Adversarial Queueing Theory with Setups

Marcos Kiwi Mauricio Soto Christopher Thraves

Journée GANG , Paris / February 2010

Queueing Theory - Jackson Networks

- J stations
- Poisson process arrivals
- Exponential unit mean time service
- FIFO "service discipline"
- Transition matrix $(p_{i,j})$
- Leaving probability: $p_{i,0} = 1 - \sum_{j=1}^{J}(p_{i,j})$

Queueing Theory - Jackson Networks

- J stations
- Poisson process arrivals
- Exponential unit mean time service
- FIFO "service discipline"
- Transition matrix $(p_{i,j})$
- Leaving probability: $p_{i,0} = 1 - \sum_{j=1}^{J}(p_{i,j})$

Queueing Theory - Jackson Networks

- J stations
- Poisson process arrivals
- Exponential unit mean time service
- FIFO "service discipline"
- Transition matrix $(p_{i,j})$
- Leaving probability: $p_{i,0} = 1 - \sum_{j=1}^{J}(p_{i,j})$

Queueing Theory - Jackson Networks

- J stations
- Poisson process arrivals
- Exponential unit mean time service
- FIFO “service discipline”
- Transition matrix $(p_{i,j})$
- Leaving probability: $p_{i,0} = 1 - \sum_{j=1}^{J}(p_{i,j})$

Queueing Theory - Jackson Networks

- J stations
- Poisson process arrivals
- Exponential unit mean time service
- FIFO “service discipline”
- Transition matrix $(p_{i,j})$
- Leaving probability: $p_{i,0} = 1 - \sum_{j=1}^{J}(p_{i,j})$

Queueing Theory - Jackson Networks

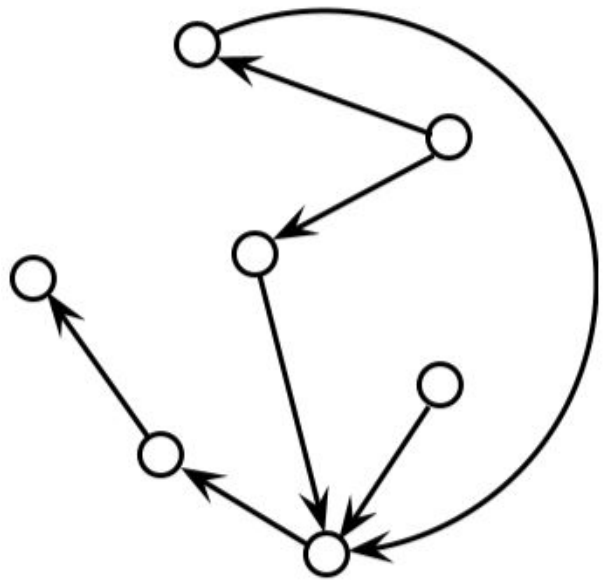
- J stations
- Poisson process arrivals
- Exponential unit mean time service
- FIFO “service discipline”
- Transition matrix $(p_{i,j})$
- Leaving probability: $p_{i,0} = 1 - \sum_{j=1}^{J}(p_{i,j})$

Adversarial Queueing Theory [Borodin et al., 2001]

- Underlying directed graph
- Discrete unit time
- Adversarial Injections (path)
- Packet absorbed at destination
- Unit edge capacity
- Protocol resolves congestion
- Stability: Bounded queue size

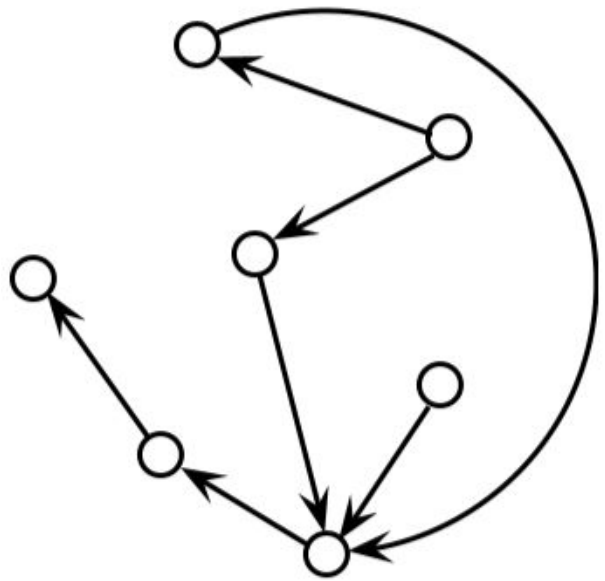
Adversarial Queueing Theory [Borodin et al., 2001]

