$(0) \Rightarrow (5, 6)$$

- Constant link rates:

$$c_{i,j} = 1 \text{ packet/slot}, \ \forall i,j.$$

Foundation of Network Coding – Single Session



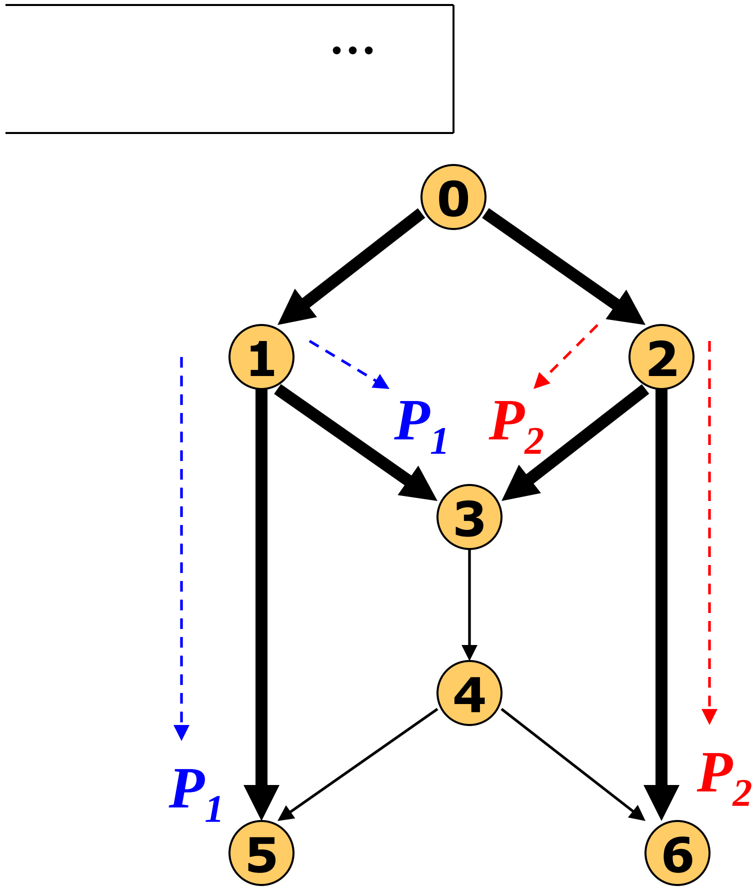
- Network with a single multicast:

$$(0) \Rightarrow (5, 6)$$

- Constant link rates:

$$c_{i,j} = 1 \text{ packet/slot}, \ \forall i,j.$$

Foundation of Network Coding – Single Session



- Network with a single multicast:

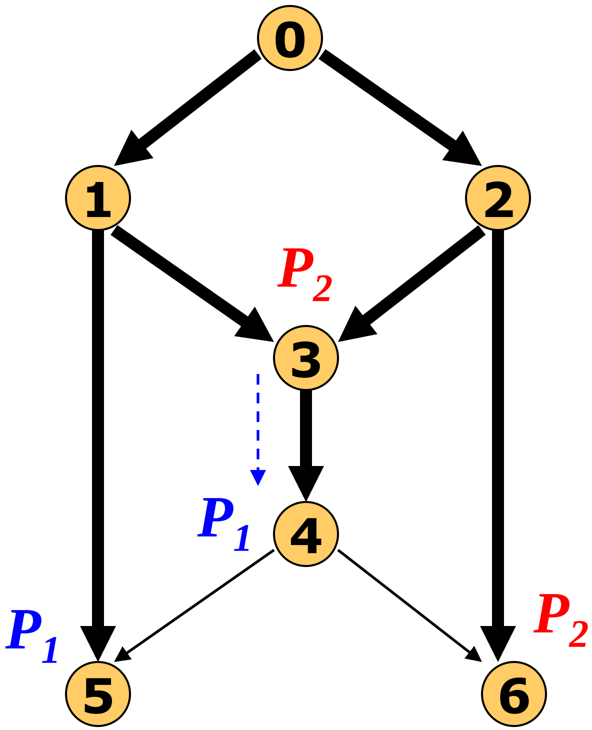
$$(0) \Rightarrow (5, 6)$$

- Constant link rates:

$$c_{i,j} = 1 \text{ packet/slot}, \ \forall i,j.$$

Foundation of Network Coding – Single Session

…



- Network with a single multicast:

$$(0) \Rightarrow (5, 6)$$

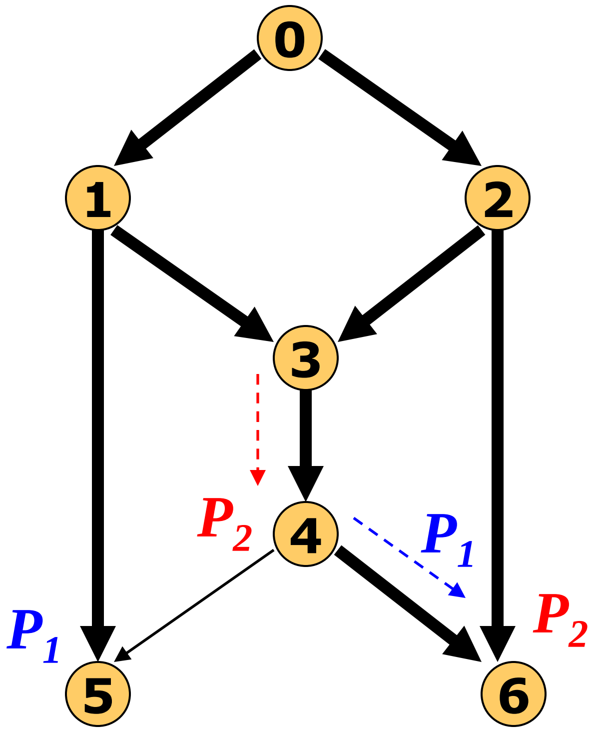
- Constant link rates:

$$c_{i,j} = 1 \text{ packet/slot}, \ \forall\, i,j.$$

- If only Routing is allowed…

Foundation of Network Coding – Single Session

…



- Network with a single multicast:

$$(0) \Rightarrow (5, 6)$$

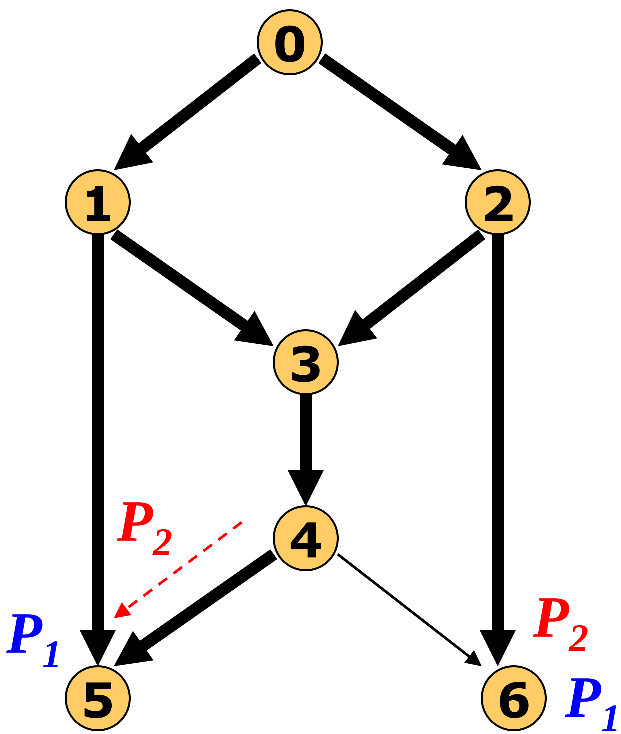
- Constant link rates:

$$c_{i,j} = 1 \text{ packet/slot}, \ \forall i,j.$$

- If only Routing is allowed…

Foundation of Network Coding – Single Session

…



- Network with a single multicast:

$$(0) \Rightarrow (5, 6)$$

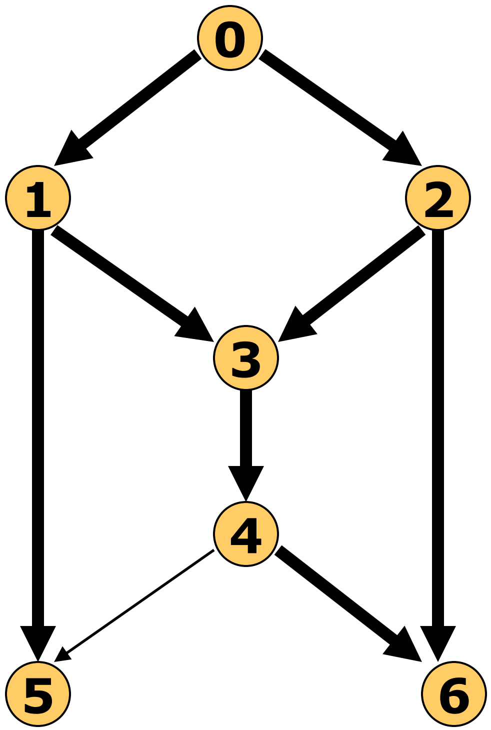
- Constant link rates:

$$c_{i,j} = 1 \text{ packet/slot}, \;\; \forall\, i,j.$$

- If only Routing is allowed…

Foundation of Network Coding – Single Session

…



- Network with a single multicast:

$$(0) \Rightarrow (5, 6)$$

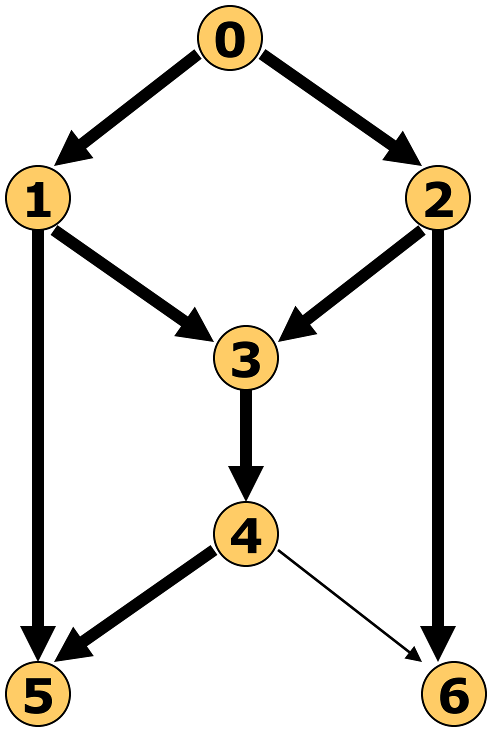
- Constant link rates:

$$c_{i,j} = 1 \text{ packet/slot}, \;\; \forall\, i,j.$$

- If only Routing is allowed…

Foundation of Network Coding – Single Session

…



- Network with a single multicast:

  $(0) \Rightarrow (5, 6)$

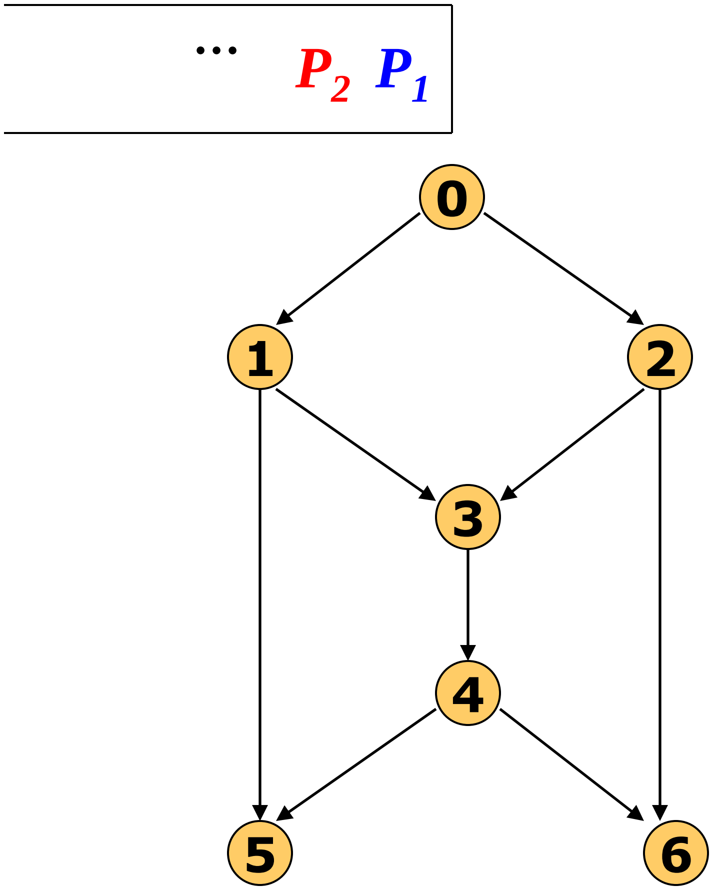
- Constant link rates:

  $c_{i,j} = 1$ ***packet/slot,*** $\forall\, i,j.$

- If only Routing is allowed…

  - Each receiver gets ***1.5 packets per slot.***
  - Link ***(3,4)*** is the bottleneck

Foundation of Network Coding – Single Session



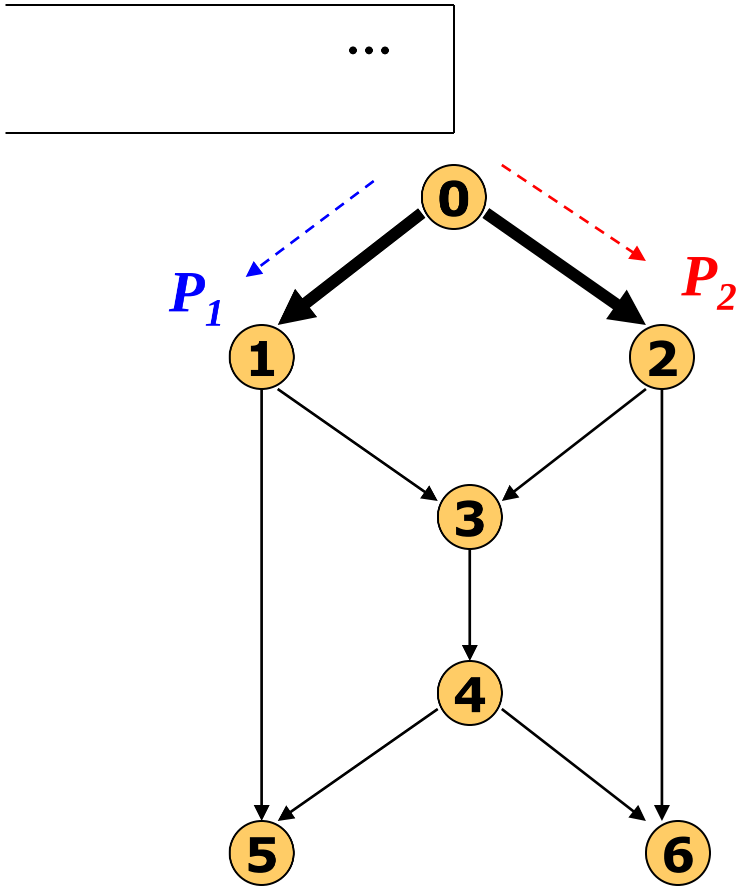
- Network with a single multicast:

$$(0) \Rightarrow (5, 6)$$

- Constant link rates:

$$c_{i,j} = 1 \text{ packet/slot}, \ \forall\, i,j.$$

Foundation of Network Coding – Single Session



- Network with a single multicast:

$$(0) \Rightarrow (5, 6)$$

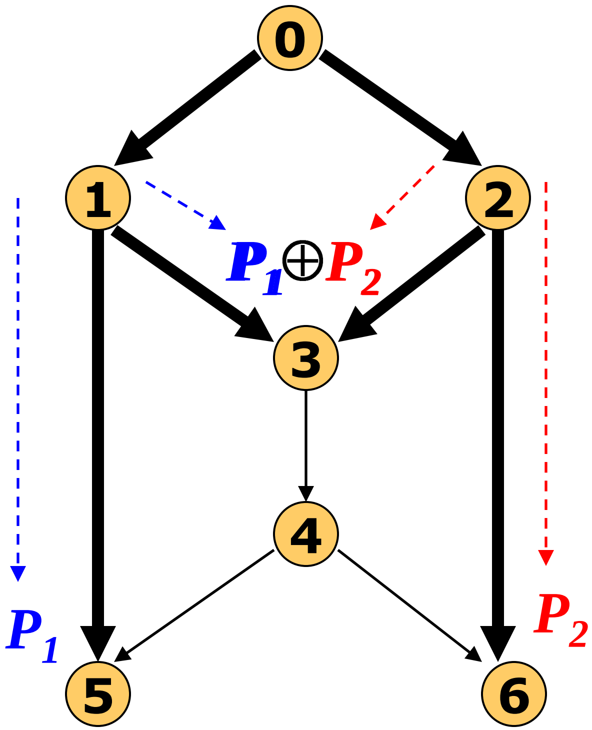
- Constant link rates:

$$c_{i,j} = 1 \text{ packet/slot}, \;\; \forall\, i,j.$$

- If Coding is allowed…

Foundation of Network Coding – Single Session

…



- Network with a single multicast:

$$(0) \Rightarrow (5, 6)$$

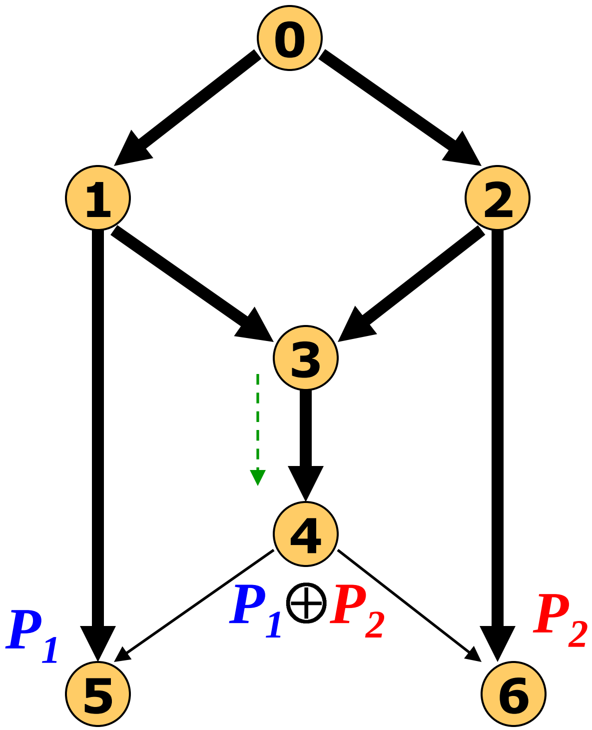
- Constant link rates:

$$c_{i,j} = 1 \text{ packet/slot}, \ \forall i,j.$$

- If Coding is allowed…

Foundation of Network Coding – Single Session

…



- Network with a single multicast:

