Optimization and Algorithms for Resource Allocation in Wireless Networks

A. Eryilmaz
(*Ohio State University*)

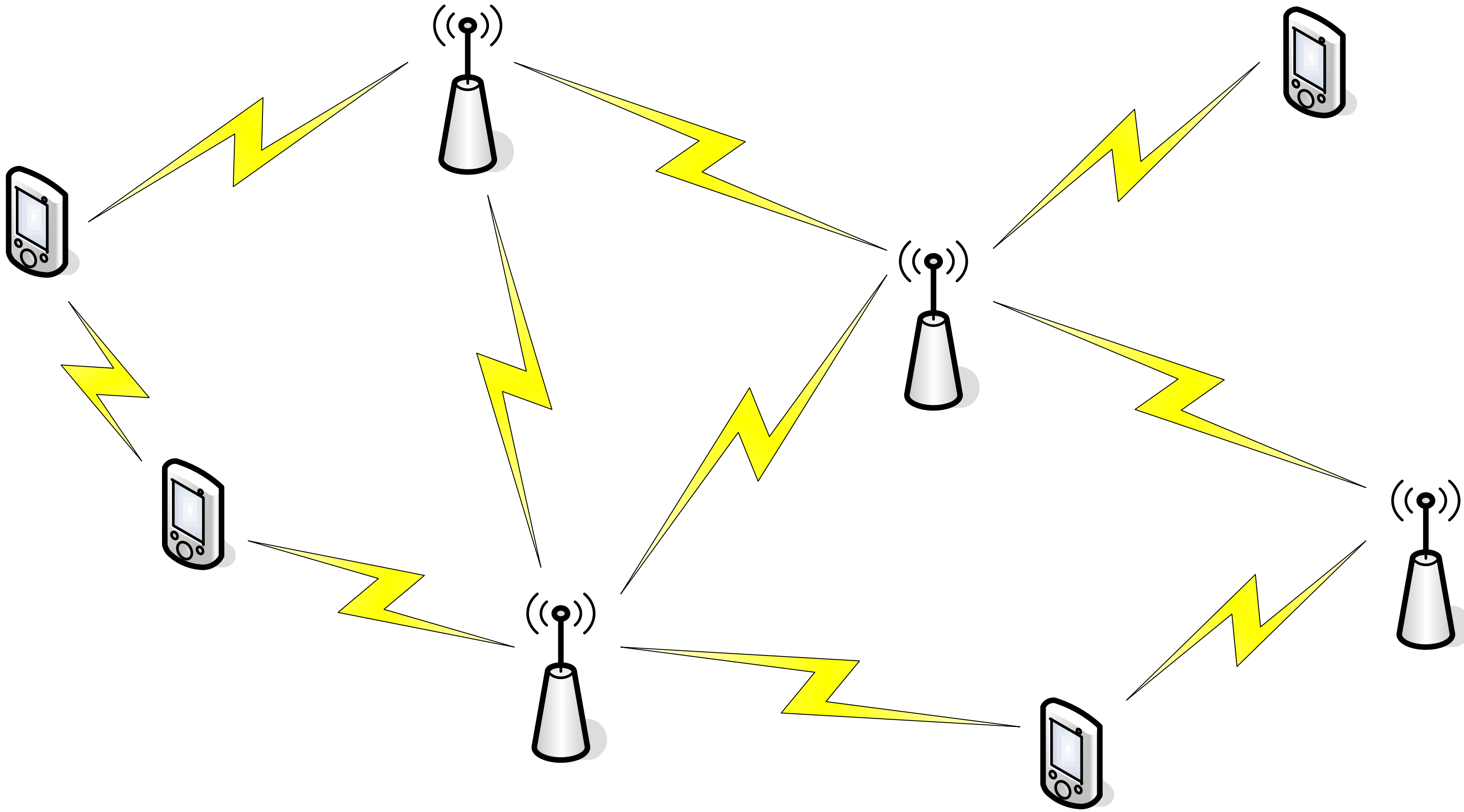
R. Srikant
(*UIUC*)

Softcopy available at: http://www.ece.osu.edu/~eryilmaz/Sigmetrics2008Tutorial.pdf

Outline

- Introduction
- Two key problems
  - Architecture for fair resource allocation (Srikant)
  - Distributed Algorithms (Eryilmaz)
- Open Issues

Multihop wireless network



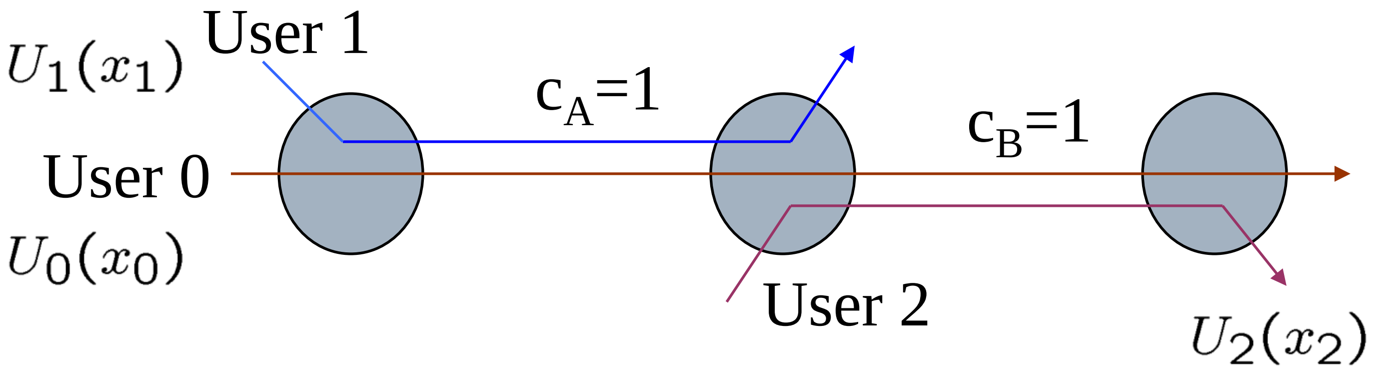
Multihop wireless network

- Different types of traffic sharing the wireless network:
  - Unicast and multicast
  - Short flows and long flows
  - Elastic and Inelastic
  - Real-time (with delay & jitter requirements) and non-real-time
- Need an *efficient protocol stack* to allocate resources between these different types of flows.

Resource Allocation

- **Design an optimal protocol stack architecture for networks with unicast and multicast flows**
  - Functional decomposition: what should the end users do? what should the network do? what should the nodes in the network do?
  - We only consider long elastic unicast and multicast flows.
  - Short flows and inelastic flows can be given higher priority and will act like stochastic fluctuations in the channel model.
- **Literature:**
  - Optimization & stochastic networks for unicast traffic: Eryilmaz & Srikant '05, '06; Stolyar '05, '06; Neely, Modiano & Li '05; Bui, Srikant & Stolyar '08
  - Optimization ideas for unicast traffic: Lin & Shroff '04, Chiang '04, Lai, Pachalidis & Starobinski '05.