- Underlying directed graph
- Discrete unit time
- Adversarial Injections (path)
- Packet absorbed at destination
- Unit edge capacity
- Protocol resolves congestion
- Stability: Bounded queue size



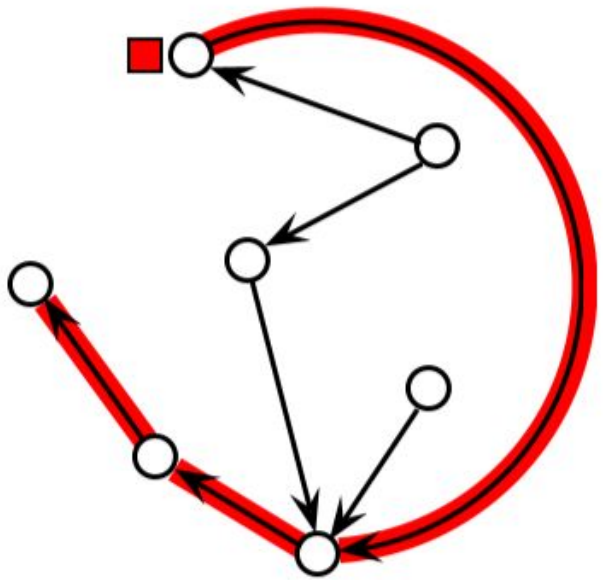
Adversarial Queueing Theory [Borodin et al., 2001]

- Underlying directed graph
- Discrete unit time
- Adversarial Injections (path)
- Packet absorbed at destination
- Unit edge capacity
- Protocol resolves congestion
- Stability: Bounded queue size



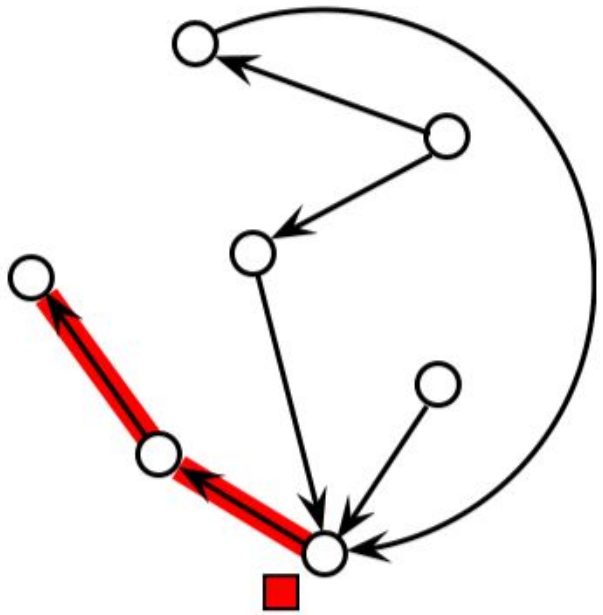
Adversarial Queueing Theory [Borodin et al., 2001]

- Underlying directed graph
- Discrete unit time
- Adversarial Injections (path)
- Packet absorbed at destination
- Unit edge capacity
- Protocol resolves congestion
- Stability: Bounded queue size



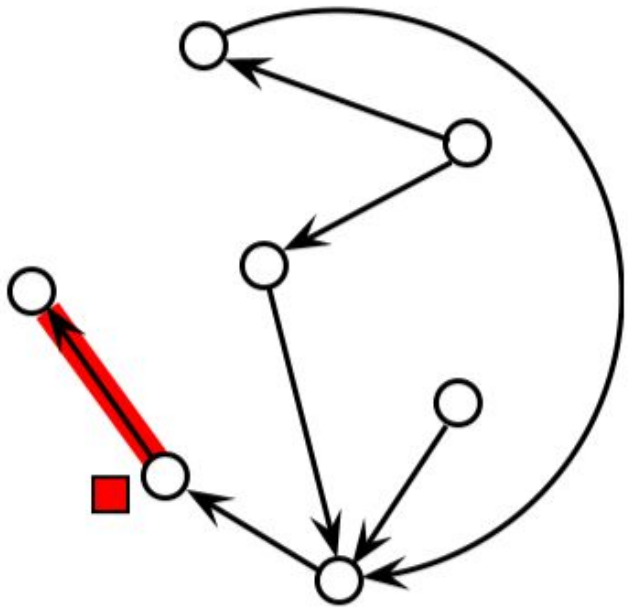
Adversarial Queueing Theory [Borodin et al., 2001]