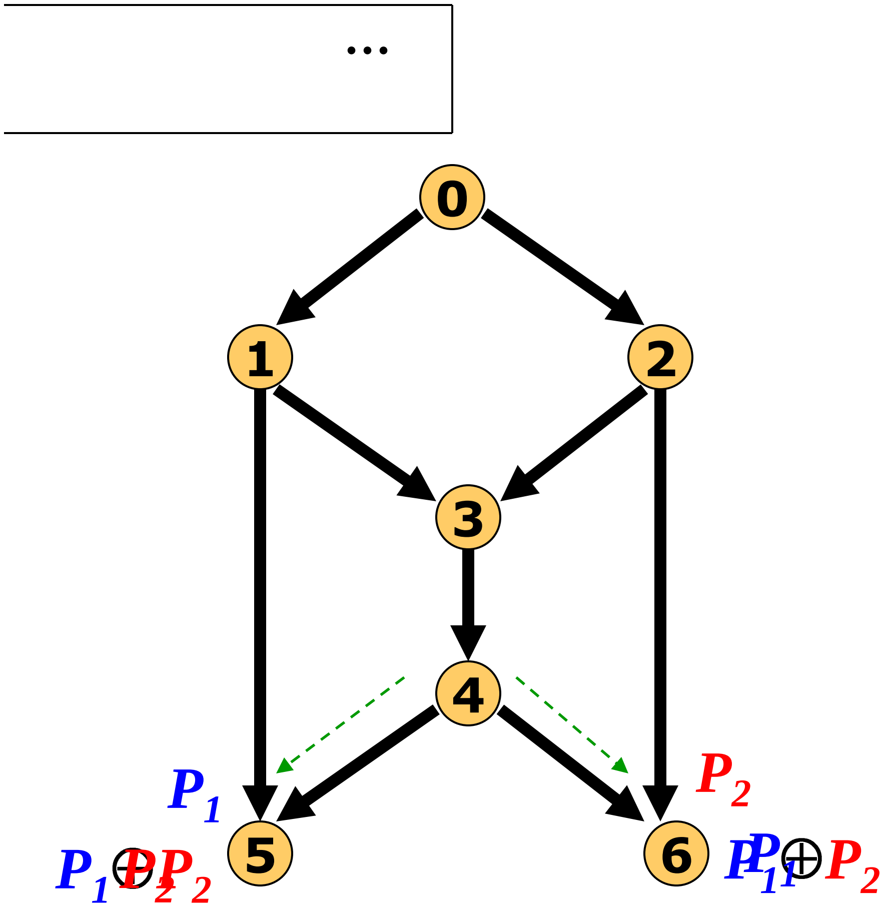
$$(0) \Rightarrow (5, 6)$$

- Constant link rates:

$$c_{i,j} = 1 \text{ packet/slot}, \ \forall i,j.$$

- If Coding is allowed…

Foundation of Network Coding – Single Session



- Network with a single multicast:

$$(0) \Rightarrow (5, 6)$$

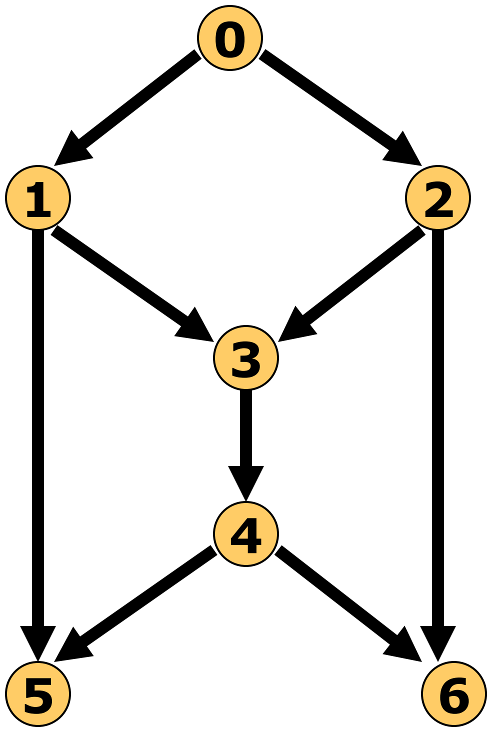
- Constant link rates:

$$c_{i,j} = 1 \text{ packet/slot}, \ \forall\, i,j.$$

- If Coding is allowed…

Foundation of Network Coding – Single Session

…



- Network with a single multicast:

$$(0) \Rightarrow (5, 6)$$

- Constant link rates:

$$c_{i,j} = 1 \text{ packet/slot}, \ \forall\, i,j.$$

- If Coding is allowed…

  - Each receiver gets **_2 packets per slot_**.
  - Link ***(3,4)*** is no longer the bottleneck.

Background

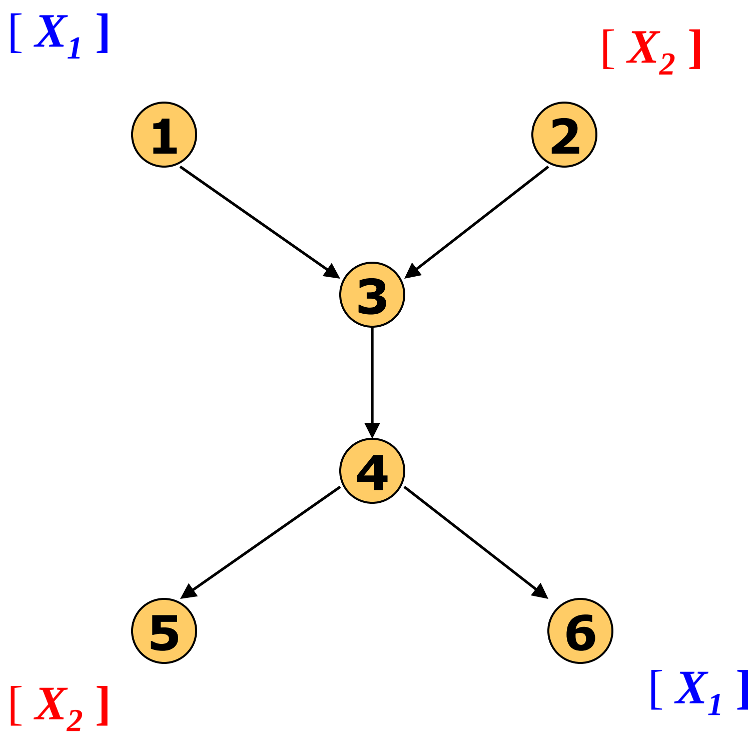
- By allowing algebraic **mixing** of packets, network coding can increase the session throughput
- [Ahlswede et al. (IT `00)] proved that for a single multicast session, network coding achieves the maximum possible rate allowed in the network
- [Koetter and Médard (ToN `03)] put network coding in a beautiful algebraic framework
- Nice random/deterministic algorithms exist for serving a single multicast session [Ho et al. (Thesis `04), Jaggi et al. (IT `05)]
- Linear network coding is sufficient to achieve maximum rate for a single multicast [Li et al. (IT `03)]

ALL FOR A SINGLE SESSION NETWORK CODING !!

Network Coding – Multiple Sessions

- If multiple sessions exist in the network, should we always code across sessions?

NO!



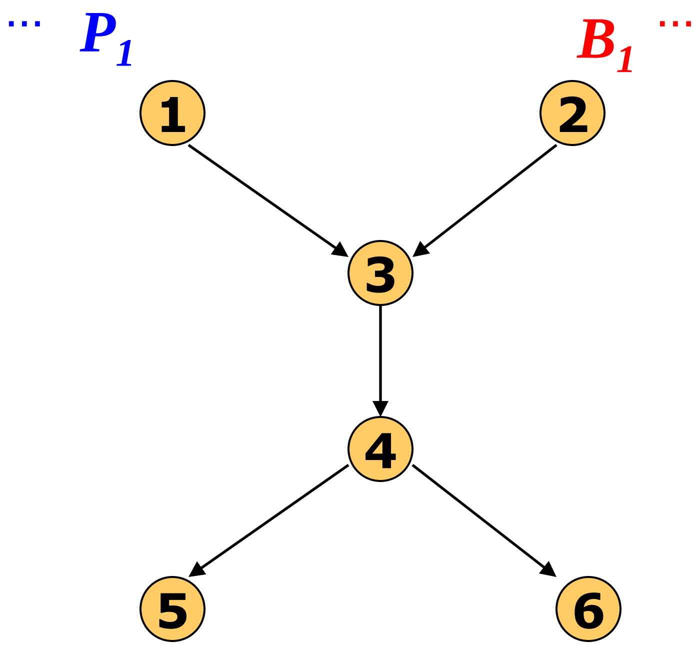
- (1) $\Rightarrow$ (6)
- (2) $\Rightarrow$ (5)

- $c_{15} = c_{26} = 0$

Network Coding – Multiple Sessions

- If multiple sessions exist in the network, should we always code across sessions?

NO!

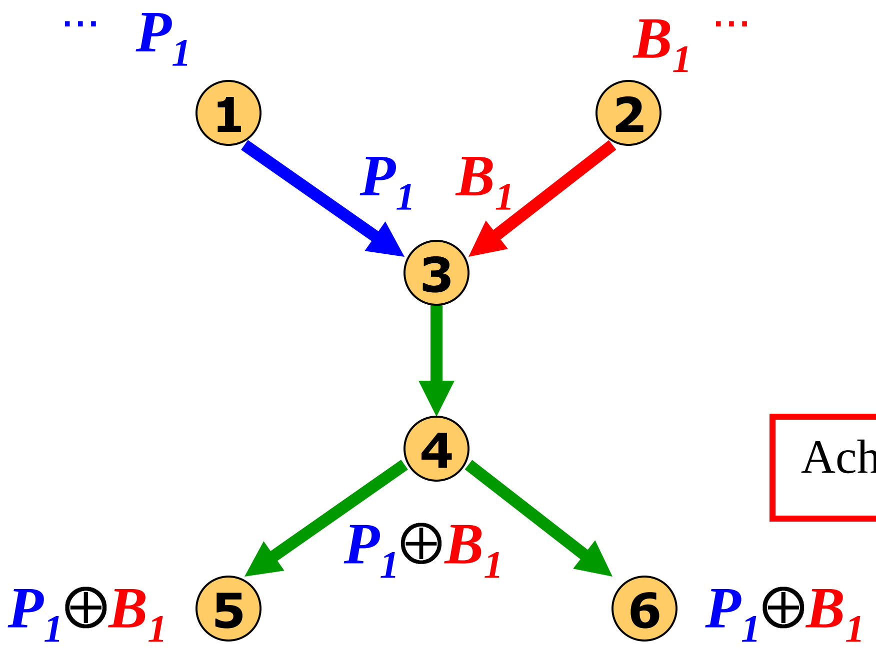


- (1) ⇒ (6)
- (2) ⇒ (5)
- $c_{15} = c_{26} = 0$

Network Coding – Multiple Sessions

- If multiple sessions exist in the network, should we always code across sessions?

NO!

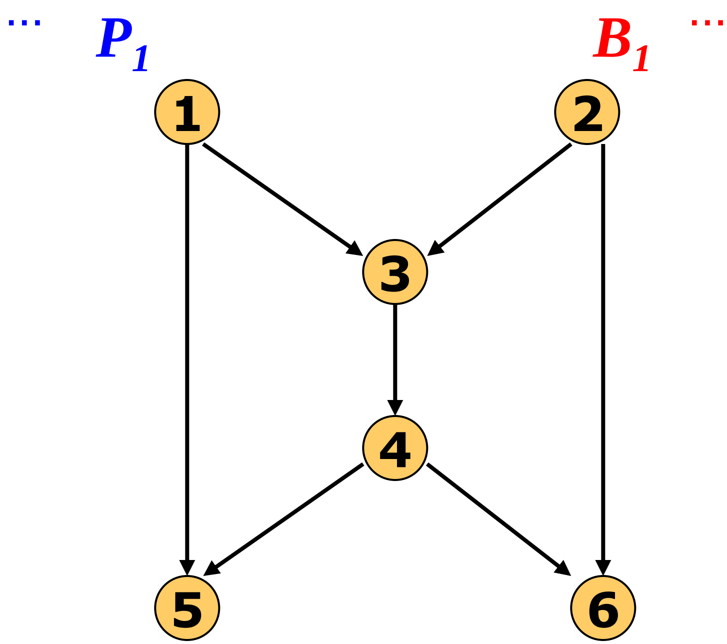


- *(1) ⇒ (6)*
- *(2) ⇒ (5)*
- $c_{15} = c_{26} = 0$

Achieved rate = ***ZERO !***

Network Coding – Multiple Sessions

- Should we never code across sessions?

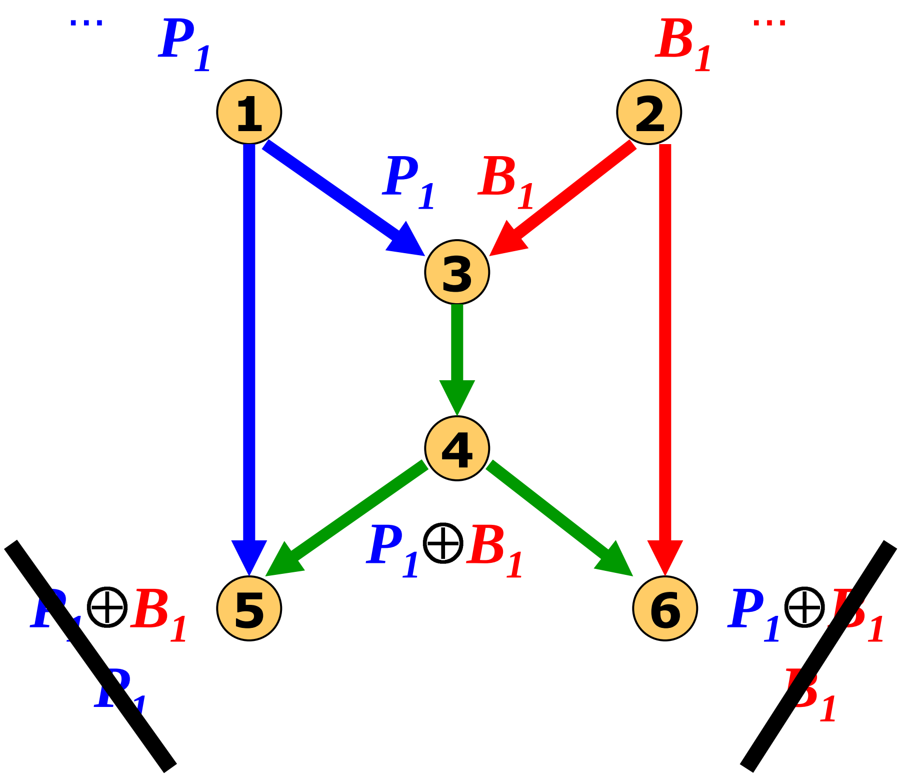


NO!

- $(1) \Rightarrow (6)$
- $(2) \Rightarrow (5)$
- $c_{15} = c_{26} = 1$

Network Coding – Multiple Sessions

- Should we never code across sessions?

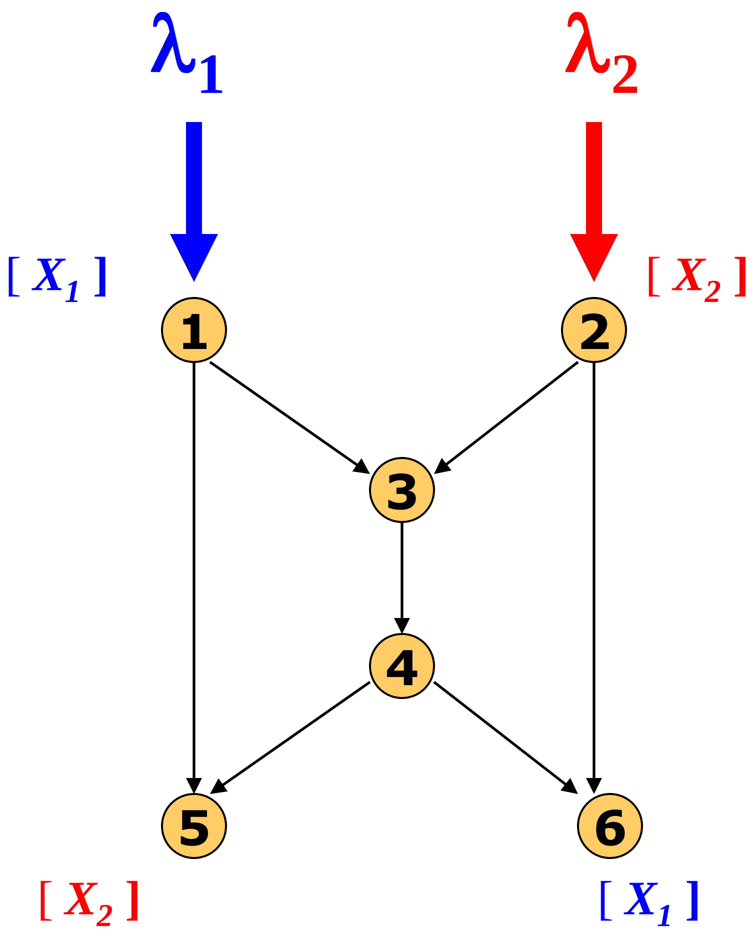


NO!

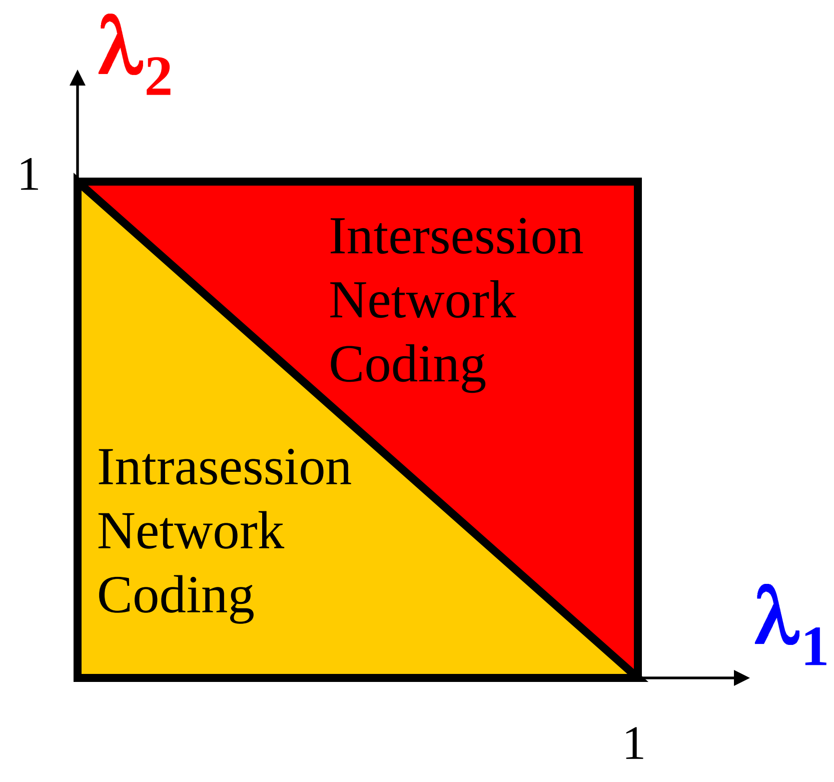
- *(1) ⇒ (6)*
- *(2) ⇒ (5)*
- $c_{15} = c_{26} = 1$

Achieved rate =
1 packet/slot/session

Illustration of Gains for the Butterfly



- $(1) \Rightarrow (6)$ & $(2) \Rightarrow (5)$
- c_{ij} = 1 for all (i, j).



