Throughput-Optimal Algorithms for Generalized Network-Flow Problems

Abhishek Sinha
Laboratory for Information and Decision Systems

Talk at: TIFR

December 29, 2016

Outline

1. Introduction
2. Optimal Broadcasting in a DAG
3. Algorithm for Generalized Flow
4. Simulation Results
5. Conclusion

Outline

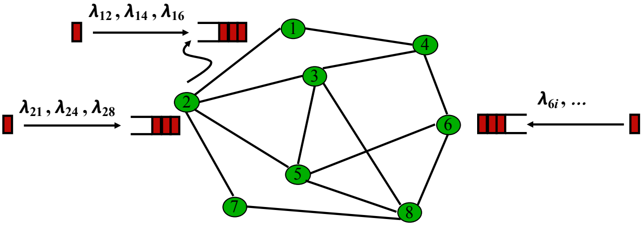
1. Introduction
2. Optimal Broadcasting in a DAG
3. Algorithm for Generalized Flow
4. Simulation Results
5. Conclusion

Introduction

- We consider the Generalized Flow Problem, where the packets are to be routed in a wireless network from their source to the destination(s) at the maximum rate possible.
- Packets may belong to different sessions including unicast, broadcast, multicast or anycast.
- A *Fundamental* problem with wide ranging applications: Internet routing, in-network function computations, live multi-media streaming, military communications
- Contributions of my thesis:
  - Broadcast: A specialized dynamic algorithm that solves the optimal broadcasting problem
  - Generalized Flow: A single general algorithmic paradigm that solves *all flow problems*.

Network Model

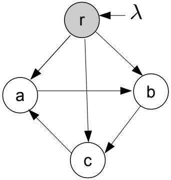
- Multi-hop wireless network given by the graph $\mathcal{G}(V, E)$.
- Time-slotted system
- Stochastic arrivals- i.i.d. process with arrival vector $\boldsymbol{\lambda}$
  - $\boldsymbol{\lambda}$ is, in general, unknown in advance



The network $\mathcal{G}(V, E)$

Wireless Interference Model

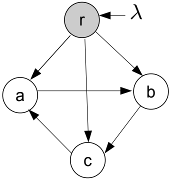
- Due to Wireless interference, only a subset of links can be activated at a slot. The set of all feasible link activations is given by $\mathcal{M}$.
  - Example: (1) For primary interference constraints: $\mathcal{M}$ is the set of all Matchings
  - (2) For wired network : $\mathcal{M}$ is the set of all subsets of links (no interference)



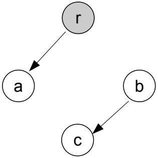
A wireless network

Wireless Interference Model

- Due to Wireless interference, only a subset of links can be activated at a slot. The set of all feasible link activations is given by $\mathcal{M}$.
    - Example: (1) For primary interference constraints: $\mathcal{M}$ is the set of all Matchings
    - (2) For wired network : $\mathcal{M}$ is the set of all subsets of links (no interference)



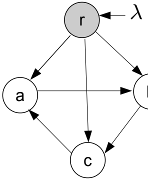
A wireless network



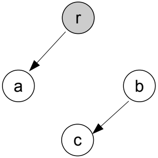
Activation $\boldsymbol{s}_1$

Wireless Interference Model

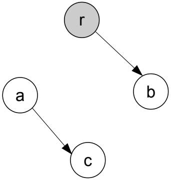
- Due to Wireless interference, only a subset of links can be activated at a slot. The set of all feasible link activations is given by $\mathcal{M}$.
  - Example: (1) For primary interference constraints: $\mathcal{M}$ is the set of all Matchings
  - (2) For wired network : $\mathcal{M}$ is the set of all subsets of links (no interference)



A wireless network



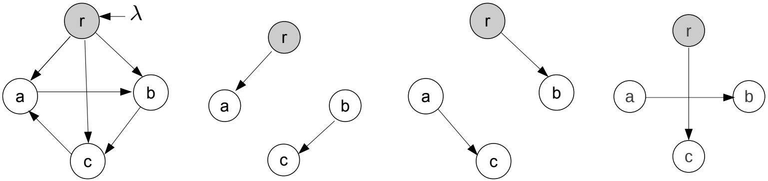
Activation $\boldsymbol{s}_1$



Activation $\boldsymbol{s}_2$

Wireless Interference Model

- Due to Wireless interference, only a subset of links can be activated at a slot. The set of all feasible link activations is given by $\mathcal{M}$.
  - Example: (1) For primary interference constraints: $\mathcal{M}$ is the set of all Matchings
  - (2) For wired network : $\mathcal{M}$ is the set of all subsets of links (no interference)



A wireless network — Activation $\boldsymbol{s}_1$ — Activation $\boldsymbol{s}_2$ — Activation $\boldsymbol{s}_3$

A wireless network and its feasible link activations under primary interference constraints

- The links may also be time-varying $\{ON, OFF\}$ according to a stationary ergodic process.

Classification of Network Flow Problems

Traffic Classes: Traffic Class c has arrival rate λ_c, source node S^c, destination node(s) D^c, where

- Unicast: Single source, single destination

$$\mathcal{S} = \{s\}, \mathcal{D} = \{d\}$$

- Multicast: Single source, multiple destinations

$$\mathcal{S} = \{s\}, \mathcal{D} \subset V \setminus \{s\}$$

- Broadcast: Single source, all destinations

$$\mathcal{S} = \{s\}, \mathcal{D} = V \setminus \{s\}$$

- Anycast: Single source, choice of one among multiple alternative destinations

$$\mathcal{S} = \{s\}, \mathcal{D} = v_1 \oplus v_2 \oplus \ldots \oplus v_k$$

- All of the above problems with multiple sources

Problem Statement

Design a routing and scheduling policy that supports all arrival rates $\boldsymbol{\lambda}$ within the network stability region.

- Including an arbitrary mix of unicast, multicast, broadcast traffic

Formally, let $R_\pi^{(c)}(T)$ denote the number of packets received by all destinations of class c upto time T under the policy π. Our objective is to find a policy π such that the following holds:

$$\liminf_{T \nearrow \infty} \frac{R_\pi^{(c)}(T)}{T} = \lambda^c, \ \forall c \in \mathcal{C} \quad \text{w.p. } 1$$

For all arrival rates $\boldsymbol{\lambda}$ in the interior of the stability region $\boldsymbol{\Lambda}$ of the network.

Problem Statement

Design a routing and scheduling policy that supports all arrival rates $\boldsymbol{\lambda}$ within the network stability region.

- Including an arbitrary mix of unicast, multicast, broadcast traffic

Formally, let $R_\pi^{(c)}(T)$ denote the number of packets received by all destinations of class c upto time T under the policy π. Our objective is to find a policy π such that the following holds:

$$\liminf_{T \nearrow \infty} \frac{R_\pi^{(c)}(T)}{T} = \lambda^c, \ \forall c \in \mathcal{C} \quad \text{w.p. } 1$$

For all arrival rates $\boldsymbol{\lambda}$ in the interior of the stability region $\boldsymbol{\Lambda}$ of the network.

We will design two distinct optimal policies.