- Underlying directed graph
- Discrete unit time
- Adversarial Injections (path)
- Packet absorbed at destination
- Unit edge capacity
- Protocol resolves congestion
- Stability: Bounded queue size



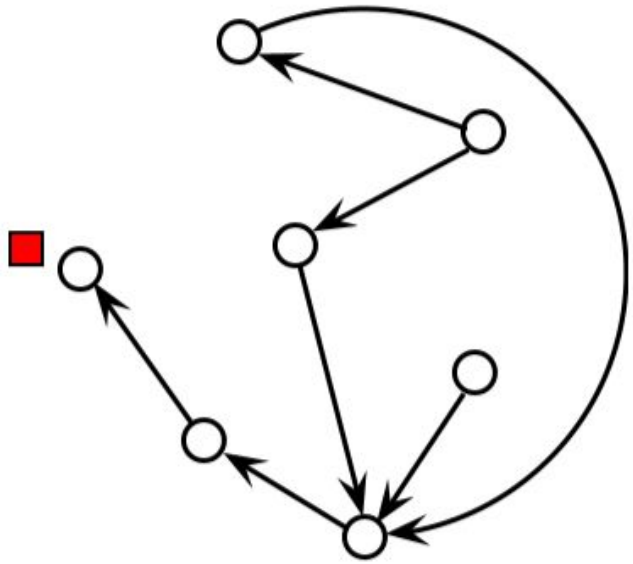
Adversarial Queueing Theory [Borodin et al., 2001]

- Underlying directed graph
- Discrete unit time
- Adversarial Injections (path)
- Packet absorbed at destination
- Unit edge capacity
- Protocol resolves congestion
- Stability: Bounded queue size



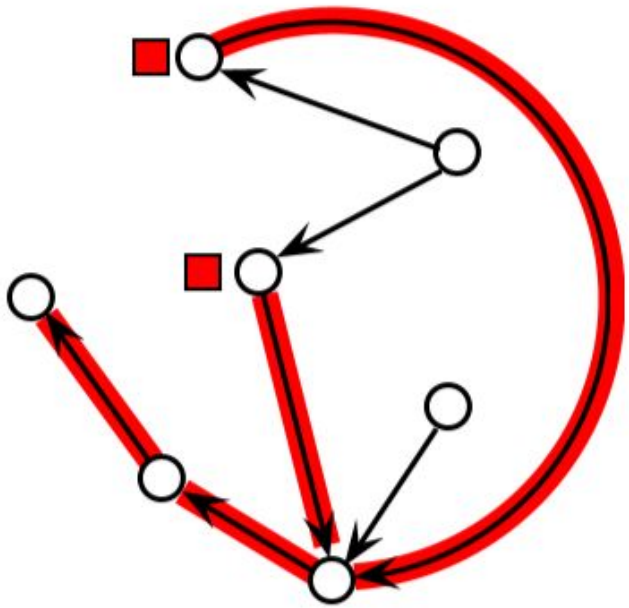
Adversarial Queueing Theory [Borodin et al., 2001]

- Underlying directed graph
- Discrete unit time
- Adversarial Injections (path)
- Packet absorbed at destination
- Unit edge capacity
- Protocol resolves congestion
- Stability: Bounded queue size



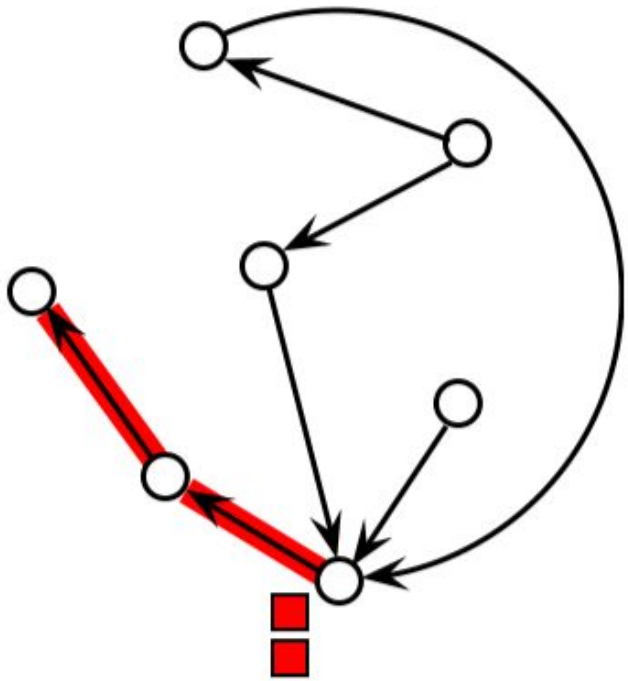
Adversarial Queueing Theory [Borodin et al., 2001]