Goal

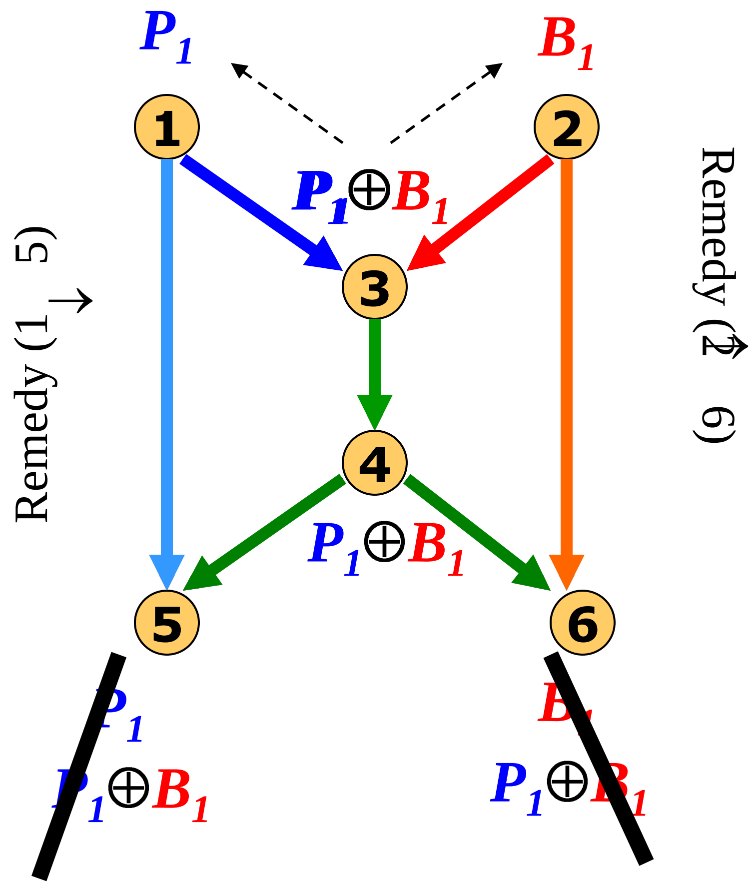
- Multiple sessions must co-exist
- Considerable throughput gains can be achieved through coding across sessions [Katabi et al. (Allerton `05)]
- We aim to
  - develop practical methods for making intersession coding decisions for general stochastic networks with unknown topology or statistics
  - guarantee provably good performance

Intersession Network Coding - Challenges

- The topology of the network is not known
- The link quality fluctuates with an unknown mean
- Session arrivals are unknown and stochastic
- The problem of intersession coding is **difficult to solve even for a genie** that has all the information about the network and sessions!!
- The capacity region is inter-session coding is not known

Observation – Effect of Coding

- *(1)* ⇒ *(6)*
- *(2)* ⇒ *(5)*



- Coding at ***(3)*** creates:

- One ***Poisoned*** Multicast

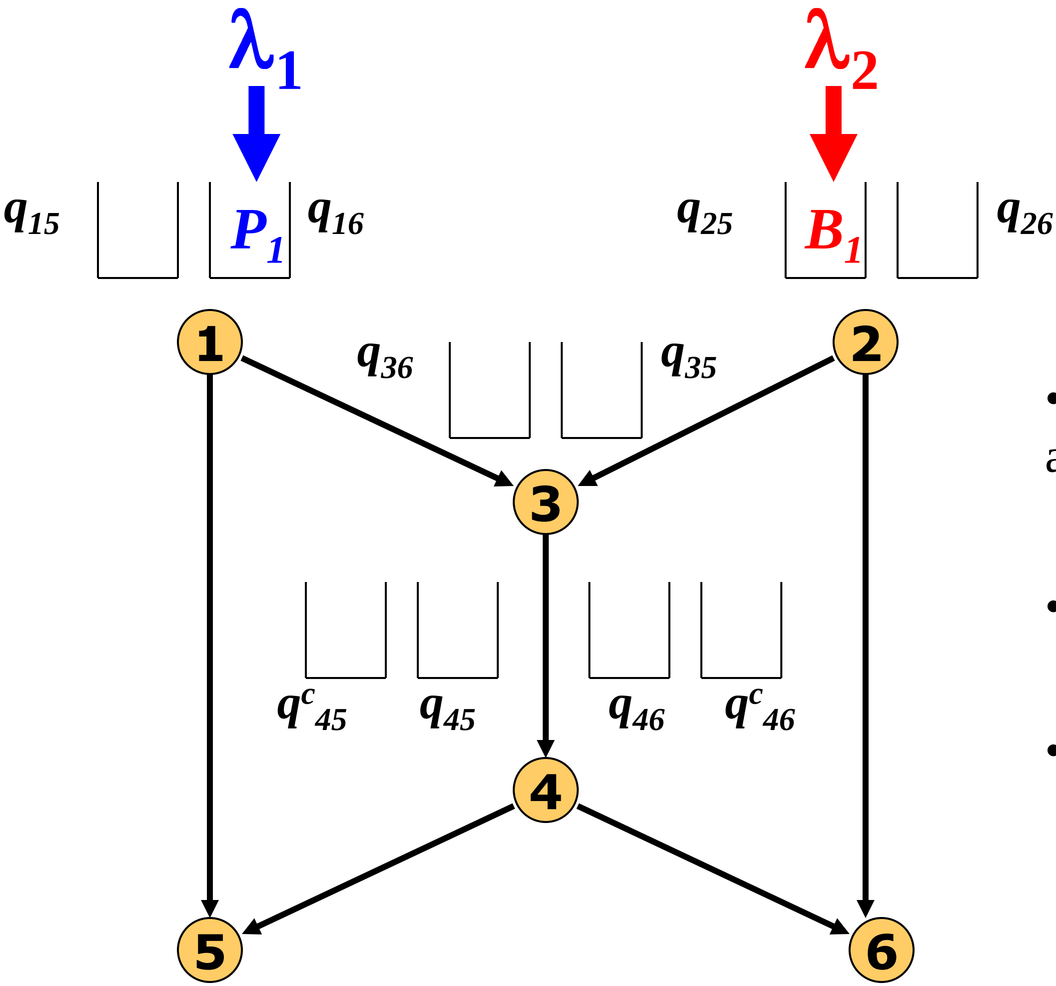
  (3) ⇒ ***(5, 6)***

- Two ***Remedy*** Unicasts

  (1) ⇒ ***(5)***

  (2) ⇒ ***(6)***

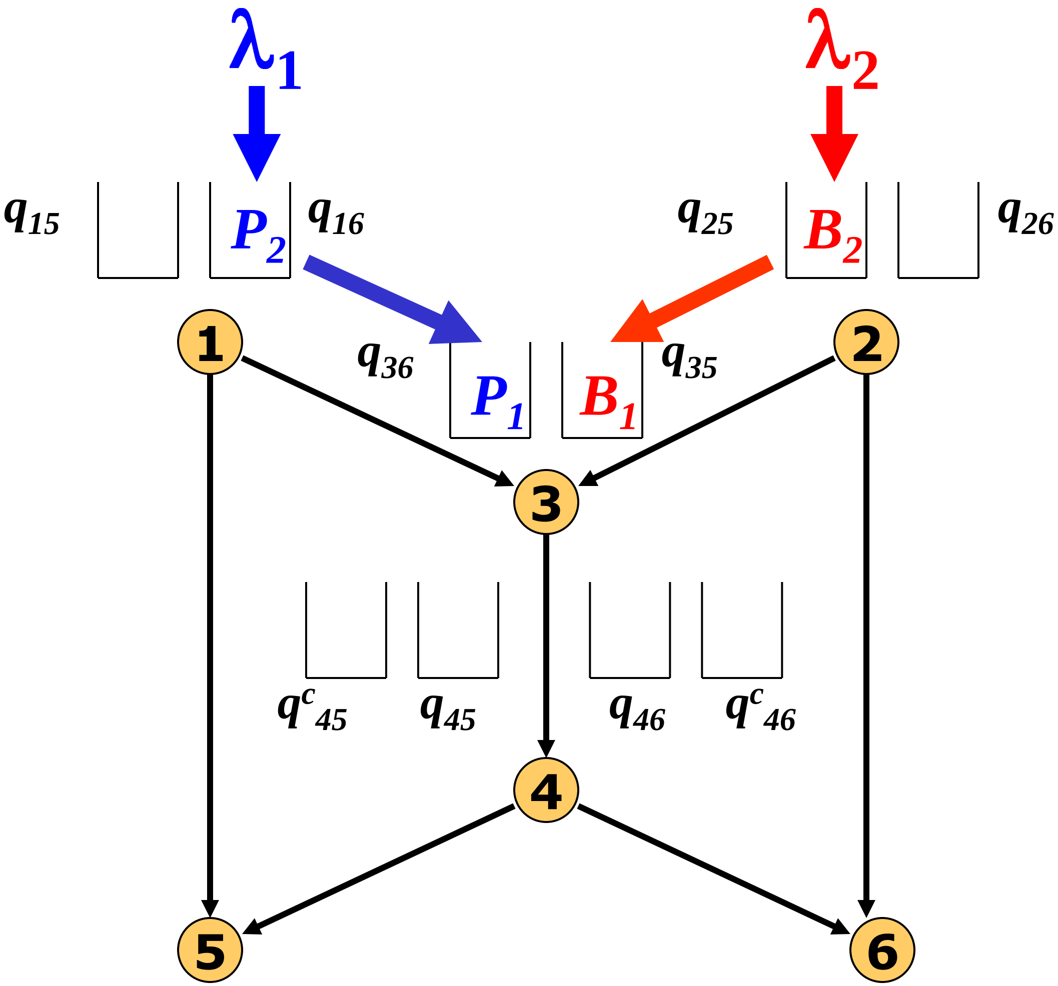
Queueing Architecture



- *(1)* $\Rightarrow$ *(6)*
- *(2)* $\Rightarrow$ *(5)*

- q_{ij} holds packets for node j at node i.
- q^c holds coded packets.
- Need queues for remedy packets.

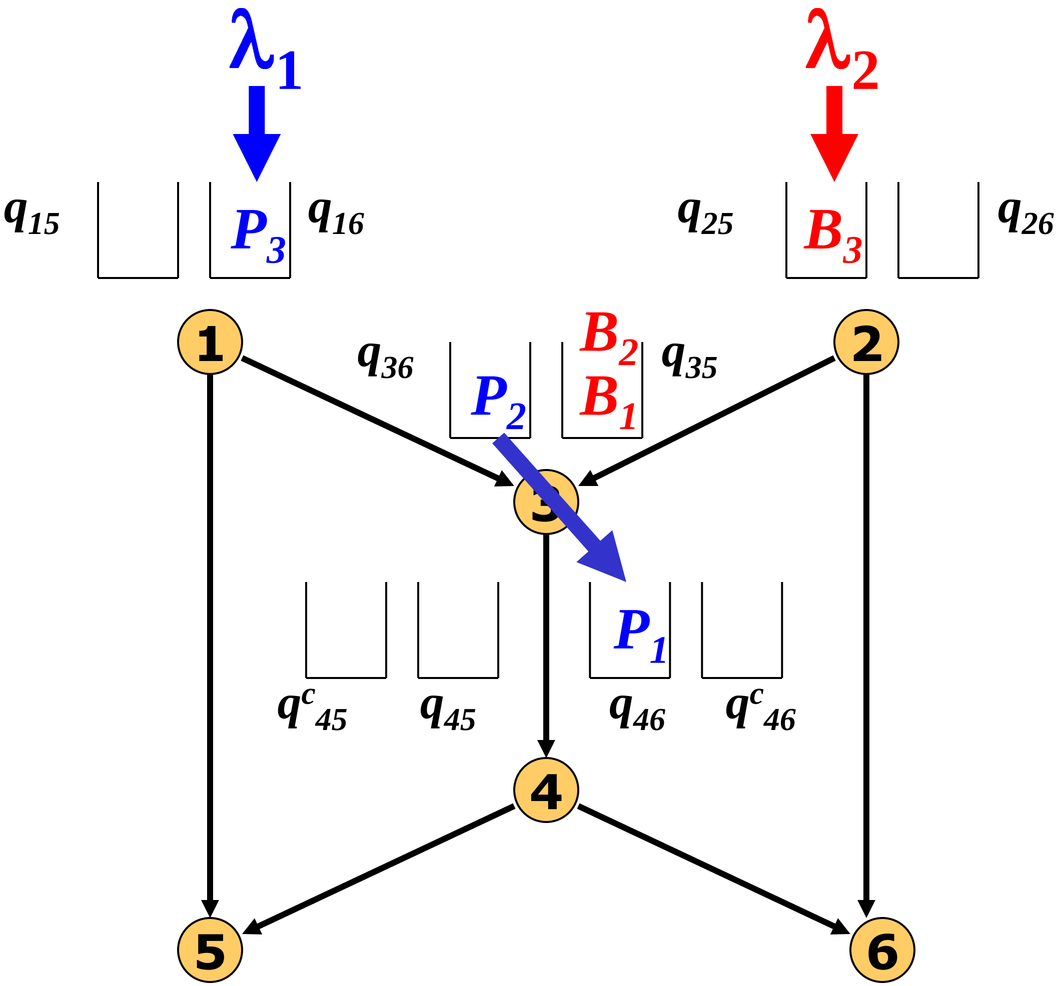
Queue Evolution



- *(1) ⇒ (6)*
- *(2) ⇒ (5)*

- ROUTING ONLY!

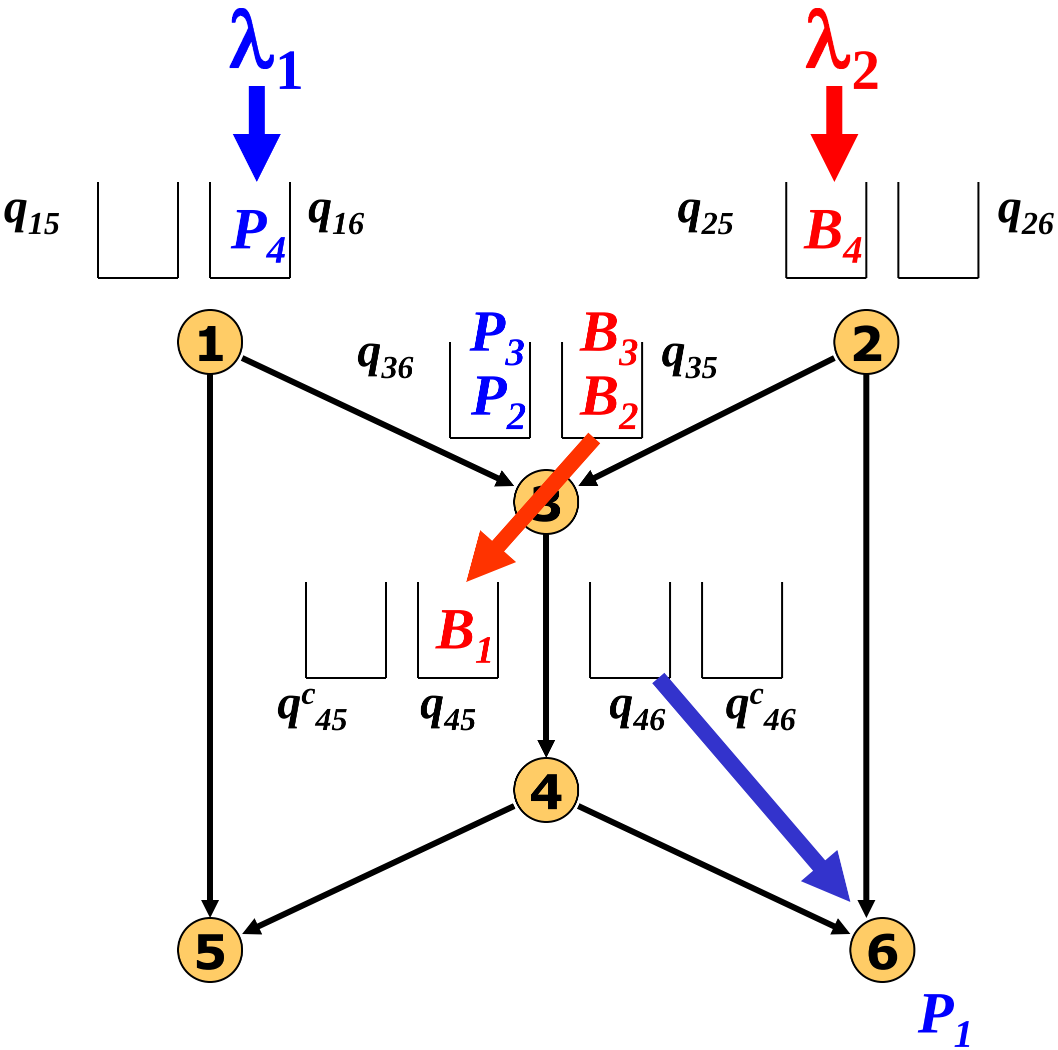
Queue Evolution



- *(1) ⇒ (6)*
- *(2) ⇒ (5)*

- ROUTING ONLY!

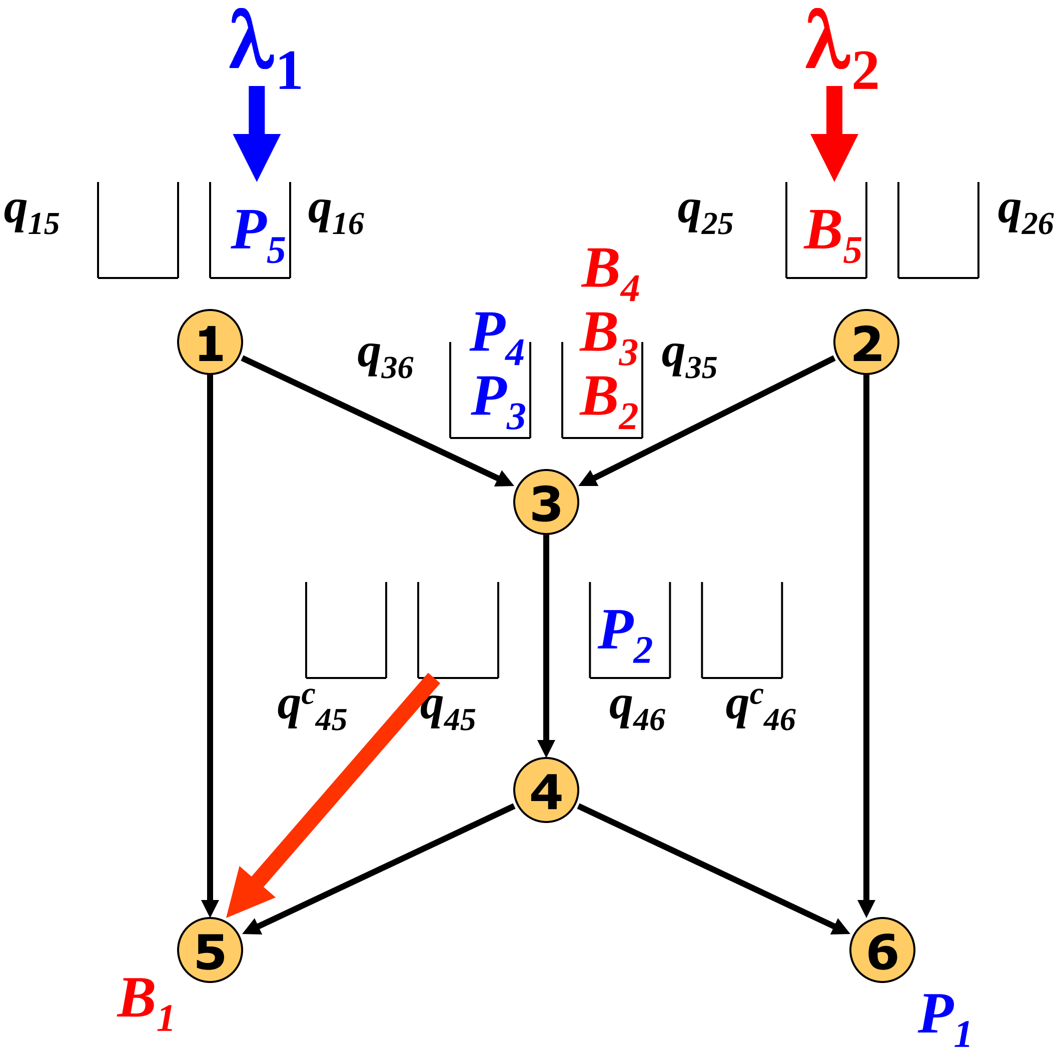
Queue Evolution



- *(1) ⇒ (6)*
- *(2) ⇒ (5)*

- ROUTING ONLY!

