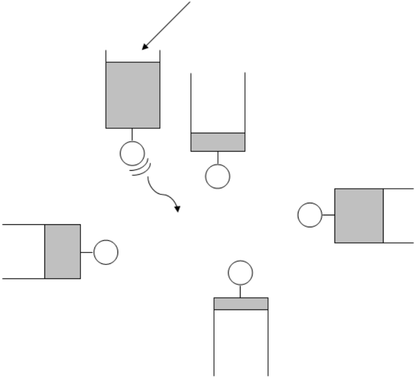
Positive Recurrent Medium Access Algorithm

Devavrat Shah* Jinwoo Shin† Prasad Tetali†

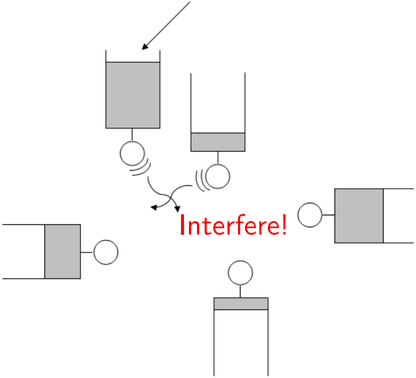
*LIDS, Massachusetts Institute of Technology

†Algorithms & Randomness Center, Georgia Institute of Technology

Medium Access Control (MAC) in Wireless Network



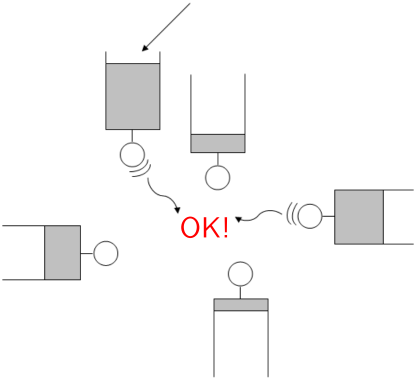
Medium Access Control (MAC) in Wireless Network



Constraints

- Interfering nodes can not transmit simultaneously.

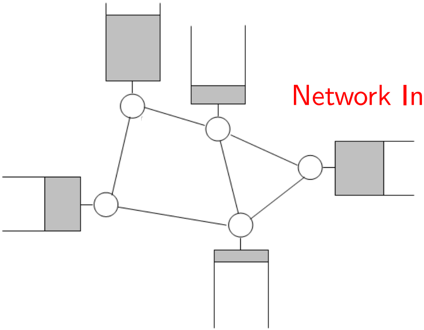
Medium Access Control (MAC) in Wireless Network



Constraints

- Interfering nodes can not transmit simultaneously.

Medium Access Control (MAC) in Wireless Network

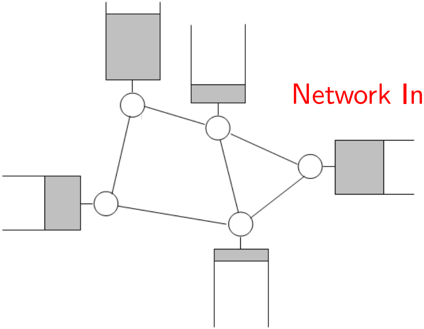


Network Interference Graph

Constraints

- Interfering nodes can not transmit simultaneously.
- Nodes have only "local" information
    - Is any interfering neighbor transmitting ?

Medium Access Control (MAC) in Wireless Network

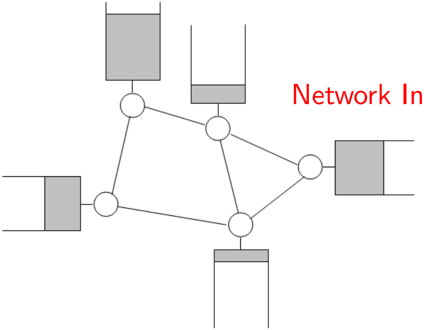


Network Interference Graph

Question

- Which nodes should transmit simultaneously using local information.
- So that performance is not compromised.