Three-node wireless network



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

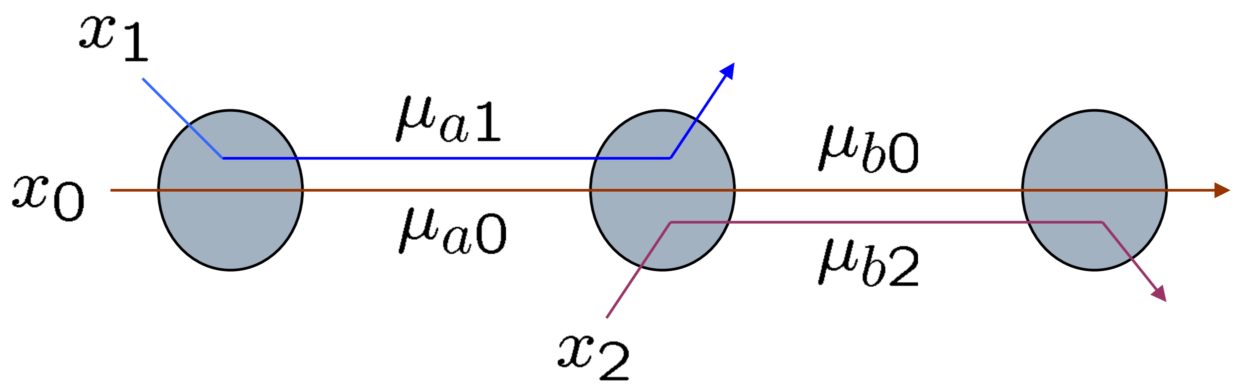
$$\max_{x,\mu \geq 0} \sum_i U_i(x_i)$$

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



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

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

Lagrangian Decomposition

Congestion control:

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

$\Rightarrow$ User 0: $\max_{x_0 \geq 0} U_0(x_0) - p_{a0}x_0$

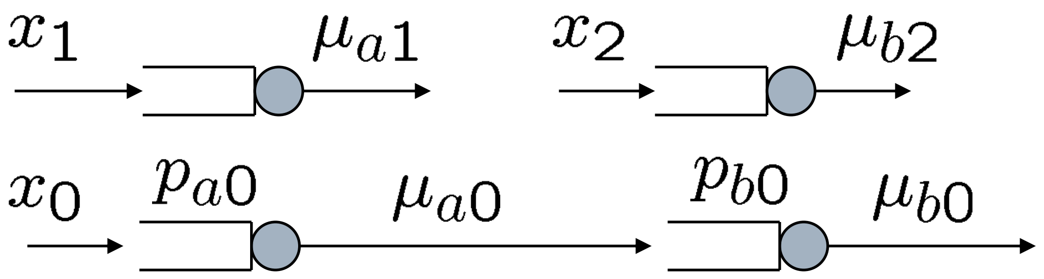
...

MAC or Scheduling:

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

Solution is an extreme point!

Resource Constraints and Queue Dynamics



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

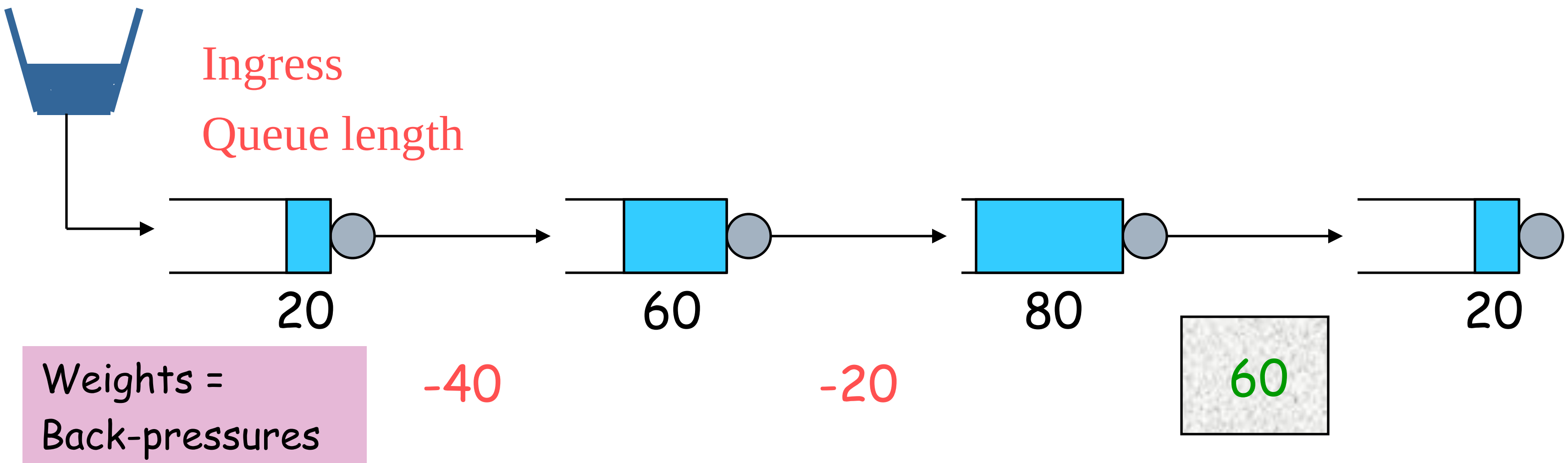
subject to

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

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

Ingress Queue-based Congestion Control

Congestion Control for flow *f* : Decrease transmission rate if *ingress queue length* is large.



- Back-pressure algorithm controls congestion in the interior of the network
- Thus, unlike the TCP protocol in the Internet, source does not have to react to end-to-end congestion

Queueing and Optimization

- Each constraint is represented by a queue:

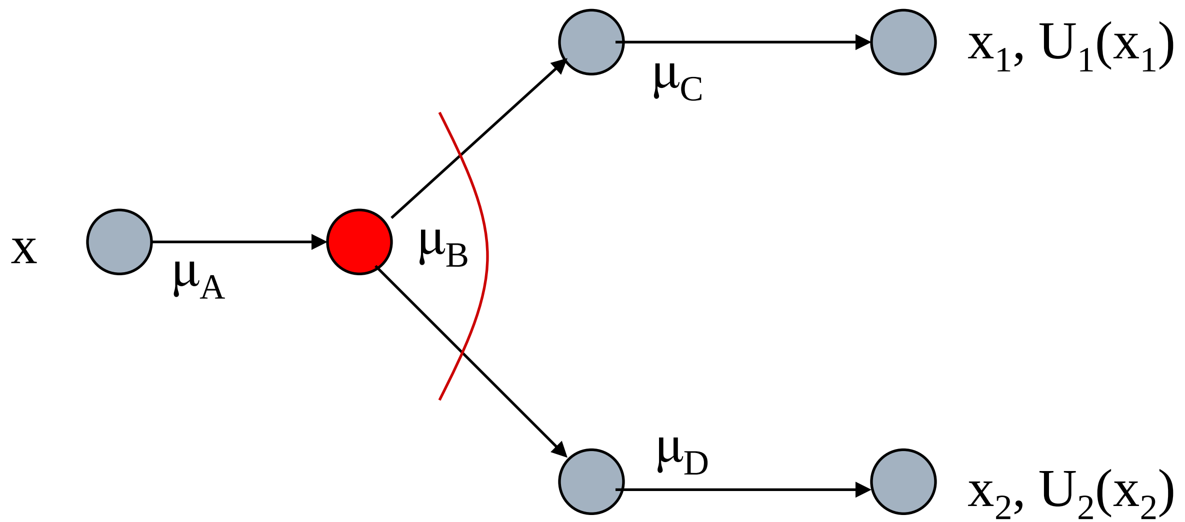
$$y \le x$$

- Stability of the queue implies constraint is satisfied and vice-versa
- Proofs don't immediately follow from dual decomposition theory
- Stochastic networks: Theory extends to general stochastic networks; the optimization problem is formulated in terms of "averages."