- First, we consider the Broadcast problem on a Wireless Directed Acyclic Graph (DAG)
- Finally, we will consider the general problem stated above

Outline

1. Introduction
2. Optimal Broadcasting in a DAG
3. Algorithm for Generalized Flow
4. Simulation Results
5. Conclusion

The Broadcast Capacity λ^*

Observation : From source $\mathtt{r}$ to each node $\mathtt{t} \neq \mathtt{r}$,

$$\lambda^* \leq \textsc{Max-Flow}(\mathtt{r} \to \mathtt{t})$$

The Broadcast Capacity λ^*

Observation : From source $\mathtt{r}$ to each node $\mathtt{t} \neq \mathtt{r}$,

$$\lambda^* \leq \textsc{Max-Flow}(\mathtt{r} \to \mathtt{t})$$

Thus,

$$\lambda^* \leq \min_{\mathtt{t} \in V \setminus \{\mathtt{r}\}} \textsc{Max-Flow}(\mathtt{r} \to \mathtt{t})$$

The Broadcast Capacity λ^*

Observation : From source $\mathtt{r}$ to each node $\mathtt{t} \neq \mathtt{r}$,

$$\lambda^* \leq \textsc{Max-Flow}(\mathtt{r} \to \mathtt{t})$$

Thus,

$$\lambda^* \leq \min_{\mathtt{t} \in V \setminus \{\mathtt{r}\}} \textsc{Max-Flow}(\mathtt{r} \to \mathtt{t})$$

Edmond's Tree-Packing Theorem [1965]

- The above inequality is satisfied with equality.
- There exist λ^* edge-disjoint spanning trees to achieve the broadcast capacity.

The Broadcast Capacity λ^*

Observation : From source $\mathtt{r}$ to each node $\mathtt{t} \neq \mathtt{r}$,

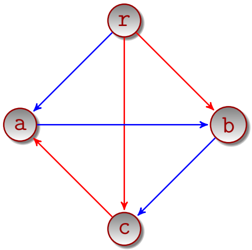
$$\lambda^* \leq \text{MAX-FLOW}(\mathtt{r} \to \mathtt{t})$$

Thus,

$$\lambda^* \leq \min_{\mathtt{t} \in V \setminus \{\mathtt{r}\}} \text{MAX-FLOW}(\mathtt{r} \to \mathtt{t})$$

Edmond's Tree-Packing Theorem [1965]

- The above inequality is satisfied with equality.
- There exist λ^* edge-disjoint spanning trees to achieve the broadcast capacity.



The Broadcast Capacity λ^*

Observation : From source $\mathtt{r}$ to each node $\mathtt{t} \neq \mathtt{r}$,

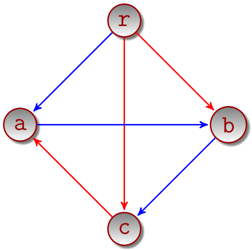
$$\lambda^* \leq \textsc{Max-Flow}(\mathtt{r} \to \mathtt{t})$$

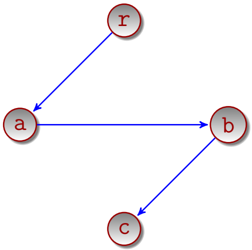
Thus,

$$\lambda^* \leq \min_{\mathtt{t} \in V \setminus \{\mathtt{r}\}} \textsc{Max-Flow}(\mathtt{r} \to \mathtt{t})$$

Edmond's Tree-Packing Theorem [1965]

- The above inequality is satisfied with equality.
- There exist λ^* edge-disjoint spanning trees to achieve the broadcast capacity.





The Broadcast Capacity λ^*

Observation : From source $\mathtt{r}$ to each node $\mathtt{t} \neq \mathtt{r}$,

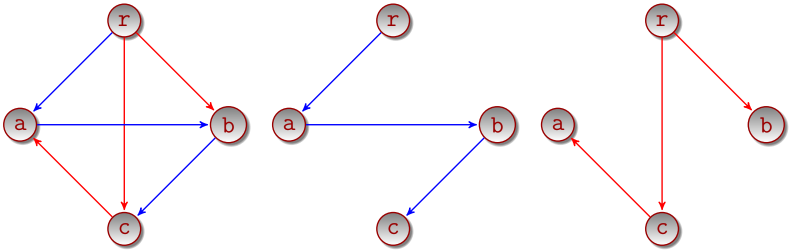
$$\lambda^* \leq \textsc{Max-Flow}(\mathtt{r} \to \mathtt{t})$$

Thus,

$$\lambda^* \leq \min_{\mathtt{t} \in V \setminus \{\mathtt{r}\}} \textsc{Max-Flow}(\mathtt{r} \to \mathtt{t})$$

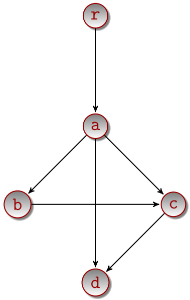
Edmond's Tree-Packing Theorem [1965]

- The above inequality is satisfied with equality.
- There exist λ^* edge-disjoint spanning trees to achieve the broadcast capacity.



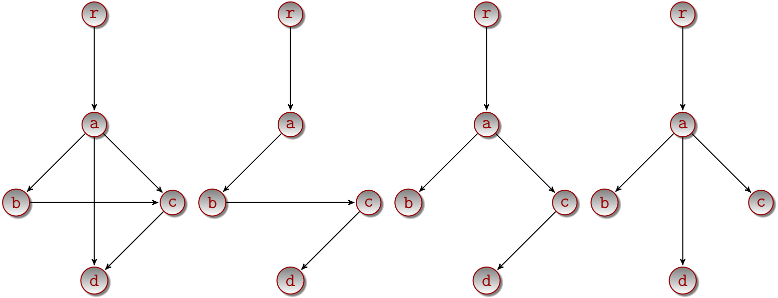
Prior Works

- Pre-computes the set of all spanning trees offline in wireline networks [Sarkar and Tassiulas, Trans. on IT 2002].



Prior Works

- Pre-computes the set of all spanning trees offline in wireline networks [Sarkar and Tassiulas, Trans. on IT 2002].



  - Impractical for large and time-varying networks. ✗
- Wireless case is studied with a fixed activation schedule only [Towsley and Twigg, 2008].

Feasible Policy Space Π

A feasible broadcast policy $\pi \in \Pi$ executes following two actions at every slot t :

- Link Activation $\pi(\mathcal{A})$: Activate a subset of links (e.g., a matching) subject to the underlying interference constraints.
- Packet Scheduling $\pi(\mathcal{S})$: Transmit packets over the set of activated links.

Feasible Policy Space Π

A feasible broadcast policy $\pi \in \Pi$ executes following two actions at every slot t :

- Link Activation $\pi(\mathcal{A})$: Activate a subset of links (e.g., a matching) subject to the underlying interference constraints.
- **Packet Scheduling $\pi(\mathcal{S})$:** Transmit packets over the set of activated links.

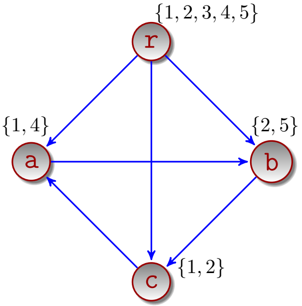
1. # of packets present at each subset of nodes

Feasible Policy Space Π

A feasible broadcast policy $\pi \in \Pi$ executes following two actions at every slot t :

- Link Activation $\pi(\mathcal{A})$: Activate a subset of links (e.g., a matching) subject to the underlying interference constraints.
- **Packet Scheduling $\pi(\mathcal{S})$:** Transmit packets over the set of activated links.

