



THE OHIO STATE
UNIVERSITY

IEEE ISIT 2016

Optimizing Data Freshness, Throughput, and Delay in Multi-Server Information-Update Systems

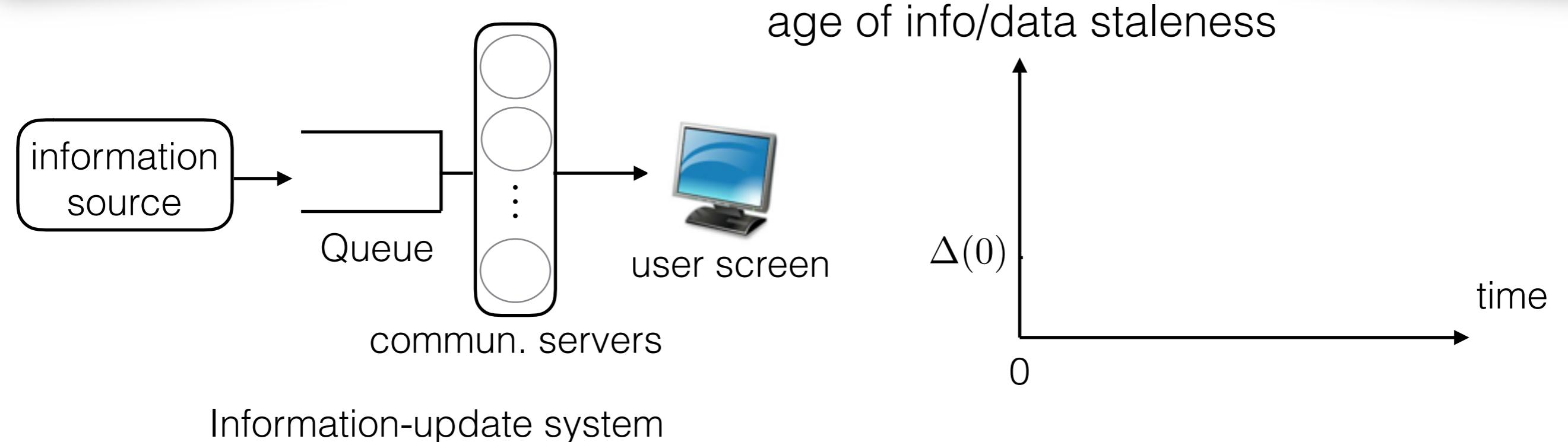
Ahmed M. Bedewy

Joint work with Yin Sun, Ness B. Shroff

Departments of ECE,
The Ohio State University

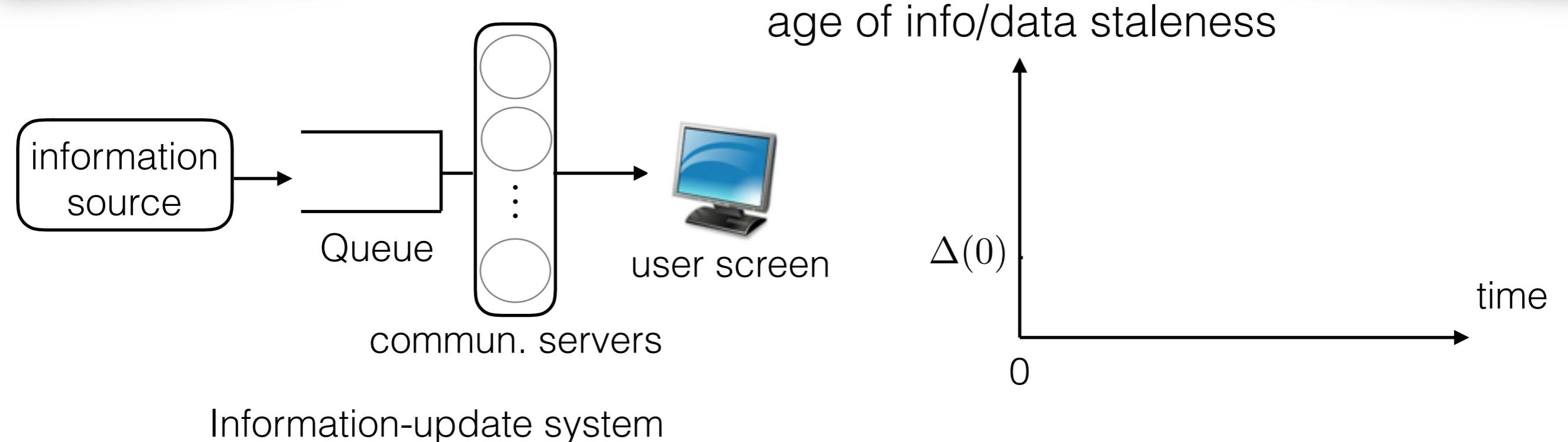
Nov. 7th 2016

What is the Age of Information?



- A stream of messages generated at an information source
- To be sent to a destination via **multiple** communication channels/servers
- Update i is generated at time s_i and delivered at time D_i

What is the Age of Information?

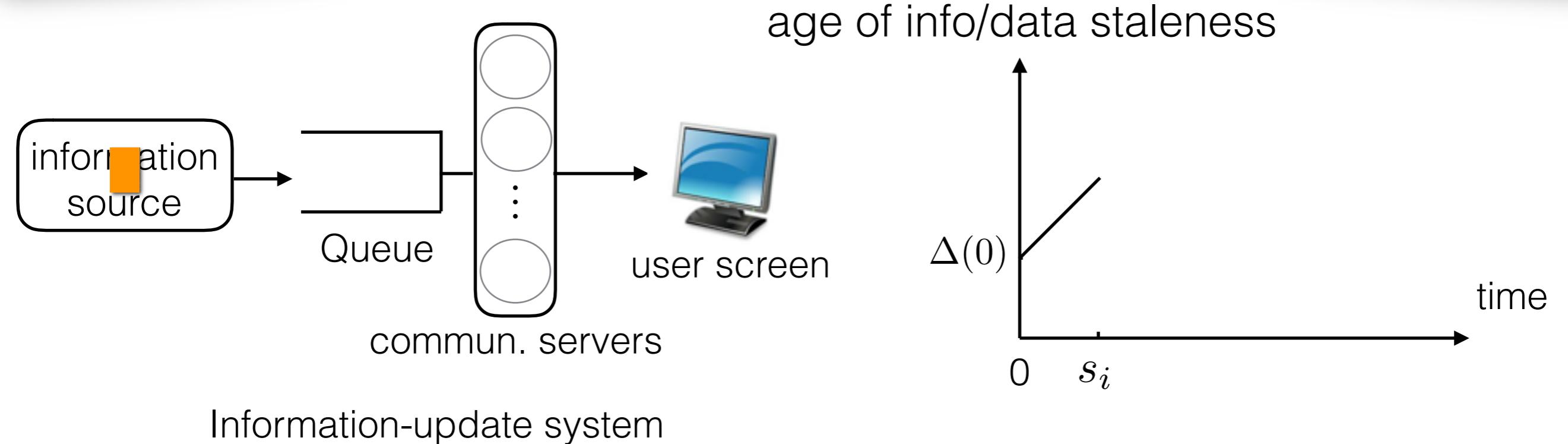


- A stream of messages generated at an information source
- To be sent to a destination via **multiple** communication channels/servers
- Update i is generated at time s_i and delivered at time D_i

Definition: at any time t , **the age-of-information** $\Delta(t)$ is the “age” of the freshest message available at the destination

$$\Delta(t) = t - \max\{s_i : D_i \leq t\}$$

What is the Age of Information?

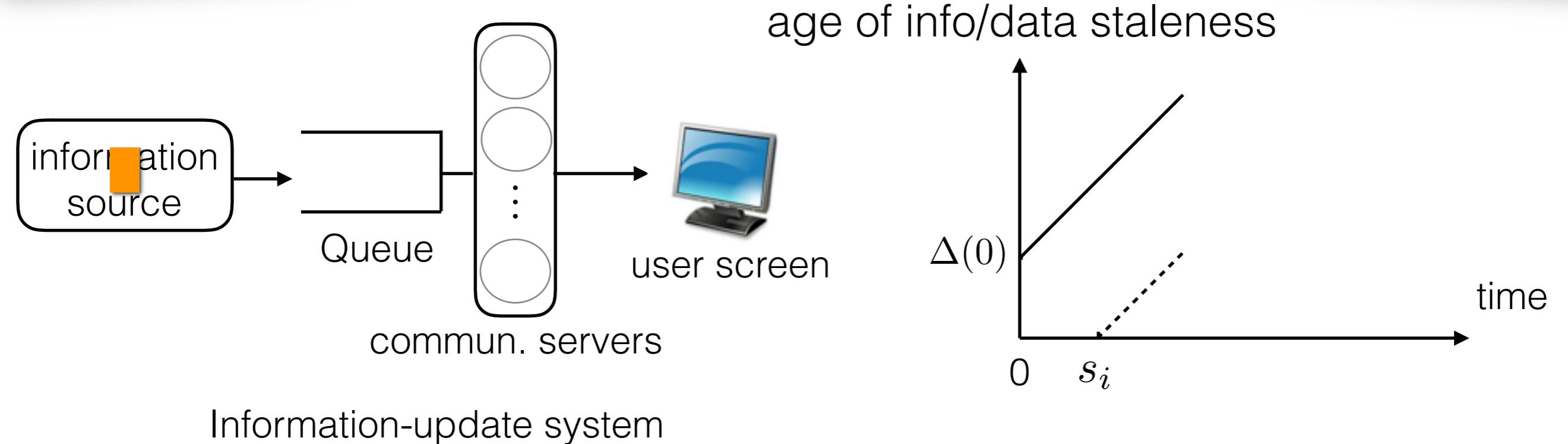


- A stream of messages generated at an information source
- To be sent to a destination via **multiple** communication channels/servers
- Update i is generated at time s_i and delivered at time D_i

Definition: at any time t , **the age-of-information** $\Delta(t)$ is the “age” of the freshest message available at the destination

$$\Delta(t) = t - \max\{s_i : D_i \leq t\}$$

What is the Age of Information?