Medium Access Control (MAC) in Wireless Network

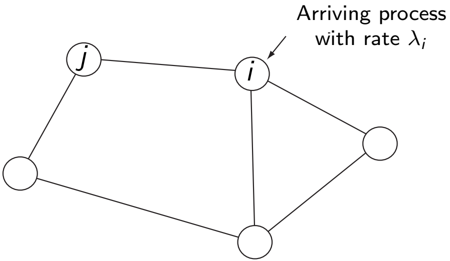


Network Interference Graph

Goal

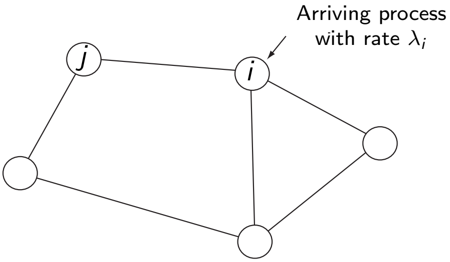
- Design an ‘distributed’, ‘efficient’ scheduling algorithm of ‘high performance’
    - Decides transmission of non-interfering nodes

Model



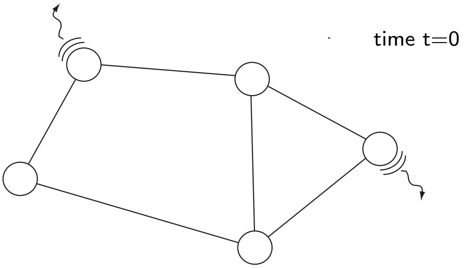
- Network interference graph $G = (V, E)$ with n queues as nodes
  - $E = \{(i, j) : i \text{ and } j \text{ cannot transmit simultaneously}\}$.
  - A packet arrives at queue i with probability λ_i at time $t \in \mathbb{Z}_+$.

Model



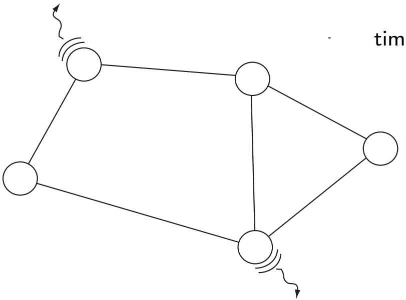
- Network interference graph $G = (V, E)$ with n queues as nodes
  - $E = \{(i, j) : i \text{ and } j \text{ cannot transmit simultaneously}\}$.
  - A packet arrives at queue i with probability λ_i at time $t \in \mathbb{Z}_+$.
- Scheduling algorithm: at each time instance $t \in \mathbb{Z}_+$
  - Selects non-interfering queues (to transmit) i.e. an independent set of G.
  - A packet in each selected queue departs (or serviced) from the network.

Model



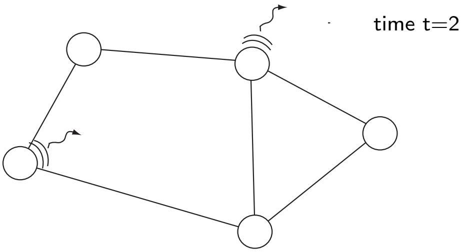
- Network interference graph $G = (V, E)$ with n queues as nodes
  - $E = \{(i, j) : i \text{ and } j \text{ cannot transmit simultaneously}\}$.
  - A packet arrives at queue i with probability λ_i at time $t \in \mathbb{Z}_+$.
- Scheduling algorithm: at each time instance $t \in \mathbb{Z}_+$
  - Selects non-interfering queues (to transmit) i.e. an independent set of G.
  - A packet in each selected queue departs (or serviced) from the network.

Model



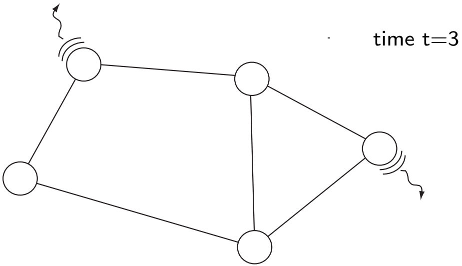
- Network interference graph $G = (V, E)$ with n queues as nodes
  - $E = \{(i, j) : i \text{ and } j \text{ cannot transmit simultaneously}\}$.
  - A packet arrives at queue i with probability λ_i at time $t \in \mathbb{Z}_+$.
- Scheduling algorithm: at each time instance $t \in \mathbb{Z}_+$
  - Selects non-interfering queues (to transmit) i.e. an independent set of G.
  - A packet in each selected queue departs (or serviced) from the network.