- Underlying directed graph
- Discrete unit time
- Adversarial Injections (path)
- Packet absorbed at destination
- Unit edge capacity
- Protocol resolves congestion
- Stability: Bounded queue size



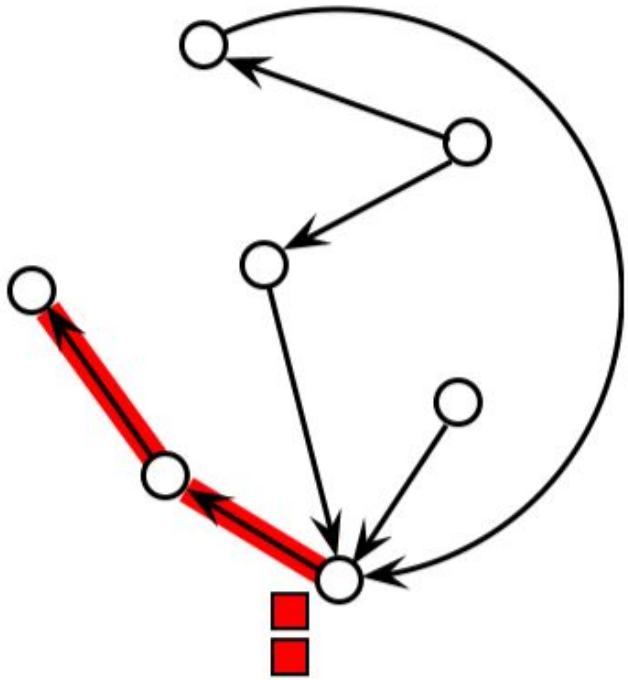
Adversarial Queueing Theory [Borodin et al., 2001]

- Underlying directed graph
- Discrete unit time
- Adversarial Injections (path)
- Packet absorbed at destination
- Unit edge capacity
- Protocol resolves congestion
- Stability: Bounded queue size



Adversarial Queueing Theory [Borodin et al., 2001]

- Underlying directed graph
- Discrete unit time
- Adversarial Injections (path)
- Packet absorbed at destination
- Unit edge capacity
- Protocol resolves congestion
- Stability: Bounded queue size



The Multi-type Adversarial Model (1)

- Underlying directed graph
- Discrete unit time
- Adversarial Injections (path,type)
- Packet absorbed at destination
- Unit edge capacity
- Protocol resolves congestion
- Stability: Bounded queue size
- Set $\mathcal{I}$ of packet classes
- Set-up time when class change

The Multi-type Adversarial Model (1)

- Underlying directed graph
- Discrete unit time
- Adversarial Injections (path,type)
- Packet absorbed at destination
- Unit edge capacity
- Protocol resolves congestion
- Stability: Bounded queue size
- Set $\mathcal{I}$ of packet classes
- Set-up time when class change

The Multi-type Adversarial Model (1)

- Underlying directed graph
- Discrete unit time
- Adversarial Injections (path,type)
- Packet absorbed at destination
- Unit edge capacity
- Protocol resolves congestion
- Stability: Bounded queue size
- Set $\mathcal{I}$ of packet classes
- Set-up time when class change

Formally...

- Adversary injects packets to the networks specifying
  - Path to follow
  - Packet type
- $A_{e,i}(t_1, t_2)$: number of packets of type i injected in the interval $[t_1, t_2)$ using edge e.
- $A_e(t_1, t_2) = \sum_{i=1}^{|\mathcal{I}|} A_{e,i}(t_1, t_2)$, $A_e(t) = A_e(0, t)$.

Load constraint ($0 < r \leq 1$, $b > 0$)

$$\forall e \in E, \forall [t_1, t_2) \quad A_e(t_1, t_2) \leq r \cdot (t_2 - t_1) + b.$$

- Change deserved packets implies a set-up cost $\Delta > 0$
- $D_{e,i}(t_1, t_2)$: number of packets of type i deserved by edge e in the interval $[t_1, t_2)$

Set-up constraint ($\Delta > 0$)

$$\forall e \in E, t \geq 0, i \in \mathcal{I} \quad D_{e,i}(t, t+1) > 0 \Longrightarrow \forall j \neq i \, , D_{e,j}(t, t+\Delta) = 0$$

Formally...

