IEEE INFOCOM 2020 AoI Workshop
Virtual Conference

TEMPLE UNIVERSITY

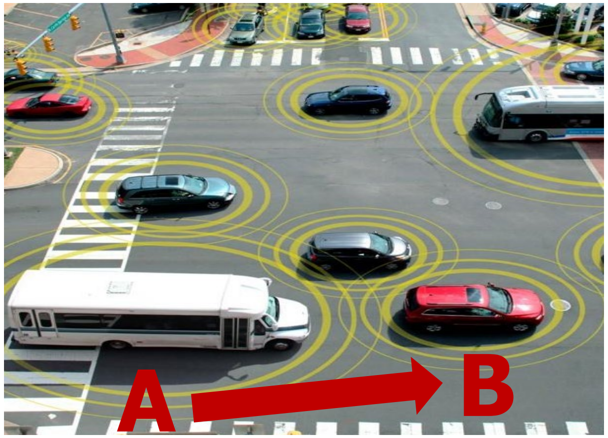
Anti-Aging Scheduling in Single-Server Queues: A Systematic and Comparative Study

Zhongdong Liu
Center for Networked Computing (CNC)
Department of Computer & Information Sciences
Temple University

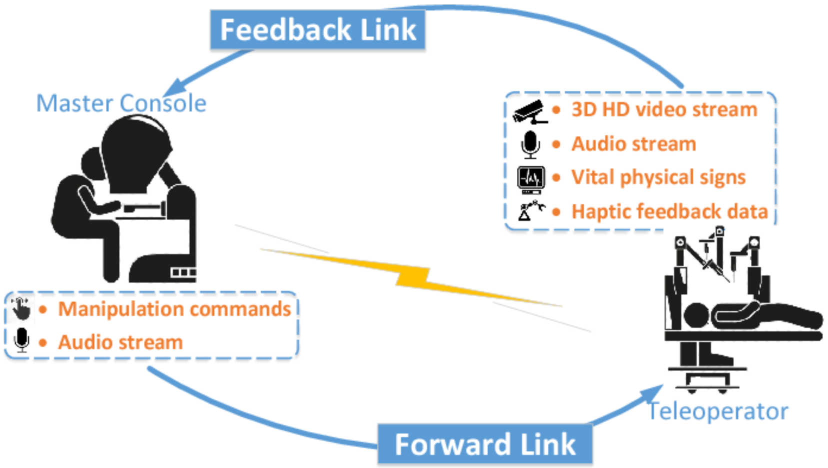
Joint work with
Liang Huang (Zhejiang University of Technology), **Bin Li** (University of Rhode Island) and **Bo Ji** (Temple University)

Freshness Matters

- Real-time services are ubiquitous
  - Intelligent transportation systems & vehicular networks
  - Sensor networks (for environment/health monitoring), wireless channel feedback, news feeds, weather updates, fare aggregating, etc.



(a) Intelligent vehicular networks

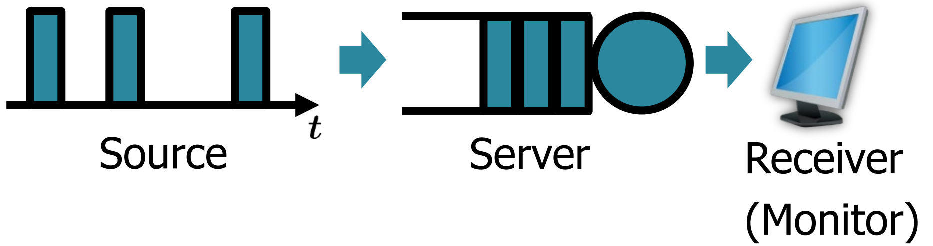


(b) Sensor networks

Age of Information: Definition

- A simple abstract model
  - Source/Server/Receiver (Monitor)
  - Performance of interest:

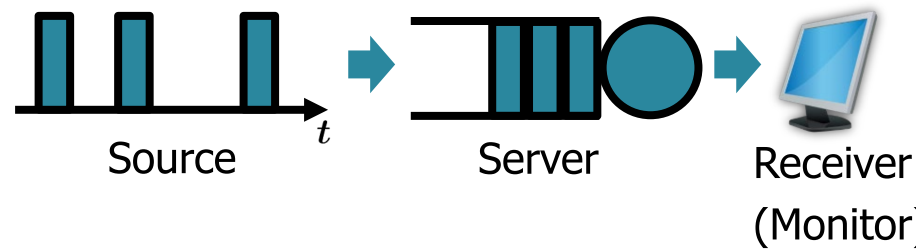
    freshness of update at the monitor



Age of Information: Definition

- A simple abstract model
  - Source/Server/Receiver (Monitor)
  - Performance of interest:

    freshness of update at the monitor



- Definition

  the time elapsed since generation

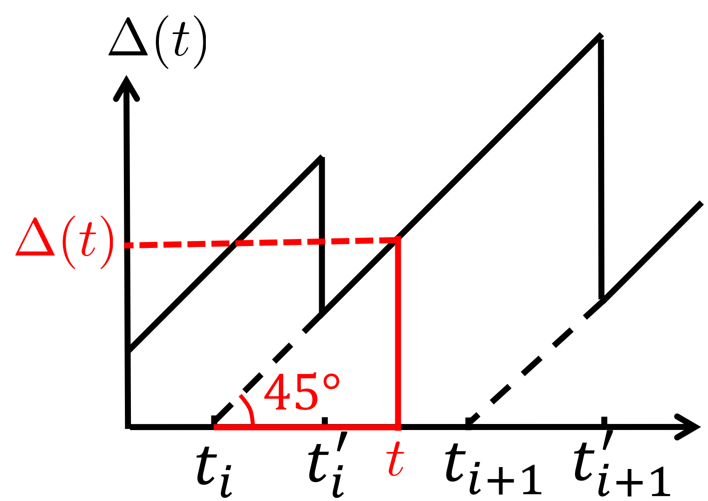
  the latest

  - At time t, the **Age-of-information (AoI)** $\Delta(t)$ is the "age" of the "youngest" update that was delivered to the receiver before time t