Multicast Session: One-to-Many

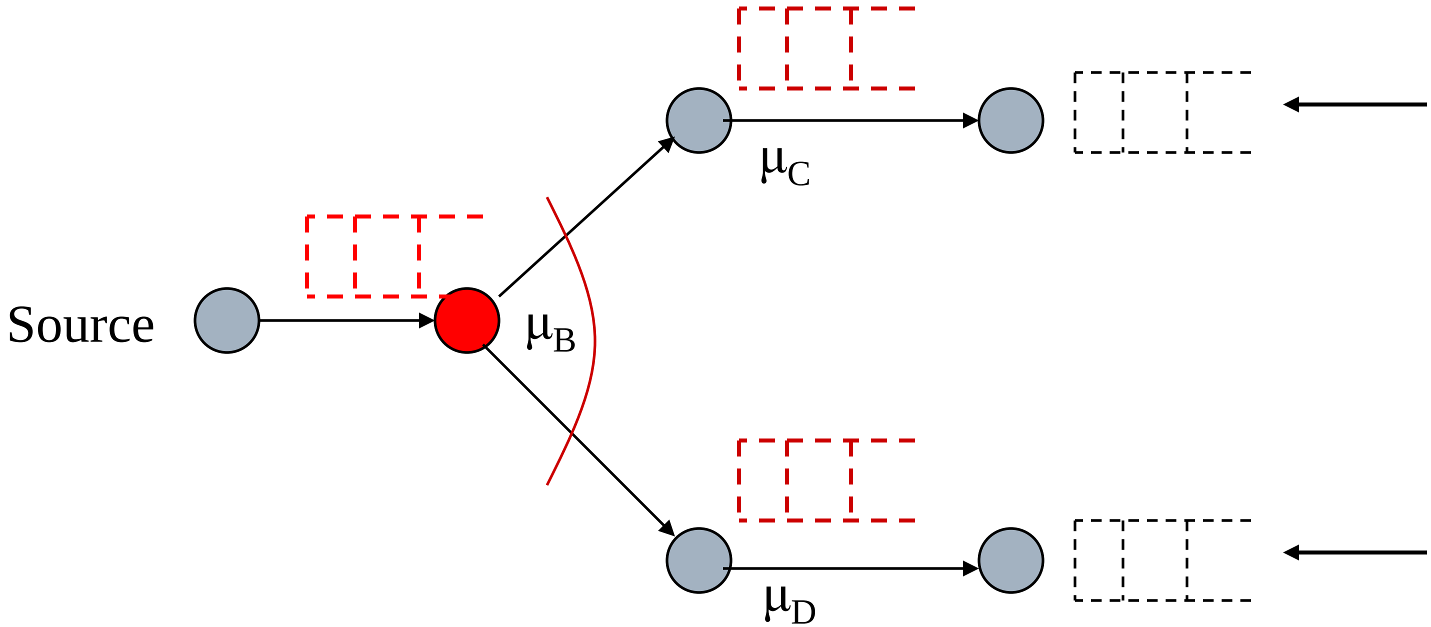
- Assume fixed routing: multicast trees are given.
- Single-rate multicast:
  - All receivers have to receive at the same rate.
  - Those above results can be easily extended, with some modifications to the back-pressure algorithm.
- Multi-rate multicast?
  - Each receiver can choose to receive at a different depending upon the congestion in their neighborhood (Internet: Deb & Srikant; Kar, Sarkar & Tassiulas).
    - Very important in wireless networks; otherwise, all rates may become zero frequently
  - Implemented using layered video coding, for example. Each receiver can subscribe to a subset of the layers.

Multi-rate multicast



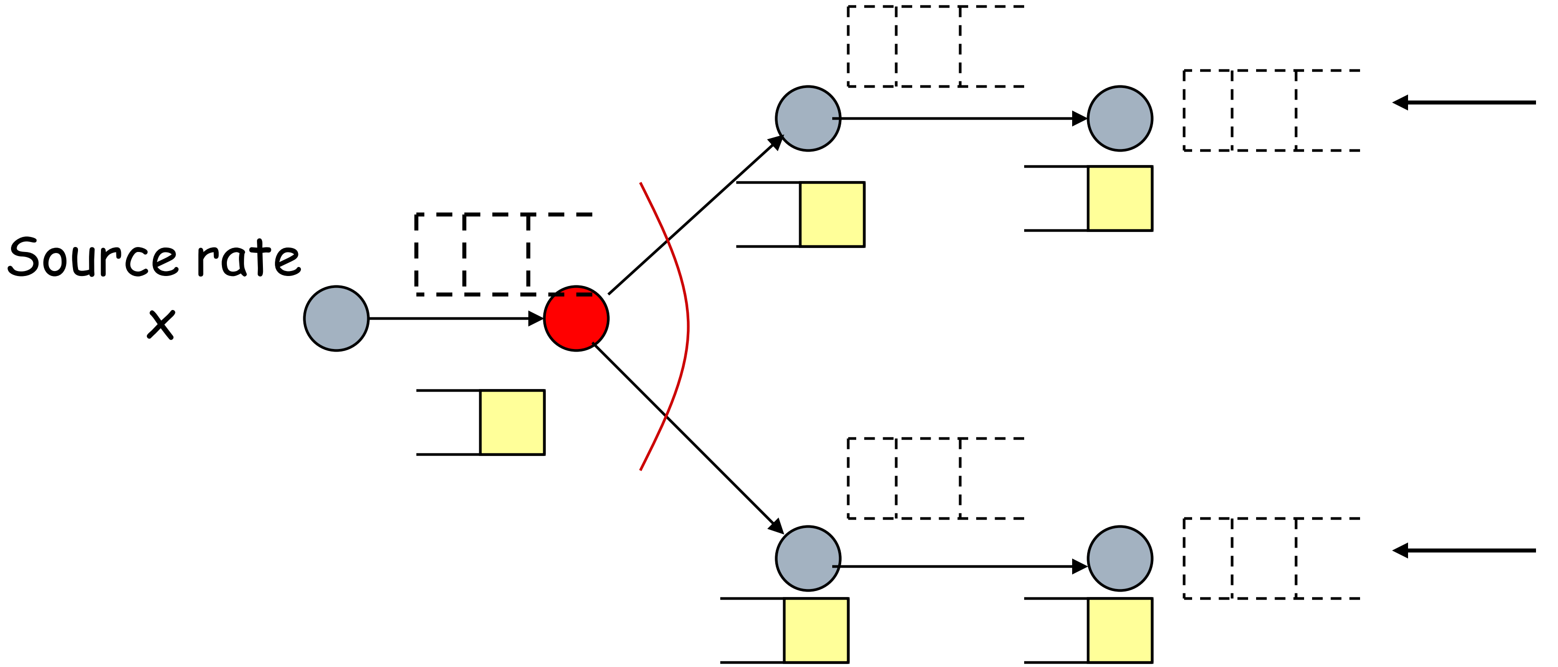
- One sender, two receivers
- Example of constraint: $\mu_B \geq \max\{\mu_C, \mu_D\}$

Solution: Multi-rate multicast



- Constraint: $\mu_B \geq \max\{\mu_C, \mu_D\}$
- A fictitious queueing network sending fictitious packets in the opposite direction enforces the constraints.
- The red queue doesn't behave like a normal queue: its arrival rate in a time slot is the maximum (not the sum) of the departure rates from the two blue queues.