Model



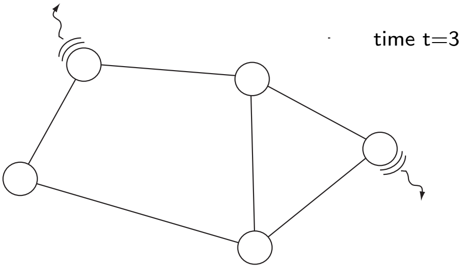
- Network interference graph $G = (V, E)$ with n queues as nodes
  - $E = \{(i, j) : i \text{ and } j \text{ cannot transmit simultaneously}\}$.
  - A packet arrives at queue i with probability λ_i at time $t \in \mathbb{Z}_+$.
- Scheduling algorithm: at each time instance $t \in \mathbb{Z}_+$
  - Selects non-interfering queues (to transmit) i.e. an independent set of G.
  - A packet in each selected queue departs (or serviced) from the network.

Model



- Network interference graph $G = (V, E)$ with n queues as nodes
  - $E = \{(i, j) : i \text{ and } j \text{ cannot transmit simultaneously}\}$.
  - A packet arrives at queue i with probability λ_i at time $t \in \mathbb{Z}_+$.
- Scheduling algorithm: at each time instance $t \in \mathbb{Z}_+$
  - Selects non-interfering queues (to transmit) i.e. an independent set of G.
  - A packet in each selected queue departs (or serviced) from the network.

Model



- Network interference graph $G = (V, E)$ with n queues as nodes
  - $E = \{(i, j) : i \text{ and } j \text{ cannot transmit simultaneously}\}$.
  - A packet arrives at queue i with probability λ_i at time $t \in \mathbb{Z}_+$.
- Scheduling algorithm: at each time instance $t \in \mathbb{Z}_+$
  - Selects non-interfering queues (to transmit) i.e. an independent set of G.
  - A packet in each selected queue departs (or serviced) from the network.
  - Our primary interest is to design a distributed algorithm minimizing cooperations.

Queueing Evolution

Notations

- $\mathbf{Q}(t) = [Q_i(t)]$ be the queue-sizes at time t.
- $\sigma_i(t) = \begin{cases} 1 & \text{if queue } i \text{ is transmitting (successfully) at time } t \\ 0 & \text{otherwise} \end{cases}.$
  - $\boldsymbol{\sigma}(t) = [\sigma_i(t)] \in \mathcal{I}(G) :=$ Collection of all independent sets in G.

$$Q_i(t+1) = Q_i(t) + A_i(t) - \sigma_i(t) \cdot 1_{\{Q_i(t)>0\}}$$

- $\mathbf{A}(t) = [A_i(t)]$ be the number of arrival packets at queue i at time t.
  - $\mathbb{E}[A_i(t)] = \lambda_i$.

Performance Metric

No common notion in the literature

- Want queues to be kept small under large arrival rate $\boldsymbol{\lambda} = [\lambda_i]$.
- In our model, $\boldsymbol{\lambda} \in \mathrm{conv}(\mathcal{I}(G))$.
  - Otherwise, queues should grow linearly over time.

Performance Metric

No common notion in the literature

- Want queues to be kept small under large arrival rate $\boldsymbol{\lambda} = [\lambda_i]$.
- In our model, $\boldsymbol{\lambda} \in \text{conv}(\mathcal{I}(G))$.
  - Otherwise, queues should grow linearly over time.

Scheduling algorithm is

- Positive recurrent if Markov chain $(\mathbf{Q}(t), \boldsymbol{\sigma}(t))$ is pos. rec. for $\boldsymbol{\lambda} \in \text{conv}^o(\mathcal{I}(G))$.
  - Hence, queues remain finite with probability 1.
- Rate stable if $\mathbf{Q}(t)/t \to 0$ with probability 1 for $\boldsymbol{\lambda} \in \text{conv}^o(\mathcal{I}(G))$.