  - If update i is generated at t_i and delivered at t_i'

$$\boxed{\Delta(t) = t - \max\{t_i : t_i' \le t\}}$$

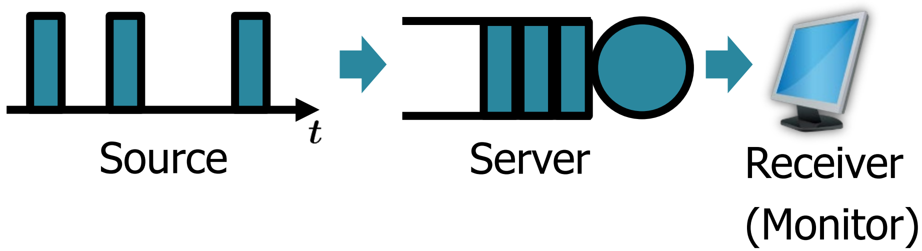
  - AoI grows linearly and drops upon new update delivered



Age of Information: Definition

- A simple abstract model
  - Source/Server/Receiver (Monitor)
  - Performance of interest:

    freshness of update at the monitor



- Definition

  the time elapsed since generation

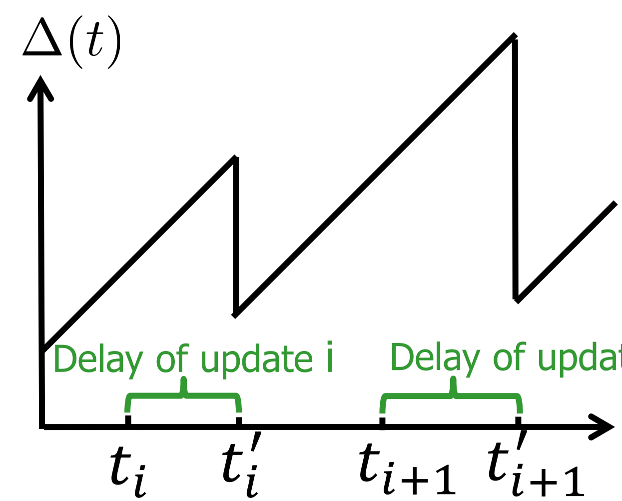
  the latest

  - At time t, the **Age-of-information (AoI)** $\Delta(t)$ is the "age" of the "youngest" update that was delivered to the receiver before time t

  - If update i is generated at t_i and delivered at t_i'

$$\Delta(t) = t - \max\{t_i : t_i' \leq t\}$$

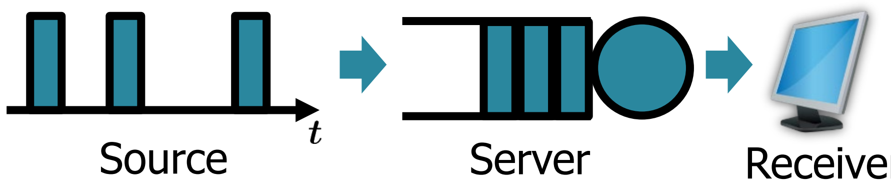
  - AoI grows linearly and drops upon new update delivered



Age of Information: Definition

- A simple abstract model
  - Source/Server/Receiver (Monitor)
  - Performance of interest:

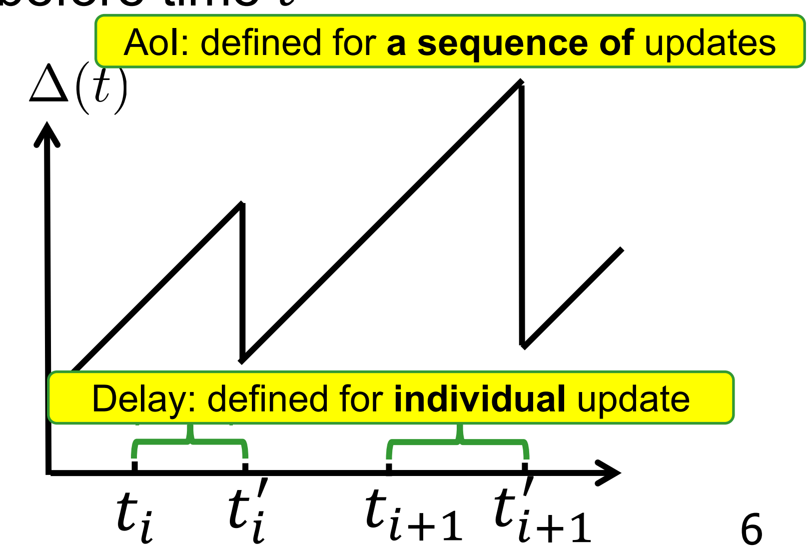
    freshness of update at the monitor



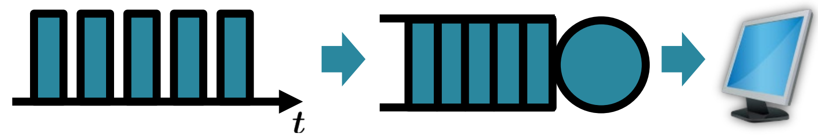
- Definition
  - the time elapsed since generation
  - the latest
  - At time t, the **Age-of-information (AoI)** $\Delta(t)$ is the "age" of the "youngest" update that was delivered to the receiver before time t
  - If update i is generated at t_i and delivered at t_i'