- An arbitrary $\pi(\mathcal{S})$ is hard to describe : state-space[1] grows exponentially!



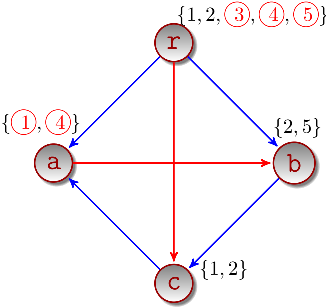
1. # of packets present at each subset of nodes

Feasible Policy Space Π

A feasible broadcast policy $\pi \in \Pi$ executes following two actions at every slot t :

- Link Activation $\pi(\mathcal{A})$: Activate a subset of links (e.g., a matching) subject to the underlying interference constraints.
- **Packet Scheduling $\pi(\mathcal{S})$:** Transmit packets over the set of activated links.

- An arbitrary $\pi(\mathcal{S})$ is hard to describe : state-space[1] grows exponentially!



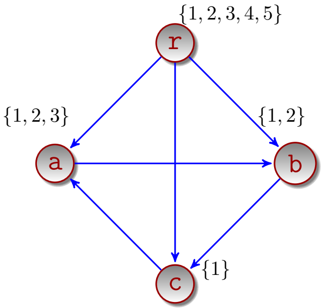
1. # of packets present at each subset of nodes

Policy-space $\Pi^{\text{in-order}} \subset \Pi$

- To simplify $\pi(\mathcal{S})$, we consider the sub-space $\Pi^{\text{in-order}} \subset \Pi$, in which all packets are delivered to every node *in-order*.

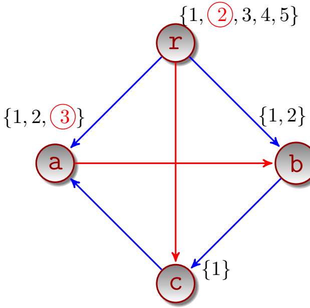
Policy-space $\Pi^{\text{in-order}} \subset \Pi$

- To simplify $\pi(\mathcal{S})$, we consider the sub-space $\Pi^{\text{in-order}} \subset \Pi$, in which all packets are delivered to every node *in-order*.



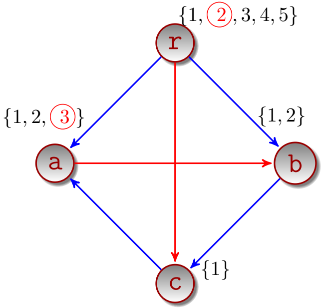
Policy-space $\Pi^{\text{in-order}} \subset \Pi$

- To simplify $\pi(\mathcal{S})$, we consider the sub-space $\Pi^{\text{in-order}} \subset \Pi$ in which all packets are delivered to every node *in-order*.



Policy-space $\Pi^{\text{in-order}} \subset \Pi$

- To simplify $\pi(\mathcal{S})$, we consider the sub-space $\Pi^{\text{in-order}} \subset \Pi$ in which all packets are delivered to every node *in-order*.



Thus, the network state can be compactly represented by the vector

$$\boldsymbol{S}(t) = \{R_1(t), R_2(t), \ldots, R_n(t)\}$$

Where $R_i(t)$ is the total number of packets received by node i up to time t.

Policy sub-space Π^*

Question: Does the reduced policy-space $\Pi^{\mathrm{in-order}}$ contain an optimal policy?

Policy sub-space Π^*

Question: Does the reduced policy-space $\Pi^{\mathrm{in-order}}$ contain an optimal policy?

We show that $\Pi^{\text{in-order}}$ does contain a optimal policy for DAGs.
To prove this, we consider a restricted subspace $\Pi^* \subset \Pi^{\text{in-order}}$ defined as follows :

Policy sub-space Π^*

Question: Does the reduced policy-space $\Pi^{\mathrm{in-order}}$ contain an optimal policy?

We show that $\Pi^{\text{in-order}}$ does contain a optimal policy for DAGs.
To prove this, we consider a restricted subspace $\Pi^* \subset \Pi^{\text{in-order}}$ defined as follows :

$\Pi^* \subset \Pi^{\text{in-order}} \subset \Pi$

For all $\pi \in \Pi^*$, a packet p is eligible for transmission to node n iff all in-neighbors of node n contain the packet p.

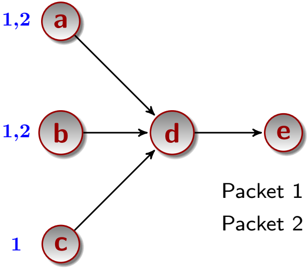
Policy sub-space Π^*

Question: Does the reduced policy-space $\Pi^{\text{in-order}}$ contain an optimal policy?

We show that $\Pi^{\text{in-order}}$ does contain a optimal policy for DAGs.
To prove this, we consider a restricted subspace $\Pi^* \subset \Pi^{\text{in-order}}$ defined as follows :

$\Pi^* \subset \Pi^{\text{in-order}} \subset \Pi$

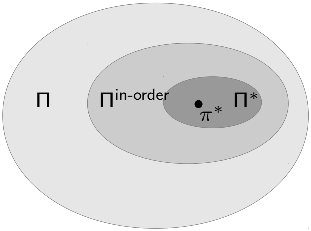
For all $\pi \in \Pi^*$, a packet p is eligible for transmission to node n iff all in-neighbors of node n contain the packet p.



Packet 1 $\rightarrow$ node d ✓
Packet 2 $\rightarrow$ node d ✗

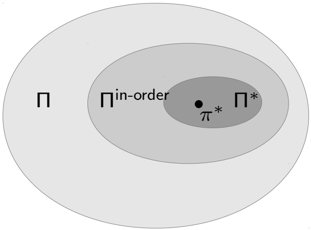
In Summary: Policy hierarchies

All policies are un-restricted for link-activations ($\pi(\mathcal{A})$)



In Summary: Policy hierarchies

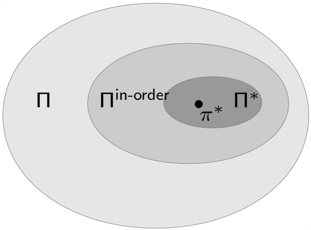
All policies are un-restricted for link-activations ($\pi(\mathcal{A})$)



Π: all policies that perform arbitrary packet-forwarding

In Summary: Policy hierarchies

All policies are un-restricted for link-activations ($\pi(\mathcal{A})$)

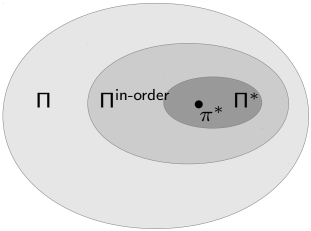


Π: all policies that perform arbitrary packet-forwarding

$\Pi^{\text{in-order}}$: policies that enforce in-order packet-forwarding

In Summary: Policy hierarchies

All policies are un-restricted for link-activations ($\pi(\mathcal{A})$)



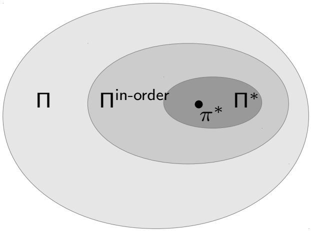
Π: all policies that perform arbitrary packet-forwarding

$\Pi^{\text{in-order}}$: policies that enforce in-order packet-forwarding

Π^*: policies that allow reception only if all in-neighbors have received the specific packet

In Summary: Policy hierarchies

All policies are un-restricted for link-activations ($\pi(\mathcal{A})$)



Π: all policies that perform arbitrary packet-forwarding

$\Pi^{\text{in-order}}$: policies that enforce in-order packet-forwarding

Π^*: policies that allow reception only if all in-neighbors have received the specific packet

Punchline : There exists $\pi^* \in \Pi^*$, which is optimal for DAG!