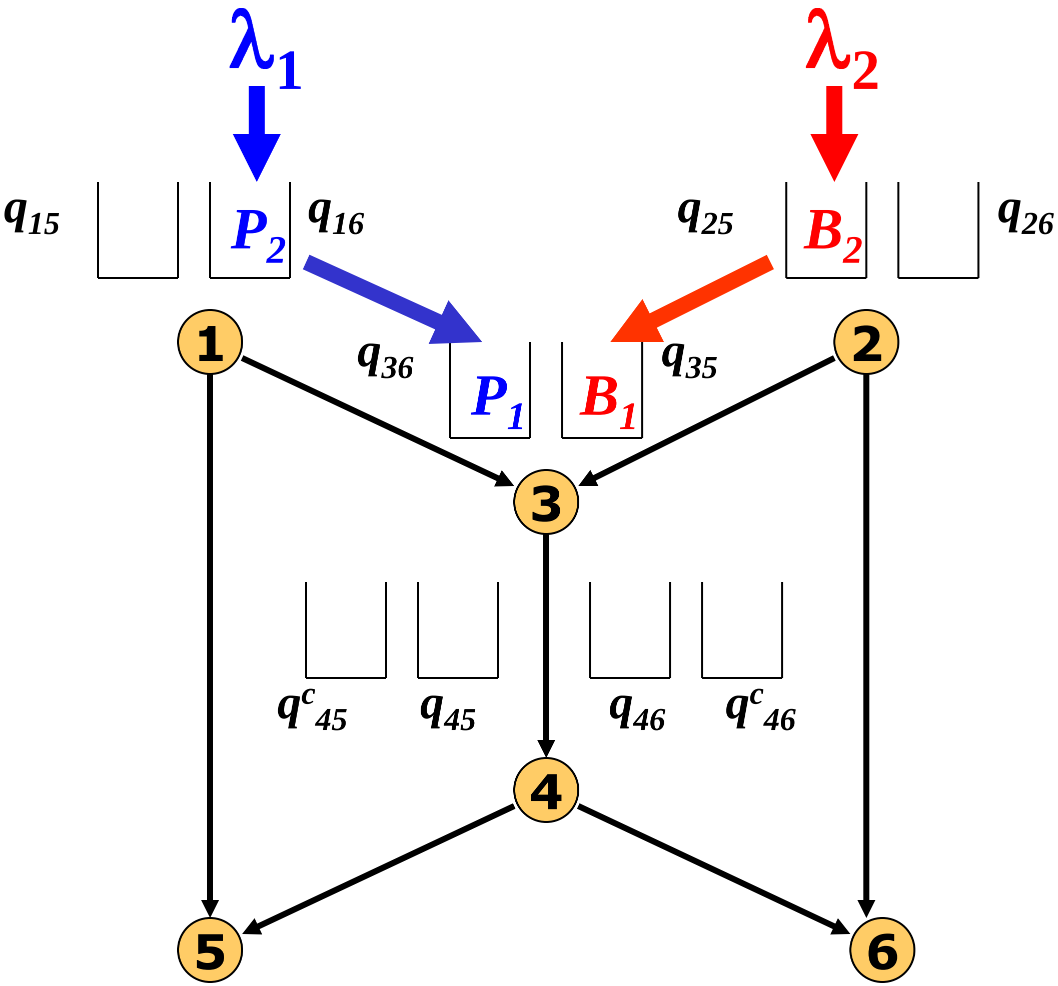
Queue Evolution



- *(1)* $\Rightarrow$ *(6)*
- *(2)* $\Rightarrow$ *(5)*

- ROUTING ONLY!

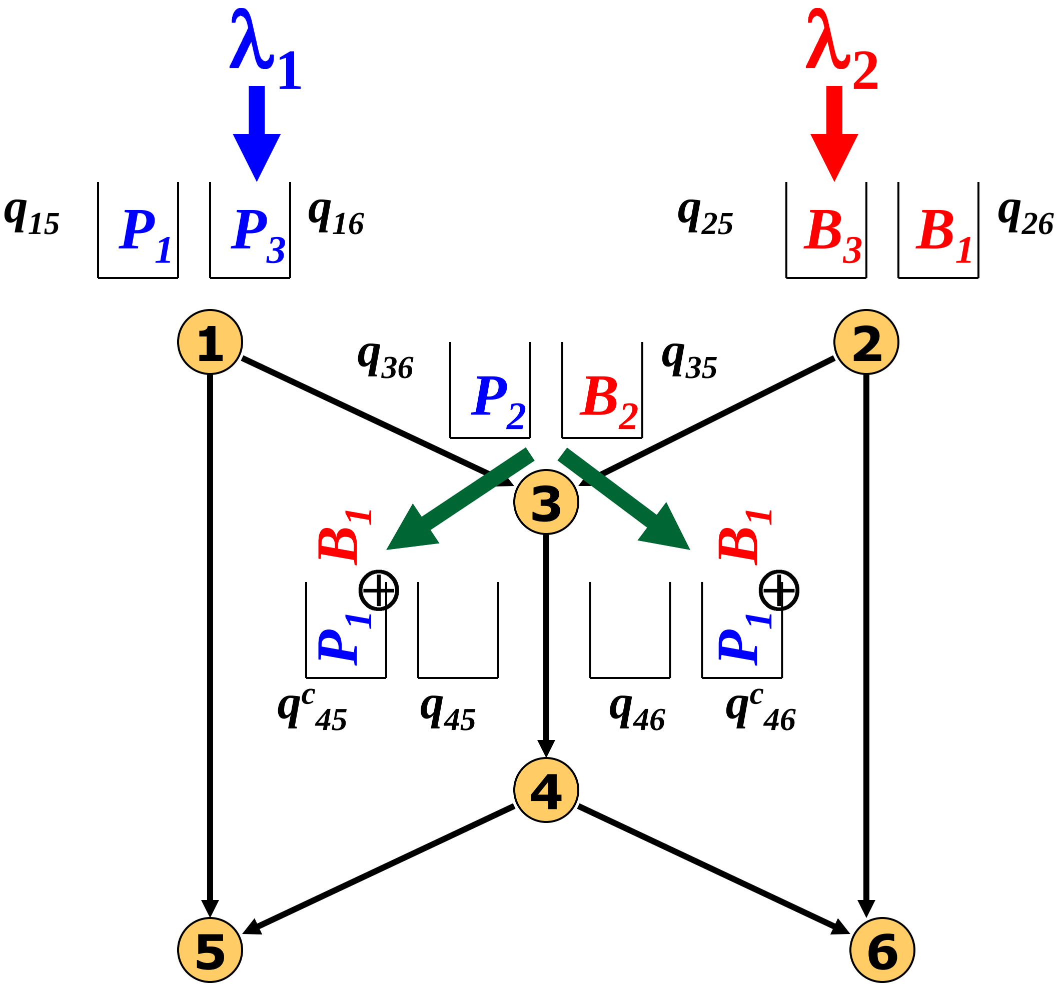
Queue Evolution



- $(1) \Rightarrow (6)$
- $(2) \Rightarrow (5)$

- CODING ONLY!

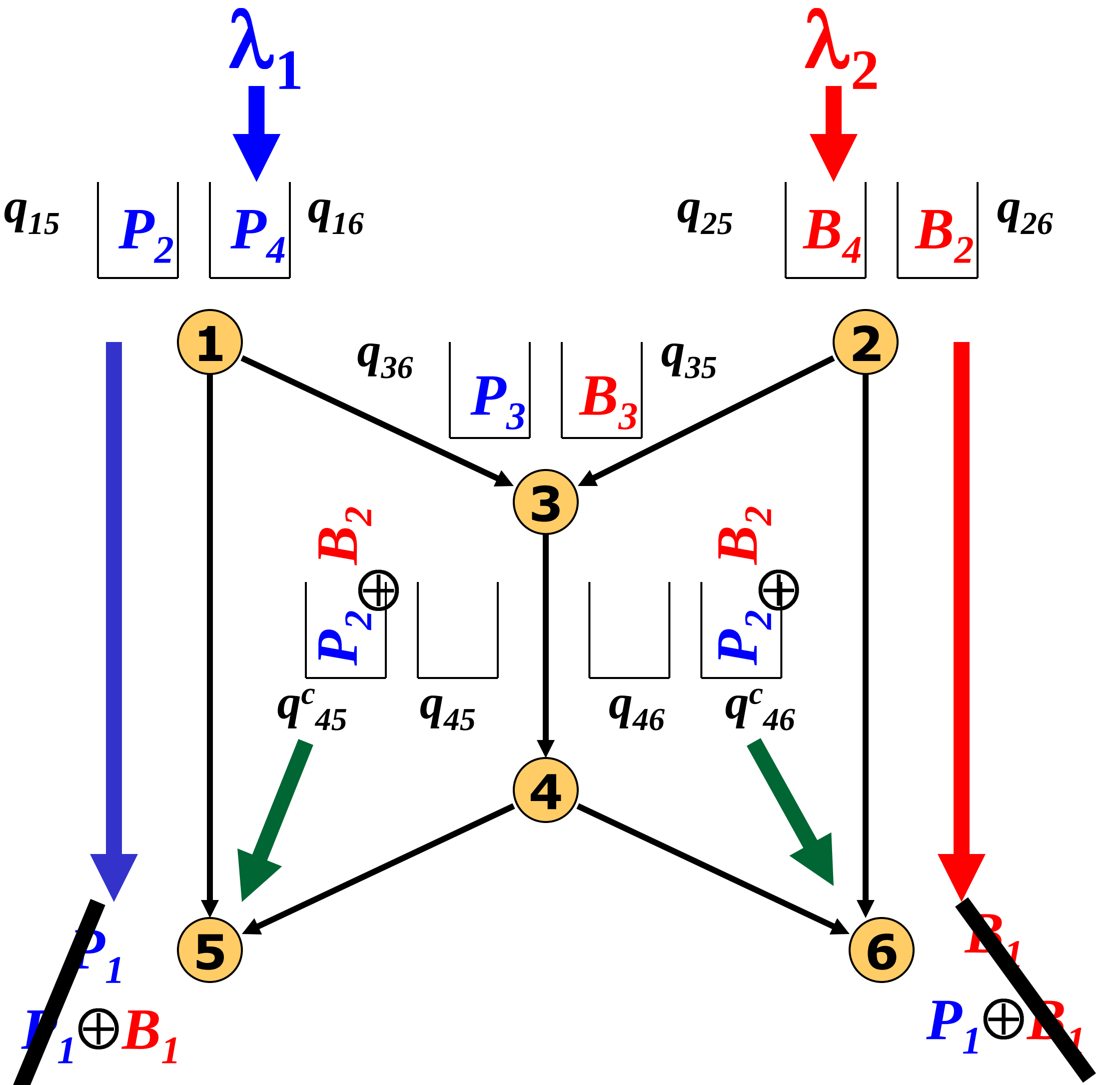
Queue Evolution



- *(1) ⇒ (6)*
- *(2) ⇒ (5)*

- CODING ONLY!

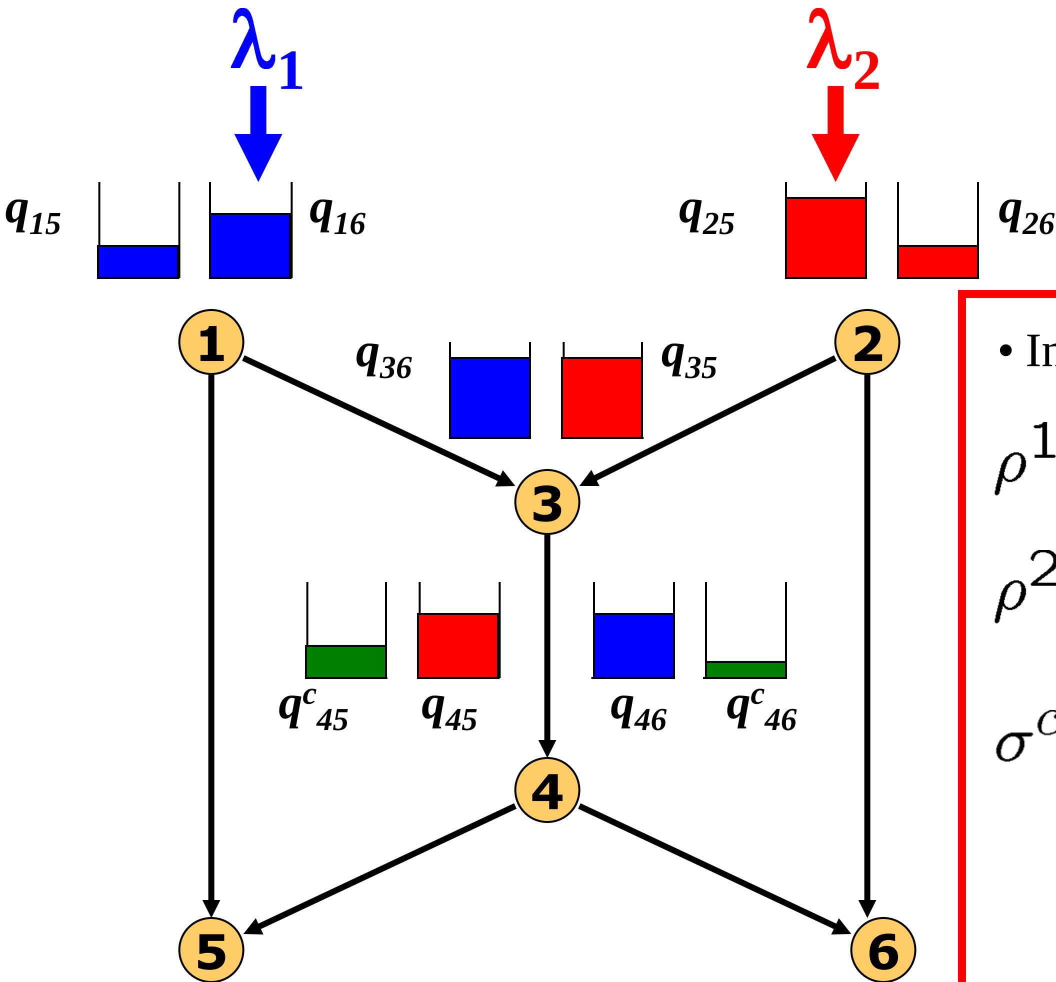
Queue Evolution



- *(1) ⇒ (6)*
- *(2) ⇒ (5)*

- CODING ONLY!

Dynamic Algorithm Description



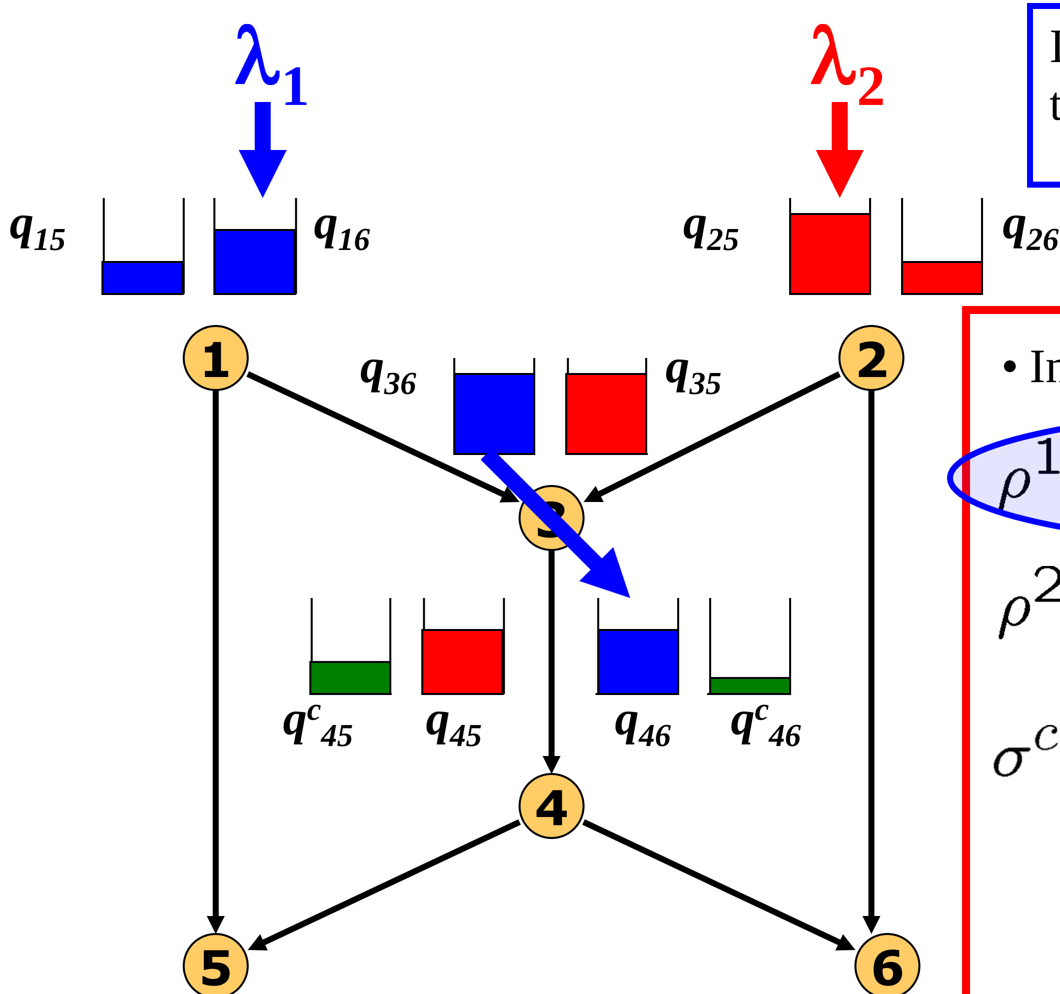
- In slot t, Node ***(3)*** computes:

$$\rho^1[t] = q_{36}[t] - q_{46}[t]$$

$$\rho^2[t] = q_{35}[t] - q_{45}[t]$$

$$\begin{aligned}\sigma^c[t] &= q_{36}[t] + q_{35}[t] \\ &- (q^c_{46}[t] + q_{26}[t] \\ &\quad +q^c_{45}[t] + q_{15}[t])\end{aligned}$$