- Adversary injects packets to the networks specifying
  - Path to follow
  - Packet type
- $A_{e,i}(t_1, t_2)$: number of packets of type i injected in the interval $[t_1, t_2)$ using edge e.
- $A_e(t_1, t_2) = \sum_{i=1}^{|\mathcal{I}|} A_{e,i}(t_1, t_2)$, $A_e(t) = A_e(0, t)$.

Load constraint ($0 < r \leq 1$, $b > 0$)

$$\forall e \in E, \forall [t_1, t_2) \quad A_e(t_1, t_2) \leq r \cdot (t_2 - t_1) + b.$$

- Change deserved packets implies a set-up cost $\Delta > 0$
- $D_{e,i}(t_1, t_2)$: number of packets of type i deserved by edge e in the interval $[t_1, t_2)$

Set-up constraint ($\Delta > 0$)

$$\forall e \in E, t \geq 0, i \in \mathcal{I} \quad D_{e,i}(t, t+1) > 0 \Longrightarrow \forall j \neq i \, , D_{e,j}(t, t+\Delta) = 0$$

Formally...

- Adversary injects packets to the networks specifying
  - Path to follow
  - Packet type
- $A_{e,i}(t_1, t_2)$: number of packets of type i injected in the interval $[t_1, t_2)$ using edge e.
- $A_e(t_1, t_2) = \sum_{i=1}^{|\mathcal{I}|} A_{e,i}(t_1, t_2)$, $A_e(t) = A_e(0, t)$.

Load constraint ($0 < r \leq 1$, $b > 0$)

$$\forall e \in E, \forall [t_1, t_2) \quad A_e(t_1, t_2) \leq r \cdot (t_2 - t_1) + b.$$

- Change deserved packets implies a set-up cost $\Delta > 0$
- $D_{e,i}(t_1, t_2)$: number of packets of type i deserved by edge e in the interval $[t_1, t_2)$

Set-up constraint ($\Delta > 0$)

$$\forall e \in E, t \geq 0, i \in \mathcal{I} \quad D_{e,i}(t, t+1) > 0 \Longrightarrow \forall j \neq i \, , D_{e,j}(t, t+\Delta) = 0$$

Formally...

- Adversary injects packets to the networks specifying
  - Path to follow
  - Packet type
- $A_{e,i}(t_1, t_2)$: number of packets of type i injected in the interval $[t_1, t_2)$ using edge e.
- $A_e(t_1, t_2) = \sum_{i=1}^{|\mathcal{I}|} A_{e,i}(t_1, t_2)$, $A_e(t) = A_e(0, t)$.

Load constraint ($0 < r \leq 1$, $b > 0$)

$$\forall e \in E, \forall [t_1, t_2) \quad A_e(t_1, t_2) \leq r \cdot (t_2 - t_1) + b.$$

- Change deserved packets implies a set-up cost $\Delta > 0$
- $D_{e,i}(t_1, t_2)$: number of packets of type i deserved by edge e in the interval $[t_1, t_2)$

Set-up constraint ($\Delta > 0$)

$$\forall e \in E, t \geq 0, i \in \mathcal{I} \quad D_{e,i}(t, t+1) > 0 \Longrightarrow \forall j \neq i \, , D_{e,j}(t, t+\Delta) = 0$$

Formally...

- Adversary injects packets to the networks specifying
  - Path to follow
  - Packet type
- $A_{e,i}(t_1, t_2)$: number of packets of type i injected in the interval $[t_1, t_2)$ using edge e.
- $A_e(t_1, t_2) = \sum_{i=1}^{|\mathcal{I}|} A_{e,i}(t_1, t_2)$, $A_e(t) = A_e(0, t)$.

Load constraint ($0 < r \leq 1$, $b > 0$)

$$\forall e \in E, \forall [t_1, t_2) \quad A_e(t_1, t_2) \leq r \cdot (t_2 - t_1) + b.$$

- Change deserved packets implies a set-up cost $\Delta > 0$
- $D_{e,i}(t_1, t_2)$: number of packets of type i deserved by edge e in the interval $[t_1, t_2)$

Set-up constraint ($\Delta > 0$)

$$\forall e \in E, t \geq 0, i \in \mathcal{I} \quad D_{e,i}(t, t+1) > 0 \Longrightarrow \forall j \neq i \, , D_{e,j}(t, t+\Delta) = 0$$

Formally...