System-dynamics under Π^*

State Variables:

For each node $j \in V \setminus \{r\}$ define

$$
\begin{aligned}
X_j(t) &= \min_{i:(i,j)\in E} \Big(R_i(t) - R_j(t)\Big) \quad \textit{(Relative deficiency)} \\
i_t^*(j) &= \arg \min_{i:(i,j)\in E} \Big(R_i(t) - R_j(t)\Big)
\end{aligned}
$$

System-dynamics under Π^*

State Variables:

For each node $j \in V \setminus \{r\}$ define

$$
\begin{aligned}
X_j(t) &= \min_{i:(i,j)\in E} \Big(R_i(t) - R_j(t) \Big) \quad \textit{(Relative deficiency)} \\
i_t^*(j) &= \arg \min_{i:(i,j)\in E} \Big(R_i(t) - R_j(t) \Big)
\end{aligned}
$$

Theorem 1: Queuing Dynamics

For any policy $\pi \in \Pi^*$ the dynamics of the state variables $\{\boldsymbol{X}(t)\}_{t\geq 0}$ satisfies a Lindley recursion.

System-dynamics under Π^*

State Variables:

For each node $j \in V \setminus \{r\}$ define

$$X_j(t) = \min_{i:(i,j)\in E} \Big(R_i(t) - R_j(t)\Big) \quad \textit{(Relative deficiency)}$$

$$i_t^*(j) = \arg \min_{i:(i,j)\in E} \Big(R_i(t) - R_j(t)\Big)$$

Theorem 1: Queuing Dynamics

For any policy $\pi \in \Pi^*$ the dynamics of the state variables $\{\boldsymbol{X}(t)\}_{t\geq 0}$ satisfies a Lindley recursion.

Theorem 2: Stability $\implies$ Efficiency

Under Π^*, any algorithm stabilizing $\boldsymbol{X}(t)$ is a broadcast algorithm in a DAG.

Intuition : The state-vector $\boldsymbol{X}(t)$ mathematically corresponds to "queue-sizes" in the traditional queuing network.

Max-Weight Policy for Stabilizing $\boldsymbol{X}(t)$

Consider a policy $\pi^* \in \Pi^*$, for which $\pi^*(\mathcal{A})$ is obtained by minimizing the drift of the Lyapunov function $L(\boldsymbol{X}(t)) = \sum_{j \in V \setminus \{r\}} X_j^2(t)$

- To each edge $(i,j) \in E$, assign a weight $W_{ij}(t)$, where

$$W_{ij}(t) = \Big(X_j(t) - \sum_{k: j = i_t^*(k)} X_k(t)\Big) \mathbb{1}(\boldsymbol{\sigma}_{ij}(t) = 1)$$

Max-Weight Policy for Stabilizing $\boldsymbol{X}(t)$

Consider a policy $\pi^* \in \Pi^*$, for which $\pi^*(\mathcal{A})$ is obtained by minimizing the drift of the Lyapunov function $L(\boldsymbol{X}(t)) = \sum_{j \in V \setminus \{r\}} X_j^2(t)$

- To each edge $(i,j) \in E$, assign a weight $W_{ij}(t)$, where

$$W_{ij}(t) = \Big(X_j(t) - \sum_{k: j = i_t^*(k)} X_k(t)\Big) \mathbb{1}(\boldsymbol{\sigma}_{ij}(t) = 1)$$

- Choose a *max-weight* activation with weights $\boldsymbol{W}(t)$

$$\boldsymbol{\mu}^{\pi^*}(t) \in \arg\max_{\boldsymbol{\mu} \in \boldsymbol{C} \odot \mathcal{S}} \boldsymbol{\mu} \cdot \boldsymbol{W}(t)$$

Max-Weight Policy for Stabilizing $\boldsymbol{X}(t)$

Consider a policy $\pi^* \in \Pi^*$, for which $\pi^*(\mathcal{A})$ is obtained by minimizing the drift of the Lyapunov function $L(\boldsymbol{X}(t)) = \sum_{j \in V \setminus \{r\}} X_j^2(t)$

- To each edge $(i,j) \in E$, assign a weight $W_{ij}(t)$, where

$$W_{ij}(t) = \Big(X_j(t) - \sum_{k: j = i_t^*(k)} X_k(t)\Big) \mathbb{1}(\boldsymbol{\sigma}_{ij}(t) = 1)$$

- Choose a *max-weight* activation with weights $\boldsymbol{W}(t)$

$$\boldsymbol{\mu}^{\pi^*}(t) \in \arg \max_{\boldsymbol{\mu} \in \boldsymbol{C} \odot \mathcal{S}} \boldsymbol{\mu} \cdot \boldsymbol{W}(t)$$

Theorem 3: Throughput-Optimality of π^*

The policy π^* is an optimal broadcast policy for a DAG, i.e. for $\lambda < \lambda^*$

$$\liminf_{t \to \infty} \frac{R_i^{\pi^*}(t)}{t} = \lambda, \quad \forall i \in V \quad \text{w.p. } 1$$

Characterization of Broadcast-Capacity of Wireless DAGs

The policy yields constructive proof of the following:

Theorem 4: Capacity characterization

$$\lambda^*_{\mathrm{DAG}} = \max_{\boldsymbol{\beta}_\sigma \in \mathrm{conv}(\mathcal{M}_\sigma)} \min_j \boldsymbol{u}_j \cdot \left(\sum_\sigma p(\boldsymbol{\sigma}) \boldsymbol{\beta}_\sigma \right)$$

where $\boldsymbol{u}_j$ is the cut-vector separating the node j from the rest of the network.

Characterization of Broadcast-Capacity of Wireless DAGs

The policy yields constructive proof of the following:

Theorem 4: Capacity characterization

$$\lambda^*_{\mathrm{DAG}} = \max_{\boldsymbol{\beta}_\sigma \in \mathrm{conv}(\mathcal{M}_\sigma)} \min_j \boldsymbol{u}_j \cdot \Big(\sum_\sigma p(\boldsymbol{\sigma}) \boldsymbol{\beta}_\sigma \Big)$$

where $\boldsymbol{u}_j$ is the cut-vector separating the node j from the rest of the network.

This yields:

Theorem 5: Efficient Computation

λ^*_{DAG} may be computed in poly-time for primary interference constraint and polynomially many network-configurations.