Dynamic Algorithm Description



If $\rho^1 = \max\{\rho^1, \rho^2, \sigma^c, 0\}$
then ***SERVE FLOW 1.***

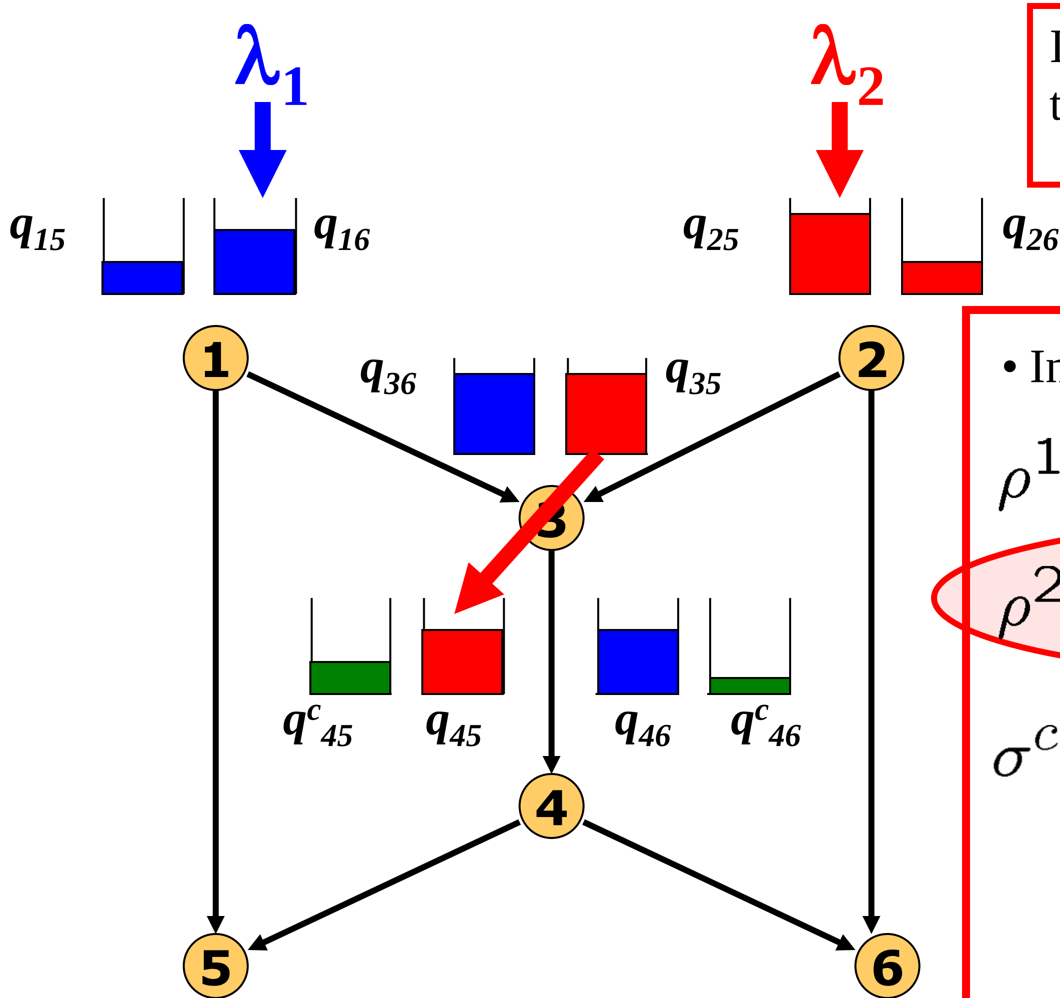
- In slot t, Node ***(3)*** computes:

$$\rho^1[t] = q_{36}[t] - q_{46}[t]$$

$$\rho^2[t] = q_{35}[t] - q_{45}[t]$$

$$\begin{aligned}\sigma^c[t] &= q_{36}[t] + q_{35}[t] \\ &- (q^c_{46}[t] + q_{26}[t] \\ &\quad + q^c_{45}[t] + q_{15}[t])\end{aligned}$$

Dynamic Algorithm Description



If $\rho^2 = max \{\rho^1, \rho^2, \sigma^c, 0\}$
then ***SERVE FLOW 2***.

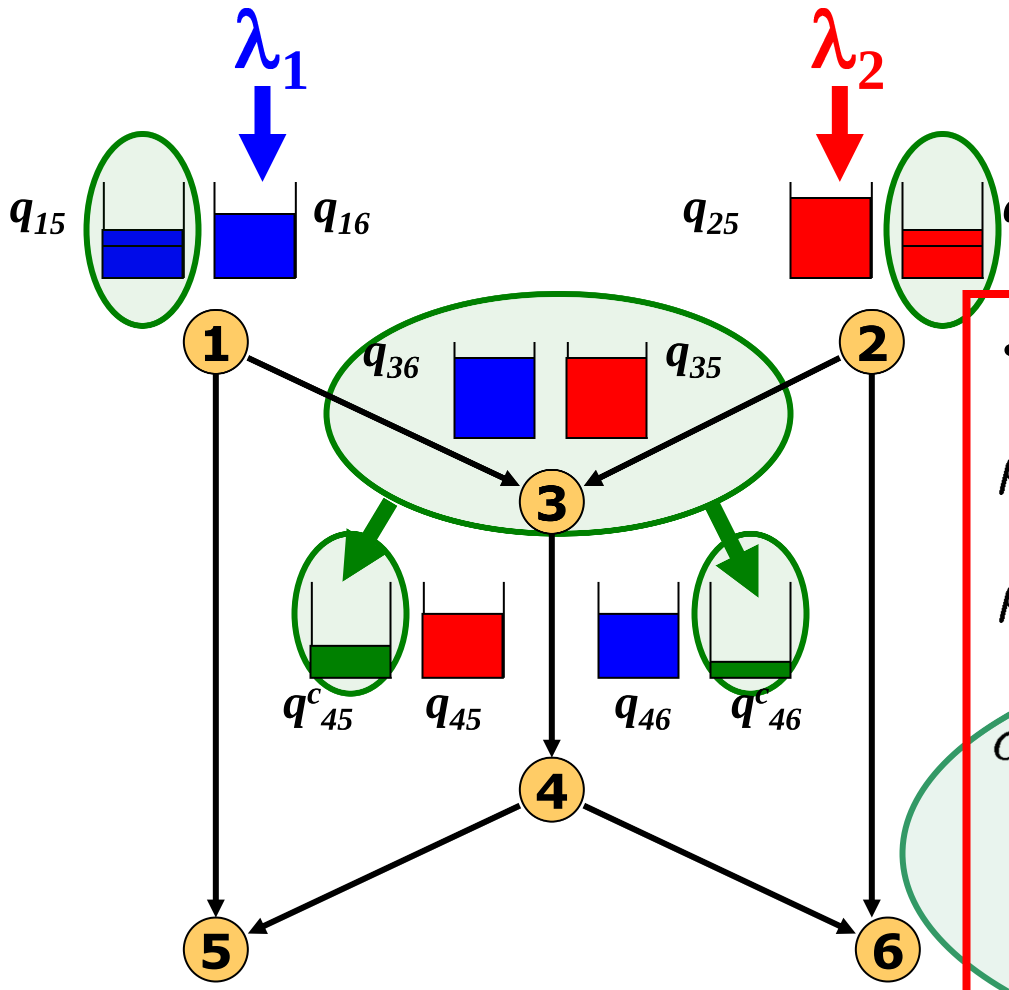
- In slot t, Node ***(3)*** computes:

$$\rho^1[t] = q_{36}[t] - q_{46}[t]$$

$$\rho^2[t] = q_{35}[t] - q_{45}[t]$$

$$\begin{aligned}\sigma^c[t] &= q_{36}[t] + q_{35}[t] \\ &- (q^c_{46}[t] + q_{26}[t] \\ &\quad + q^c_{45}[t] + q_{15}[t])\end{aligned}$$

Dynamic Algorithm Description



If σ^c = *max* $\{\rho^1, \rho^2, \sigma^c, 0\}$
then ***MIX FLOWS***.

- In slot t, Node ***(3)*** computes:

$$\rho^1[t] = q_{36}[t] - q_{46}[t]$$

$$\rho^2[t] = q_{35}[t] - q_{45}[t]$$

$$\begin{aligned}\sigma^c[t] &= q_{36}[t] + q_{35}[t] \\ &- (q^c_{46}[t] + q_{26}[t] \\ &\quad + q^c_{45}[t] + q_{15}[t])\end{aligned}$$

Achieving the Capacity Region

Theorem: Our dynamic algorithm achieves any rate within the capacity region of the butterfly network.

Proof (Outline): $\{(\boldsymbol{q}[t], \boldsymbol{q}^c[t])\}_t$ forms an irreducible, aperiodic Markov Chain

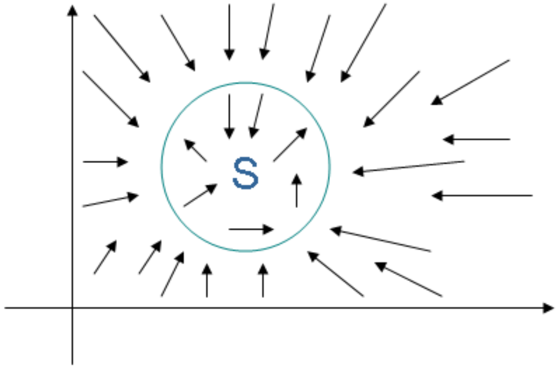
Foster – Lyapunov criterion : Suppose Markov Chain ***q[t]*** is irreducible and aperiodic. Let ***V(q)*** be a function such that $V(q) \geq 0 \ \forall \ q$ and

There exists a finite set ***S*** such that, for $\Delta V(q[t]) := V(q[t+1]) - V(q[t])$

(i) $E[\Delta V(q[t]) \mid q[t]] \leq -\varepsilon$ for $q[t] \in S^c$

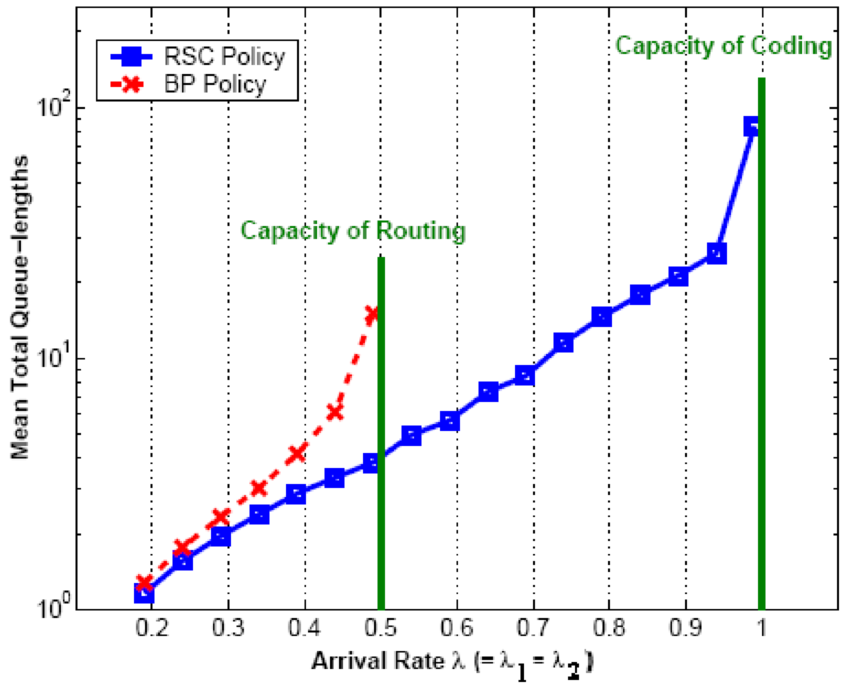
(ii) $E[\Delta V(q[t]) \mid q[t]] \leq M$ for $q[t] \in S$

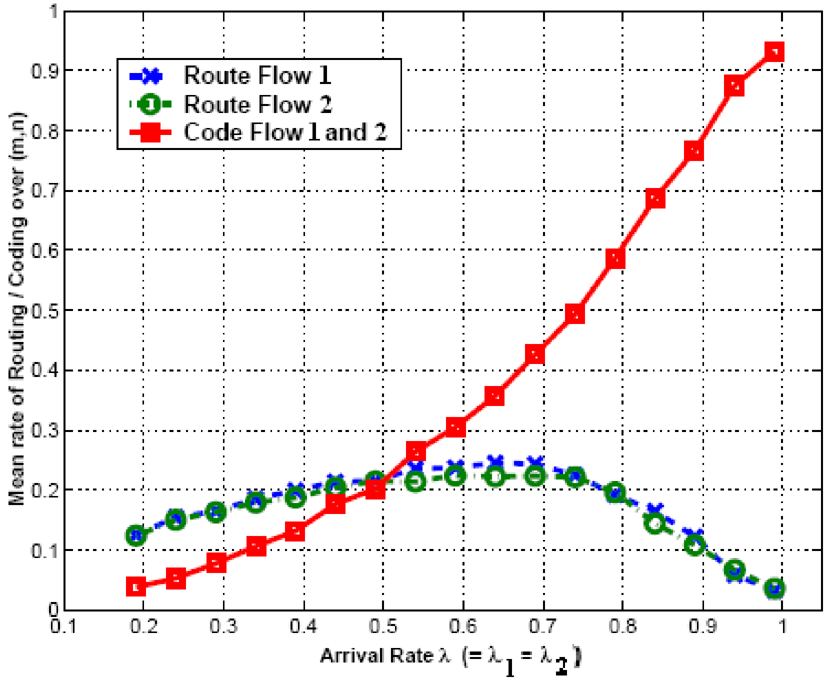
Then, the Markov Chain is positive recurrent (stable).



Let $V(\mathrm{q}) = \sum_{i,j} \left[(q_{i,j})^2 + (q^c_{i,j})^2 \right]$ to prove stability

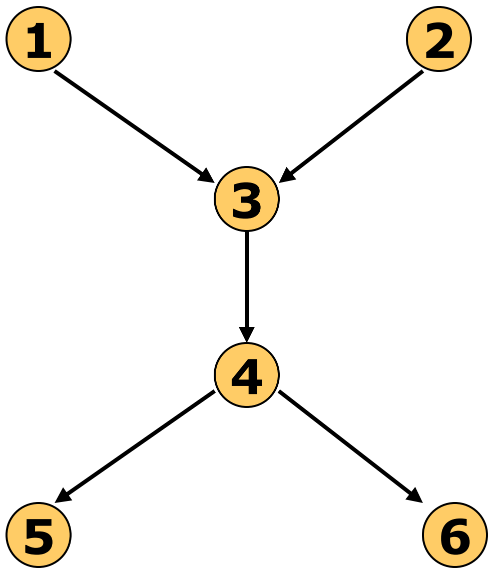
Numerical Results – Butterfly Network

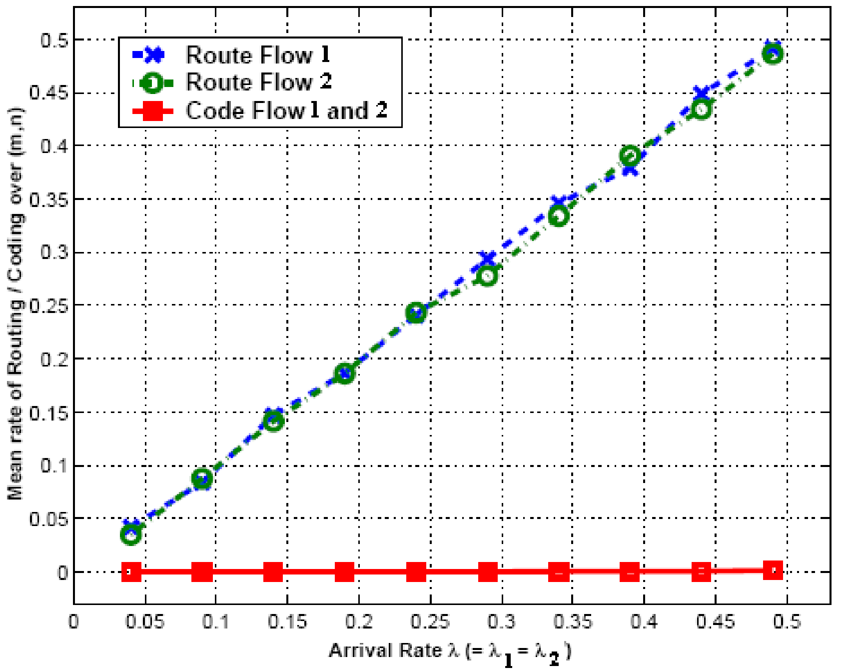




- As the arrival rates of the flows increase, the coding decision dominates the routing decisions, to achieve better performance

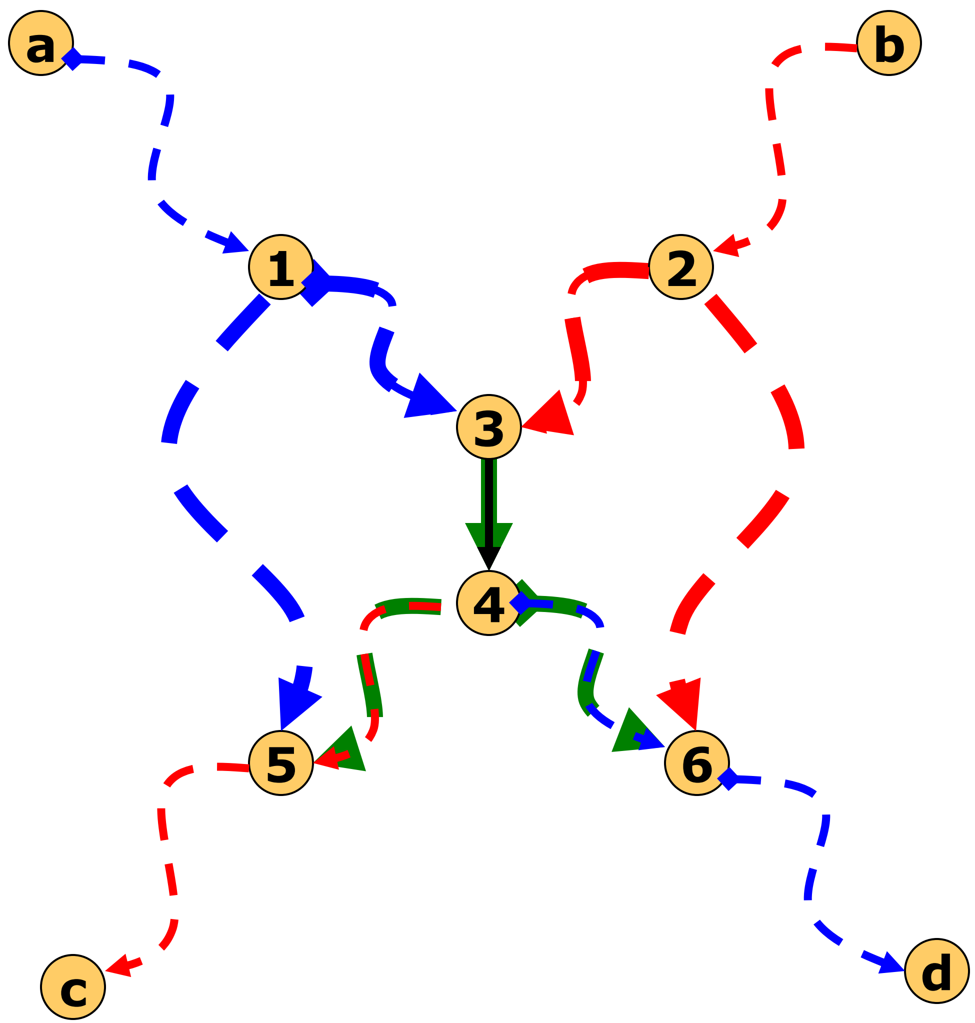
Results – Skeleton of the Butterfly Network





- The policy never performs coding decisions to guarantee decodability at the receivers
- Our dynamic policy adapts its decision dynamically to achieve the best throughput performance

Extension to General Networks - Idea



- *(a)* $\Rightarrow$ *(d)*
- *(b)* $\Rightarrow$ *(c)*

- **Idea**:

  To seek butterflies of various sizes using our dynamic algorithm

TRLKM Region [Traskov et al. (ISIT `06)]:
An achievable rate region with intersession network coding based on superimposing all possible butterflies in the network.