Prior MAC Algorithms

Two Recent Research Directions

I. Starting from [Jiang and Walrand 08]
   - Based on arrival-rate information $\boldsymbol{\lambda}$

II. Starting from [Rajagopalan, Shah and Shin 09]
   - Based on queueing information $\mathbf{Q}(t)$

MAC Algorithm: Example

Each individual queue i makes her own decision as

> Attempt to transmit with probability $p_i(t)$ at time t.

MAC Algorithm: Example

Each individual queue i makes her own decision as

> Attempt to transmit with probability $p_i(t)$ at time t.

- The transmission of i is successful at time t i.e. $\sigma_i(t) = 1$ if
  - i attempts to transmit at time t &
  - no collision i.e. no (interfering) neighbors of i attempts to transmit at time t

MAC Algorithm: Example

Each individual queue i makes her own decision as

Attempt to transmit with probability $p_i(t)$ at time t.

- The transmission of i is successful at time t i.e. $\sigma_i(t) = 1$ if
  - i attempts to transmit at time t &
  - no collision i.e. no (interfering) neighbors of i attempts to transmit at time t
- Known to be positive recurrent if
  - $p_i(t)$ is some polynomial of $\#$ prior consecutive collisions of i &
  - interference graph G is complete [Hastad et al. 96], bipartite [Goldberg et al. 99].

MAC Algorithm: Example

Each individual queue i makes her own decision as

> Attempt to transmit with probability $p_i(t)$ at time t.

- The transmission of i is successful at time t i.e. $\sigma_i(t) = 1$ if
  - i attempts to transmit at time t &
  - no collision i.e. no (interfering) neighbors of i attempts to transmit at time t
- Known to be positive recurrent if
  - $p_i(t)$ is some polynomial of $\#$ prior consecutive collisions of i &
  - interference graph G is complete [Hastad et al. 96], bipartite [Goldberg et al. 99].
- Open question: How about general G?

MAC Algorithm: Example

Each individual queue i makes her own decision as

> Attempt to transmit with probability $p_i(t)$ at time t.

- The transmission of i is successful at time t i.e. $\sigma_i(t) = 1$ if
  - i attempts to transmit at time t &
  - no collision i.e. no (interfering) neighbors of i attempts to transmit at time t
- Known to be positive recurrent if
  - $p_i(t)$ is some polynomial of # prior consecutive collisions of i &
  - interference graph G is complete [Hastad et al. 96], bipartite [Goldberg et al. 99].
- Open question: How about general G?
  - Next: some positive answers utilizing additional local information

MAC Algorithm using Carrier Sensing (CSMA)

Each individual queue i makes her own decision as

Attempt to transmit with probability $p_i(t)$ at time t.

$$p_i(t) = \begin{cases} 0 & \text{if some (interfering) neighbor attempted at time } t-1 \\ 1 - \frac{1}{W_i(t)} & \text{else if } i \text{ attempted (to transmit) at time } t-1 \\ \frac{1}{2} & \text{otherwise} \end{cases}.$$

- Carrier Sensing Information
  - Knowledge whether neighbors attempted to transmit (at the previous time-slot).

MAC Algorithm using Carrier Sensing (CSMA)

Each individual queue i makes her own decision as

Attempt to transmit with probability $p_i(t)$ at time t.

$$p_i(t) \;=\; \begin{cases} 0 & \text{if some (interfering) neighbor attempted at time } t-1 \\ 1 - \frac{1}{W_i(t)} & \text{else if } i \text{ attempted (to transmit) at time } t-1 \\ \frac{1}{2} & \text{otherwise} \end{cases}.$$

- Carrier Sensing Information
  - Knowledge whether neighbors attempted to transmit (at the previous time-slot).
- Next: Two known successful designs of $W_i(t)$
  - Using arrival-rate information $\boldsymbol{\lambda}$ [Jiang and Walrand 08]
  - Using queueing information $\mathbf{Q}(t)$ [Rajagopalan, Shah and Shin 09]