- Adversary injects packets to the networks specifying
  - Path to follow
  - Packet type
- $A_{e,i}(t_1, t_2)$: number of packets of type i injected in the interval $[t_1, t_2)$ using edge e.
- $A_e(t_1, t_2) = \sum_{i=1}^{|\mathcal{I}|} A_{e,i}(t_1, t_2)$, $A_e(t) = A_e(0, t)$.

Load constraint ($0 < r \leq 1$, $b > 0$)

$\forall e \in E, \forall [t_1, t_2) \quad A_e(t_1, t_2) \leq r \cdot (t_2 - t_1) + b.$

- Change deserved packets implies a set-up cost $\Delta > 0$
- $D_{e,i}(t_1, t_2)$: number of packets of type i deserved by edge e in the interval $[t_1, t_2)$

Set-up constraint ($\Delta > 0$)

$\forall e \in E, t \geq 0, i \in \mathcal{I} \quad D_{e,i}(t, t+1) > 0 \Longrightarrow \forall j \neq i \, , D_{e,j}(t, t+\Delta) = 0$

Formally...

- Adversary injects packets to the networks specifying
  - Path to follow
  - Packet type
- $A_{e,i}(t_1, t_2)$: number of packets of type i injected in the interval $[t_1, t_2)$ using edge e.
- $A_e(t_1, t_2) = \sum_{i=1}^{|\mathcal{I}|} A_{e,i}(t_1, t_2)$, $A_e(t) = A_e(0, t)$.

Load constraint ($0 < r \leq 1$, $b > 0$)

$\forall e \in E, \forall [t_1, t_2) \quad A_e(t_1, t_2) \leq r \cdot (t_2 - t_1) + b.$

- Change deserved packets implies a set-up cost $\Delta > 0$
- $D_{e,i}(t_1, t_2)$: number of packets of type i deserved by edge e in the interval $[t_1, t_2)$

Set-up constraint ($\Delta > 0$)

$\forall e \in E, t \geq 0, i \in \mathcal{I} \quad D_{e,i}(t, t+1) > 0 \Longrightarrow \forall j \neq i \, , D_{e,j}(t, t+\Delta) = 0$

Formally...

- Adversary injects packets to the networks specifying
  - Path to follow
  - Packet type
- $A_{e,i}(t_1, t_2)$: number of packets of type i injected in the interval $[t_1, t_2)$ using edge e.
- $A_e(t_1, t_2) = \sum_{i=1}^{|\mathcal{I}|} A_{e,i}(t_1, t_2)$, $A_e(t) = A_e(0, t)$.

Load constraint ($0 < r \leq 1$, $b > 0$)

$\forall e \in E, \forall [t_1, t_2) \quad A_e(t_1, t_2) \leq r \cdot (t_2 - t_1) + b.$

- Change deserved packets implies a set-up cost $\Delta > 0$
- $D_{e,i}(t_1, t_2)$: number of packets of type i deserved by edge e in the interval $[t_1, t_2)$

Set-up constraint ($\Delta > 0$)

$\forall e \in E, t \geq 0, i \in \mathcal{I} \quad D_{e,i}(t, t+1) > 0 \Longrightarrow \forall j \neq i \, , D_{e,j}(t, t+\Delta) = 0$

Formally...

- Adversary injects packets to the networks specifying
  - Path to follow
  - Packet type
- $A_{e,i}(t_1, t_2)$: number of packets of type i injected in the interval $[t_1, t_2)$ using edge e.
- $A_e(t_1, t_2) = \sum_{i=1}^{|\mathcal{I}|} A_{e,i}(t_1, t_2)$, $A_e(t) = A_e(0, t)$.

Load constraint ($0 < r \leq 1$, $b > 0$)

$\forall e \in E, \forall [t_1, t_2) \quad A_e(t_1, t_2) \leq r \cdot (t_2 - t_1) + b.$

- Change deserved packets implies a set-up cost $\Delta > 0$
- $D_{e,i}(t_1, t_2)$: number of packets of type i deserved by edge e in the interval $[t_1, t_2)$

Set-up constraint ($\Delta > 0$)

$\forall e \in E, t \geq 0, i \in \mathcal{I} \quad D_{e,i}(t, t+1) > 0 \Longrightarrow \forall j \neq i \, , D_{e,j}(t, t+\Delta) = 0$