$$\boxed{\Delta(t) = t - \max\{t_i : t_i' \le t\}}$$

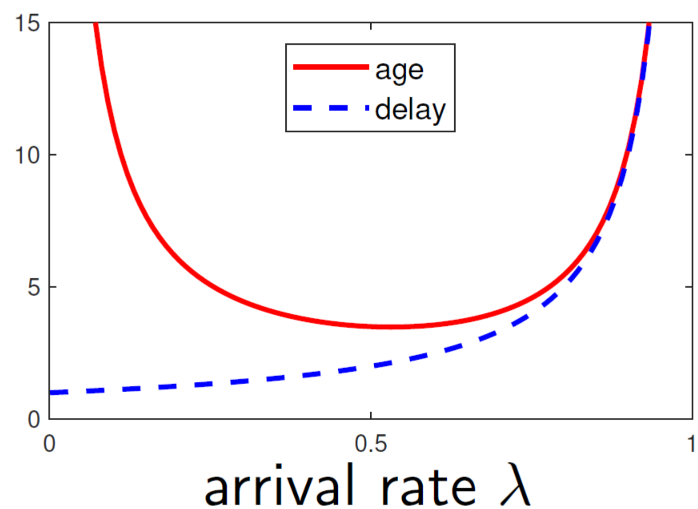
  - AoI grows linearly and drops upon new update delivered



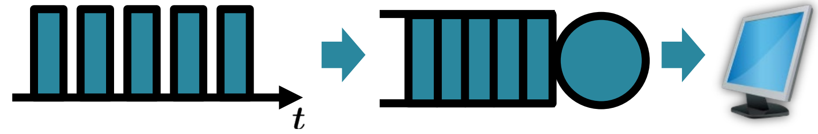
AoI vs. Delay



- Low arrival rate
  - Empty buffer → low delay
  - Infrequent updates → large interarrival time & high AoI
- Large arrival rate
  - Full buffer → high delay
  - Become stale while waiting→ high AoI



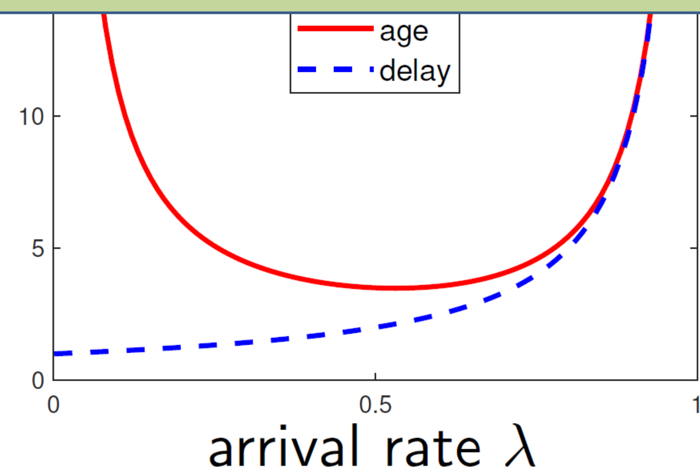
AoI vs. Delay



- Low arrival rate
- Large arrival rate

In M/M/1 FCFS queues: [Kaul et al.,12]

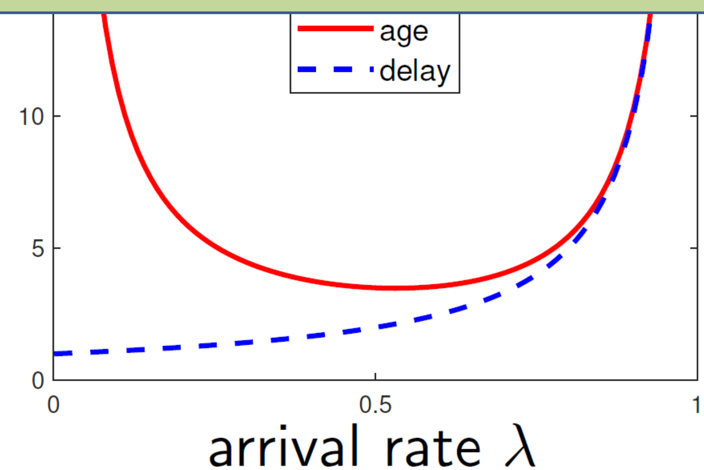
- AoI first decreases, then increases with arrival rate
- Delay increases with arrival rate



AoI vs. Delay

AoI depends on both
- **Queueing delay** (how fast to deliver)
- **Inter-arrival time** (how often to generate)

In M/M/1 FCFS queues: [Kaul et al.,12]
- AoI first decreases, then increases with arrival rate
- Delay increases with arrival rate