CSMA I using Arrival-rate Information

Theorem (Jiang and Walrand 08)

For given arrival rate $\boldsymbol{\lambda} \in conv^o(\mathcal{I}(G))$, *there exists* $\mathbf{W}^* = \mathbf{W}^*(\lambda, G)$ *such that*

$$\textit{CSMA using } W_i(t) = W_i^* \textit{ is rate stable.}$$

CSMA I using Arrival-rate Information

Theorem (Jiang and Walrand 08)

For given arrival rate $\boldsymbol{\lambda} \in \mathit{conv}^o(\mathcal{I}(G))$, *there exists* $\mathbf{W}^* = \mathbf{W}^*(\lambda, G)$ *such that*

$$\textit{CSMA using } W_i(t) = W_i^* \textit{ is rate stable.}$$

- To find W_i^* at node i (in a distributed manner, without message-passing)
  - Require appropriate updating rule/period of $W_i(t)$
  - So that $W_i(t)$ converges to W_i^*.

CSMA I using Arrival-rate Information

Theorem (Jiang and Walrand 08)

For given arrival rate $\boldsymbol{\lambda} \in conv^o(\mathcal{I}(G))$, there exists $\mathbf{W}^* = \mathbf{W}^*(\lambda, G)$ such that

$$\textit{CSMA using } W_i(t) = W_i^* \textit{ is rate stable.}$$

- To find W_i^* at node i (in a distributed manner, without message-passing)
  - Require appropriate updating rule/period of $W_i(t)$
  - So that $W_i(t)$ converges to W_i^*.
  - Design such an updating period for rate stability.
    [Jiang, Shah, Shin and Walrand 09]

CSMA I using Arrival-rate Information

Theorem (Jiang and Walrand 08)

For given arrival rate $\boldsymbol{\lambda} \in conv^o(\mathcal{I}(G))$, there exists $\mathbf{W}^* = \mathbf{W}^*(\lambda, G)$ such that

$$\textit{CSMA using } W_i(t) = W_i^* \textit{ is rate stable.}$$

- To find W_i^* at node i (in a distributed manner, without message-passing)
  - Require appropriate updating rule/period of $W_i(t)$
  - So that $W_i(t)$ converges to W_i^*.
  - Design such an updating period for rate stability.
    [Jiang, Shah, Shin and Walrand 09]

- However, should assume that $\boldsymbol{\lambda}$ is possible to estimate.
  - In practice, $\boldsymbol{\lambda}$ is difficult to collect/know in many applications.

CSMA I using Arrival-rate Information

Theorem (Jiang and Walrand 08)

For given arrival rate $\boldsymbol{\lambda} \in conv^o(\mathcal{I}(G))$, there exists $\mathbf{W}^* = \mathbf{W}^*(\lambda, G)$ such that

$$\textit{CSMA using } W_i(t) = W_i^* \textit{ is rate stable.}$$

- To find W_i^* at node i (in a distributed manner, without message-passing)
  - Require appropriate updating rule/period of $W_i(t)$
  - So that $W_i(t)$ converges to W_i^*.
  - Design such an updating period for rate stability.
    [Jiang, Shah, Shin and Walrand 09]

- However, should assume that $\boldsymbol{\lambda}$ is possible to estimate.
  - In practice, $\boldsymbol{\lambda}$ is difficult to collect/know in many applications.

- Question: Possible to design $\mathbf{W}(t)$ using queueing information?

CSMA II using Queueing Information

Theorem (Shah and Shin 10)

CSMA is positive recurrent if

$$W_i(t) \;=\; \max\left\{\log Q_i(t)\;,\; e^{\sqrt{\log\,\log Q_{\max}(t)}}\;\right\},$$

where $Q_{\max}(t) = \max_i Q_i(t)$.

CSMA II using Queueing Information

Theorem (Shah and Shin 10)

CSMA is positive recurrent if

$$W_i(t) \; = \; \max\left\{\log Q_i(t)\,,\; e^{\sqrt{\log\,\log Q_{\max}(t)}}\right\},$$

where $Q_{\max}(t) = \max_i Q_i(t)$.