Real Packet Generation

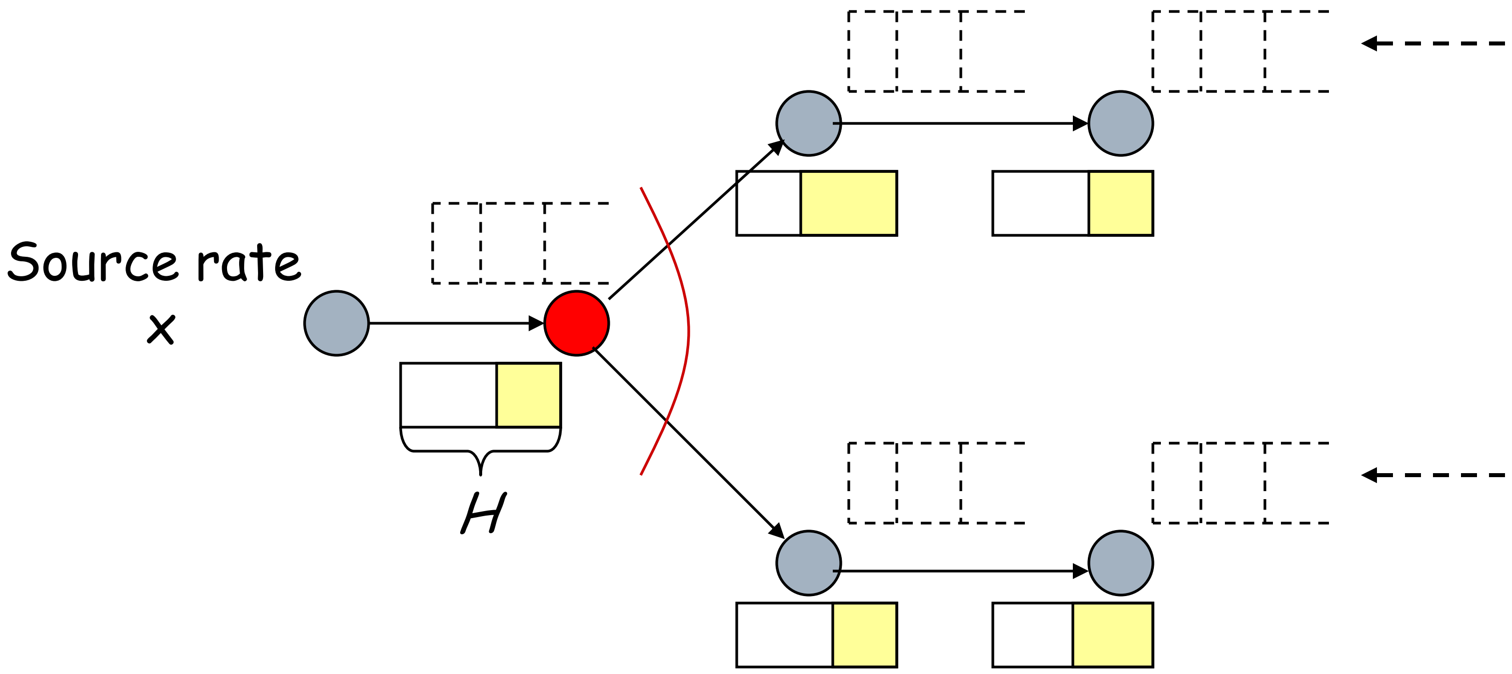


- Source can send a packet for every token, or can generate 9 packets for every 10 tokens received.
- Tokens inform the source of the amount of resources reserved for it.
- Source can use this information, but sends at a smaller rate to ensure the stability of the real queues (yellow).

Result I

- Token generation rate at each destination is equal to the solution of the optimization problem.
- Fraction of packets reaching each destination is close to the token generation rate of the destination.
- The Markov chain of the shadow queues and the real queues is positive recurrent.
- The first moments of the queue lengths exist and are finite.

Finite-buffer queues



- Real packet generation policy: transmit as many real packets at the source as tokens received; no thinning as before.

- Result II: When the buffer size is large, the received rate at each receiver is close to the token generation rate.

Summary of Architectural Results

- End users perform congestion control.
- Network allocates resources using the back-pressure algorithm:
  - Shadow queues are necessary to enforce multicast constraints.
  - Shadow packets serve as permits or tokens to generate packets.
  - Shadow network pushes packets from receivers to sources.
  - Sources send real packets in the opposite direction.
  - The back-pressure algorithm is implemented using shadow queue lengths.

Open Issues

Routing

- Back-pressure algorithm can also be used to select routes
  - No pre-defined routes necessary; automatically finds routes for each packet to maximize throughput
  - Per-packet routing at each node can result in loops, out-of-sequence delivery, etc. leading to large delays even when the network is lightly loaded. Solution?

Per-neighbor queues

- Reducing the number of queues:
  - Existing solutions require that each node (link) has to keep a separate queue for each flow (per-flow queue) or for each destination (per-destination queue)
  - Per-neighbor queues would lead to significant reduction in number of queues at each node
  - Even with fixed routing, per-destination queues lead to large delays when the number of hops is large. Per-neighbor queues may reduce the delay
  - But is the network stable?

Decentralization & Complexity

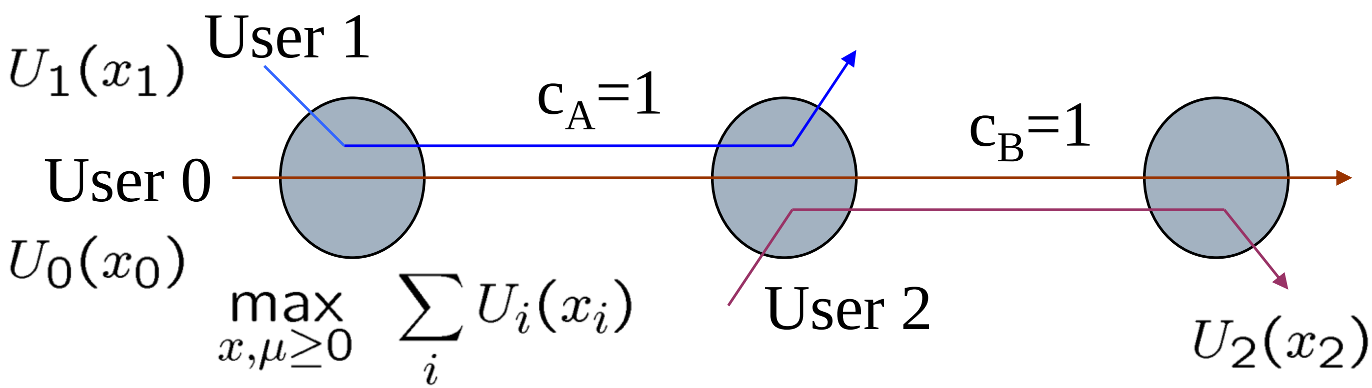
- Atilla Eryilmaz….

Distributed Algorithm Design

Atilla Eryilmaz

Ohio State University

Back to the 3 node example



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

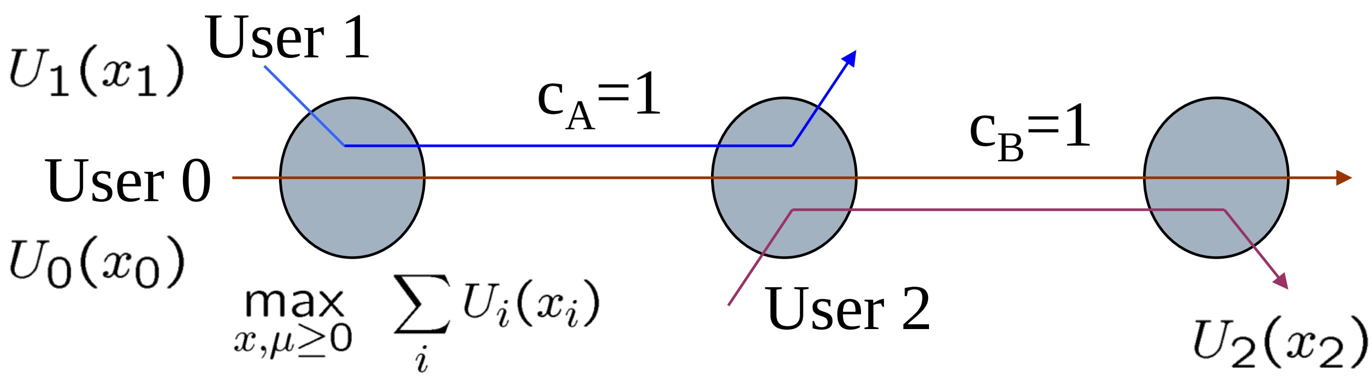
Congestion control:

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

MAC or Scheduling:

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

Back to the 3 node example



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

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

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

$$\max_{\mu\geq 0} \quad [\mu_{a0}(p_{a0} - p_{b0}) + \mu_{a1}p_{a1}] + [\mu_{b0}p_{b0} + \mu_{b2}p_{b1}]$$

$$s.t. \quad \boxed{\mu_{a0} + \mu_{a1} \leq 1, \qquad \mu_{b0} + \mu_{b2} \leq 1,} \leftarrow \text{Link Capacity Constr.}$$

$$\boxed{\mu_a + \mu_b \leq 1.} \leftarrow \text{Interference Constraints}$$

$$\max_{\mu\geq 0} \quad [\mu_a \max(p_{a0} - p_{b0}, p_{a1})] + [\mu_b \max(p_{b0}, p_{b1})]$$

$$s.t. \quad \mu_a + \mu_b \leq 1.$$

Back to the 3 node example

$$\max_{\mu \geq 0} \quad [\mu_{a0}(p_{a0} - p_{b0}) + \mu_{a1}p_{a1}] + [\mu_{b0}p_{b0} + \mu_{b2}p_{b1}]$$

$$s.t. \quad \boxed{\mu_{a0} + \mu_{a1} \leq 1, \qquad \mu_{b0} + \mu_{b2} \leq 1,}$$

$$\boxed{\mu_a + \mu_b \leq 1.}$$

Link Capacity Constr.

Interference Constraints

$$\max_{\mu \geq 0} \quad [\mu_a \max(p_{a0} - p_{b0}, p_{a1})] + [\mu_b \max(p_{b0}, p_{b1})]$$

$$s.t. \quad \mu_a + \mu_b \leq 1.$$

$w_a = \max(p_{a0} - p_{b0}, p_{a1})$

$w_b = \max(p_{b0}, p_{b1})$

$$\max_{\mu \geq 0} \quad [\mu_a w_a] + [\mu_b w_b]$$

$$s.t. \quad \mu_a + \mu_b \leq 1.$$

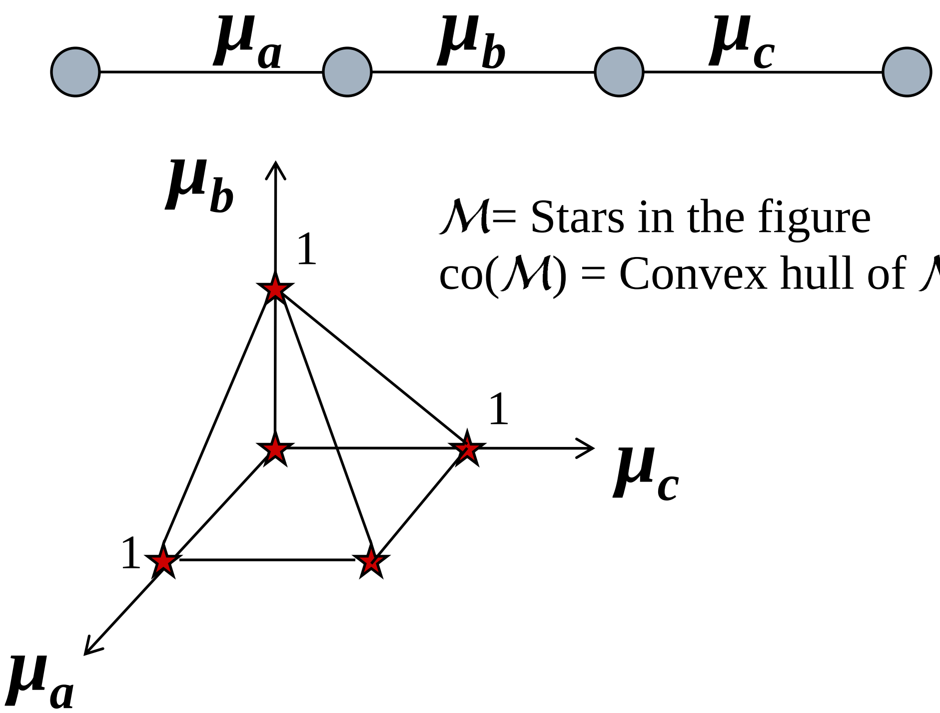
Generalized

$$\max \quad \sum_l \mu_l w_l$$

$$s.t. \quad \mu \in \mathcal{M}$$

Interference constraint determines the set of allowable link rates

Elaboration on $\mathcal{M}$ - Primary Interference

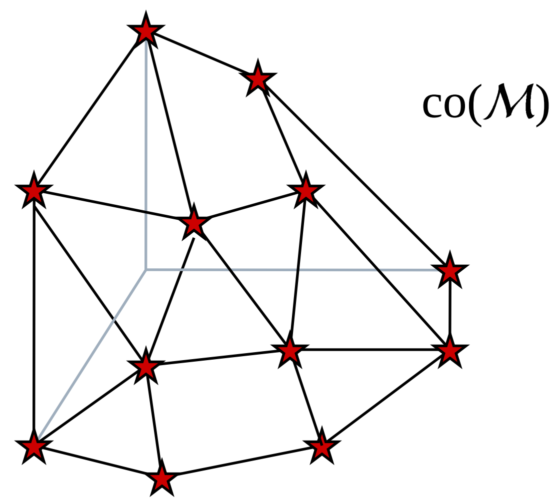


$\mathcal{M}$= Stars in the figure
co($\mathcal{M}$) = Convex hull of $\mathcal{M}$

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

<u>K^{th} order interference model</u>:
Any two active links must be separated by $\geq$ K links

- In general $\mathcal{M}$ is a complex set of rate allocations that depends on the topology and interference model
- Thus, it is very difficult to compute $\sum_l \mu_l w_l$ over $\mathcal{M}$



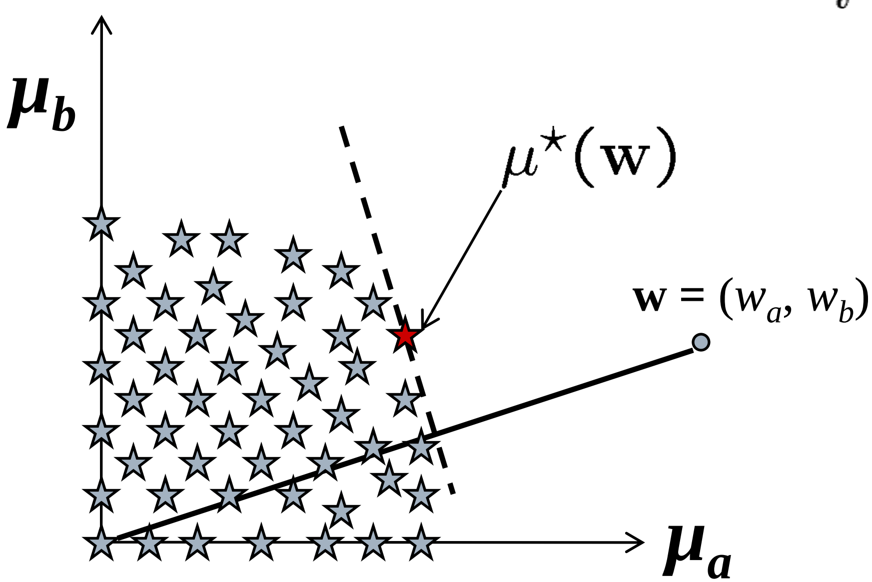
Goal

- Given the locally computable link weights ***w***, we aim to (approximately) solve

$$\max \sum_l \mu_l w_l \qquad \text{s.t.} \qquad \mu \in \mathcal{M}$$

distributively and with low-complexity operations.