Anti-Aging Scheduling

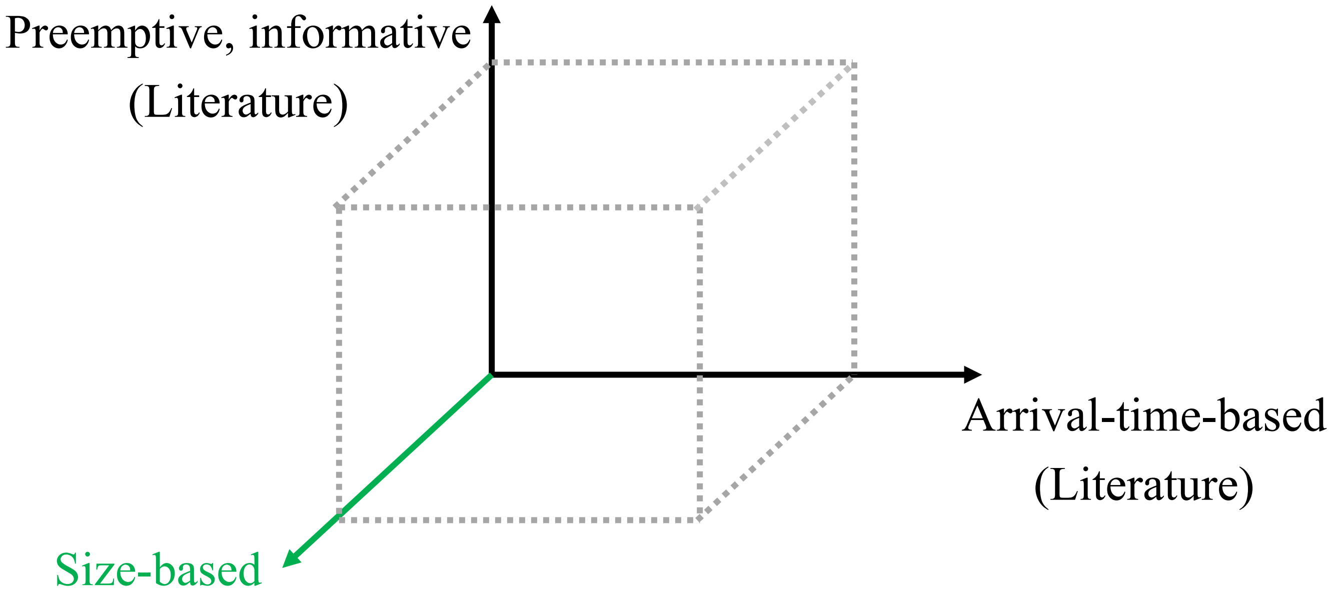
- Motivation & Position

Preemptive, informative
(Literature)

Arrival-time-based
(Literature)