- Myopic & robust against the arrival assumption.

CSMA II using Queueing Information

Theorem (Shah and Shin 10)

CSMA is positive recurrent if

$$W_i(t) \;=\; \max\left\{\log Q_i(t)\;,\; e^{\sqrt{\log\,\log Q_{\max}(t)}}\;\right\},$$

where $Q_{\max}(t) = \max_i Q_i(t)$.

- Myopic & robust against the arrival assumption.
- To compute $W_i(t)$, it requires to know global information $Q_{\max}(t)$.

CSMA II using Queueing Information

Theorem (Shah and Shin 10)

CSMA is positive recurrent if

$$W_i(t) \;=\; \max\left\{\log Q_i(t)\;,\; e^{\sqrt{\log \max_{j\in\mathcal{N}(i)} W_j(t-1)}}\right\},$$

where $Q_{\max}(t) = \max_i Q_i(t)$.

- Myopic & robust against the arrival assumption.
- To compute $W_i(t)$, it requires to know global information $Q_{\max}(t)$.

CSMA II using Queueing Information

Theorem (Shah and Shin 10)

CSMA is positive recurrent if

$$W_i(t) \;=\; \max\left\{\log Q_i(t)\;,\; e^{\sqrt{\log \max_{j\in\mathcal{N}(i)} W_j(t-1)}}\right\},$$

where $Q_{\max}(t) = \max_i Q_i(t)$*.*

- Myopic & robust against the arrival assumption.
- To compute $W_i(t)$, it requires to know global information $Q_{\max}(t)$.
  - Still require some explicit message passing (minimal though)

CSMA II using Queueing Information

Theorem (Shah and Shin 10)

CSMA is positive recurrent if

$$W_i(t) \;=\; \max\left\{\log Q_i(t)\;,\; e^{\sqrt{\log \max_{j \in \mathcal{N}(i)} W_j(t-1)}}\right\},$$

where $Q_{\max}(t) = \max_i Q_i(t)$.

- Myopic & robust against the arrival assumption.
- To compute $W_i(t)$, it requires to know global information $Q_{\max}(t)$.
  - Still require some explicit message passing (minimal though)

Main Result of This Talk

- We revise the algorithm so that it does not require such message passing.

CSMA II using Queueing Information

Theorem (Shah and Shin 10)

CSMA is positive recurrent if

$$W_i(t) \ = \ \max\left\{\log Q_i(t)\ ,\ e^{\sqrt{\log \max_{j\in\mathcal{N}(i)} W_j(t-1)}}\right\},$$

where $Q_{\max}(t) = \max_i Q_i(t)$*.*

- Myopic & robust against the arrival assumption.
- To compute $W_i(t)$, it requires to know global information $Q_{\max}(t)$.
  - Still require some explicit message passing (minimal though)

Main Result of This Talk

- We revise the algorithm so that it does not require such message passing.
  - Motivation : Possible to learn $W_j(t)$ using carrier sensing without message passing?

Main Issue: Online Learning Problem

How to learn weights of neighbors without message passing?

- Equivalent question: how to learn access probabilities of (interfering) neighbors?
  - since recall access probability $p_i(t) = 1 - \frac{1}{W_i(t)}$.

Main Issue: Online Learning Problem

How to learn weights of neighbors without message passing?

- Equivalent question: how to learn access probabilities of (interfering) neighbors?
  - since recall access probability $p_i(t) = 1 - \frac{1}{W_i(t)}$.

- May be possible using carrier sensing information since

Access probability $\approx$ How often attempt $\approx$ Carrier Sensing

Main Issue: Online Learning Problem

How to learn weights of neighbors without message passing?

- Equivalent question: how to learn access probabilities of (interfering) neighbors?
  - since recall access probability $p_i(t) = 1 - \frac{1}{W_i(t)}$.

- May be possible using carrier sensing information since

Access probability $\approx$ How often attempt $\approx$ Carrier Sensing

- Non-trivial online learning problem since
  - $W_j(t)$ is changing

Main Issue: Online Learning Problem

How to learn weights of neighbors without message passing?