Scheduling Protocol

1. Packet scheduling policy specifies which packet waiting at the edge's queue is to be served.
   Ex. First-In-First-Out, Nearest-To-Go, Farthest-To-Go, Shortest-In-System, Longest-In-System
2. Switching policy specifies whether to keep servicing a type to start servicing a new packet type.
   Ex. Largest-Queue, Round-Robin, First-In-First-Out, Priority

Greedy scheduling protocol

A scheduling protocol is greedy if all non-empty edges are is set-up time or serving a packet

Sensible Policies [Dai and Jennnings. 2004]

For S switching policy and $\theta > 0$, the sensible policy S_θ picks according to S among types with more than θ packets

Scheduling Protocol

1. Packet scheduling policy specifies which packet waiting at the edge's queue is to be served.
   Ex. First-In-First-Out, Nearest-To-Go, Farthest-To-Go, Shortest-In-System, Longest-In-System
2. Switching policy specifies whether to keep servicing a type to start servicing a new packet type.
   Ex. Largest-Queue, Round-Robin, First-In-First-Out, Priority

Greedy scheduling protocol

A scheduling protocol is greedy if all non-empty edges are is set-up time or serving a packet

Sensible Policies [Dai and Jennnings. 2004]

For S switching policy and $\theta > 0$, the sensible policy S_θ picks according to S among types with more than θ packets

Scheduling Protocol

1. Packet scheduling policy specifies which packet waiting at the edge's queue is to be served.
   Ex. First-In-First-Out, Nearest-To-Go, Farthest-To-Go, Shortest-In-System, Longest-In-System
2. Switching policy specifies whether to keep servicing a type to start servicing a new packet type.
   Ex. Largest-Queue, Round-Robin, First-In-First-Out, Priority

Greedy scheduling protocol

A scheduling protocol is greedy if all non-empty edges are is set-up time or serving a packet

Sensible Policies [Dai and Jennnings. 2004]

For S switching policy and $\theta > 0$, the sensible policy S_θ picks according to S among types with more than θ packets

Scheduling Protocol

1. Packet scheduling policy specifies which packet waiting at the edge's queue is to be served.
   Ex. First-In-First-Out, Nearest-To-Go, Farthest-To-Go, Shortest-In-System, Longest-In-System
2. Switching policy specifies whether to keep servicing a type to start servicing a new packet type.
   Ex. Largest-Queue, Round-Robin, First-In-First-Out, Priority

Greedy scheduling protocol

A scheduling protocol is greedy if all non-empty edges are is set-up time or serving a packet

Sensible Policies [Dai and Jennnings. 2004]

For S switching policy and $\theta > 0$, the sensible policy S_θ picks according to S among types with more than θ packets

The Goal of the game

Stability

A network system $(G, \Delta, (P, S); A)$ is stable if for every initial network configuration exists $M > 0$ such that the number of packets in the networks is bounded by M for every time $t \geq 0$.

Network G is stable for a protocol class $\mathcal{S}$ with respect to adversary class $\mathcal{A}$ if $\forall \Delta \geq 0, (P, S) \in \mathcal{S}$ and $A \in \mathcal{A}$, there exists $\theta \geq 0$ for which $(G, \Delta, (P, S_\theta); A)$ is stable.

Universal Stability

G is universally stable with respect to the adversary class $\mathcal{A}$, if G is stable for $\mathcal{S}$ with respect to $\mathcal{A}$ where $\mathcal{S}$ denotes the class of greedy scheduling protocols.

The Goal of the game

Stability

A network system $(G, \Delta, (P, S); A)$ is stable if for every initial network configuration exists $M > 0$ such that the number of packets in the networks is bounded by M for every time $t \geq 0$.

Network G is stable for a protocol class $\mathcal{S}$ with respect to adversary class $\mathcal{A}$ if $\forall \Delta \geq 0, (P, S) \in \mathcal{S}$ and $A \in \mathcal{A}$, there exists $\theta \geq 0$ for which $(G, \Delta, (P, S_\theta); A)$ is stable.

Universal Stability

G is universally stable with respect to the adversary class $\mathcal{A}$, if G is stable for $\mathcal{S}$ with respect to $\mathcal{A}$ where $\mathcal{S}$ denotes the class of greedy scheduling protocols.

Related Works

- Borodin et al. 2001
  - DAG ($r = 1$)
  - Ring ($r = 1$, LIS) but Ring ($r = 1$, FTG).
  - NTG is unstable
- Andrews et al. 2001
  - Ring ($0 < r < 1$)
  - FTG, NTS, LIS, SIS
  - FIFO, LIFO, NTG, FFS
  - Universal Stability decidable for undirected graph
- Goel, 2001
  - Characterization of universal stable networks via minors

Directed Acyclic Graph

Theorem

Acyclic digraphs are universally stable.