The proof uses an LP formulation of the above and the well-known poly-time separating oracle for the matching polytope.

Summary of the results so far . . .

- We focussed exclusively on the Broadcast Problem and designed a Throughput-optimal policy for Wireless DAGs
- The algorithm used the idea of *in-order* delivery of packets in a cleverly chosen restricted policy space.
- The Wireless Broadcast capacity of DAGs is explicitly characterized.
- We have also designed different throughput-optimal broadcast policy for arbitrary networks, using the new concept of Backpressure on sets. (not discussed in this talk)
- In the rest of the talk, we will consider the generalized network flow problem.

Outline

1. Introduction
2. Optimal Broadcasting in a DAG
3. Algorithm for Generalized Flow
4. Simulation Results
5. Conclusion

What's new? (Preview)

The new policy is named Universal Max-Weight Algorithm (UMW). It is different from the classical algorithms (such as, *Backpressure*) in the following aspects:

What's new? (Preview)

The new policy is named Universal Max-Weight Algorithm (UMW). It is different from the classical algorithms (such as, *Backpressure*) in the following aspects:

- Instead of making routing decisions for each packet hop-by-hop, UMW dynamically chooses routes of each packet right at the source.

What's new? (Preview)

The new policy is named Universal Max-Weight Algorithm (UMW). It is different from the classical algorithms (such as, *Backpressure*) in the following aspects:

- Instead of making routing decisions for each packet hop-by-hop, UMW dynamically chooses routes of each packet right at the source.
- Instead of taking control actions based on queue lengths $\boldsymbol{Q}(t)$ (closed-loop control), UMW is oblivious to the queues (semi open-loop control).

What's new? (Preview)

The new policy is named Universal Max-Weight Algorithm (UMW). It is different from the classical algorithms (such as, *Backpressure*) in the following aspects:

- Instead of making routing decisions for each packet hop-by-hop, UMW dynamically chooses routes of each packet right at the source.
- Instead of taking control actions based on queue lengths $\boldsymbol{Q}(t)$ (closed-loop control), UMW is oblivious to the queues (semi open-loop control).
- UMW maintains a m-dimensional vector at the source, called virtual queue-lengths $\tilde{\boldsymbol{Q}}(t)$, which are used for making routing and scheduling decisions.

What's new? (Preview)

The new policy is named Universal Max-Weight Algorithm (UMW). It is different from the classical algorithms (such as, *Backpressure*) in the following aspects:

- Instead of making routing decisions for each packet hop-by-hop, UMW dynamically chooses routes of each packet right at the source.
- Instead of taking control actions based on queue lengths $\boldsymbol{Q}(t)$ (closed-loop control), UMW is oblivious to the queues (semi open-loop control).
- UMW maintains a m-dimensional vector at the source, called virtual queue-lengths $\tilde{\boldsymbol{Q}}(t)$, which are used for making routing and scheduling decisions.
- The virtual queues $\tilde{\boldsymbol{Q}}(t)$ correspond to a precedence-relaxed network with dynamics given by a Lindley recursion.

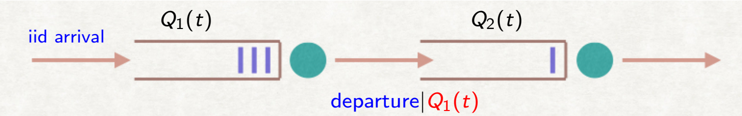
What's new? (Preview)

The new policy is named Universal Max-Weight Algorithm (UMW). It is different from the classical algorithms (such as, *Backpressure*) in the following aspects:

- Instead of making routing decisions for each packet hop-by-hop, UMW dynamically chooses routes of each packet right at the source.
- Instead of taking control actions based on queue lengths $\boldsymbol{Q}(t)$ (closed-loop control), UMW is oblivious to the queues (semi open-loop control).
- UMW maintains a *m*-dimensional vector at the source, called virtual queue-lengths $\tilde{\boldsymbol{Q}}(t)$, which are used for making routing and scheduling decisions.
- The virtual queues $\tilde{\boldsymbol{Q}}(t)$ correspond to a precedence-relaxed network with dynamics given by a Lindley recursion.
- UMW uses the solution of some standard combinatorial problems (e.g., Shortest Path, MST, Steiner Tree, MCDS..) on a graph weighted by the dynamic virtual queues.

Design of UMW: Motivation and Insight

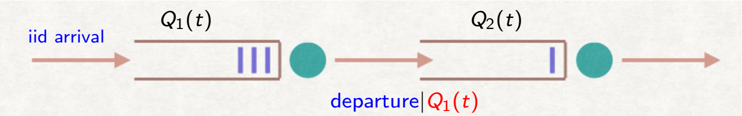
- **Observation:** Because of coupling, networked queues are harder to analyze and control. This is because queuing process, described by Skorokhod maps are fundamentally non-linear in nature.



IID arrivals to Q_1 causes correlated arrivals to Q_2

Design of UMW: Motivation and Insight

- **Observation:** Because of coupling, networked queues are harder to analyze and control. This is because queuing process, described by Skorokhod maps are fundamentally non-linear in nature.

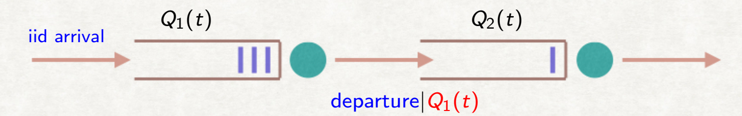


IID arrivals to Q_1 causes correlated arrivals to Q_2

- This motivates us to obtain a relaxed system, which is easier to analyze, yet, preserves some fundamental characteristics we are interested in (e.g., stability) .

Design of UMW: Motivation and Insight

- **Observation:** Because of coupling, networked queues are harder to analyze and control. This is because queuing process, described by Skorokhod maps are fundamentally non-linear in nature.



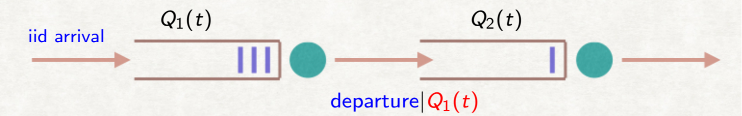
IID arrivals to Q_1 causes correlated arrivals to Q_2

- This motivates us to obtain a relaxed system, which is easier to analyze, yet, preserves some fundamental characteristics we are interested in (e.g., stability) .

Question: How to obtain a good relaxation ? Which constraints to relax ?

Design of UMW: Motivation and Insight

- **Observation:** Because of coupling, networked queues are harder to analyze and control. This is because queuing process, described by Skorokhod maps are fundamentally non-linear in nature.



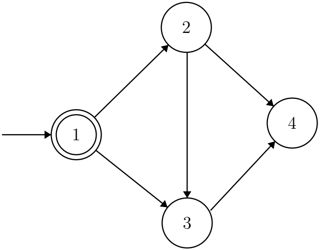
IID arrivals to Q_1 causes correlated arrivals to Q_2

- This motivates us to obtain a relaxed system, which is easier to analyze, yet, preserves some fundamental characteristics we are interested in (e.g., stability) .