Theorem [E., Lun `07]: Our dynamic algorithm for general networks supports any arrival rate that lies within the TRLKM region.

Remarks

- Our algorithm
  - is the first work that provides a practical algorithm for performing intersession coding decisions for general topologies
  - introduces an original ***queueing architecture***
  - describes a ***simple decision rule*** for linear intersession coding across sessions
  - ***unifies the class of backpressure policies***
  - ***applies to general topologies***, wireless/sensor networks
  - achieves the full capacity in the butterfly case
  - ***performs provably good*** in general topologies

Open Issues

Extending the Framework for

- Short-term optimality: include short-term Quality-of-Service (QoS) constraints such as delay, overflow-probabilities etc.
- Low-complexity and Distributed Implementation:
  - Development of network algorithms in the class of randomized strategies with favorable qualities
  - Development of dynamic strategies in the context of random access schedulers
- Managing Dynamics: Scalable and high-performance network algorithms under dynamic network conditions

Architectures and Algorithms for Wireless Networks

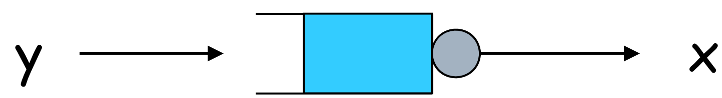
R. Srikant
Department of ECE & CSL
ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Collaborators: Atilla Eryilmaz (OSU), Juan Jose Jaramillo, Jian Ni, Bo Tan (Illinois), Shihuan Liu and Lei Ying (Iowa State)

Recap: Queueing and Optimization

- Each constraint is represented by a queue:

$$y \leq x$$



- Stability of the queue implies constraint is satisfied and vice-versa; resource allocation is some form of the Maxweight algorithm with queue lengths as weights
  - Dual formulation reveals the form of the MaxWeight algorithm

- The expected queue length is proportional to the Lagrange multiplier for this constraint (ϵ: step-size parameter):

$$q(k+1)=[q(k)+\epsilon\ (y(k)-a(k))]^+$$

Typical Theorem

- J^* is the optimal value of the objective of the deterministic problem
- J_{st} is the long-run average objective in the real system, which is usually stochastic (stochastic arrivals, stochastic channels, etc.)
- Theorem: Using the Lyapunov function $\sum_l q_l^2$

$$\mathbb{E}(J_{st}) \leq J^* + K\epsilon; \quad \mathbb{E}(\sum_l q_l) = O(1/\epsilon)$$

for some constant K

Issues

- All constraints formulated in terms of long-term averages

- Does this mean only long-lived elastic flows can be modeled using this framework?

- We will present two applications which can be modeled using this framework:

  - Packets with deadlines: constraint in terms of lower bounds long-run fraction of packets delivered before deadline expiry, i.e., a certain % of packets have to served before deadline expires

  - A mixture of long-lived and short-lived flows: Short-lived flows bring a finite number of packets and depart when their packets are delivered.

Application I: Per-packet Deadlines

- Consider an ad hoc network consisting of L links

- Time is divided into frames of T slots each (Hou, Borkar, Kumar, ‘09)

- QoS requirement for link l: fraction of packets lost due to deadline expiry has to be less than or equal to p_l

Schedule for Each Frame

	Time Slot 1	Time Slot 2	.	.	Time Slot T
Link 1	1 (ON)	0	0	1	1
Link 2	1	0	1	0	0
.	0 (OFF)	1	0	0	1
.	0	1	0	0	1
Link L	0	1	1	0	0

- In each time slot, select a set of links to be ON, while satisfying interference constraints
- Thus, a schedule is an LxT matrix of 1s and 0s

Problem: Find a schedule in each frame such that the QoS constraints are satisfied for each link

An Optimization Formulation

- $S_{lk} = 1$ if link l is scheduled in time slot k

- A_l: Number of arrivals to link l in a frame, a random variable, with mean λ_l

- Constraint: Average number of slots allocated must be greater than or equal to the QoS requirement for each link l

$$\boxed{E[\min(\textstyle\sum_k S_{lk}, A_l)] \geq \lambda_l(1-p_l)}$$

- A dummy optimization problem (A is some constant):

$$\boxed{\max A}$$

Fictitious Queue

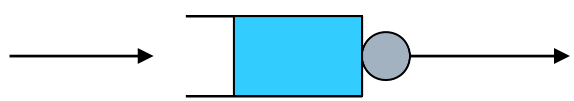
- Recall $y \leq x$ corresponds to

- Similarly,

$$E[\min(\textstyle\sum_k S_{lk}, A_l)] \geq \lambda_l(1-p_l)$$

corresponds to

Upon each packet arrival to link l, add a packet to this queue with prob. $(1-p_l)$



Remove packet from the queue every time a packet is successfully scheduled

Deficit counter: Keeps track of deficit in QoS

Optimal Schedule

- d_l: deficit of link l

- Choose a schedule at each frame to maximize

$$\sum_l d_l \left(\sum_k S_{lk}\right)$$

subject to $\sum_l S_{lk} \leq A_l$

- The constraint simply states the the number of slots allocated to link l in a frame should not be greater than the number of arrivals in the frame
- This is simply the MaxWeight algorithm where the deficits are used as weights, instead of queue lengths

Resource Allocation

- Beyond just meeting constraints: allocate extra resources to meet some fairness constraint

$$\max \sum_l w_l \left(\sum_k S_{lk}\right)$$

subject to $\quad E[\min(\sum_k S_{lk}, A_l)] \geq \lambda_l(1-p_l)$

- Optimal Solution: Choose S in each frame to maximize

$$\sum_l (w_l + \epsilon\, d_l)\left(\sum_k S_{lk}\right)$$

Formulation with Elastic Flows

- x_{li} is the transmission rate of an inelastic flow and x_{le} is the transmission rate of an elastic flow

- Associating utility functions with inelastic flows, the objective becomes

$$\max \sum_l w_l x_{li} + \sum_l U_l (x_{le})$$

- In the schedule, distinguish between the transmission of an elastic flow packet and an inelastic flow packets

- Write down constraints as before to complete the problem formulation

Solution with Elastic Flows

- At each link, maintain a queue q_l for elastic flows and a deficit counter d_l for elastic flows:

- Scheduling algorithm:

$$\text{Max } \sum_l (w_l + \epsilon\, d_l)(\sum_k S_{lki}) + \epsilon\, q_l (\sum_k S_{lke})$$

- Congestion control for elastic flows: Choose x_{le} such that

$$\text{Max } U_l(x_{le}) - q_l\, x_{le}$$

Theorem

- Result 1:

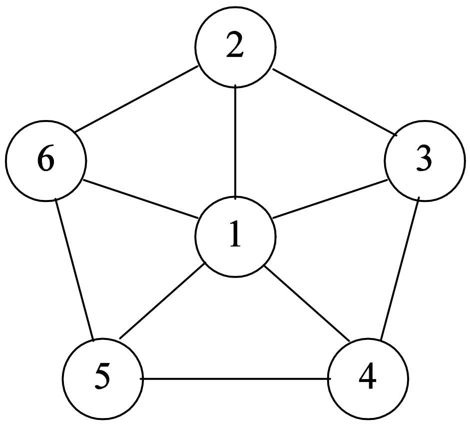
$$E(\sum_l U_l (x_{le}) + w_l x_{li}) - \sum_l U_l (x_{le}^*) + w_l x_{li}^* = O(\epsilon)$$

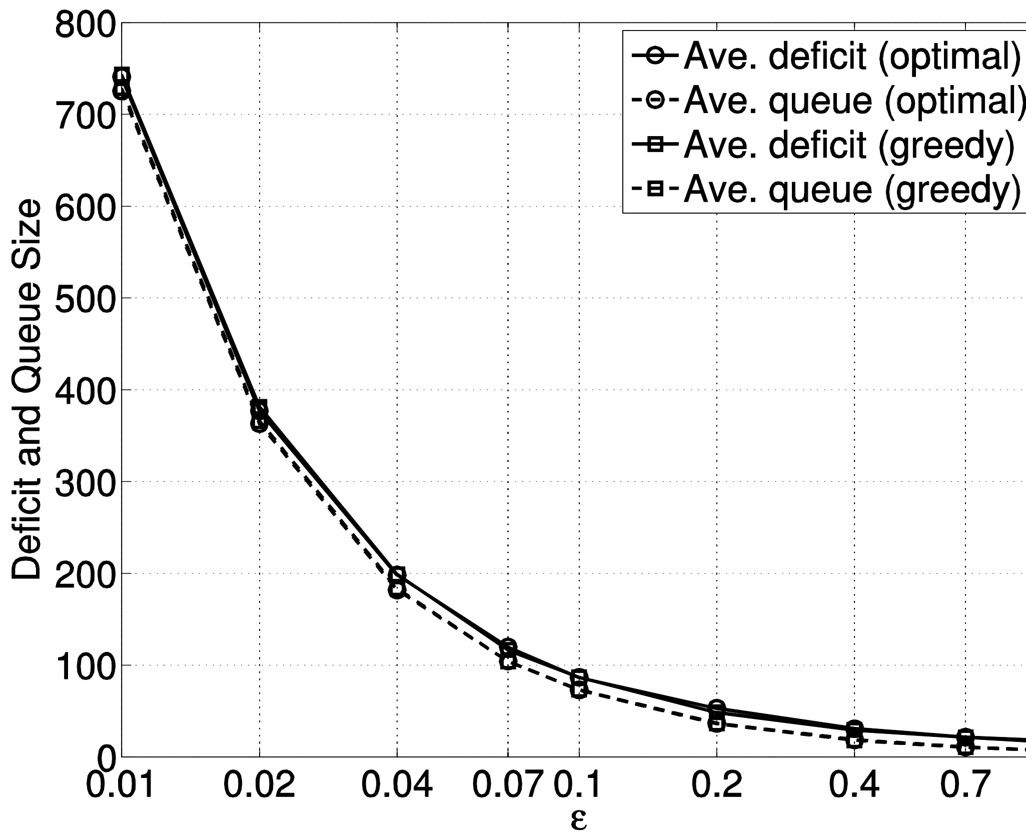
- Result 2:

$$E(\sum_l q_l + d_l) = O(1/\epsilon)$$

ϵ provides a tradeoff between optimality and queue lengths and deficits.

Simulations





Application II: Short-Lived Flows

- Model: A base station transmitting to a number of receivers

- The base station can transmit to only one user at a time

- Classical Model: a fixed number of users, say N

- Each user's channel can be in one of many states:
  - $\mathbb{R}_i(t)$: Rate at which the base station can transmit to User i if it chooses to schedule user I

- Classical problem: Which user should the base station select for transmission at each time instant?

Classical Solution

- Suppose that the goal is to maximize network throughput:
  - i.e., the queues in the network must be stable as long as the arrival rates lie within the capacity region of the system

- (Tassiulas-Ephremides '92): Transmit to user i such that

$$i \in \arg\max_j q_j(t)\, R_j(t)$$

- Solution can be derived from optimization considerations as mentioned earlier

New Model: Short-lived Flows

- What if the number of flows in the network is not fixed?
  - Each flow arrives with a finite number of bits. Departs when all of its bits are served
  - Flows arrive according to some stochastic process (Poisson, Bernoulli, etc.)

- Since the number of bits in each flow is finite, need a new notion of stability since queues cannot become large
  - Need the number of flows to be "finite" in some sense

Van de Ven, Borst, Shneer '09: The MaxWeight algorithm need not be stabilizing: the number of flows can become infinite even when the load lies within the capacity region

Necessary condition for stability

- Suppose each channel has a maximum rate $\mathbb{R}^{max}$

- A necessary condition for stability:
  - F: File size, a random variable. Min expected number of time slots (workload) required to serve a file is

$$E(\lceil F/\mathbb{R}^{max} \rceil)$$

  - λ: Rate of flow arrivals (number of flows per time slot)

  - Necessary condition for stability :

$$\boxed{\lambda \; E(\lceil F/\mathbb{R}^{max} \rceil) < 1}$$

Scheduling Algorithm

- Transmit to the user with the best rate at each time instant, $\text{Max}_i\, R_i(t)$
- Does not even consider queue lengths in making scheduling decisions
- Why does it work?
  - When the number of flows in the network is large, some flow must have a rate equal to R^{max} with high probability
  - Thus, we schedule users when their channel condition is the best; therefore, we use the minimum number of time slots to serve a user

Short-Lived and Long-Lived Flows

- Now consider the situation where there are some long-lived (persistent) flows in the networks

- Why does this model make sense, after all, every flow has a finite size in reality?
  - File sizes are heavy-tailed: from the point of view of small-sized flows, the large-sized flows can be thought as persistent

- For simplicity, we will consider the case of one long-lived flow which generates packets at rate ν packets per time slot

- Solution: using an optimization formulation

Capacity constraints

- R_c: rate at which the long-lived flow can be served when its channel state is c (a random variable)
- π_c: probability that the long-lived channel state is c
- p_c: probability of serving the long-flow in state c
- Constraints:
  - Long-lived flows: $\nu \leq \sum_c \pi_c \, p_c \, R_c$
  - Short-lived flows: $\lambda \, E(\lceil F / R^{max} \rceil) \leq \sum_c \pi_c (1-p_c)$

Optimization Interpretation

- Lagrange multiplier of $\nu \leq \sum_c \pi_c \, p_c \, R_c$
  - Left-hand side is packet arrival rate, right hand side is packet departure rate of long-lived flows
  - So, the Lagrange multiplier is (proportional to) the queue length of long-lived flows

- Lagrange multiplier of $\lambda \; \mathbb{E}(\lceil \mathbb{F}/\mathbb{R}^{\max} \rceil) \leq \sum_c \pi_c \, (1\text{-}p_c)$
  - Left-hand side is number of slots (workload) required to serve short-lived flows, the right-hand side is the number of slots available
  - So, the Lagrange multiplier is (proportional to) the minimum number of slots required (workload) to serve the short-lived flows in the solution

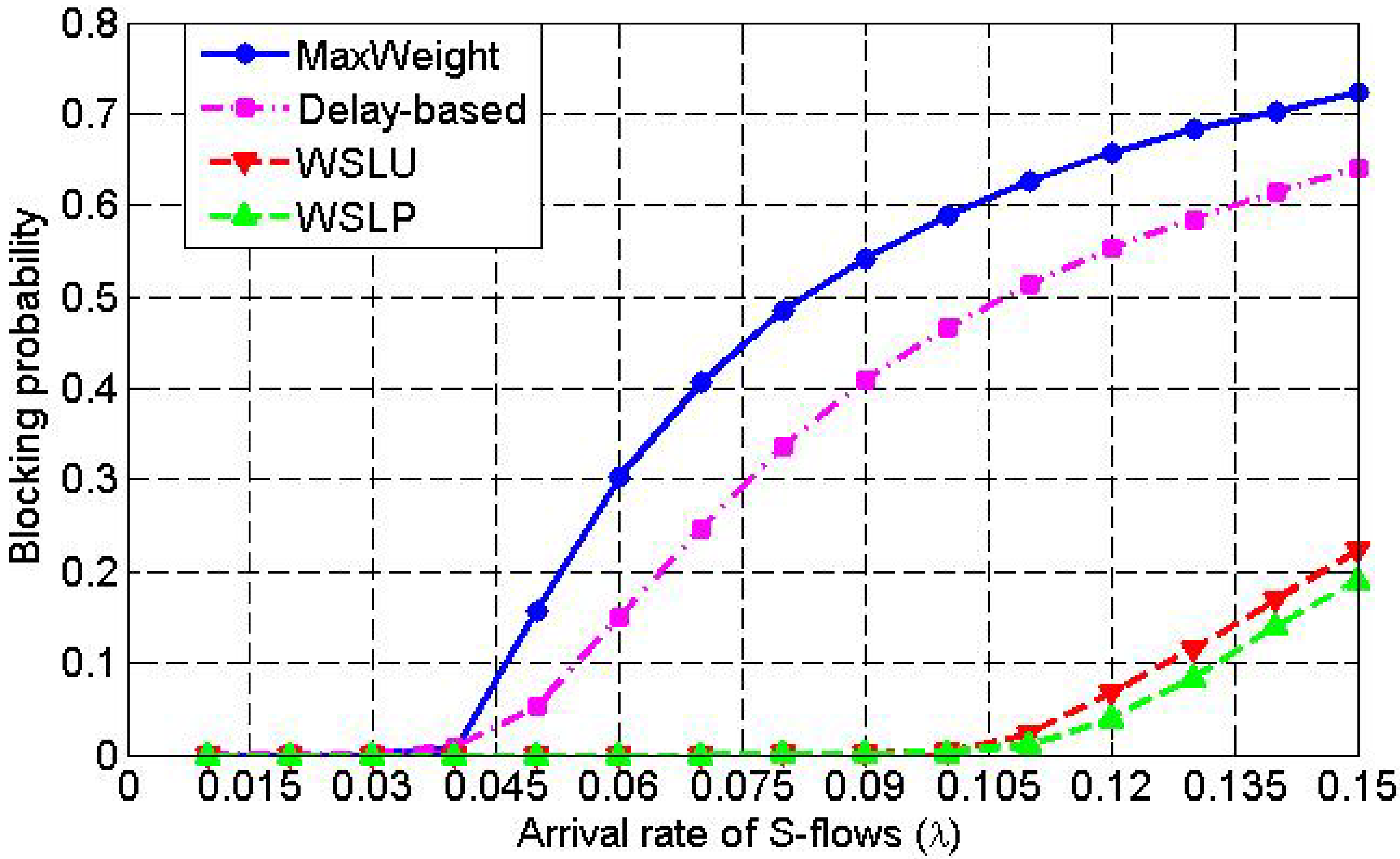
Optimization Solution

- If the workload of short-lived flows is larger than the queue length of the long-lived flow, then serve a short-lived flow
  - Choose the flow with the best channel condition

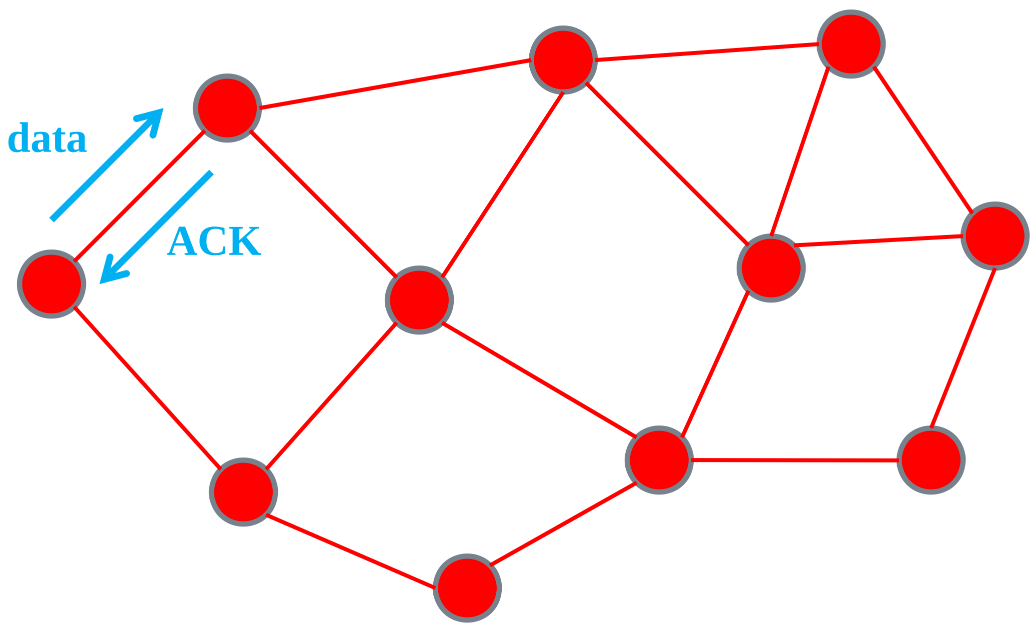
- Else, serve the long-lived flow

- Extensions:
  - More than one long-lived flow
  - Different short-lived flows have different $\mathbb{R}^{max}$
  - The $\mathbb{R}^{max}$'s are unknown; learn it, by using the best channel condition seen by each flow so far

Simulations



Distributed Implementation: Model



e.g., 2-hop interference model

- Links may not be able to transmit simultaneously due to interference.
- Scheduling algorithm determines which links transmit at each time instant.
- Performance metrics: **throughput and delay**.