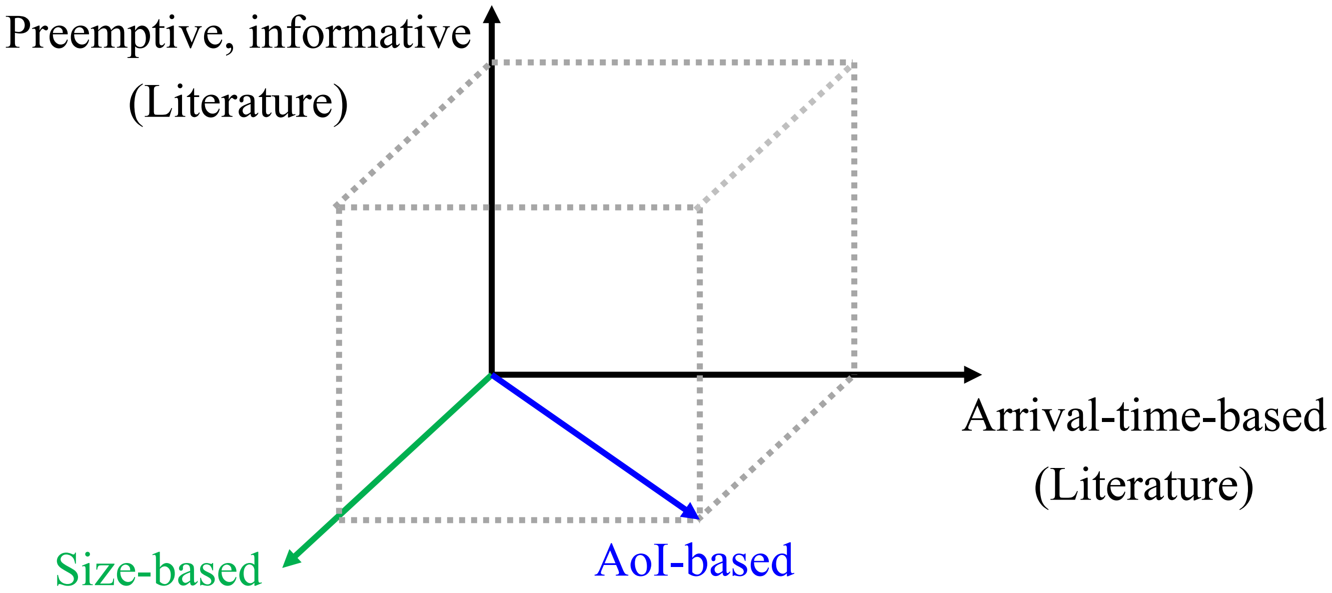
Anti-Aging Scheduling

- Motivation & Position



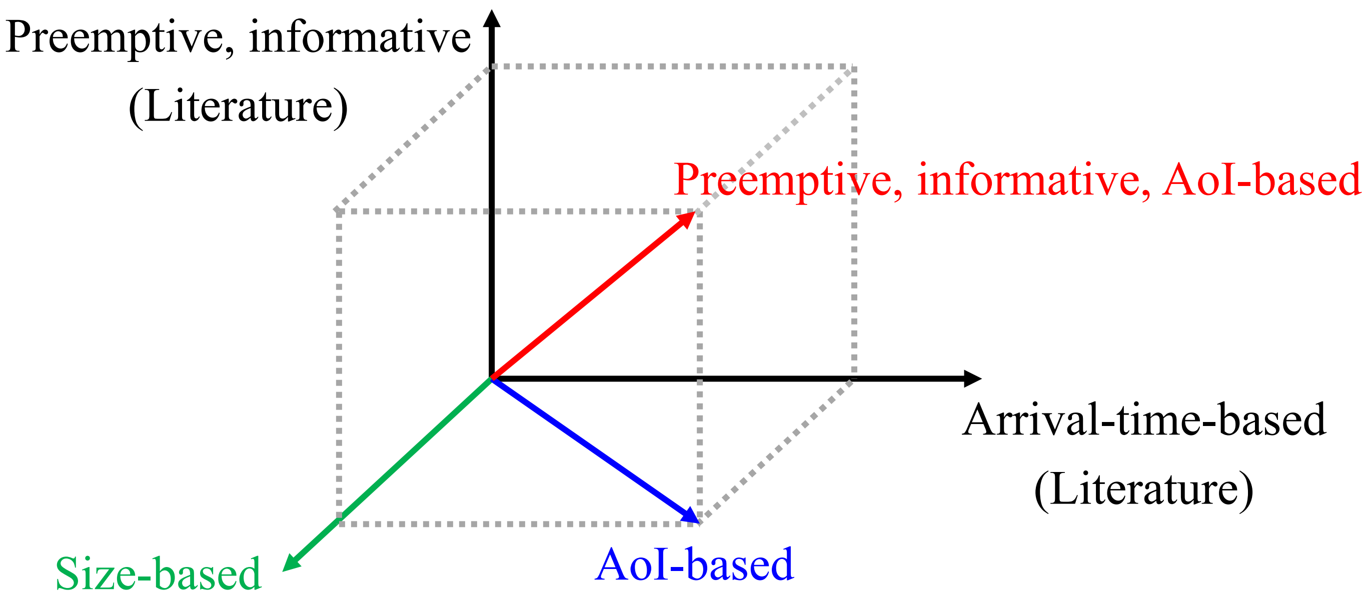
Anti-Aging Scheduling

- Motivation & Position



Anti-Aging Scheduling

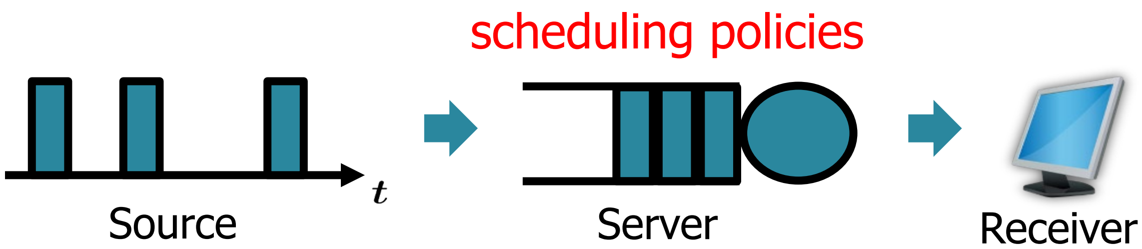
- Motivation & Position



- Contributions
  - Investigate the impact of scheduling policies on the AoI performance
  - Summarize useful guidelines for the design of AoI-efficient policies
  - Equivalence between some size-based and AoI-based policies

Anti-Aging Scheduling

- System model: G/G/1



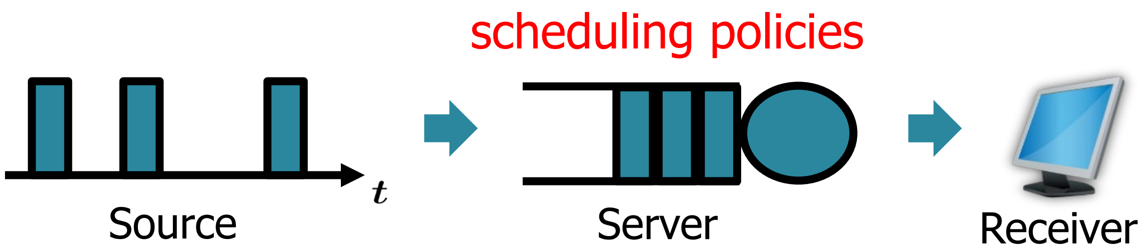
(avg. arrival rate: λ) (avg. service rate: μ)

S:job size=service requirement

$$E[S] = \frac{1}{\mu}$$

Anti-Aging Scheduling

- System model: G/G/1

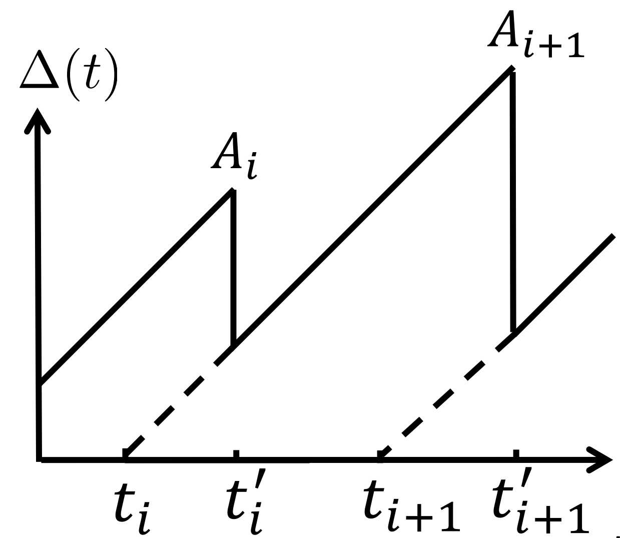


(avg. arrival rate: λ)　　(avg. service rate: μ)

S:job size=service requirement

$$E[S] = \frac{1}{\mu}$$

- Performance metrics
  - Time Average AoI: $\Delta = \lim_{t \to \infty} \frac{1}{t} \int_0^t \Delta(\tau)\, d\tau$
  - Average Peak AoI (PAoI): $E[A] = \frac{1}{N} \sum_{i=0}^{N} A_i$



Sized-based Policies

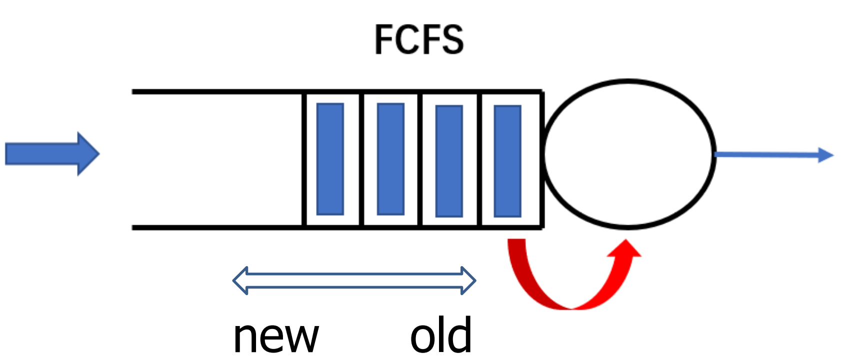
- Common scheduling policies

	Non-preemptive	Preemptive
Blind to Size	FCFS (First-Come-First-Served) LCFS (Last-Come-First-Served) RANDOM (Random-Order-Service)	PS (Processor-Sharing) LCFS_P
Uses Size	**SJF** (Shortest-Job-First)	**SJF_P** **SRPT** (Shortest-Remaining-Processing-Time)