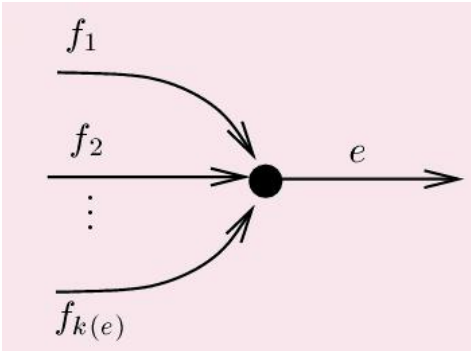
Proof:

$Q_{e,i}(t)$: number of packets of type i at edge e at time step t.

$Q_e(t) = \sum_{i \in \mathcal{I}} Q_{e,i}(t)$.

Fix θ such that $r(\theta + 2\Delta) + b \leq \theta$, we define

$$\psi(e) = \max\{(|\mathcal{I}| + 1)\theta, Q_e(0)\} + \sum_{j=1}^{k} \psi(f_j)\,.$$



where $f_1, \ldots, f_k$ incident edges to e's tail.

$N_e(t)$: number of packets that want use e in t.

Claim: $N_e(t) \leq \psi(e)$ for al $t = k(\theta + 2\Delta)$, $k \in \mathbb{N}$.

$N_e(t') \leq N_e(t) + r(\theta + 2\Delta) + b$ for $t \leq t' < t + (\theta + 2\Delta)$.

Directed Acyclic Graph (2)

Induction on distance of the tail of e to a source of G.

$Q_{e,i}(t) \geq \theta$ for some $i \in \mathcal{I}$:

$$N_e(t + (\theta + 2\Delta)) \leq N_e(t).$$

$Q_{e,i}(t) < \theta$ for all $i \in \mathcal{I}$:

$$N_e(t+(\theta+2\Delta)) \leq Q_e(t)+r(\theta+2\Delta)+b+\sum_{j=1}^{k} N_{f_j}(t) \leq (|\mathcal{I}|+1)\theta+\sum_{j=1}^{k} \psi_{f_j}(t)$$

Unidirectional Ring

The n-node directed ring network is universally stable.

Characterization of Universal Stable Networks

Lemma [Goel, 2001]

Let G_1, G_2 be universally stable digraphs and G formed by adding directed edges from G_1 to G_2 (bridges)then G is universally stable

Proof:
A packet that requires crossing a bridge during $[t_1, t_2)$ can be:

1. injected in $[t_1, t_2)$
2. a packet from G_1
3. packets initial configuration

At most $r(t_2 - t_1) + (b + C_1 + C_e)$.

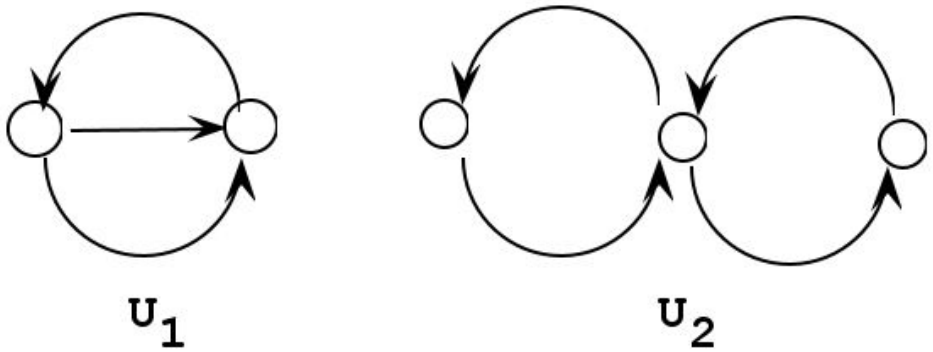
Corollary

In the AQT with setup model, a digraph G is universally stable iff all its strongly connected components are universally stable.

Characterization of Universal Stable Networks (2)

Alvarez et al. 2004

- (1)edge subdivision and (2) cycle subdivision operations to non-stable digraphs yields non-stable digraphs.
- $\varepsilon(G)$ denote the collection of digraphs obtained from G through subdivision operations
- U_1, U_2 are instable



Proposition

In the AQT model with setups, a digraph is universally stable iff it does not contain as a subgraph a digraph in $\varepsilon(U_1) \cup \varepsilon(U_2)$.

FIN