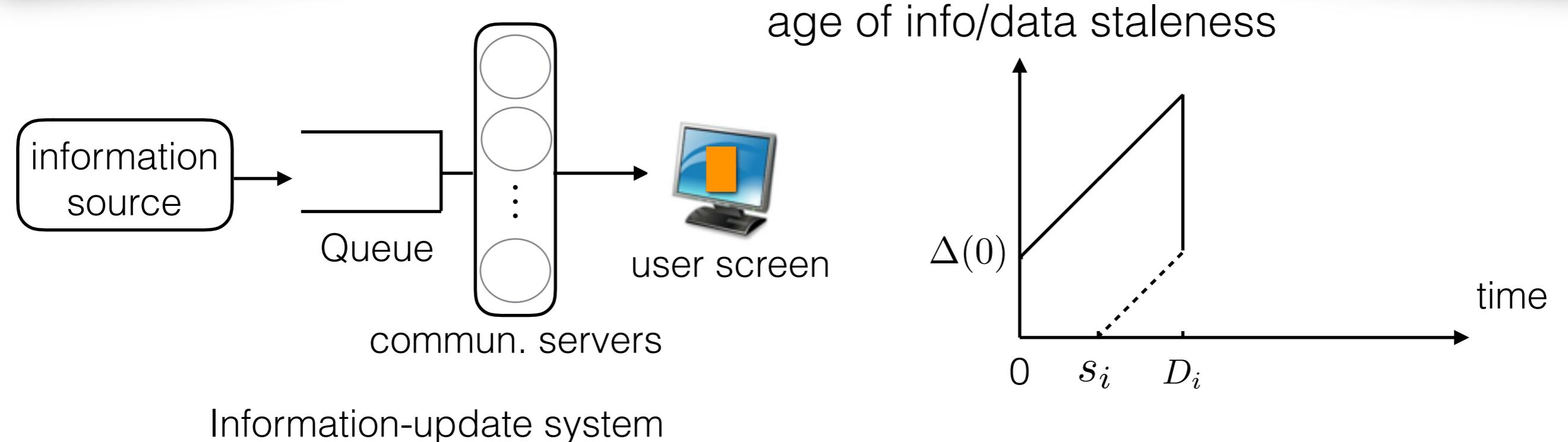


- A stream of messages generated at an information source
- To be sent to a destination via **multiple** communication channels/servers
- Update i is generated at time s_i and delivered at time D_i

Definition: at any time t , **the age-of-information** $\Delta(t)$ is the “age” of the freshest message available at the destination

$$\Delta(t) = t - \max\{s_i : D_i \leq t\}$$

What is the Age of Information?

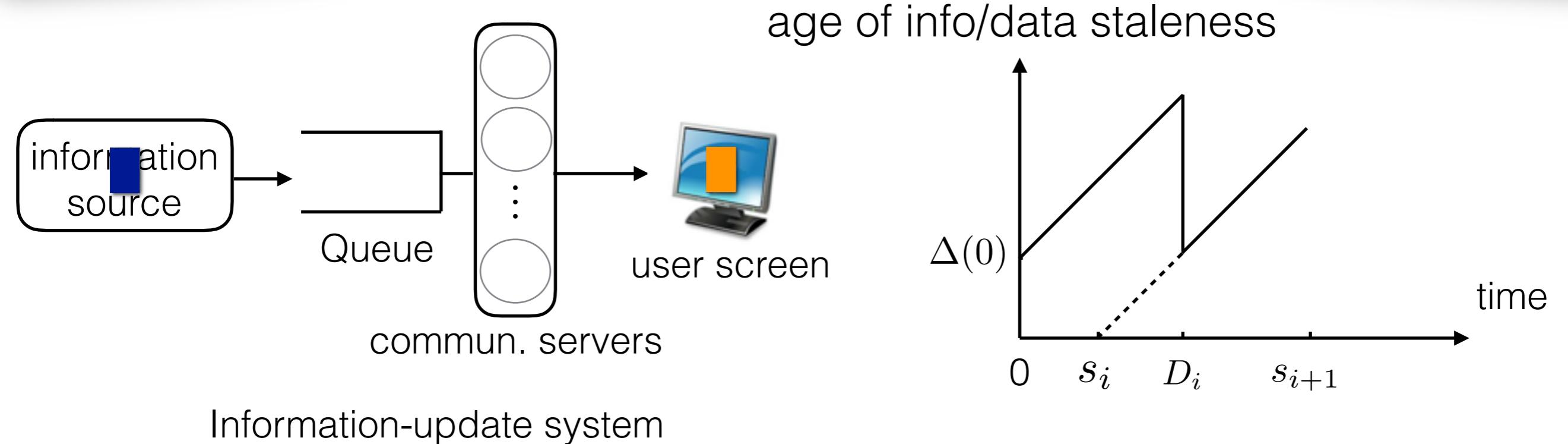


- A stream of messages generated at an information source
- To be sent to a destination via **multiple** communication channels/servers
- Update i is generated at time s_i and delivered at time D_i

Definition: at any time t , **the age-of-information** $\Delta(t)$ is the “age” of the freshest message available at the destination

$$\Delta(t) = t - \max\{s_i : D_i \leq t\}$$

What is the Age of Information?

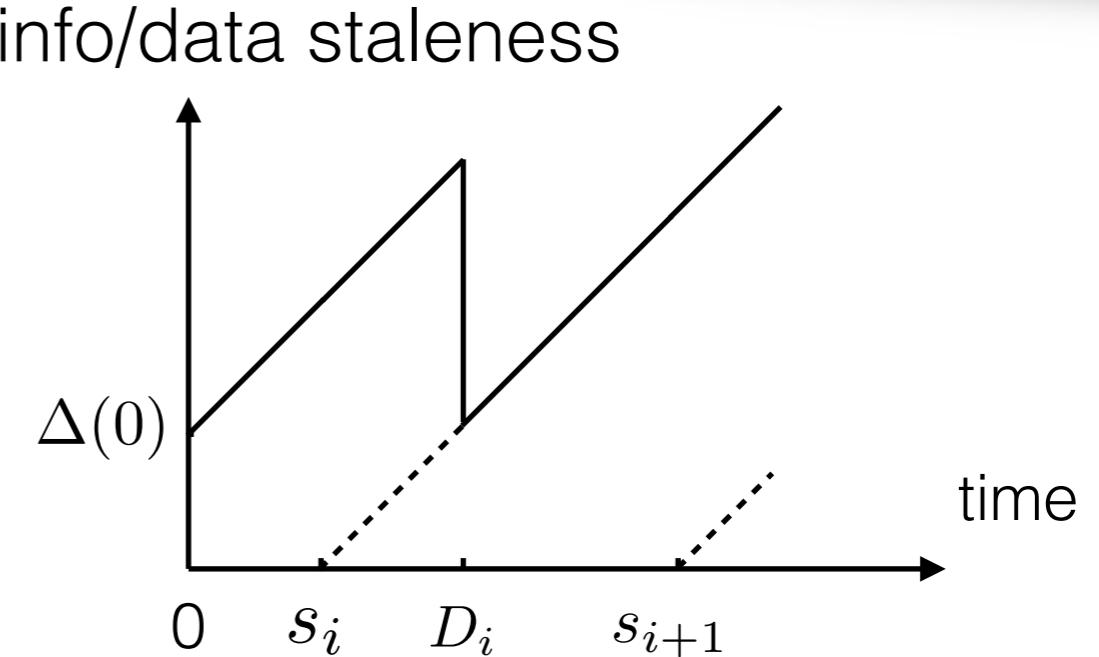
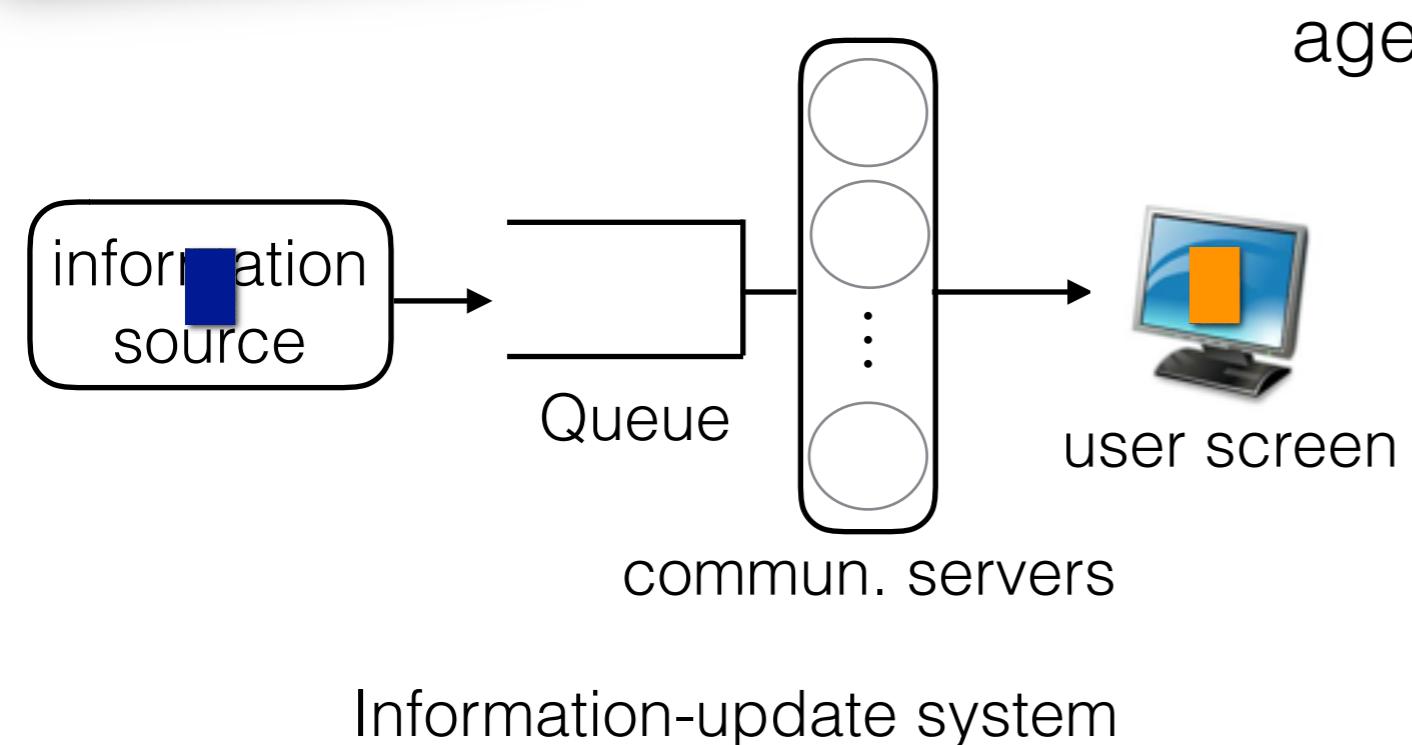