- Equivalent question: how to learn access probabilities of (interfering) neighbors?
  - since recall access probability $p_i(t) = 1 - \frac{1}{W_i(t)}$.

- May be possible using carrier sensing information since

Access probability $\approx$ How often attempt $\approx$ Carrier Sensing

- Non-trivial online learning problem since
  - $W_j(t)$ is changing
  - # samples is affected by many random environments

Main Issue: Online Learning Problem

How to learn weights of neighbors without message passing?

- Equivalent question: how to learn access probabilities of (interfering) neighbors?
  - since recall access probability $p_i(t) = 1 - \frac{1}{W_i(t)}$.

- May be possible using carrier sensing information since

Access probability $\approx$ How often attempt $\approx$ Carrier Sensing

- Non-trivial online learning problem since
  - $W_j(t)$ is changing
  - # samples is affected by many random environments
  - how much error is allowed for positive recurrence?

New Algorithm with Learning

Recall the original positive recurrent algorithm

- When the medium is free, each queue i attempts to transmit with probability

$$1 - \frac{1}{W_i(t)} = 1 - \frac{1}{\max\left\{\log Q_i(t)\,,\ e^{\sqrt{\log \max_{j \in \mathcal{N}(i)} W_j(t-1)}}\right\}}.$$

New Algorithm with Learning

Revise the original positive recurrent algorithm using estimator L_j^i

- When the medium is free, each queue i attempts to transmit with probability

$$1 - \frac{1}{W_i(t)} = 1 - \frac{1}{\max\left\{\log Q_i(t)\,,\ e^{\sqrt{\log \max_{j\in\mathcal{N}(i)} L_j^i(t)}}\right\}}.$$

New Algorithm with Learning

Revise the original positive recurrent algorithm using estimator L_j^i

- When the medium is free, each queue i attempts to transmit with probability

$$1 - \frac{1}{W_i(t)} = 1 - \frac{1}{\max\left\{\log Q_i(t)\,,\ e^{\sqrt{\log \max_{j \in \mathcal{N}(i)} L_j^i(t)}}\right\}}.$$

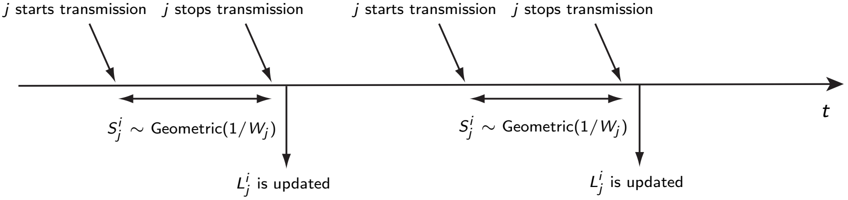
In addition, each queue i maintains $L_j^i(t)$, $S_j^i(t)$ for $j \in \mathcal{N}(i)$: for some function g

- If $\sigma_j(t-1) = 1$, $S_j^i(t) = S_j^i(t-1) + 1$.
- Else if $S_j^i(t-1) > 0$, $S_j^i(t) = 0$ and

$$L_j^i(t) = \begin{cases} L_j^i(t-1) + \Delta & \text{if } S_j^i(t-1) \geq L_j^i(t-1) \\ L_j^i(t-1) - \Delta & \text{otherwise} \end{cases}, \qquad \text{where } \Delta = \frac{1}{g(L_j^i(t-1))}.$$

$\star$ $L_j^i(t)$ and $S_j^i(t)$ are long-term and temporal estimators of $W_j(t)$ at queue i, respectively.