- Let $\mu^\star(\mathbf{w}) \in \arg\max \sum_l \mu_l w_l \qquad \text{s.t.} \quad \mu \in \mathcal{M}$



- Different values of **w** lead to different μ* (**w**)
- For large values of **w** bounded changes in **w** has little effect on μ* (**w**)

Different Approaches

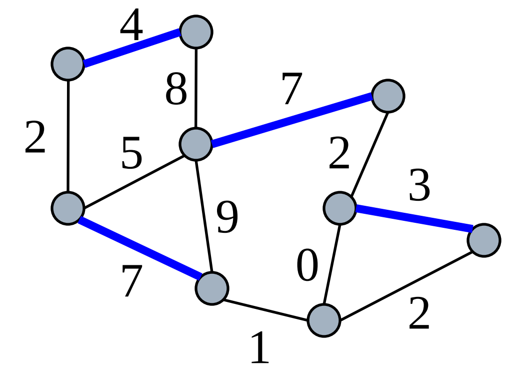
1. Greedy Algorithms [Lin, Shroff `05, Chaporkar, Kar, Sarkar `05, Wu, Srikant `05, Changhee-Joo et al. `07,`08, Brezinski, Zussman, Modiano `07,`08, etc.]
   - Choose link rates greedily from the largest weight to the smallest weight
   - In general, can only guarantee a fraction of the capacity region
   - Achieves full capacity for some cases
2. Pick and Compare Algorithms [Tassiulas `98, Modiano, Shah, Zussman `06, Eryilmaz, Ozdaglar, Modiano `07, `08, Sanghavi, Bui, Srikant `07, etc.]
   - Gradually improves the chosen link rates to get to the optimum over time
   - Can achieve full capacity for general topologies and interference
   - Higher complexity than greedy
3. Random Access Algorithms [Gupta, Stolyar `06, Gupta, Lin, Srikant `07, Rasool, Lin `07, Stolyar `07, Marbach, Eryilmaz, Ozdaglar `07, etc.]
   - Uses Aloha-like methods together with queue-lengths to adjust attempt probabilities
   - Achieves a fraction of the capacity region
   - Lowest complexity
4. Others [Shah `04, Deb et al. `06, …]

1. Greedy Maximal Matching (GMM)

Procedure:

1. Pick the link with the greatest weight
2. Eliminate all links that interfere with the selected link
3. Pick the link with the greatest weight in the remaining graph
4. Repeat.

Primary Interference Model



MaxWeight = 21

GMM Weight = 16

- In general, one can state that (for primary interference model)

$$\text{GMM Weight} \geq \text{Max Weight} / 2$$

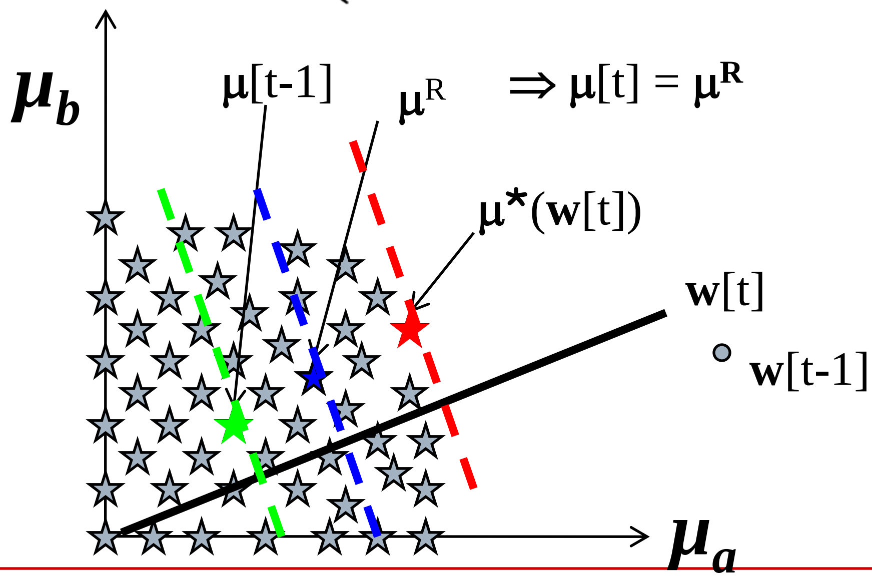
- In practice, GMM works much better than this lower bound
- Several works show that GMM can achieve full performance in the network satisfies certain properties

A Summary of Results on GMM

- [Preis `99], [Hoepman `04] – GMM can be implemented distributively
- [Dimakis, Walrand `05] found conditions under which GMM achieves full efficiency
- [Brzezinski, Zussman, Modiano `06, `08], [Joo, Lin, Shroff `07, `08] built on the work of Dimakis et al.
  - to translate the conditions into specific network topologies,
  - to develop schemes that can guarantee optimal performance,
  - to extend the conditions and study the worst case performance under various interference models
- [Joo et al. `08] showed that the worst case efficiency of GMM is between 1/6 and 1/3 for general K^{th} order *geometric network graphs*.
- A variant of GMM is the *Maximal Matching* (MM) algorithm which selects a random matching over the *non-zero-weighted* links
- [Chaporkar, Kar, Sarkar `05] and [Wu, Srikant, Perkins `06] studied the worst case performance of MM and showed that in the worst case it may perform very poorly

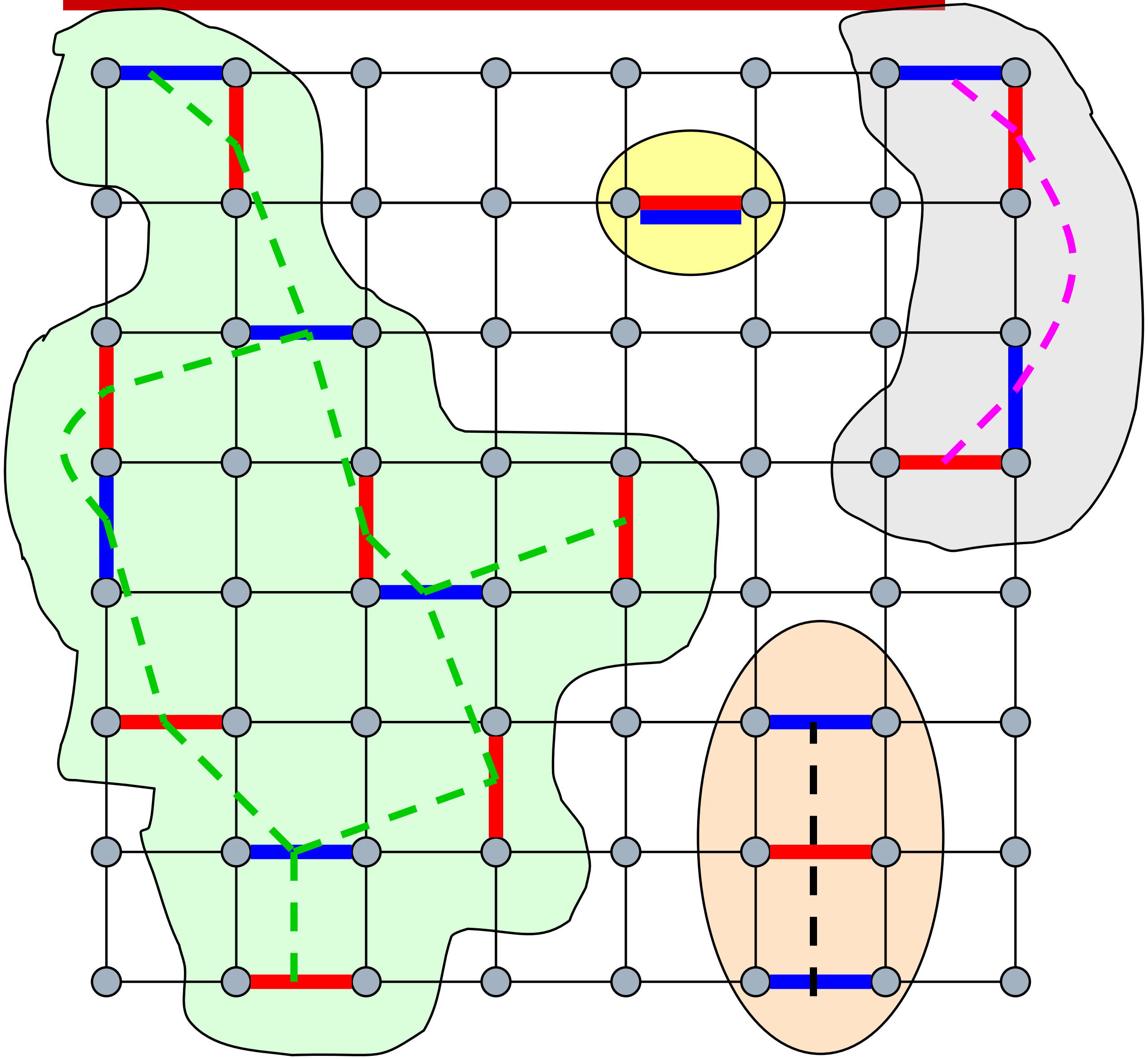
2. Pick and Compare Algorithm (PCA)