- A stream of messages generated at an information source
- To be sent to a destination via **multiple** communication channels/servers
- Update i is generated at time s_i and delivered at time D_i

Definition: at any time t , **the age-of-information** $\Delta(t)$ is the “age” of the freshest message available at the destination

$$\Delta(t) = t - \max\{s_i : D_i \leq t\}$$

What is the Age of Information?

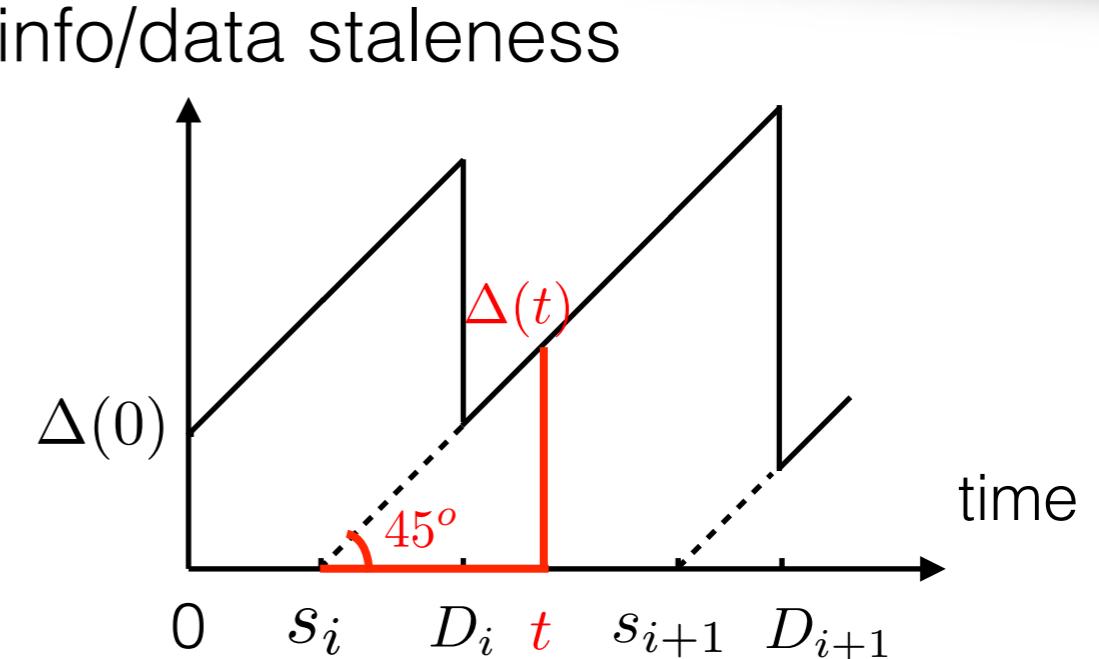
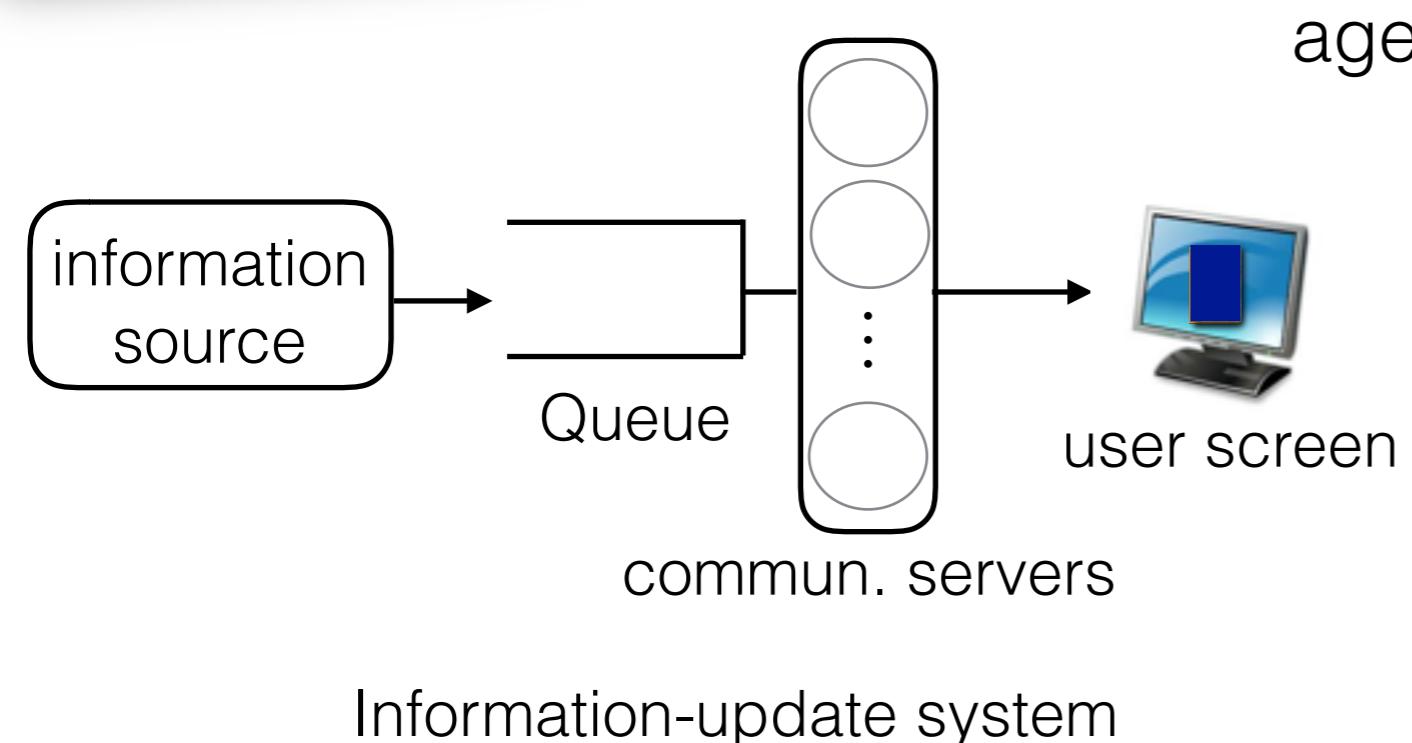


- A stream of messages generated at an information source
- To be sent to a destination via **multiple** communication channels/servers
- Update i is generated at time s_i and delivered at time D_i

Definition: at any time t , **the age-of-information** $\Delta(t)$ is the “age” of the freshest message available at the destination

$$\Delta(t) = t - \max\{s_i : D_i \leq t\}$$

What is the Age of Information?