Quick Recap

- MaxWeight Scheduling (centralized, high complexity):
  - Associate a weight with each link, equal to its queue length.
  - Find schedule **x** which maximizes w(**x**); w(**x**): weight of a schedule **x** is the sum of the weights of the links in the schedule.
  - Throughput Optimal

- Useful Observation [Eryilmaz-S.-Perkins'05]: MaxWeight is throughput-optimal even under the following modification: pick a schedule of weight sufficiently close to max-weight schedule with high probability, going to one as the weight of the MWS goes to infinity.

Low-Complexity Schedules

- Maximal Scheduling
  - A schedule is maximal if no additional link can be added to it without violating the interference constraints.
  - May only achieve a small fraction of the capacity region.
- Greedy Maximal Scheduling (GMS)
  - Sequentially add a link with the longest queue to the schedule until it is maximal (Dimakis-Walrand, Joo-Lin-Shroff, Zussman-Modiano, Leconte-Ni-S.).
  - In general may only achieve a fraction of the capacity region (throughput optimal if local pooling condition is satisfied).
  - Performance close to MWS in simulations.

Random-Access Algorithms

- Aloha (Slotted)
  - Transmit a fresh packet at the beginning of the next slot; if collision, retransmit the packet in each subsequent slot with probability p until success.
  - Only efficient under light traffic.

- Carrier Sense Multiple Access (CSMA)
  - Sense the channel before transmission. If busy, keep silent; if free, attempt to join the schedule after a random backoff time.
  - Continuous-Time CSMA Model (no collisions)
    - Boorstyn et al.('87): distribution over schedules has a product-form.
    - Jiang-Walrand('08), Rajagopalan-Shah-Shin('08): CSMA can achieve max throughput if mean backoff time is updated based on link weight.
  - Other Related Works
    Marbach-Eryilmaz-Ozdaglar('07), Liu-Yi-Proutiere-Chiang-Poor('08)

Goal

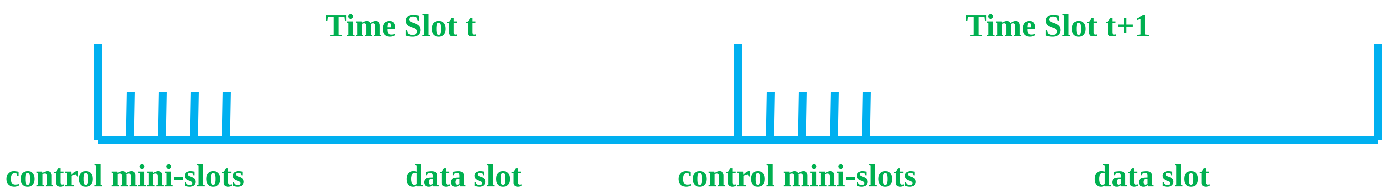
- Design a distributed algorithm which picks schedule $\boldsymbol{x}$ with probability

$$\pi(x) = \frac{e^{w(x)}}{Z}$$

- Note: algorithm picks schedules of large weights with high probability, as required.
- Focus today: Discrete-time model which allows the algorithm to be combined with heuristics leading to dramatic delay reduction; explicitly takes into account collisions

Modeling Assumption

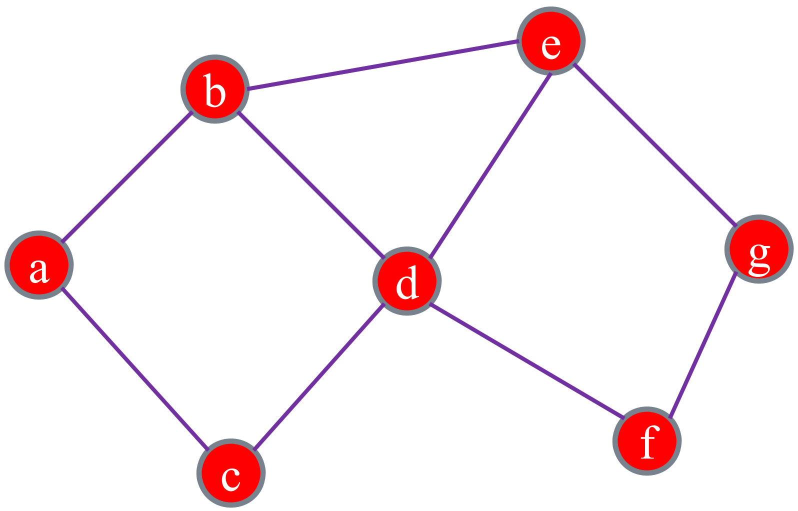
- Divide each time slot into a control slot and a data transmission slot:



  - Links contend in control mini-slots to determine a collision-free schedule in the data slot.
  - Collisions are allowed in the control mini-slots.

Interference Graph

schedule x = {**a**, **d**, **g**}



- Each vertex in the interference graph represents a link in the network.
- If two links interfere with each other, they are neighbors in the interference graph.
- A feasible schedule: a set of vertices which are not neighbors of each other, i.e., they form an independent set.
- We consider one-hop traffic only.

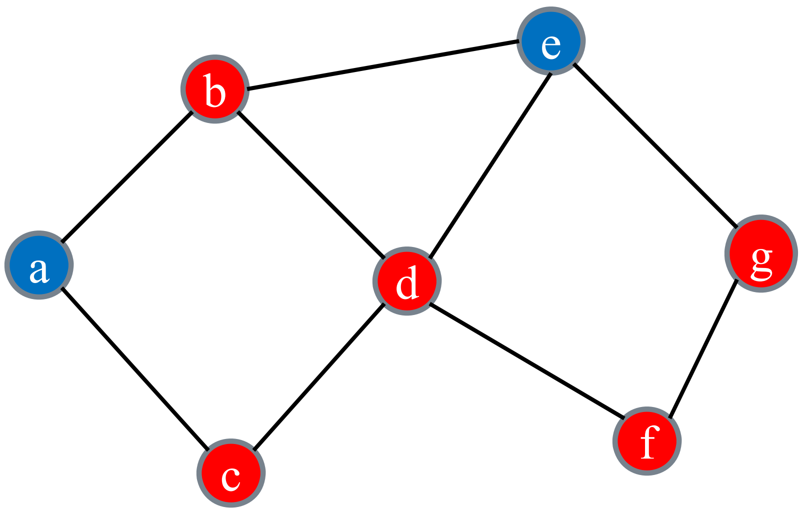
Basic Scheduling Algorithm

- Step 1. In control slot t, select a "decision schedule" **m**(t): a set of links that may decide to change their state from the previous slot; other links cannot change their state.

- Step 2. For any link i in **m**(t) do
  - If no links in its conflict set **C**(i) were active in the previous data slot, link i will decide to become
    - **active** with probability p_i: $\mathbf{x_i(t)=1}$
    - **inactive** with probability $1-p_i$: $\mathbf{x_i(t)=0}$
  - Else, link i will be **inactive**: $\mathbf{x_i(t)=0}$

- Step 3. In the data slot, use **x**(t) as the transmission schedule.

Illustration



- Current schedule: {a, e}
- Decision schedule m(t)={c, f}
- Allowed decisions for links in m(t):
  - Link c, $x_c(t)=0$ (no choice)
  - Link f, $x_f(t)=1$ (w.p. p_f)
- Other links' states are unchanged.
- New schedule: x(t)={a, e, f}

Schedule Evolution Markov Chain

- If both **x**(t-1) and **m**(t) are feasible, then **x**(t) is also feasible.

- **x**(t) evolves as a discrete-time Markov chain (DTMC) (if **m**(t) is picked at random in each time slot).

- **x** can make a transition to **y** if and only if $\mathbf{x} \cup \mathbf{y}$ is feasible and there exists a decision schedule **m** such that
  $\mathbf{x} \Delta \mathbf{y} \subseteq \mathbf{m}$.

Product-Form Distribution

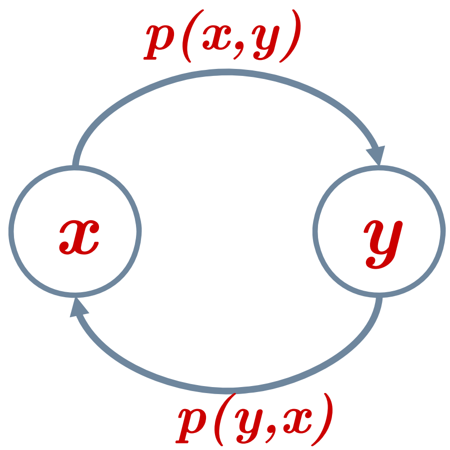
- **Proposition:** If the set of possible decision schedules includes all the links, then the DTMC is reversible and the steady-state probability of using schedule ***x*** is

$$\pi(x) = \frac{1}{Z}\prod_{i\in x}\frac{p_i}{1-p_i}$$

$$Z = \sum_{x\in M}\prod_{i\in x}\frac{p_i}{1-p_i}$$

Outline of Proof

- State **0** can reach any state $\boldsymbol{x}$ (collision-free schedule) with positive probability in a finite number of steps, and vice versa.
- Local balance equation is satisfied.



$$\pi\ (x)\ \ p(y,x) = \pi(y)\ \ p(y,x)$$

Throughput Optimality

- Choose p_i for link i (whose weight is w_i) as

$$p_i/(1-p_i)=exp(w_i),$$

then the probability of choosing a schedule x with weight $w(x)$ is given by

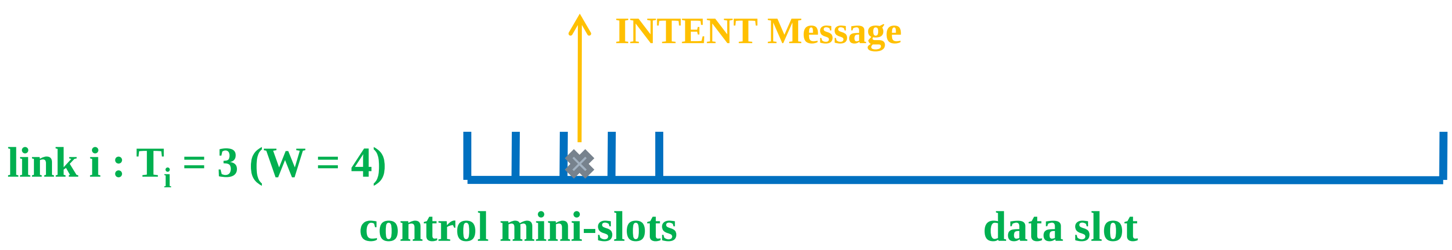
$$\boxed{\pi(x) = \frac{1}{Z}\prod_{i\in x}\frac{p_i}{1-p_i} = \frac{1}{Z}\prod_{i\in x}e^{w_i} = \frac{e^{w(x)}}{Z}}$$

Thus, a schedule of large weight is picked with high probability.

- Question: How to pick the decision schedule?

Q-CSMA

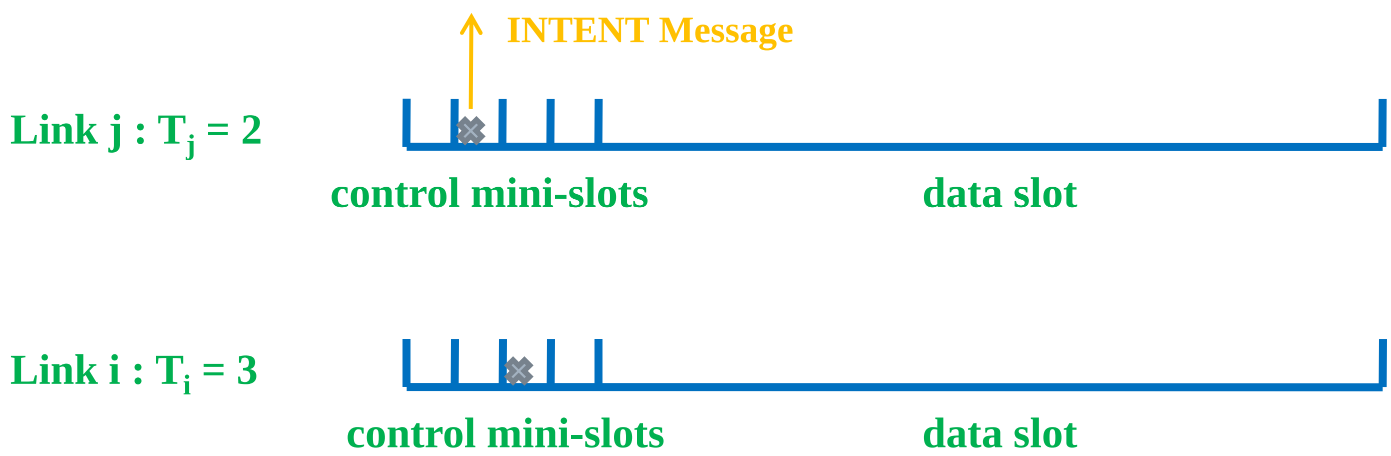
- Each time slot is divided into a data slot and control mini-slots.
- The control mini-slots are used to determine the decision schedule in a distributed manner; each link i selects a random control mini-slot T_i in $[1,W]$.



- Roughly, the idea is that a link will send a message announcing its intent to make a decision during its chosen control mini-slot if it does not hear such a message from its neighbors.

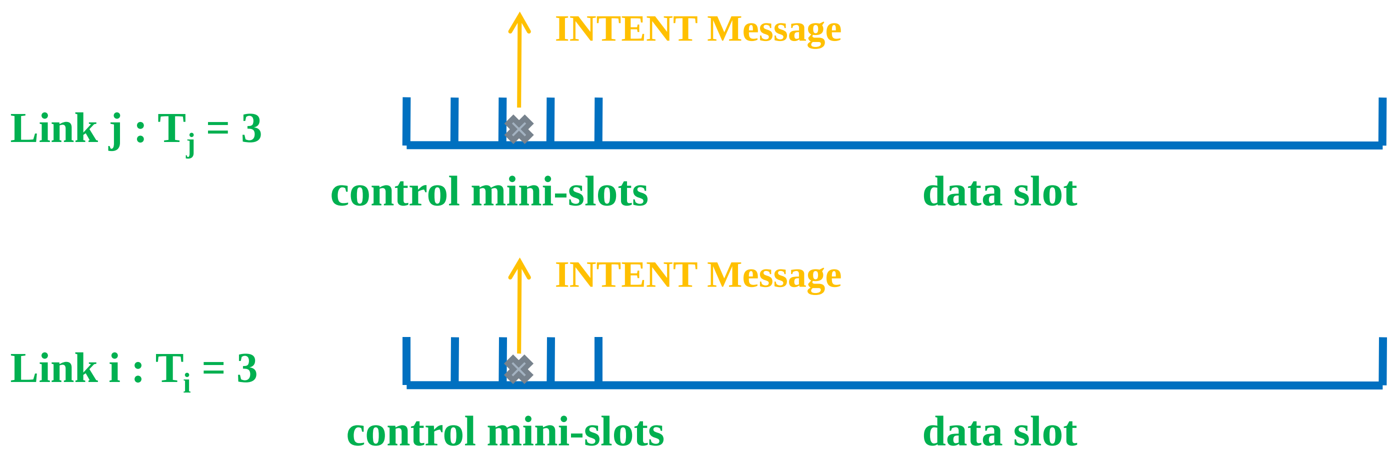
Case 1

- If link i hears an **INTENT** message from a link in its neighborhood **C**(i) before its chosen mini-slot, it will keep its state unchanged from the previous time-slot.
  - If it was **active** in the previous time slot, it will continue to be **active**; will be **inactive** otherwise.