- Procedure: At step t,
  1. Pick: Select any feasible schedule (rate allocation) $\boldsymbol{\mu}^R$ randomly s.t.
  $$P(\boldsymbol{\mu}^R = \boldsymbol{\mu}^*(\mathbf{w}[t])) \geq \delta$$
  for some $\delta > 0$.
  2. Compare: Select the allocation $\boldsymbol{\mu}[t]$ such that
  $$\mu[t] = \begin{cases} \mu[t-1], & \text{if } \sum_l \mu_l[t-1]w_l[t] > \sum_l \mu_l^R w_l[t] \\ \mu^R, & \text{if } \sum_l \mu_l[t-1]w_l[t] \leq \sum_l \mu_l^R w_l[t] \end{cases}$$



- As time progresses, $\boldsymbol{\mu}[t]$ will gradually converge toward $\boldsymbol{\mu}^*(\mathbf{w}[t])$
- **Theorem:** PCA policy will achieve full capacity region for very general scenarios
- How to operate distributively?

Distributive operation [E., Ozdaglar, Modiano `07]



- Secondary interference model
- Grid network
- Two allocations : $\mu[t-1]$, μ^R
- Goal: compute and compare the total weights of the blue and red allocations distributively
- Connect the interfering links
- Creates isolated network components
- Each component can compare the two schedules independently