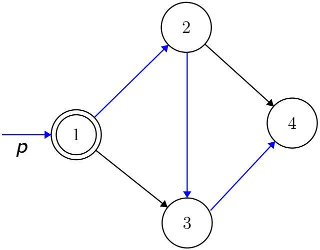
Question: How to obtain a good relaxation ? Which constraints to relax ?

Ans: The Precedence Constraints!

Precedence Relaxation: Example



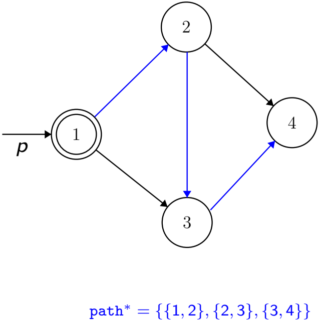
Precedence Relaxation: Example

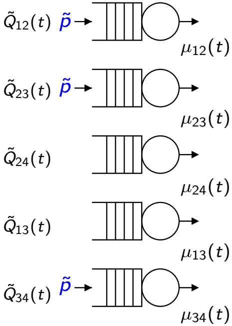


$\mathtt{path}^* = \{\{1,2\},\{2,3\},\{3,4\}\}$

Precedence: The packet p cannot be transmitted over the link $2-3$, until it has been transmitted over the link $1-2$.

Precedence Relaxation: Example





Virtual Queues

Virtual Net: Packets are replicated to the virtual queues as soon as they arrive to the source.

Virtual Queues: Operation

- Associate a virtual queue with each link of the graph.
- Upon packet arrival:
  - Determine an *optimal route* $T^*(t)$ (e.g., path, spanning tree) for the packet
  - Immediately inject a new virtual packet to each virtual queue along the route
  - This amounts to incrementing the queue counters along the route
- Serve the virtual packets at the rate $\boldsymbol{\mu}^*(t)$ as long as the corresponding virtual queues are non-empty
  - Subject to the same link scheduling constraints ($\boldsymbol{\mu}^*(t) \in \mathcal{M}$)

Virtual Queues: Operation

- Associate a virtual queue with each link of the graph.
- Upon packet arrival:
  - Determine an *optimal route* $T^*(t)$ (e.g., path, spanning tree) for the packet
  - Immediately inject a new virtual packet to each virtual queue along the route
  - This amounts to incrementing the queue counters along the route
- Serve the virtual packets at the rate $\boldsymbol{\mu}^*(t)$ as long as the corresponding virtual queues are non-empty
  - Subject to the same link scheduling constraints ($\boldsymbol{\mu}^*(t) \in \mathcal{M}$)

Question: How to design optimal controls: $T^*(t)$ and $\mu^*(t)$?

Dynamics of the Virtual Queues $\tilde{\boldsymbol{Q}}(t)$

The virtual queues can be mathematically identified with an *m*-dimensional vector taking values in $\mathbb{Z}_+^{|E|}$.

Dynamics of the Virtual Queues $\tilde{\boldsymbol{Q}}(t)$

The virtual queues can be mathematically identified with an m-dimensional vector taking values in $\mathbb{Z}_+^{|E|}$.

- Denote the (controlled) arrival to the VQ $\tilde{Q}_e$ by $\tilde{A}_e(t)$. Then, the virtual queues follow the following Lindley dynamics:

$$\tilde{Q}_e(t+1) = \big(\tilde{Q}_e(t) + \tilde{A}_e(t) - \mu_e(t)\big)_+, \quad \text{(Lindley recursion)} \tag{1}$$

- Note that, the arrivals to the virtual queues $\big(A_e(t), e \in E\big)$ are explicit control variables at the source.
- Unlike the original system, given the controls, the virtual queues are independent of each other. This makes exact analysis tractable.

Stabilizing Controls for $\tilde{\boldsymbol{Q}}(t)$: Drift Analysis

- We design the per-slot controls $\pi^{\mathtt{UMW}} \equiv \big(\boldsymbol{A}(t), \boldsymbol{\mu}(t)\big)_{t\geq 0}$, stabilizing the virtual system $\{\tilde{\boldsymbol{Q}}(t)\}_{t\geq 0}$.
- The policy consists of the routing decisions : routing $\boldsymbol{A}^{\pi}(t)$, sand scheduling $\boldsymbol{\mu}^{\pi}(t)$.
- Intuition: This control is *likely to stabilize* the physical queues as well
  - However, note that the dynamics of the physical queues depend explicitly on the packet scheduling policy (e.g., FIFO, LIFO etc.)

Stabilizing Controls for $\tilde{Q}(t)$: Drift Analysis

- We design the per-slot controls $\pi^{\texttt{UMW}} \equiv \big(\boldsymbol{A}(t), \boldsymbol{\mu}(t)\big)_{t\geq 0}$, stabilizing the virtual system $\{\tilde{\boldsymbol{Q}}(t)\}_{t\geq 0}$.
- The policy consists of the routing decisions : routing $\boldsymbol{A}^{\pi}(t)$, sand scheduling $\boldsymbol{\mu}^{\pi}(t)$.
- Intuition: This control is *likely to stabilize* the physical queues as well
  - However, note that the dynamics of the physical queues depend explicitly on the packet scheduling policy (e.g., FIFO, LIFO etc.)
- To stabilize the virtual queues, we choose the control that minimizes the drift of the Quadratic Lyapunov function of the Virtual Queues.
- ► It turns out that the controls $\boldsymbol{A}(t)$ (routing) and $\boldsymbol{\mu}(t)$ (activation) are separable and are standard combinatorial problems that can be solved at the source $\texttt{r}$.

Derivation of the Control-Policy

- Define a Quadratic Lyapunov function

$$L(\tilde{\boldsymbol{Q}}(t)) \stackrel{\text{def}}{=} \sum_{e\in E} \tilde{Q}_e^2(t)$$

- The one-slot drift of $L(\tilde{\boldsymbol{Q}})(t)$ under any admissible policy π may be computed to be

$$\begin{aligned}\Delta^\pi(t) &\stackrel{\text{def}}{=} L(\tilde{\boldsymbol{Q}}(t+1)) - L(\tilde{\boldsymbol{Q}}(t)) \\ &\leq B + 2\Big(\underbrace{\sum_{e\in E} \tilde{Q}_e(t)A(t)\mathbb{1}(e \in T^\pi(t))}_{(a)} - \underbrace{\sum_{e\in E} \tilde{Q}_e(t)\mu_e^\pi(t)}_{(b)}\Big) \qquad (2)\end{aligned}$$

Where $T^\pi(t) \in \mathcal{T}$ and $\boldsymbol{\mu}^\pi(t) \in \mathcal{M}$ are routing and activation control variables chosen for slot t.

- The drift upper-bound (2) has a nice separable form and may be minimized over the set of all feasible controls individually.

Optimal Routing Policy $T^*(t)$

Minimizing the term (a), we get the following optimal routing policy.

Optimal Routing : $T^*(t)$

$$T^*(t) \in \arg\min_{T \in \mathcal{T}} \sum_{e \in E} \tilde{Q}_e(t) \mathbb{1}(e \in T)$$

Optimal Routing Policy $T^*(t)$

Minimizing the term (a), we get the following optimal routing policy.