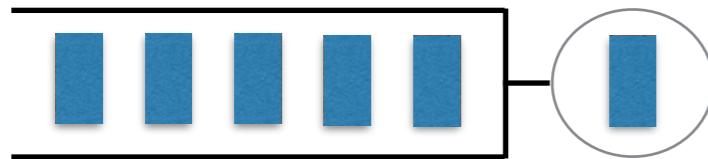


- A stream of messages generated at an information source
- To be sent to a destination via **multiple** communication channels/servers
- Update i is generated at time s_i and delivered at time D_i

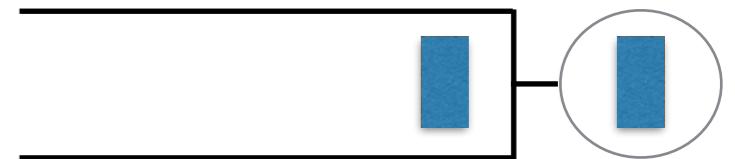
Definition: at any time t , **the age-of-information** $\Delta(t)$ is the “age” of the freshest message available at the destination

$$\Delta(t) = t - \max\{s_i : D_i \leq t\}$$

Difference between Delay & Age



High arrival rate



Low arrival rate

Age is different from delay. [Kaul, Yates, Gruteser, Infocom 2012]

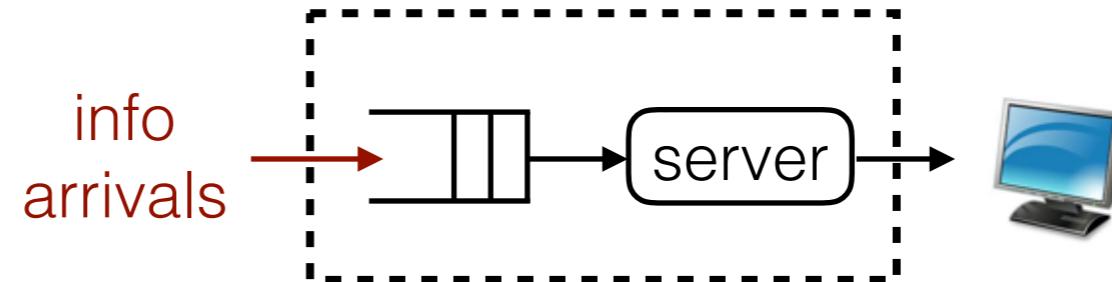
- High arrival rate: long waiting time, packets becomes stale
- Low arrival rate: short queue, but infrequent updates

} High age

Age is not monotonic wrt queue length

- For delay, Little's Law says: $L = \lambda W$
- Delay grows linearly wrt queue length

Open Questions



Enqueue-and-forward model

- **Age Characterization and Age Reduction:**
 - M/M/1, M/D/1, D/M/1 [Kaul, Yates, Gruteser 2012]
 - M/M/2 [Kam, Kompella, Ephremides 2014]
 - Multi-sources [Yates, Kaul ISIT 2012] [Huang, Modiano 2015]
 - Packet management, LCFS [Kam, Kompella, Ephremides 2013, 2014]
 - Channel state info [Costa, Valentin, Ephremides, 2015]
 - LCFS (single server) with & without preemption [Kaul, Yates, Gruteser 2012]
- **Question 1:** Which policy is **age-optimal**?

Open Questions (cont.)

Open Questions (cont.)

- **Information Updates**

- News, emails, notifications, stock quotes, ...



Open Questions (cont.)

- **Information Updates**

- News, emails, notifications, stock quotes, ...



- Users may be interested in not just the latest updates, but also past news.

Open Questions (cont.)

- **Information Updates**

- News, emails, notifications, stock quotes, ...



- Users may be interested in not just the latest updates, but also past news.
- **Question 2:** Is there a policy which simultaneously optimizes **age**, **throughput**, and **delay**?

Open Questions (cont.)

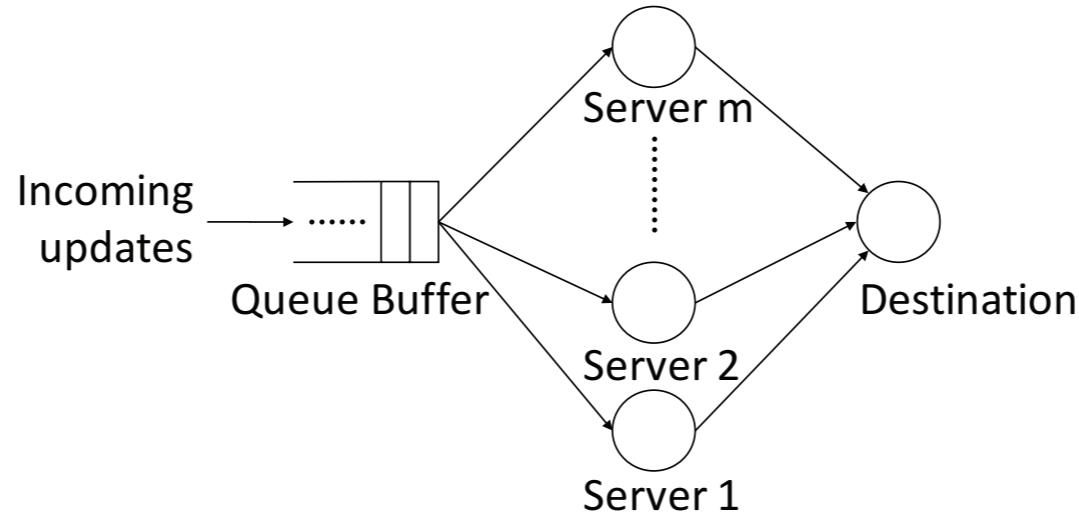
- **Information Updates**

- News, emails, notifications, stock quotes, ...



- Users may be interested in not just the latest updates, but also past news.
- **Question 2:** Is there a policy which simultaneously optimizes **age**, **throughput**, and **delay**?
 - This presentation will answer these two questions

System Model

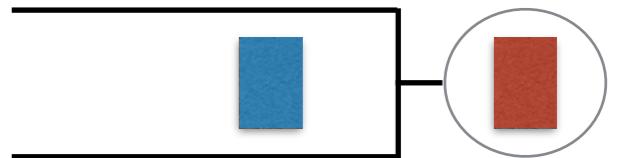


- An information-update system with
 - **m i.i.d.** servers
 - Queue with buffer size B
- Packet i is generated at time s_i , and arrives at time a_i ($s_1 \leq s_2 \leq \dots$)
 - **Arbitrary** arrival process (including **non-stationary**)
 - Update packets can arrive **out of order** (e.g., $a_i > a_{i+1}$ but $s_i < s_{i+1}$)
- The set of all causal policies is denoted by Π

Definitions

Definitions

- **Definition. Service Preemption:** At any time



Definitions

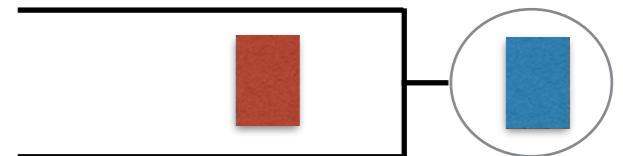
- **Definition. Service Preemption:** At any time
 - A server can switch to send any packet



Definitions

- **Definition. Service Preemption:** At any time

- A server can switch to send any packet
- The preempted packet will be stored back into the queue



Definitions

- **Definition. Service Preemption:** At any time
 - A server can switch to send any packet
 - The preempted packet will be stored back into the queue
 - To be sent at a later time when the servers are available again.



Definitions

- **Definition. Service Preemption:** At any time
 - A server can switch to send any packet
 - The preempted packet will be stored back into the queue
 - To be sent at a later time when the servers are available again.
- **Definition. Stochastic Ordering:** Let X and Y be two random variables. Then, X is said to be **stochastically smaller** than Y (denoted as $X \leq_{\text{st}} Y$), if

$$\mathbb{P}\{X > x\} \leq \mathbb{P}\{Y > x\}, \quad \forall x \in \mathbb{R}.$$



Definitions

- **Definition. Service Preemption:** At any time
 - A server can switch to send any packet
 - The preempted packet will be stored back into the queue
 - To be sent at a later time when the servers are available again.
- **Definition. Stochastic Ordering:** Let X and Y be two random variables. Then, X is said to be **stochastically smaller** than Y (denoted as $X \leq_{\text{st}} Y$), if
$$\mathbb{P}\{X > x\} \leq \mathbb{P}\{Y > x\}, \quad \forall x \in \mathbb{R}.$$
- **In other words:** $X \leq_{\text{st}} Y$ **iff** there exist two random variable \hat{X} and \hat{Y} , defined on the same probability space, such that
$$\hat{Y} =_{\text{st}} Y \quad \& \quad \hat{X} =_{\text{st}} X$$
and
$$\mathbb{P}\{\hat{X} \leq \hat{Y}\} = 1$$
(here $=_{\text{st}}$ denotes equality in law)



Definitions

Definitions

- **Definition. Stochastic Ordering of Stochastic Processes:** Two random processes $\{X(t), t \in [0, \infty)\}$ and $\{Y(t), t \in [0, \infty)\}$ satisfies