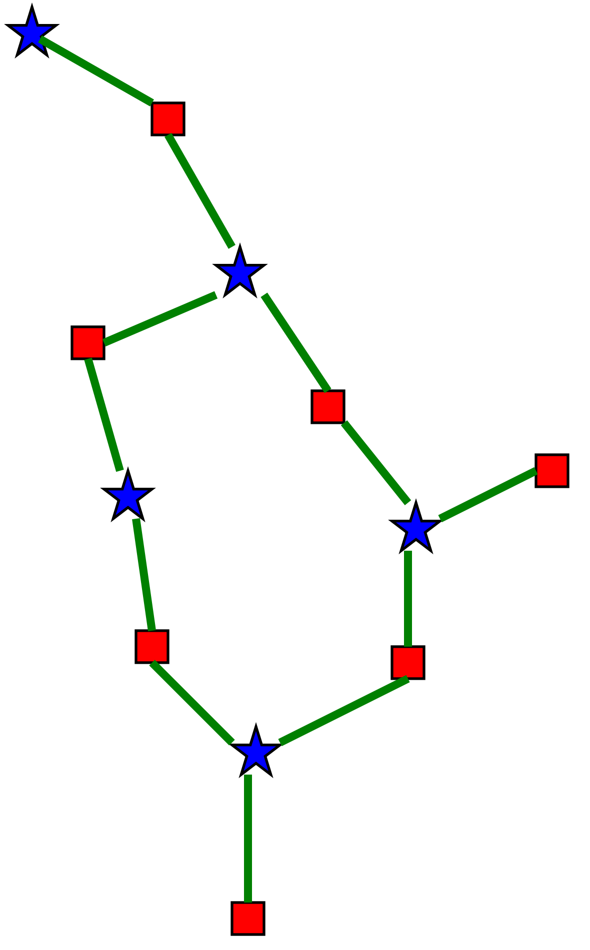
Conflict Graph

COMPARE ALGORITHM

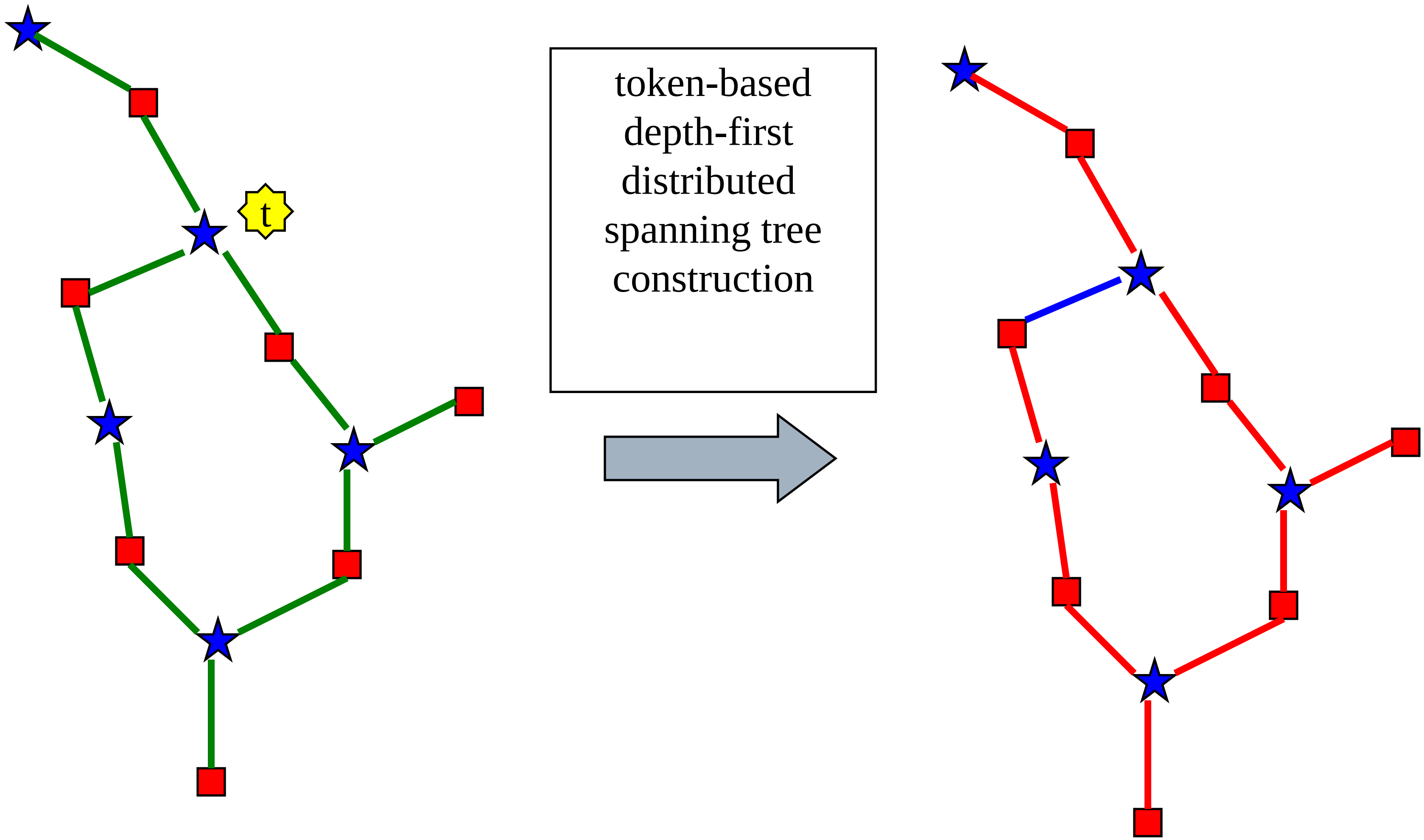
Find Spanning Tree

↓

Communicate & Decide

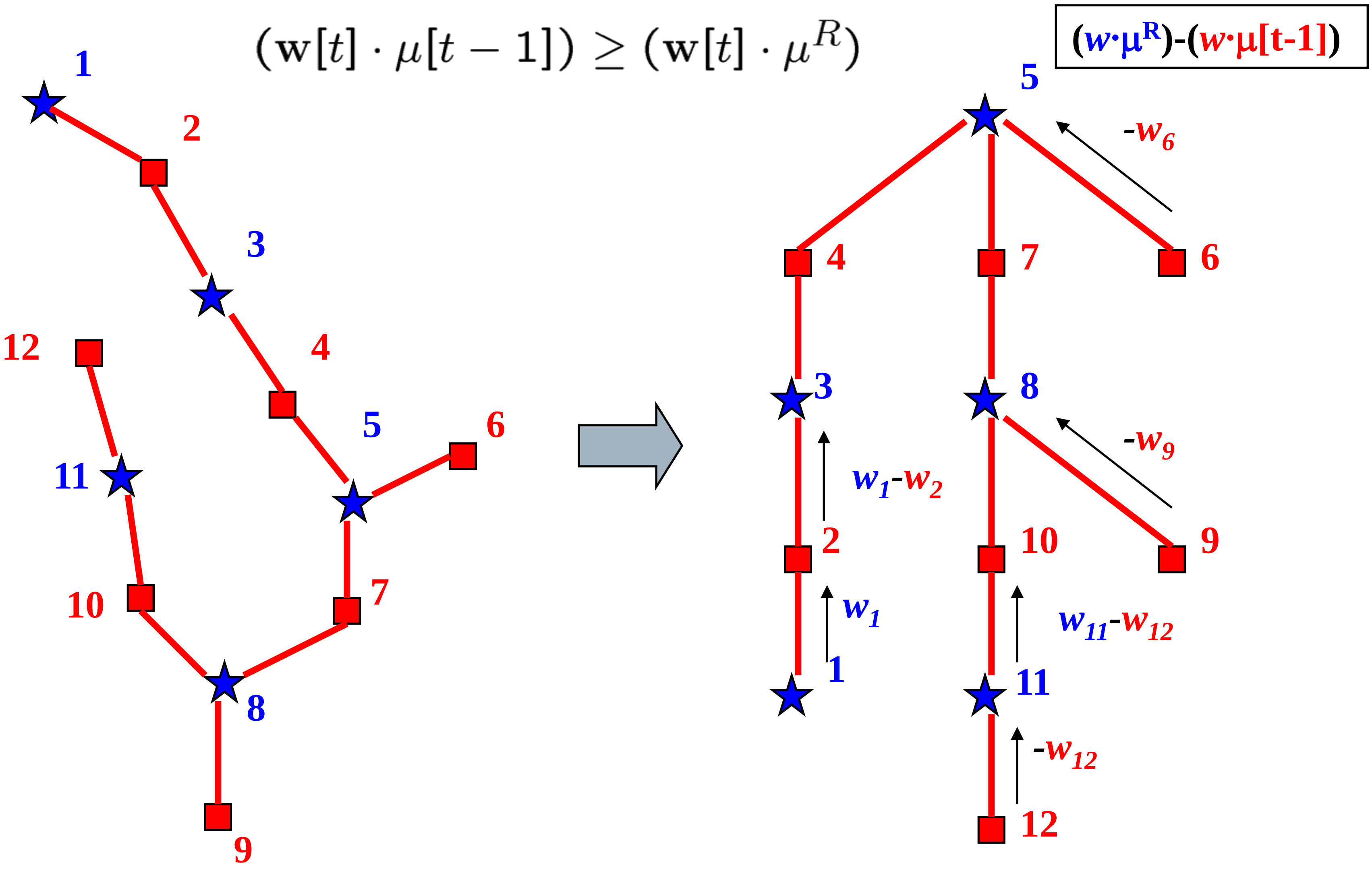


Find Spanning Tree Procedure



Communicate & Decide Procedure

$$(\mathbf{w}[t] \cdot \mu[t-1]) \geq (\mathbf{w}[t] \cdot \mu^R)$$



Results

- This algorithm achieves 100% efficiency with $O(N^3)$ time and $O(N^2)$ message exchanges in the worst case.
- [Modiano, Shah, Zussman `06] gives deterministic algorithms for the primary interference model, and gossip-based randomized algorithms for more general interference models with polynomial complexity and 100% efficiency.
- [Sanghavi, Bui, Srikant `07] focuses on primary interference model to develop O(m) complexity algorithm that achieves (m/(m+2)) fraction of the capacity region
- [Eryilmaz, Ozdaglar, Shah, Modiano `07, `08] studies the effect of the distributive implementations on the utility maximization problem, with and without imperfections and errors in the operation.

3. Random Access Algorithms

- In random access, the nodes try to capture links for transmission randomly
- Random access algorithms are amongst the lowest complexity algorithms there is
- But, with random access, collisions may be unavoidable
- Idea : Exploit locally available link weight information to set the channel access probabilities so that the link with a higher weight has a higher chance of capturing the channel, and collisions are limited
- The resulting algorithm is low complexity and amenable to distributed implementation, but is suboptimal
- Typically there is a tradeoff between the complexity and the degree of optimality of these schemes

Random Access Algorithms

- [Kar, Sarkar, Tassiulas `04], [Wang, Kar `05] proposed optimal random access schemes that use network topology information to achieve *proportional-fairness*
- [Gupta, Stolyar `05], extended the static scenario of Kar et al. to include dynamic link weights in the transmission probability selection
- [Stolyar `05], [Liu, Stolyar `07] used local queue-lengths to determine the transmission probabilities of each node, and showed that *saturation throughput region* of Aloha is supportable with their scheme
- [Marbach `04, `07] also suggested a combination of queue-length-based channel access and *active-queue-management*, and studied its stability characteristics
- [Lin, Rasool `06], [Gupta, Lin, Srikant `07] studied several other queue-length-based random access strategies with varying complexities and efficiency ratios (ranging from 1/3 to 1/2)
- [Bui, Eryilmaz, Srikant `06] studied the asynchronous implementation of a cross-layer algorithm with a random access scheduler.
- [Marbach, Eryilmaz, Ozdaglar `07] proposed and analyzed a CSMA policy with vanishing sensing time that achieves full efficiency as the network size scales

Open Problems

- Utility Maximization
  - Non-concave utility functions
  - Incorporating delay constraints
  - Network Coding
    - [Ho, Viswanathan `05, Eryilmaz, Lun `07, Ho `07, Wang, Shroff `07]
  - Rate of convergence
  - …
- Distributed Algorithm Design
  - Even Lower Complexity implementations and fundamental bounds
  - Overhead issues
  - Rate of convergence, delay performance analysis
  - Dealing with dynamics – mobility, fading
  - Asynchronous operation
  - …