New Algorithm with Learning



In addition, each queue i maintains $L_j^i(t)$, $S_j^i(t)$ for $j \in \mathcal{N}(i)$: for some function g

- If $\sigma_j(t-1) = 1$, $S_j^i(t) = S_j^i(t-1) + 1$.
- Else if $S_j^i(t-1) > 0$, $S_j^i(t) = 0$ and

$$L_j^i(t) = \begin{cases} L_j^i(t-1) + \Delta & \text{if } S_j^i(t-1) \geq L_j^i(t-1) \\ L_j^i(t-1) - \Delta & \text{otherwise} \end{cases}, \qquad \text{where } \Delta = \frac{1}{g(L_j^i(t-1))}.$$

$\star$ $L_j^i(t)$ and $S_j^i(t)$ are long-term and temporal estimators of $W_j(t)$ at queue i, respectively.

New Algorithm: Positive Recurrence

Theorem (Shah, **Shin** and Tetali)

The network Markov chain induced by the revised algorithm is positive recurrent if

$$\boldsymbol{\lambda} \in conv^{o}(\mathcal{I}(G)) \qquad and \qquad g(x) = e^{e^{\log^{1/4} x}}.$$

New Algorithm: Positive Recurrence

Theorem (Shah, **Shin** and Tetali)

The network Markov chain induced by the revised algorithm is positive recurrent if

$$\boldsymbol{\lambda} \in conv^{o}(\mathcal{I}(G)) \qquad and \qquad g(x) = e^{e^{\log^{1/4} x}}.$$

- In the proof, we use the following Lyapunov function

$$F(t) \;=\; \sum_{i} h(Q_i(t)) + \sum_{i,j} g(L_j^i(t))^2 + \sum_{i,j} g(S_j^i(t)),$$

where $h = \int \log\log$.

New Algorithm: Positive Recurrence

Theorem (Shah, **Shin** and Tetali)

The network Markov chain induced by the revised algorithm is positive recurrent if

$$\boldsymbol{\lambda} \in conv^{o}(\mathcal{I}(G)) \qquad and \qquad g(x) = e^{e^{\log^{1/4} x}}.$$

- In the proof, we use the following Lyapunov function

$$F(t) \;=\; \sum_i h(Q_i(t)) + \sum_{i,j} g(L_j^i(t))^2 + \sum_{i,j} g(S_j^i(t)),$$

where $h = \int \log\log$.

- Main additional techniques
  - Careful martingale arguments to control L_j^i.

New Algorithm: Positive Recurrence

Theorem (Shah, **Shin** and Tetali)

The network Markov chain induced by the revised algorithm is positive recurrent if

$$\boldsymbol{\lambda} \in conv^{o}(\mathcal{I}(G)) \qquad and \qquad g(x) = e^{e^{\log^{1/4} x}}.$$

- In the proof, we use the following Lyapunov function

$$F(t) \;=\; \sum_i h(Q_i(t)) + \sum_{i,j} g(L_j^i(t))^2 + \sum_{i,j} g(S_j^i(t)),$$

  where $h = \int \log\log$.

- Main additional techniques
  - Careful martingale arguments to control L_j^i.
  - 'Hitting time' of Markov chains.

Summary & Wide-applicability

In summary

- We present a myopic, positive recurrent MAC algorithm, where individual queues
  - Use only local & primitive (carrier sensing) information
  - Performs few logical operations per each time.

Summary & Wide-applicability

In summary

- We present a myopic, positive recurrent MAC algorithm, where individual queues
    - Use only local & primitive (carrier sensing) information
    - Performs few logical operations per each time.
    - It essentially simulates the max weight algorithm [Tassiulas and Ephremides 92].

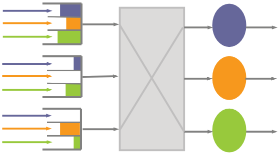
Summary & Wide-applicability

In summary

- We present a myopic, positive recurrent MAC algorithm, where individual queues
  - Use only local & primitive (carrier sensing) information
  - Performs few logical operations per each time.
  - It essentially simulates the max weight algorithm [Tassiulas and Ephremides 92].

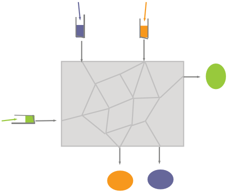
Our framework to design such algorithms has wide-applicability

- In the context of stochastic processing networks [Harrison 00].



High speed switch scheduling

(Matching constraints)



Scheduling in optical core networks

(Multi-commodity-type constraints)