$\{X(t), t \in [0, \infty)\} \leq_{\text{st}} \{Y(t), t \in [0, \infty)\}$ **iff** there exist two random processes $\{\hat{X}(t), t \in [0, \infty)\}$ and $\{\hat{Y}(t), t \in [0, \infty)\}$, defined on the same probability space, such that

$$\{\hat{Y}(t), t \in [0, \infty)\} =_{\text{st}} \{Y(t), t \in [0, \infty)\} \quad \& \quad \{\hat{X}(t), t \in [0, \infty)\} =_{\text{st}} \{X(t), t \in [0, \infty)\}$$

and

$$\mathbb{P}[\hat{X}(t) \leq \hat{Y}(t), t \in [0, \infty)] = 1$$

Definitions

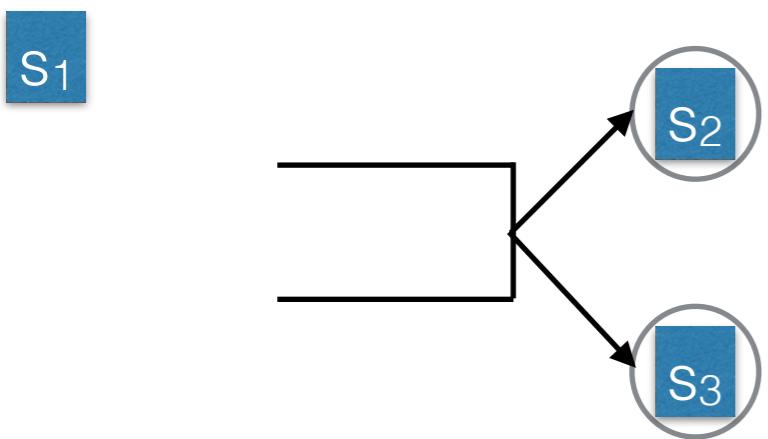
- **Definition. Stochastic Ordering of Stochastic Processes:** Two random processes $\{X(t), t \in [0, \infty)\}$ and $\{Y(t), t \in [0, \infty)\}$ satisfies $\{X(t), t \in [0, \infty)\} \leq_{\text{st}} \{Y(t), t \in [0, \infty)\}$ **iff** there exist two random processes $\{\hat{X}(t), t \in [0, \infty)\}$ and $\{\hat{Y}(t), t \in [0, \infty)\}$, defined on the same probability space, such that
$$\{\hat{Y}(t), t \in [0, \infty)\} =_{\text{st}} \{Y(t), t \in [0, \infty)\} \quad \& \quad \{\hat{X}(t), t \in [0, \infty)\} =_{\text{st}} \{X(t), t \in [0, \infty)\}$$
and
$$\mathbb{P}[\hat{X}(t) \leq \hat{Y}(t), t \in [0, \infty)] = 1$$
- **Definition. Age Optimality:** A policy $\gamma \in \Pi$ is said to be age-optimal, if for all $\pi \in \Pi$
$$\{\Delta_\gamma(t), t \in [0, \infty)\} \leq_{\text{st}} \{\Delta_\pi(t), t \in [0, \infty)\}.$$

Policy P Algorithm

- Policy P: **Preemptive Last Generated First Served (LGFS)**
- Arrival
- Departure

Policy P Algorithm

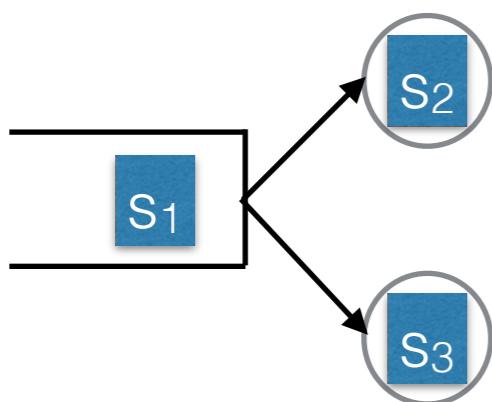
- Policy P: **Preemptive Last Generated First Served (LGFS)**
- Arrival
 - Stale packet



- Departure

Policy P Algorithm

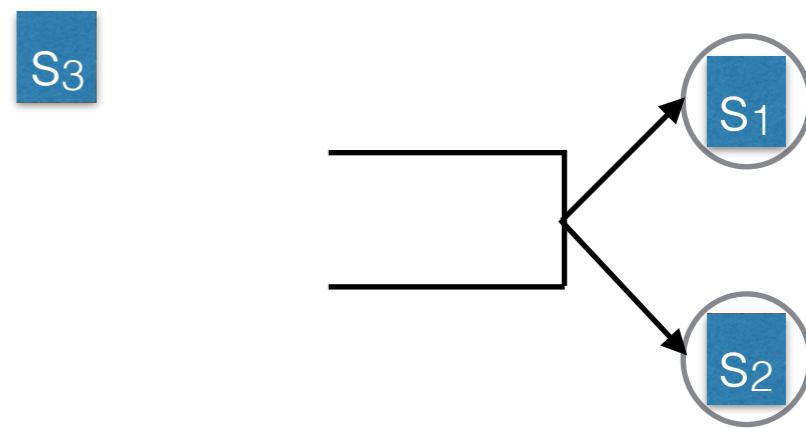
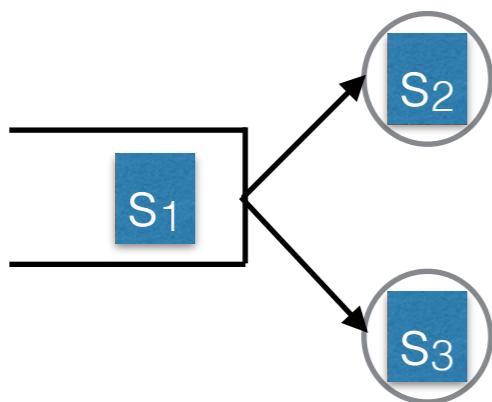
- Policy P: **Preemptive Last Generated First Served (LGFS)**
- Arrival
 - Stale packet



- Departure

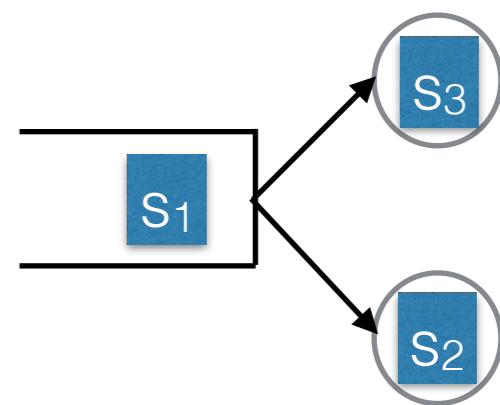
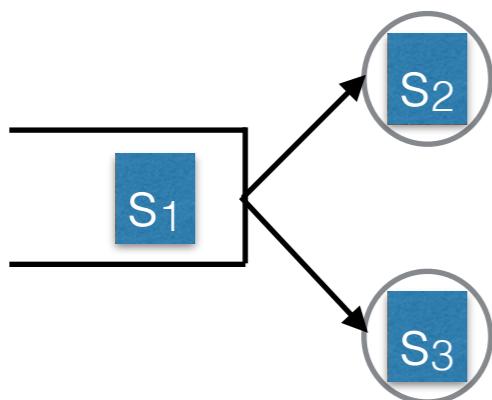
Policy P Algorithm

- Policy P: **Preemptive Last Generated First Served (LGFS)**
- Arrival
 - Stale packet
 - Fresh packet
- Departure



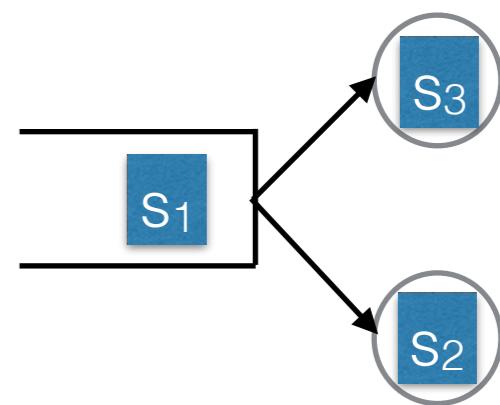
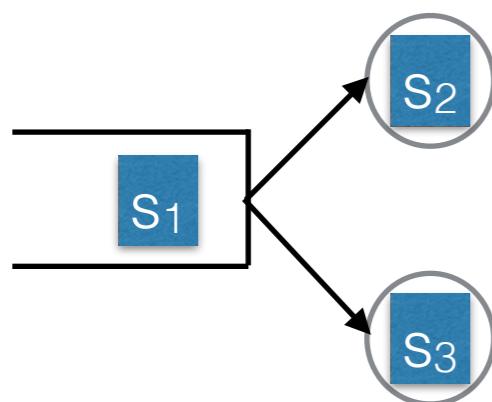
Policy P Algorithm

- Policy P: **Preemptive Last Generated First Served (LGFS)**
- Arrival
 - Stale packet
 - Fresh packet
- Departure



Policy P Algorithm

- Policy P: **Preemptive Last Generated First Served (LGFS)**
- Arrival
 - Stale packet
 - Fresh packet



- Departure

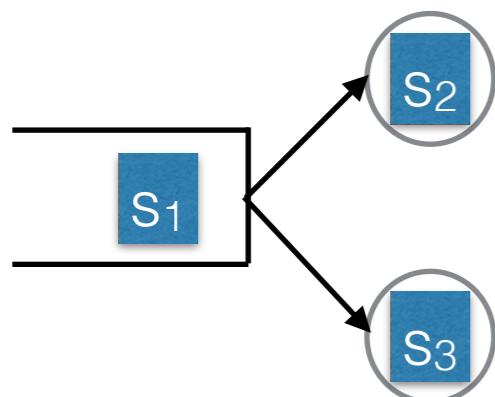
- Policy P is serving the **freshest packets** among all arrived packets.
- Preempted packet is **stored back** in the queue to preserve the **throughput**.