Sized-based Policies

- Common scheduling policies

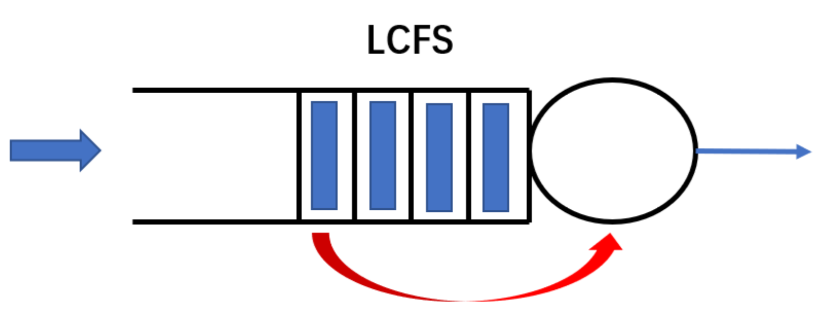
	Non-preemptive	Preemptive
Blind to Size	FCFS (First-Come-First-Served) LCFS (Last-Come-First-Served) RANDOM (Random-Order-Service)	PS (Processor-Sharing) LCFS_P
Uses Size	**SJF** (Shortest-Job-First)	**SJF_P** **SRPT** (Shortest-Remaining-Processing-Time)



Sized-based Policies

- Common scheduling policies

	Non-preemptive	Preemptive
Blind to Size	FCFS (First-Come-First-Served) LCFS (Last-Come-First-Served) RANDOM (Random-Order-Service)	PS (Processor-Sharing) LCFS_P
Uses Size	**SJF** (Shortest-Job-First)	**SJF_P** **SRPT** (Shortest-Remaining-Processing-Time)

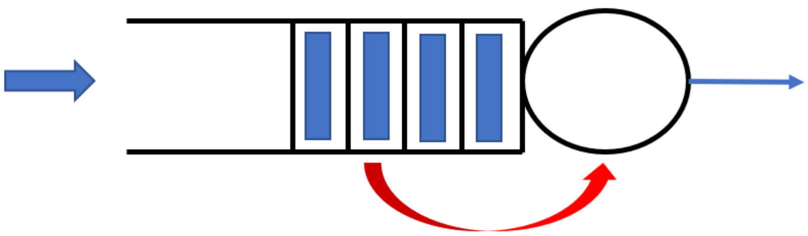


Sized-based Policies

- Common scheduling policies

	Non-preemptive	Preemptive
Blind to Size	FCFS (First-Come-First-Served) LCFS (Last-Come-First-Served) RANDOM (Random-Order-Service)	PS (Processor-Sharing) LCFS_P
Uses Size	**SJF** (Shortest-Job-First)	**SJF_P** **SRPT** (Shortest-Remaining-Processing-Time)

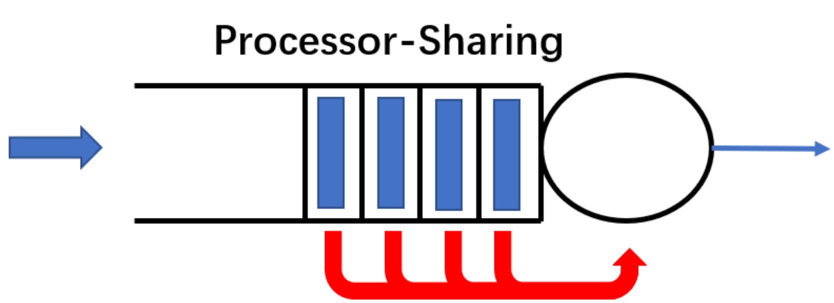
RANDOM



Sized-based Policies

- Common scheduling policies

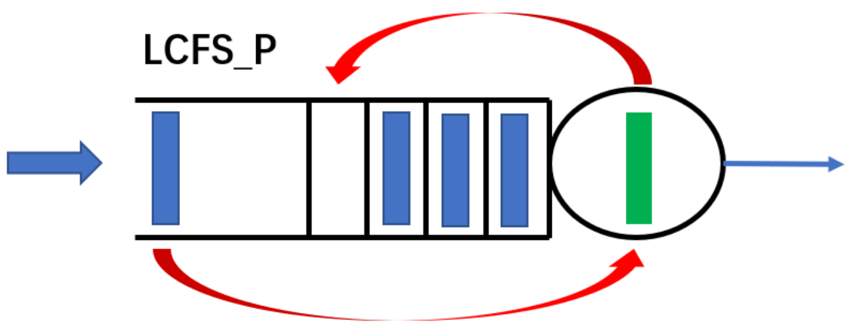
	Non-preemptive	Preemptive
Blind to Size	FCFS (First-Come-First-Served) LCFS (Last-Come-First-Served) RANDOM (Random-Order-Service)	PS (Processor-Sharing) LCFS_P
Uses Size	**SJF** (Shortest-Job-First)	**SJF_P** **SRPT** (Shortest-Remaining-Processing-Time)