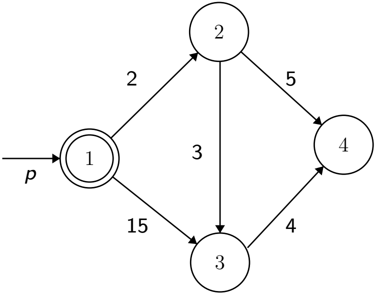
Optimal Routing : $\boldsymbol{T}^*(t)$

$$T^*(t) \in \arg\min_{T \in \mathcal{T}} \sum_{e \in E} \tilde{Q}_e(t) \mathbb{1}(e \in T)$$

Examples:

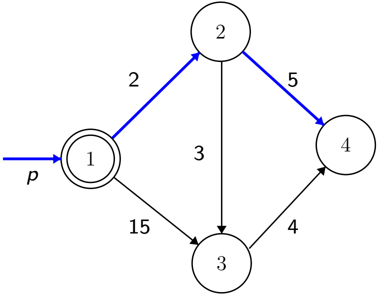
- For the unicast problem : $T^*(t)$ is the Shortest s $\rightarrow$ t path in the weighted graph $\mathcal{G}(V, E, \tilde{\boldsymbol{Q}}(t))$.
- For the broadcast problem : $T^*(t)$ is the Minimum Weight Spanning tree (MST) in the weighted graph $\mathcal{G}(V, E, \tilde{\boldsymbol{Q}}(t))$.
- For the multicast problem : $T^*(t)$ is the Minimum Weight Steiner tree in the weighted graph $\mathcal{G}(V, E, \tilde{\boldsymbol{Q}}(t))$ connecting the source nodes to the destination nodes.

Example of Optimal Routing: Unicast



Network Weighted by Virtual Queue sizes

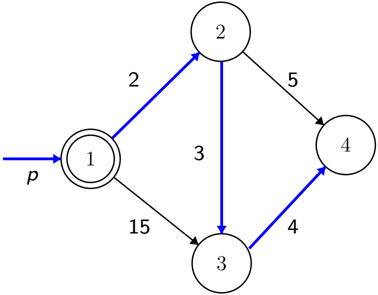
Example of Optimal Routing: Unicast



Shortest 1-4 path $= \{\{1,2\},\{2,4\}\}$

Network Weighted by Virtual Queue sizes

Example of Optimal Routing: Broadcast



Network Weighted by Virtual Queue sizes

MST rooted at 1 $= \{\{1,2\},\{2,3\},\{2,4\}\}$

Optimal Link Activations $\boldsymbol{\mu}^*(t)$

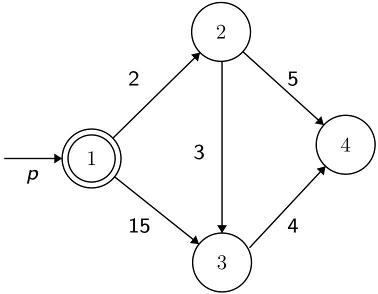
Similarly, minimizing the term (b), we obtain the following optimal activation policy

Optimal Activation: $\boldsymbol{\mu}^*(t)$

$$\boldsymbol{\mu}^*(t) \in \arg\min_{\boldsymbol{\mu}\in\mathcal{M}} \sum_e \tilde{Q}_e(t) c_e \mathbb{1}(e \in \boldsymbol{\mu})$$

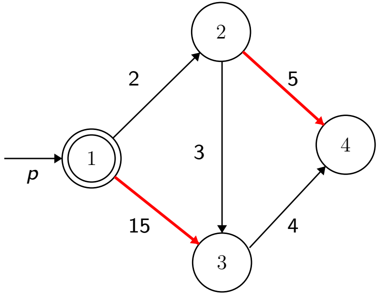
Example: For the case of wireless network with primary interference constraint, this problem corresponds to finding Max-Weight-Matching in the graph $\mathcal{G}(V, E, \boldsymbol{c} \odot \tilde{\boldsymbol{Q}}(t))$.

Example of Optimal Link Activations



Network Weighted by Virtual Queue sizes

Example of Optimal Link Activations



Network Weighted by Virtual Queue sizes

`Max Weight Matching` $= \{\{1,3\},\{2,4\}\}$

Stability of the Virtual Queue

Theorem 6: Strong Stability of $\tilde{\boldsymbol{Q}}(t)$

Under the above routing and scheduling policy, for all arrival rate $\boldsymbol{\lambda} \in \mathring{\boldsymbol{\Lambda}}$ the virtual queue process is Strongly stable and has a limiting M.G.F, i.e.,

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{e} \mathbb{E}(\tilde{Q}_e(t)) \leq B$$

and,

$$\limsup_{T\to\infty} \mathbb{E}(\exp(\theta^* \sum_{e} \tilde{Q}_e(t))) \leq C$$

for some finite B, C and strictly positive θ^*.

Stability of the Virtual Queue

Theorem 6: Strong Stability of $\tilde{\boldsymbol{Q}}(t)$

Under the above routing and scheduling policy, for all arrival rate $\boldsymbol{\lambda} \in \mathring{\boldsymbol{\Lambda}}$ the virtual queue process is Strongly stable and has a limiting M.G.F, i.e.,

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\sum_{e}\mathbb{E}(\tilde{Q}_e(t)) \leq B$$

and,

$$\limsup_{T\to\infty} \mathbb{E}(\exp(\theta^* \sum_{e} \tilde{Q}_e(t))) \leq C$$

for some finite B, C and strictly positive θ^*.

The above leads to the following :

Lemma: Sample Path bound on Virtual Queues

Under the same condition, we have

$$\sum_{e} \tilde{Q}_e(t) = \mathcal{O}(\log t), \quad \text{a.s.}$$

Relating Virtual Queues to Physical Processes : Arrivals and Service

For any edge $e \in E$, define the cumulative arrival and service processes in the interval $(t_1, t_2]$ as follows

$$\tilde{A}_e([t_1, t_2]) \stackrel{(\text{def})}{=} \sum_{\tau=t_1}^{t_2} A_e(\tau), \quad \tilde{S}_e([t_1, t_2]) \stackrel{(\text{def})}{=} \sum_{\tau=t-1}^{t_2} \mu_e(\tau)$$

Relating Virtual Queues to Physical Processes : Arrivals and Service

For any edge $e \in E$, define the cumulative arrival and service processes in the interval $(t_1, t_2]$ as follows

$$\tilde{A}_e([t_1, t_2]) \stackrel{(\text{def})}{=} \sum_{\tau=t_1}^{t_2} A_e(\tau), \quad \tilde{S}_e([t_1, t_2]) \stackrel{(\text{def})}{=} \sum_{\tau=t-1}^{t_2} \mu_e(\tau)$$

Then the Lindley recursion takes the following form (Skorokhod Map):

$$\tilde{Q}_e(t) = \sup_{0 \leq \tau \leq t} (\tilde{A}_e(\tau, t) - \tilde{S}_e(\tau, t))$$

Thus, we have for any $0 \leq \tau \leq t$:

$$\boxed{\tilde{A}_e(\tau, t) \leq \tilde{S}_e(\tau, t) + F(t), \quad F(t) = \mathcal{O}(\log t) \qquad (3)}$$