Policy P Algorithm

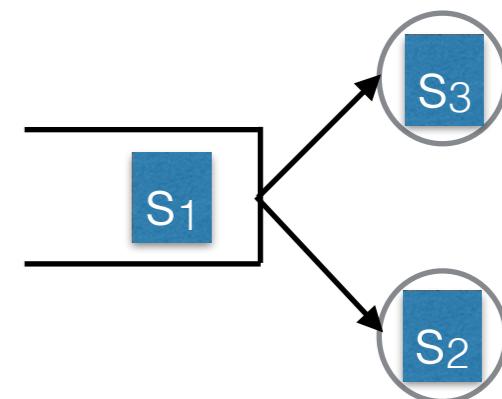
- Policy P: **Preemptive Last Generated First Served (LGFS)**

- Arrival

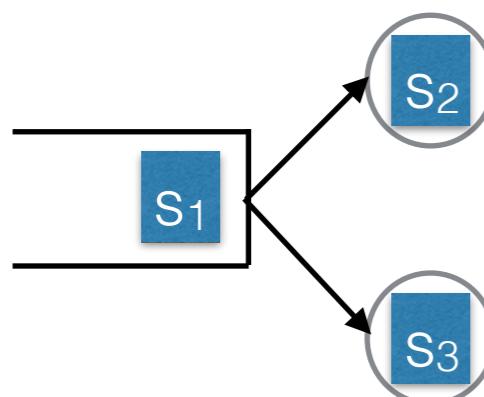
- Stale packet



- Fresh packet



- Departure



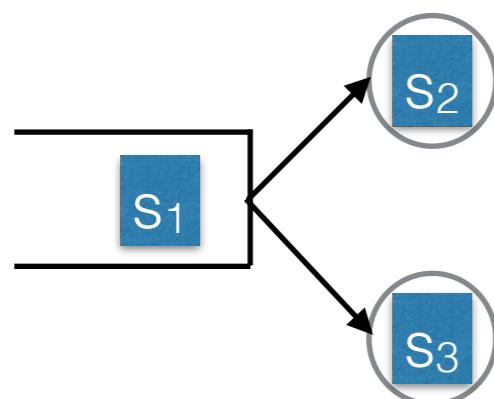
- Policy P is serving the **freshest packets** among all arrived packets.
 - Preempted packet is **stored back** in the queue to preserve the **throughput**.

Policy P Algorithm

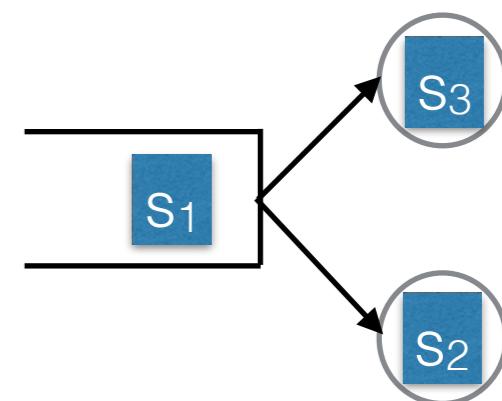
- Policy P: **Preemptive Last Generated First Served (LGFS)**

- Arrival

- Stale packet

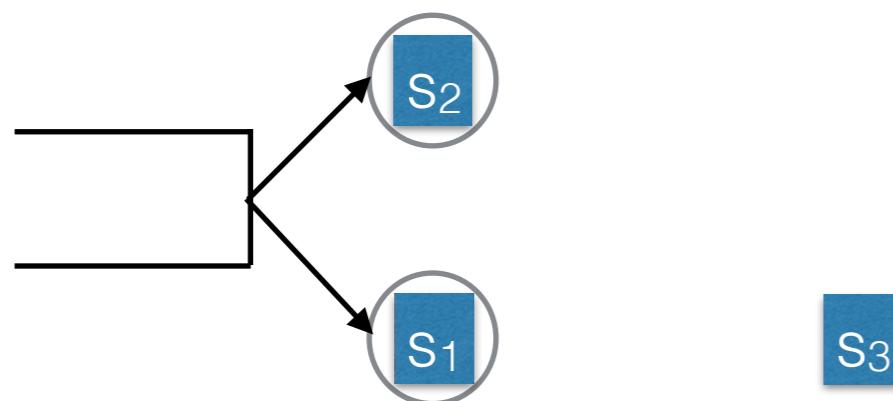


- Fresh packet



- Departure

- Policy P is serving the **freshest packets** among all arrived packets.
 - Preempted packet is **stored back** in the queue to preserve the **throughput**.



Main Theorem

Theorem 1: For

1. *i.i.d. exponential* service time distribution
2. any packet generation and arrival times
3. any buffer sizes B

The **preemptive LGFS** policy is **age-optimal**.

Main Theorem

Theorem 1: For

1. *i.i.d. exponential* service time distribution
2. any packet generation and arrival times
3. any buffer sizes B

The **preemptive LGFS** policy is **age-optimal**.

- The system state of policy π is $V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$, where

Main Theorem

Theorem 1: For 1. *i.i.d. exponential* service time distribution
2. any packet generation and arrival times
3. any buffer sizes B

The **preemptive LGFS** policy is **age-optimal**.

- The system state of policy π is $V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$, where
 - $U_\pi(t)$ is the **largest** time stamp of the **delivered packets**

Main Theorem

Theorem 1: For 1. *i.i.d. exponential* service time distribution
2. any packet generation and arrival times
3. any buffer sizes B

The **preemptive LGFS** policy is **age-optimal**.

- The system state of policy π is $V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$, where
 - $U_\pi(t)$ is the **largest** time stamp of the **delivered packets**
 - $\alpha_{i,\pi}(t)$ is the i -th smallest time stamp of the **packets being transmitted**

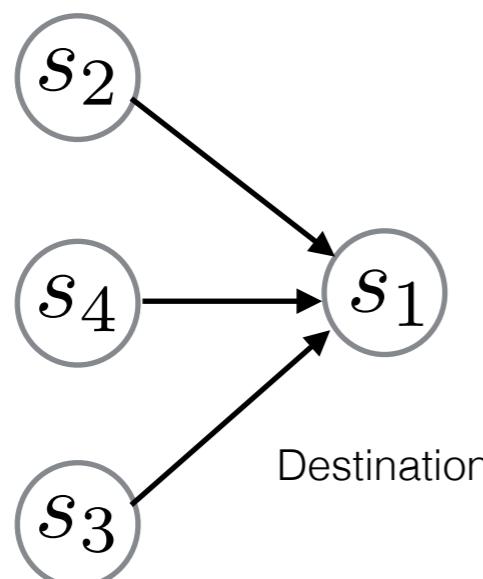
Main Theorem

Theorem 1: For

1. i.i.d. exponential service time distribution
2. any packet generation and arrival times
3. any buffer sizes B

The **preemptive LGFS** policy is **age-optimal**.

- The system state of policy π is $V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$, where
 - $U_\pi(t)$ is the **largest** time stamp of the **delivered packets**
 - $\alpha_{i,\pi}(t)$ is the i -th smallest time stamp of the **packets being transmitted**



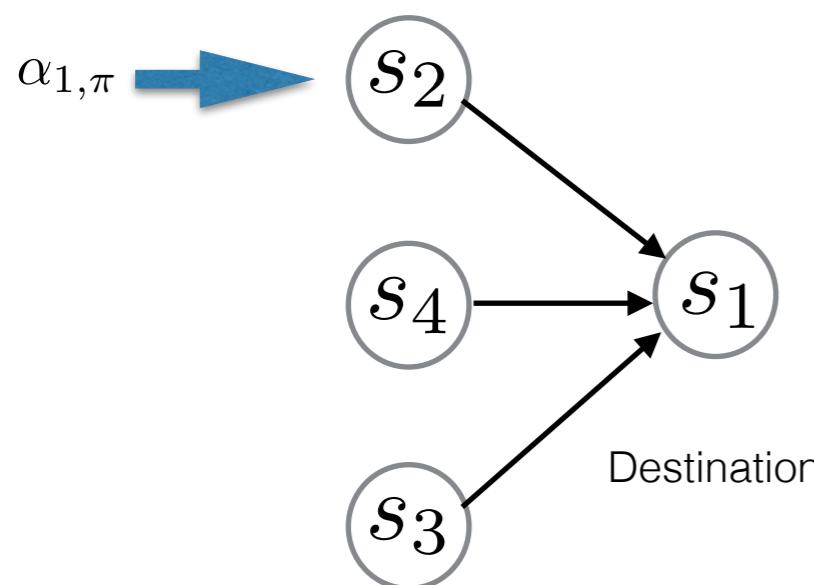
Main Theorem

Theorem 1: For

1. i.i.d. exponential service time distribution
2. any packet generation and arrival times
3. any buffer sizes B

The **preemptive LGFS** policy is **age-optimal**.

- The system state of policy π is $V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$, where
 - $U_\pi(t)$ is the **largest** time stamp of the **delivered packets**
 - $\alpha_{i,\pi}(t)$ is the i -th smallest time stamp of the **packets being transmitted**



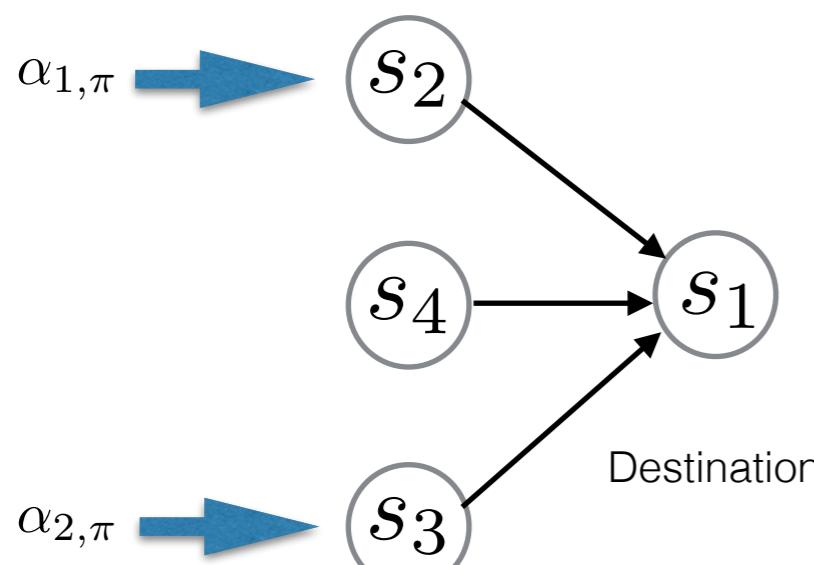
Main Theorem

Theorem 1: For

1. i.i.d. exponential service time distribution
2. any packet generation and arrival times
3. any buffer sizes B

The **preemptive LGFS** policy is **age-optimal**.