Sized-based Policies

- Common scheduling policies

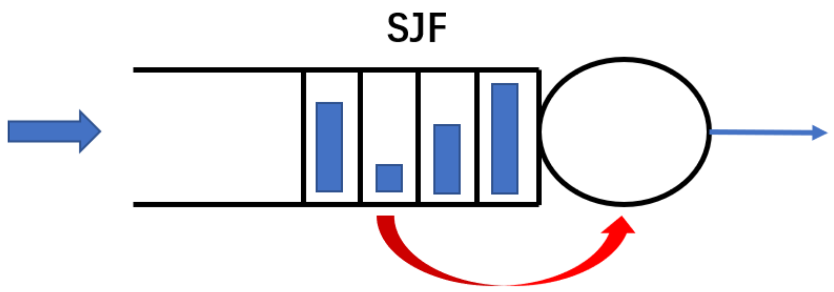
	Non-preemptive	Preemptive
Blind to Size	FCFS (First-Come-First-Served) LCFS (Last-Come-First-Served) RANDOM (Random-Order-Service)	PS (Processor-Sharing) LCFS_P
Uses Size	**SJF** (Shortest-Job-First)	**SJF_P** **SRPT** (Shortest-Remaining-Processing-Time)



Sized-based Policies

- Common scheduling policies

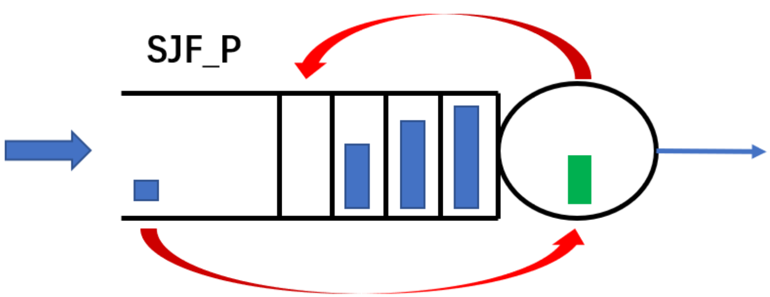
	Non-preemptive	Preemptive
Blind to Size	FCFS (First-Come-First-Served) LCFS (Last-Come-First-Served) RANDOM (Random-Order-Service)	PS (Processor-Sharing) LCFS_P
Uses Size	**SJF** (Shortest-Job-First)	**SJF_P** **SRPT** (Shortest-Remaining-Processing-Time)



Sized-based Policies

- Common scheduling policies

	Non-preemptive	Preemptive
Blind to Size	FCFS (First-Come-First-Served) LCFS (Last-Come-First-Served) RANDOM (Random-Order-Service)	PS (Processor-Sharing) LCFS_P
Uses Size	**SJF** (Shortest-Job-First)	**SJF_P** **SRPT** (Shortest-Remaining-Processing-Time)



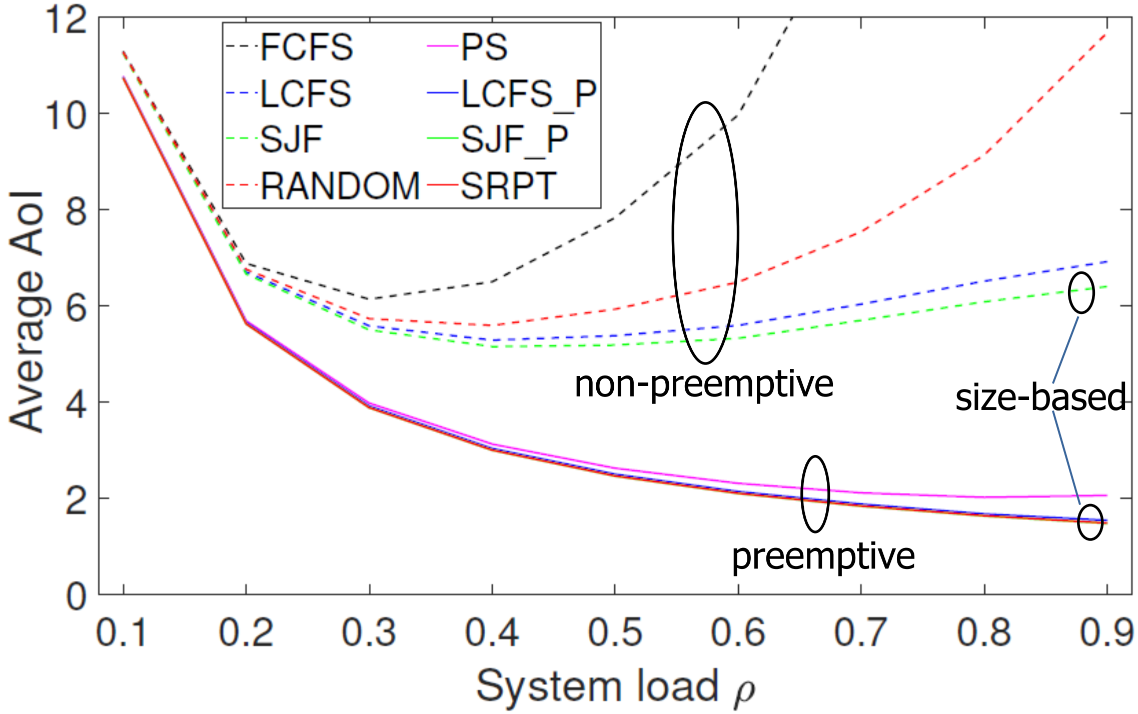
Sized-based Policies

- Common scheduling policies

	Non-preemptive	Preemptive
Blind to Size	FCFS (First-Come-First-Served) LCFS (Last-Come-First-Served) RANDOM (Random-Order-Service)	PS (Processor-Sharing) LCFS_P
Uses Size	**SJF** (Shortest-Job-First)	**SJF_P** **SRPT** (Shortest-Remaining-Processing-Time)

Sized-based Policies

- Average AoI performance (with update size info.)