Optimal Control : Packet Scheduling

- How do we decide which packet to transmit over a link at any given time slot?
  - Why does it matter? Cant we just use FCFS?
- Nearest to Origin (NTO) policy [Gamarnik, 1998]
- Extended Nearest to Origin policy (ENTO): When multiple packets contend for an edge, schedule the one which has traversed the least number of edges
  - Extension of NTO to general flow problems

Optimal Control : Packet Scheduling

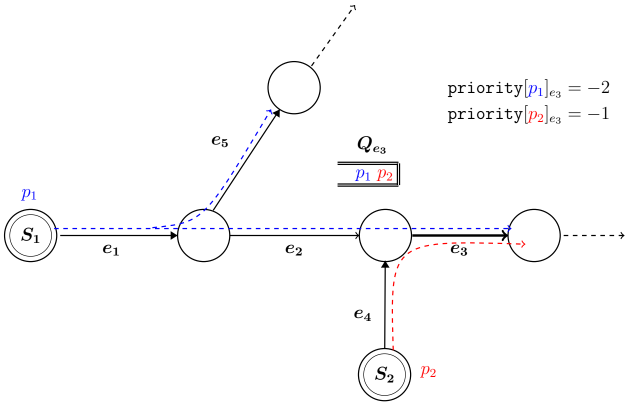
- How do we decide which packet to transmit over a link at any given time slot?
  - Why does it matter? Cant we just use FCFS?
- Nearest to Origin (NTO) policy [Gamarnik, 1998]
- Extended Nearest to Origin policy (ENTO): When multiple packets contend for an edge, schedule the one which has traversed the least number of edges
  - Extension of NTO to general flow problems

Theorem 7: Stability of the Physical Queues

The overall UMW policy is throughput-optimal.

Proof uses the previous almost sure arrival bound on a typical sample path with an inductive argument on the edges.

ENTO: Example

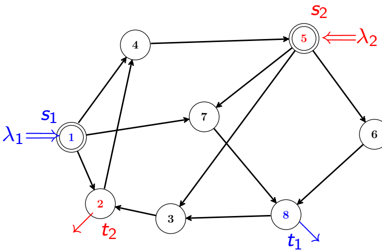


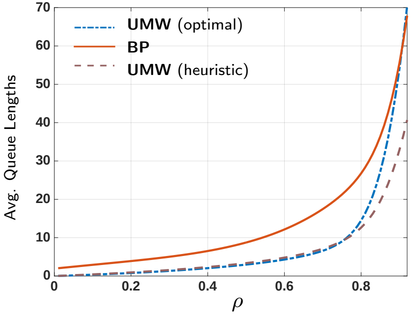
Packet p_1 has higher priority than p_2 to cross e_3 as it has traversed less number of edges

Outline

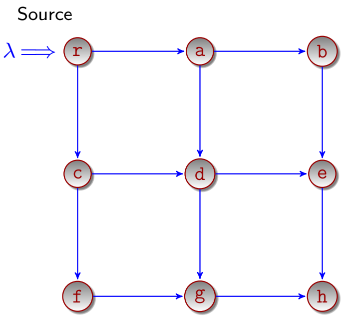
1. Introduction
2. Optimal Broadcasting in a DAG
3. Algorithm for Generalized Flow
4. **Simulation Results**
5. Conclusion

Multi-commodity Flow Simulation

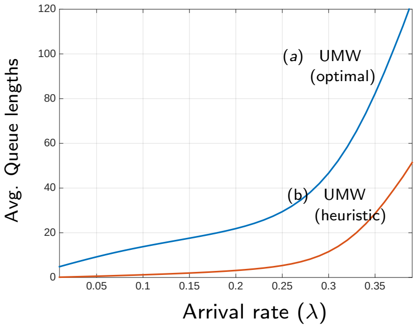




Broadcasting Simulation: Static Network

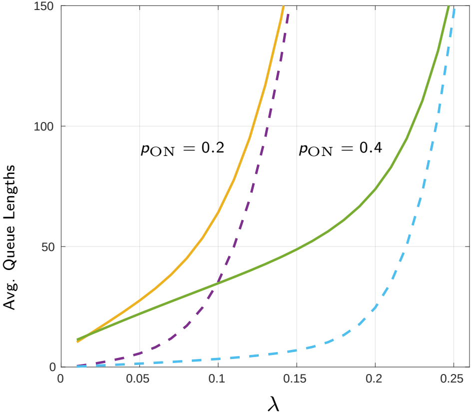


Wireless Network



Performance of the Proposed Policy

Broadcasting: Time Varying Network



Comparison of the time-averaged total queue lengths under the optimal (solid line) and heuristic (dashed line) UMW policy in the time-varying grid network (with parameter p_{ON}), for the broadcast problem.

Generality of the Virtual Queue Paradigm: Most Recent Results

- The algorithmic paradigm that we introduced in our thesis turns out to be pretty general.
- Recently, we deveoped an optimal broadcast policy for wireless networks with point-to-multi-point links
- Here instead of trees, the "routes" consists of Minimum Connected Dominating Sets (MCDS)
- This framework could also be extended to Network Utility Maximization Problems.

Outline

1. Introduction
2. Optimal Broadcasting in a DAG
3. Algorithm for Generalized Flow
4. Simulation Results
5. **Conclusion**

Conclusion

- Our understanding of network control theory has progressed enormously over the past 25 years, starting with the seminal work of Tassiulas and Ephremides in 1992.
- Universal-Max-Weight (UMW) is throughput-optimal and can be used in a wide range of network flow problems
- UMW solves the important open problem of Generalized Flow and eliminates many of the drawbacks of the classical Back-Pressure algorithm
- Open Problem: Does the algorithm remain optimal when used in conjunction with the Physical Queue lengths, instead of the virtual queue-lengths ?
  - Empirical evidence suggest yes
  - Leads to a more efficient practical implementation of UMW

References

Accepted:

1. **A. Sinha** and E. Modiano, Optimal control for generalized network-flow problems, in IEEE INFOCOM '17.
2. **A. Sinha**, G. Paschos, C. P. Li, and E. Modiano, Throughput-optimal multihop broadcast on directed acyclic wireless networks, in IEEE/ACM Transactions on Networking
3. **A. Sinha**, L. Tassiulas, and E. Modiano, Throughput-optimal broadcast in wireless networks with dynamic topology, in ACM MobiHoc '16 (**Best Paper Award**).
4. **A Sinha**, G. Paschos, and E. Modiano, Throughput-optimal multi-hop broadcast algorithms, in ACM MobiHoc '16
5. **A. Sinha**, G. Paschos, C.-p. Li, and E. Modiano, Throughput-optimal broadcast on directed acyclic graphs, in IEEE INFOCOM '15

Under submission:

1. **A. Sinha** and E. Modiano, Optimal control for generalized network-flow problems, in IEEE/ACM Transactions on Networking
2. **A. Sinha** and E. Modiano, Throughput-Optimal Broadcast in Wireless Networks with Point-to-Multipoint Transmissions, in ACM MobiHoc '17,
3. **A. Sinha**, L. Tassiulas, and E. Modiano, Throughput-optimal broadcast in wireless networks with dynamic topology, in IEEE Trans. on Mobile Computing
4. **A Sinha**, G. Paschos, and E. Modiano, Throughput-optimal multi-hop broadcast algorithms, in IEEE/ACM Transactions on Networking