- The system state of policy π is $V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$, where
 - $U_\pi(t)$ is the **largest** time stamp of the **delivered packets**
 - $\alpha_{i,\pi}(t)$ is the i -th smallest time stamp of the **packets being transmitted**



3-Servers system

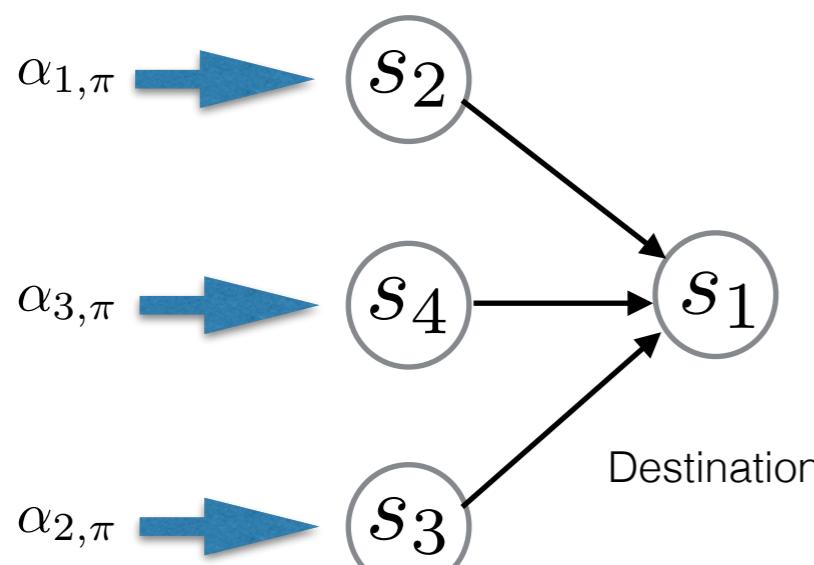
Main Theorem

Theorem 1: For

1. i.i.d. exponential service time distribution
2. any packet generation and arrival times
3. any buffer sizes B

The **preemptive LGFS** policy is **age-optimal**.

- The system state of policy π is $V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$, where
 - $U_\pi(t)$ is the **largest** time stamp of the **delivered packets**
 - $\alpha_{i,\pi}(t)$ is the i -th smallest time stamp of the **packets being transmitted**



3-Servers system

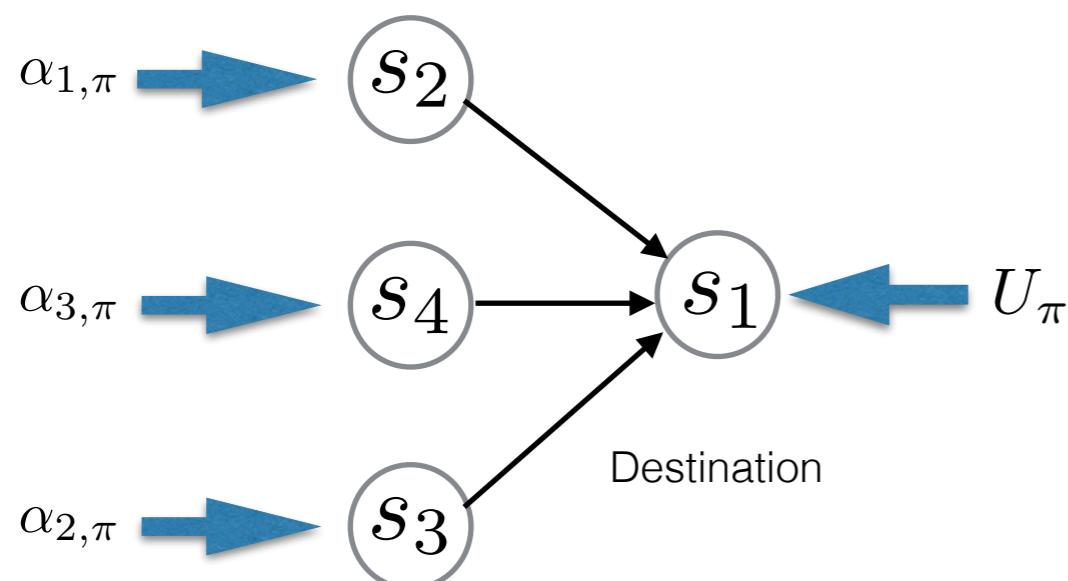
Main Theorem

Theorem 1: For

1. i.i.d. exponential service time distribution
2. any packet generation and arrival times
3. any buffer sizes B

The **preemptive LGFS** policy is **age-optimal**.

- The system state of policy π is $V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$, where
 - $U_\pi(t)$ is the **largest** time stamp of the **delivered packets**
 - $\alpha_{i,\pi}(t)$ is the i -th smallest time stamp of the **packets being transmitted**



3-Servers system

Proof sketch

$$V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$$

Proof sketch

Policy P vs policy π

$$V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$$

Proof sketch

Policy P vs policy π

$$V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$$

Step 1: Construction

- $\{\hat{V}_P(t), t \in [0, \infty)\} =_{\text{st}} \{V_P(t), t \in [0, \infty)\}.$
- $\{\hat{V}_\pi(t), t \in [0, \infty)\} =_{\text{st}} \{V_\pi(t), t \in [0, \infty)\}.$

Proof sketch

Policy P vs policy π

$$V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$$

Step 1: Construction

- $\{\hat{V}_P(t), t \in [0, \infty)\} =_{\text{st}} \{V_P(t), t \in [0, \infty)\}$.
- $\{\hat{V}_\pi(t), t \in [0, \infty)\} =_{\text{st}} \{V_\pi(t), t \in [0, \infty)\}$.

Step 2: Coupling

Policy P



Policy π



Proof sketch

Policy P vs policy π

$$V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$$

Step 1: Construction

- $\{\hat{V}_P(t), t \in [0, \infty)\} =_{\text{st}} \{V_P(t), t \in [0, \infty)\}$.
- $\{\hat{V}_\pi(t), t \in [0, \infty)\} =_{\text{st}} \{V_\pi(t), t \in [0, \infty)\}$.

Step 2: Coupling



Proof sketch

Policy P vs policy π

$$V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$$

Step 1: Construction

- $\{\hat{V}_P(t), t \in [0, \infty)\} =_{\text{st}} \{V_P(t), t \in [0, \infty)\}$.
- $\{\hat{V}_\pi(t), t \in [0, \infty)\} =_{\text{st}} \{V_\pi(t), t \in [0, \infty)\}$.

Step 2: Coupling



Proof sketch

Policy P vs policy π

$$V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$$

Step 1: Construction

- $\{\hat{V}_P(t), t \in [0, \infty)\} =_{\text{st}} \{V_P(t), t \in [0, \infty)\}$.
- $\{\hat{V}_\pi(t), t \in [0, \infty)\} =_{\text{st}} \{V_\pi(t), t \in [0, \infty)\}$.

Step 2: Coupling



Exponential service time distribution is **memory-less**

Step 3: Forward induction in time

Proof sketch

Policy P vs policy π

$$V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$$

Step 1: Construction

- $\{\hat{V}_P(t), t \in [0, \infty)\} =_{\text{st}} \{V_P(t), t \in [0, \infty)\}$.
- $\{\hat{V}_\pi(t), t \in [0, \infty)\} =_{\text{st}} \{V_\pi(t), t \in [0, \infty)\}$.

Step 2: Coupling



Exponential service time distribution is **memory-less**

Step 3: Forward induction in time

- Prove $\mathbb{P}[\hat{V}_P(t) \geq \hat{V}_\pi(t), t \in [0, \infty)] = 1$

Proof sketch

Policy P vs policy π

$$V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$$

Step 1: Construction

- $\{\hat{V}_P(t), t \in [0, \infty)\} =_{\text{st}} \{V_P(t), t \in [0, \infty)\}.$
- $\{\hat{V}_\pi(t), t \in [0, \infty)\} =_{\text{st}} \{V_\pi(t), t \in [0, \infty)\}.$

Step 2: Coupling



Exponential service time distribution is **memory-less**

Step 3: Forward induction in time

- Prove $\mathbb{P}[\hat{V}_P(t) \geq \hat{V}_\pi(t), t \in [0, \infty)] = 1$

Def.

$$\{\hat{V}_P(t), t \in [0, \infty)\} =_{\text{st}} \{V_P(t), t \in [0, \infty)\}.$$

$$\{\hat{V}_\pi(t), t \in [0, \infty)\} =_{\text{st}} \{V_\pi(t), t \in [0, \infty)\}.$$

Proof sketch

Policy P vs policy π

$$V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$$

Step 1: Construction

- $\{\hat{V}_P(t), t \in [0, \infty)\} =_{\text{st}} \{V_P(t), t \in [0, \infty)\}$.
- $\{\hat{V}_\pi(t), t \in [0, \infty)\} =_{\text{st}} \{V_\pi(t), t \in [0, \infty)\}$.