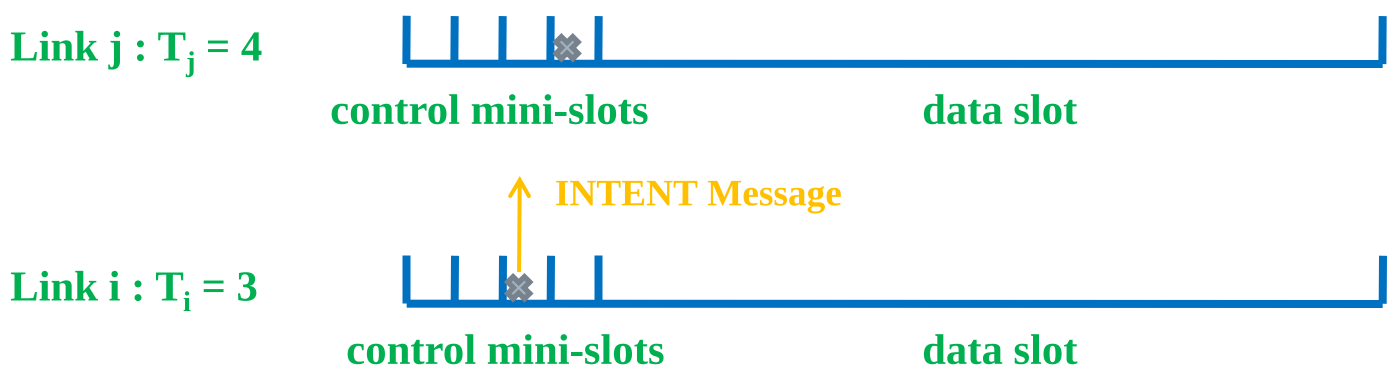
Case 2

- Otherwise, link i will broadcast an **INTENT** message to links in **C**(i) in the T_i-th control mini-slot.
- Case 2: If there is a collision, link i will not change its state.



Case 3

- If there is no collision, link i will make its decision:
  - If no links in **C**(i) were active in the previous data slot, then link i's state is chosen as follows:
    active with probability $\mathbf{p_i}$
    inactive with probability $\mathbf{1-p_i}$
  - Otherwise: **inactive**



Key Property of Q-CSMA

Proposition 2. The Q-CSMA algorithm achieves the product-form distribution if the window size $W \geq 2$.

- Any maximal schedule will be selected as the decision schedule with positive probability.
- The set of maximal schedules includes all the links.

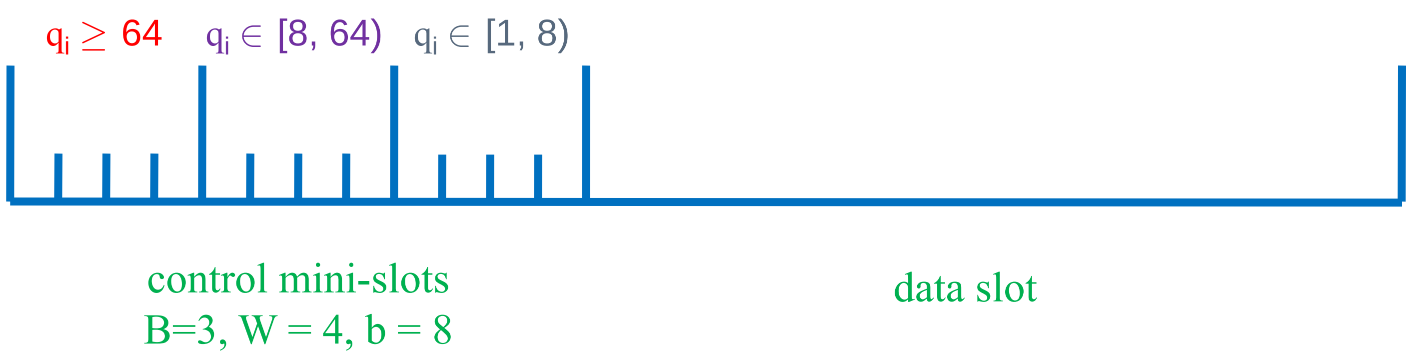
- Modification: Works even if $W=1$. A link chooses to participate in the decision schedule with probability ½. Again, one can show that the above result is still valid.

Hybrid Q-CSMA

- The delay performance of Q-CSMA can be quite bad although it is throughput-optimal.
- Question: can we enhance Q-CSMA to achieve better delay performance while retaining the throughput-optimality property?
- Idea: combine Q-CSMA with distributed approximations of Greedy Maximal Scheduling (GMS)
  - Why GMS? GMS is known to often perform as well as MWS in simulations (but is not provably throughput-optimal except in the case of small networks).

Distributed Approximation of GMS

- A control slot is divided into B frames, with each frame consisting of W mini-slots.
- Links are assigned a frame based on the log (base = b) of their queue lengths, and the W mini-slots within a frame are used to resolve contentions among links in a neighborhood.



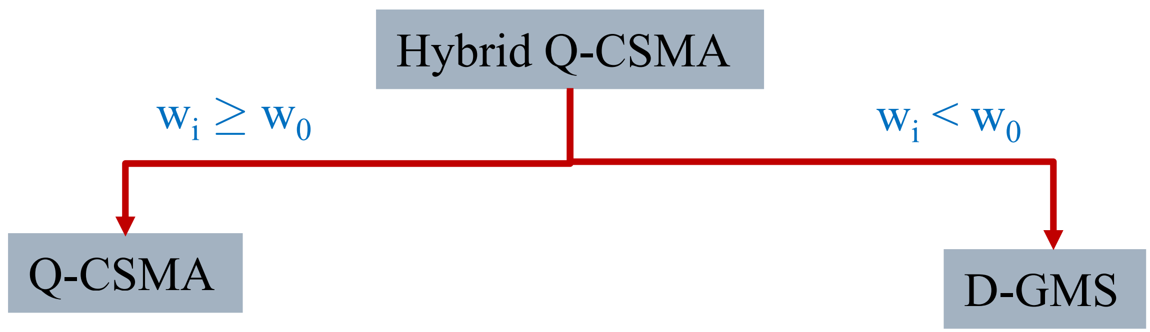
D-GMS

- At the beginning of each time slot, link i with queue length q_i selects a random control mini-slot:

$$\boxed{T_i = W \times \lfloor B - \log_b(q_i + 1) \rfloor^+ + \text{Uniform}[1, W]}$$

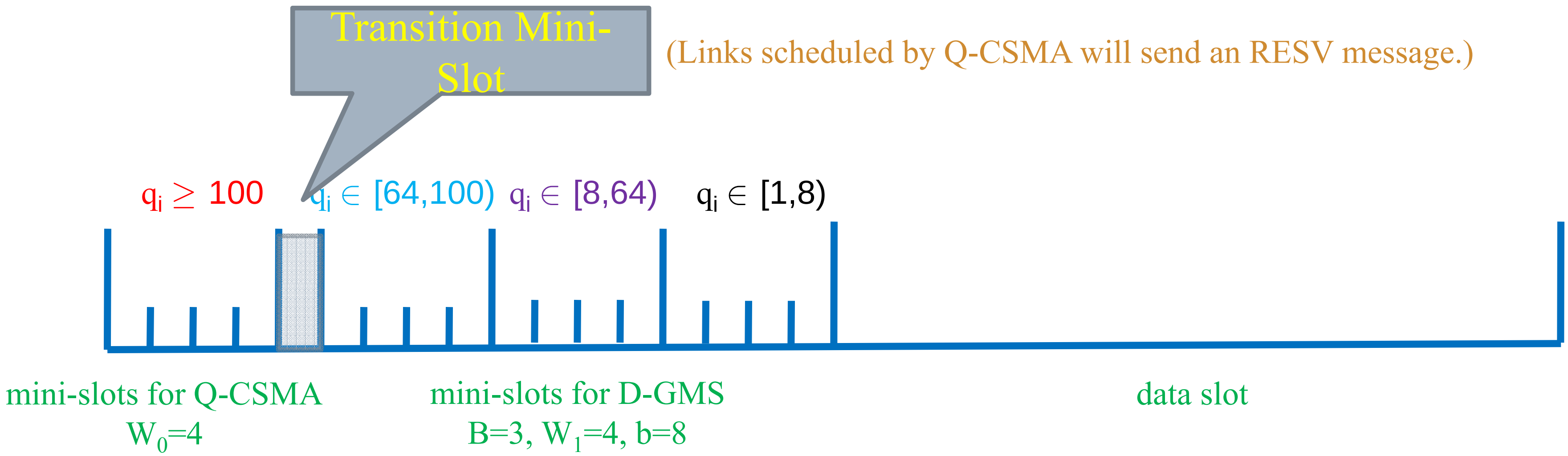
- If link i hears an **RESV** message from one of its neighbors before its chosen control mini-slot: be **inactive**.
- Otherwise, link i will broadcast an **RESV** message at the beginning of T_i-th control mini-slot:
  - collision: be **inactive**;
  - no collision: **transmit** a packet in the data slot.

Hybrid Q-CSMA



- For links with weight greater than a threshold w_0, apply Q-CSMA;
- For links with weight smaller than the threshold, apply D-GMS.

One Slot of Hybrid CSMA



Key Properties of Hybrid CSMA

- The transmission schedule for links with weight greater than the threshold is chosen according to the product-form distribution as before.

- Additional links (with weight smaller than the threshold) are added if possible using D-GMS which improves the delay performance.

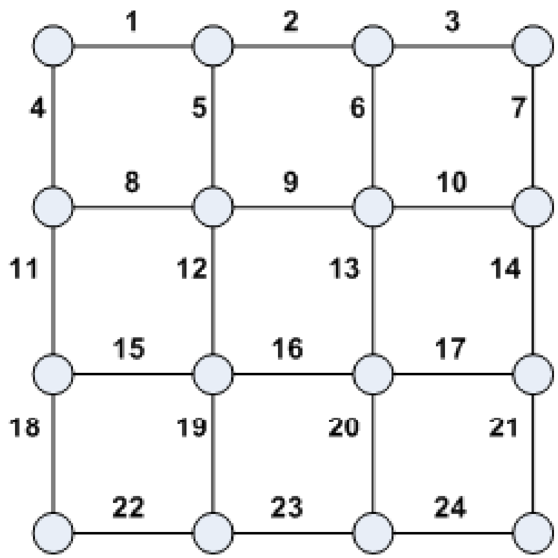
- Hybrid Q-CSMA is provably throughput-optimal (links with small weight don't matter).

Distributed Maximal Scheduling

- D-MS:
  - At the beginning of each time slot, link i selects a random control mini-slot T_i=Uniform[1,W].
  - If link i hears an **RESV** message from one of its neighbors before its chosen control mini-slot: be **inactive**.
  - Otherwise, link i will broadcast an **RESV** message at the beginning of T_i-th control mini-slot:
    - collision: be **inactive**;
    - no collision: **transmit** a packet in the data slot.

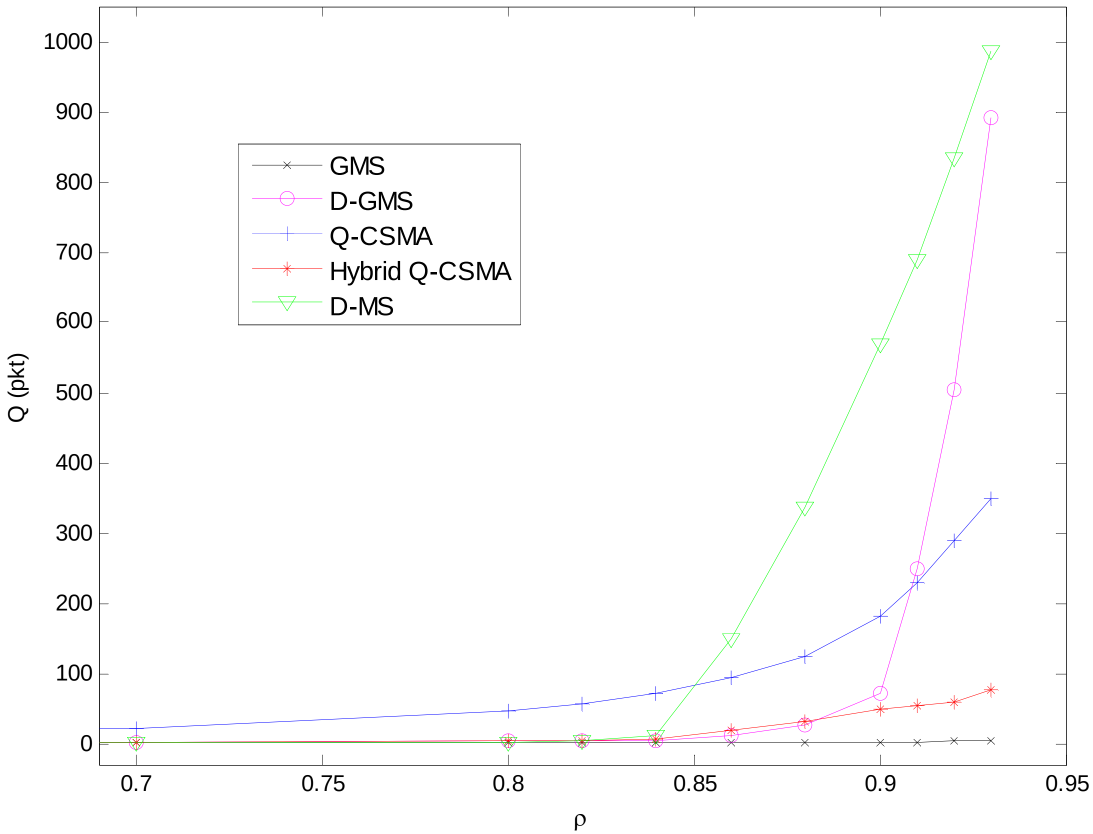
Simulation Evaluation

24-Link Grid Network
(1-hop interference model)



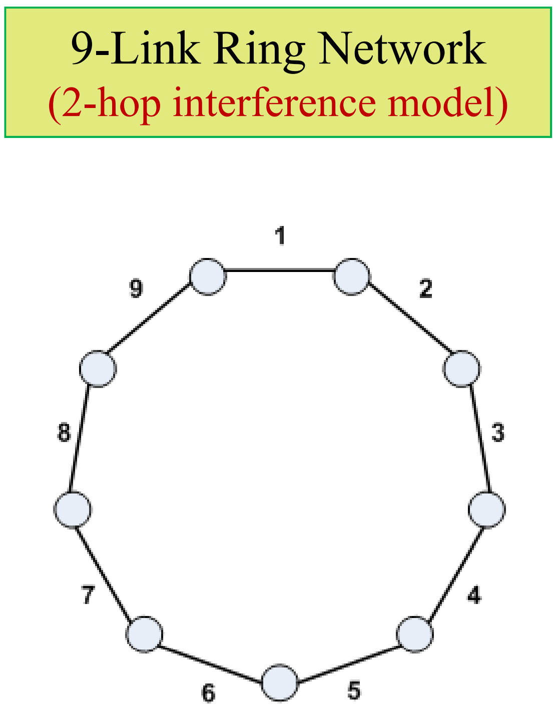
- Use a convex combination of several maximal schedules scaled by $\rho \in [0,1]$ as the arrival rate vector, ρ can be viewed as the traffic intensity.
- Compare GMS (centralized), D-GMS, D-MS, Q-CSMA, Hybrid Q-CSMA, allocate the same overhead for every distributed algorithm.
- Performance metric: **long-term average queue length per link.**

Grid Network Simulations



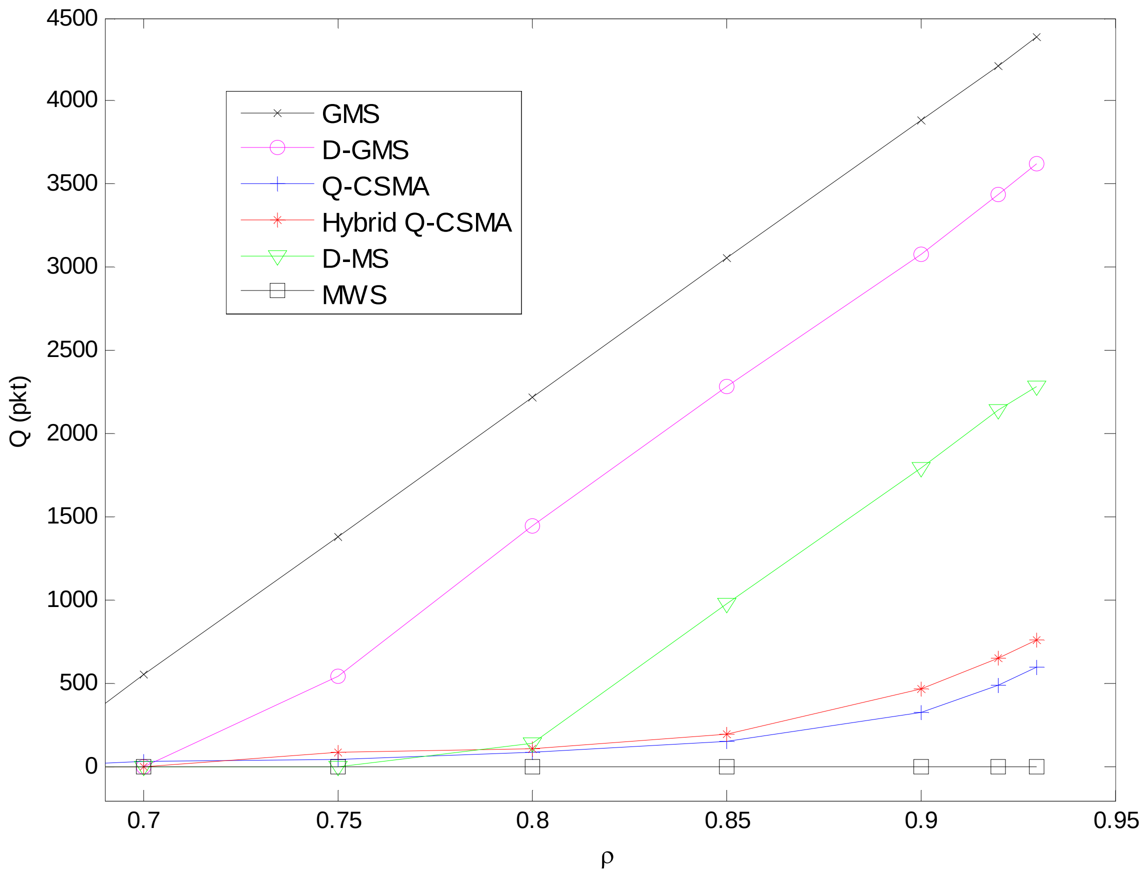
- D-GMS/D-MS have very good delay performance below a certain traffic intensity, and become unstable afterwards.
- Q-CSMA has worse delay performance than D-GMS/D-MS for small to moderate traffic intensity, and better delay performance for high traffic intensity.
- Hybrid Q-CSMA has the best delay performance among all distributed algorithms.

Ring Network



- This example shows that GMS may only achieve a fraction of the capacity region.
- Small maximal schedules: {1,5}, {2,6},{3,7},{4,8}, etc.
- Big maximal schedules: {1,4,7}, {2,5,8}, {3,6,9}.
- There exists a traffic pattern that forces GMS to use small maximal schedules only, give a service rate of $\frac{2}{9}$ packet / link.
- If we time share among big maximal schedules, then we can serve $\frac{1}{3}$ packet / link.

Ring Network Simulations



- GMS, D-GMS, and D-MS are not stable (the queue lengths increase linearly with the running time) after the traffic intensity exceeds 0.67, 0.7, and 0.8, respectively.
- Q-CSMA and Hybrid Q-CSMA have much lower delay and the queue lengths are stable.

Conclusions

- Optimization theory provides a cookbook for solving resource allocation problems in queueing networks

- Lagrange multipliers are proportional to queue lengths
  - May need to interpret queue length appropriately: e.g., deficit counters, workloads

- Resource allocation decisions are made by comparing Lagrange multipliers using the MaxWeight algorithm
  - Typically obvious when writing out the dual formulation

- Distributed Algorithms: Similar to techniques in statistical physics