Step 2: Coupling



Exponential service time distribution is **memory-less**

Step 3: Forward induction in time

- Prove $\mathbb{P}[\hat{V}_P(t) \geq \hat{V}_\pi(t), t \in [0, \infty)] = 1$

Def.

$$\{V_P(t), t \in [0, \infty)\} \geq_{\text{st}} \{V_\pi(t), t \in [0, \infty)\}. \quad \forall \pi \in \Pi$$

$$\{\hat{V}_P(t), t \in [0, \infty)\} =_{\text{st}} \{V_P(t), t \in [0, \infty)\}.$$

$$\{\hat{V}_\pi(t), t \in [0, \infty)\} =_{\text{st}} \{V_\pi(t), t \in [0, \infty)\}.$$

Proof sketch

Policy P vs policy π

$$V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$$

Step 1: Construction

- $\{\hat{V}_P(t), t \in [0, \infty)\} =_{\text{st}} \{V_P(t), t \in [0, \infty)\}$.
- $\{\hat{V}_\pi(t), t \in [0, \infty)\} =_{\text{st}} \{V_\pi(t), t \in [0, \infty)\}$.

Step 2: Coupling



Exponential service time distribution is **memory-less**

Step 3: Forward induction in time

- Prove $\mathbb{P}[\hat{V}_P(t) \geq \hat{V}_\pi(t), t \in [0, \infty)] = 1$



Def.

$$\{V_P(t), t \in [0, \infty)\} \geq_{\text{st}} \{V_\pi(t), t \in [0, \infty)\}. \quad \forall \pi \in \Pi$$

$$\{\hat{V}_P(t), t \in [0, \infty)\} =_{\text{st}} \{V_P(t), t \in [0, \infty)\}.$$

$$\{\hat{V}_\pi(t), t \in [0, \infty)\} =_{\text{st}} \{V_\pi(t), t \in [0, \infty)\}.$$



Proof sketch

Policy P vs policy π

$$V_\pi(t) = (U_\pi(t), \alpha_{1,\pi}(t), \dots, \alpha_{m,\pi}(t))$$

Step 1: Construction

- $\{\hat{V}_P(t), t \in [0, \infty)\} =_{\text{st}} \{V_P(t), t \in [0, \infty)\}$.
- $\{\hat{V}_\pi(t), t \in [0, \infty)\} =_{\text{st}} \{V_\pi(t), t \in [0, \infty)\}$.

Step 2: Coupling



Exponential service time distribution is **memory-less**

Step 3: Forward induction in time

- Prove $\mathbb{P}[\hat{V}_P(t) \geq \hat{V}_\pi(t), t \in [0, \infty)] = 1$



Def.

$$\{V_P(t), t \in [0, \infty)\} \geq_{\text{st}} \{V_\pi(t), t \in [0, \infty)\}. \quad \forall \pi \in \Pi$$

$$\{\hat{V}_P(t), t \in [0, \infty)\} =_{\text{st}} \{V_P(t), t \in [0, \infty)\}.$$

$$\{\hat{V}_\pi(t), t \in [0, \infty)\} =_{\text{st}} \{V_\pi(t), t \in [0, \infty)\}.$$



$$\{\Delta_P(t), t \in [0, \infty)\} \leq_{\text{st}} \{\Delta_\pi(t), t \in [0, \infty)\}, \quad \forall \pi \in \Pi$$

Corollaries

- **Corollary:** The age performance of the **preemptive LGFS** policy remains the same for any queue size $B \geq 0$
 - Stale packets are stored back in the queue.
- **Corollary:** For the same system setting as Theorem 1, the preemptive LGFS policy **minimizes**:
 1. The time-average age
 2. Average peak age
 3. Time-average age penalty
 - These age metrics are increasing function of the age process.

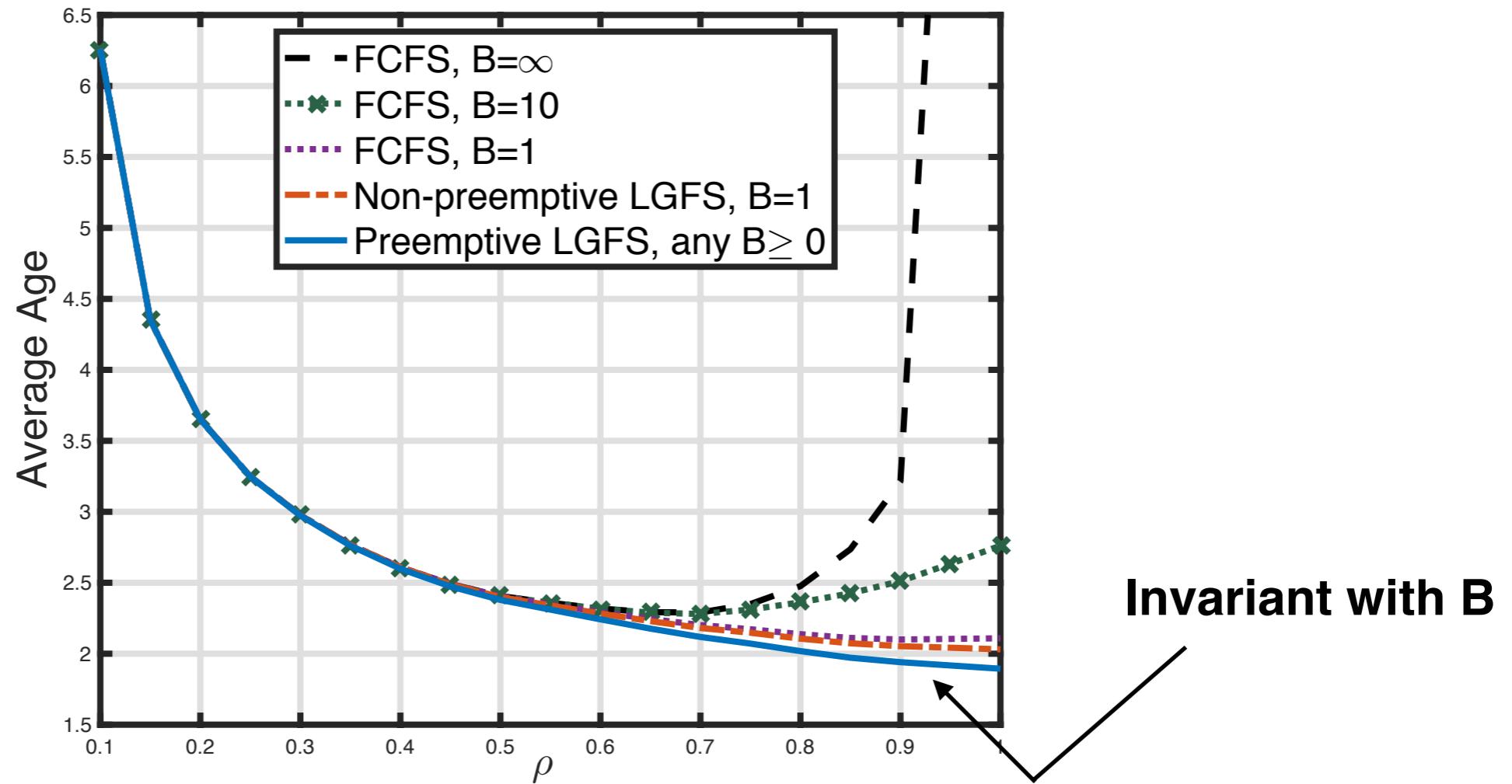
Theorem

Theorem 2: For

1. i.i.d. exponential service time distribution
2. Any packet generation and arrival times
3. Infinite buffer size $B = \infty$

The **preemptive LGFS** policy is **throughput-optimal** and **mean-delay-optimal** among all policies in Π .

Simulation Result



- $m = 5$ servers, buffer size B
- inter-generation times: *i.i.d.* **Erlang-2 distribution**
- $(a_i - s_i)$ is modeled to be either **1** or **100** with **equal probability**

Observations: 1. Preemptive LGFS **outperforms** all other policies.

2. The age performance of the preemptive LGFS is **invariant** for any B
3. **FCFS**: The age gets worse as B increases.

Summary & Future Work

Exponential service time.

- The **preemptive LGFS** optimizes **age**, **throughput** and **delay** among all causally feasible policies for
 - **Arbitrary** packet generation times s_1, s_2, \dots
 - **Arbitrary** arrival process a_1, a_2, \dots (could be **non-stationary**, **non-ergodic**, **out of order arrivals**)
 - Any number of servers

Summary & Future Work

Exponential service time.

- The **preemptive LGFS** optimizes **age**, **throughput** and **delay** among all causally feasible policies for
 - **Arbitrary** packet generation times s_1, s_2, \dots
 - **Arbitrary** arrival process a_1, a_2, \dots (could be **non-stationary**, **non-ergodic**, **out of order arrivals**)
 - Any number of servers
- Other service time distributions?

Summary & Future Work

Exponential service time.

- The **preemptive LGFS** optimizes **age**, **throughput** and **delay** among all causally feasible policies for
 - **Arbitrary** packet generation times s_1, s_2, \dots
 - **Arbitrary** arrival process a_1, a_2, \dots (could be **non-stationary**, **non-ergodic**, **out of order arrivals**)
 - Any number of servers
- Other service time distributions?

Service time dist.	Exponential	NWU Hyperexponential distribution	NBU Erlang distribution
Preemptive LGFS	Age-optimal (any arrival orders)	Not age-optimal in the same policy space	
Non-preemptive LGFS	Near age-optimal		Near age-optimal

*Thank
You!*