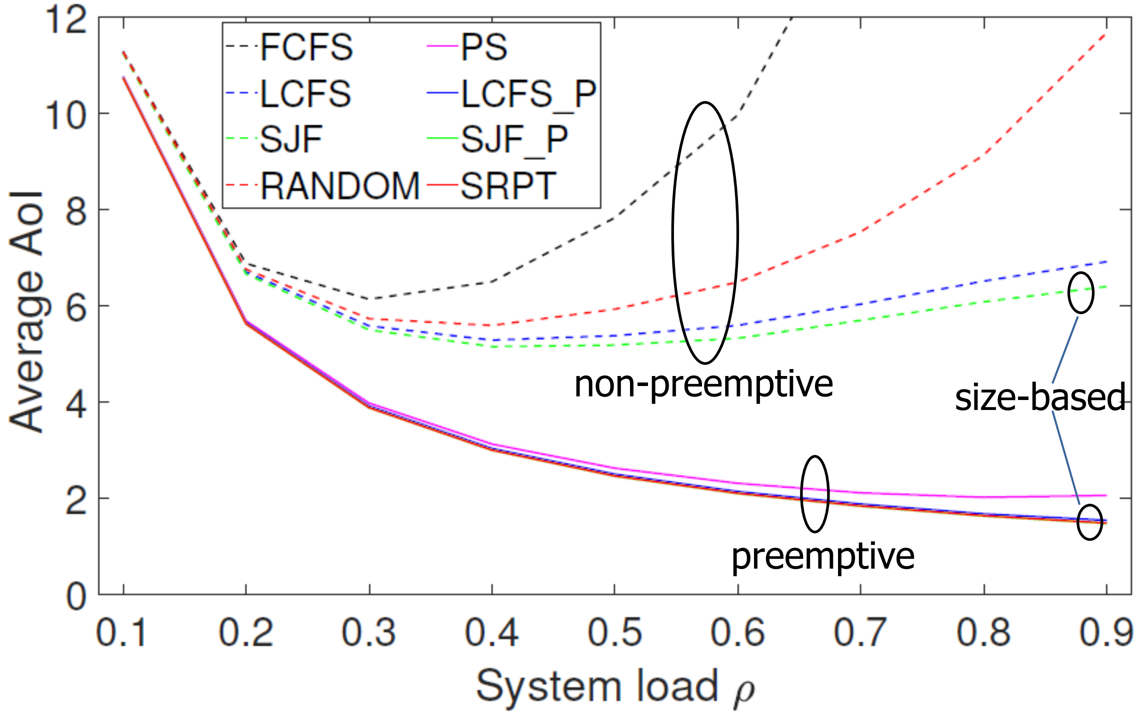


$$\text{Weibull: } \mu = 1 \text{ and } C^2 = \frac{\mathrm{Var}(S)}{\mathrm{E}[S]^2} = 10$$

Observation 1: Size-based policies > Non-size-based policies

Sized-based Policies

- Average AoI performance (with update size info.)

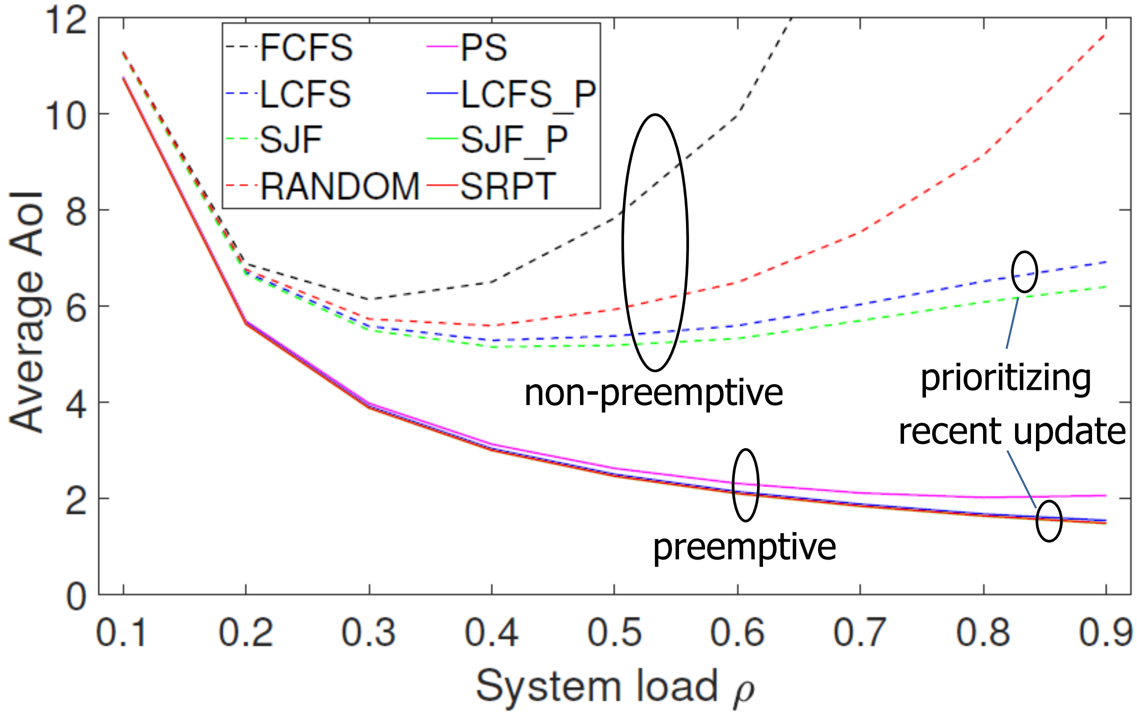


Weibull: $\mu = 1$ and $C^2 = \frac{\mathrm{Var}(S)}{\mathrm{E}[S]^2} = 10$

Guideline 1: Prioritizing updates with small size

Arrival-time-based Policies

- Average AoI performance (**without** update size info.)

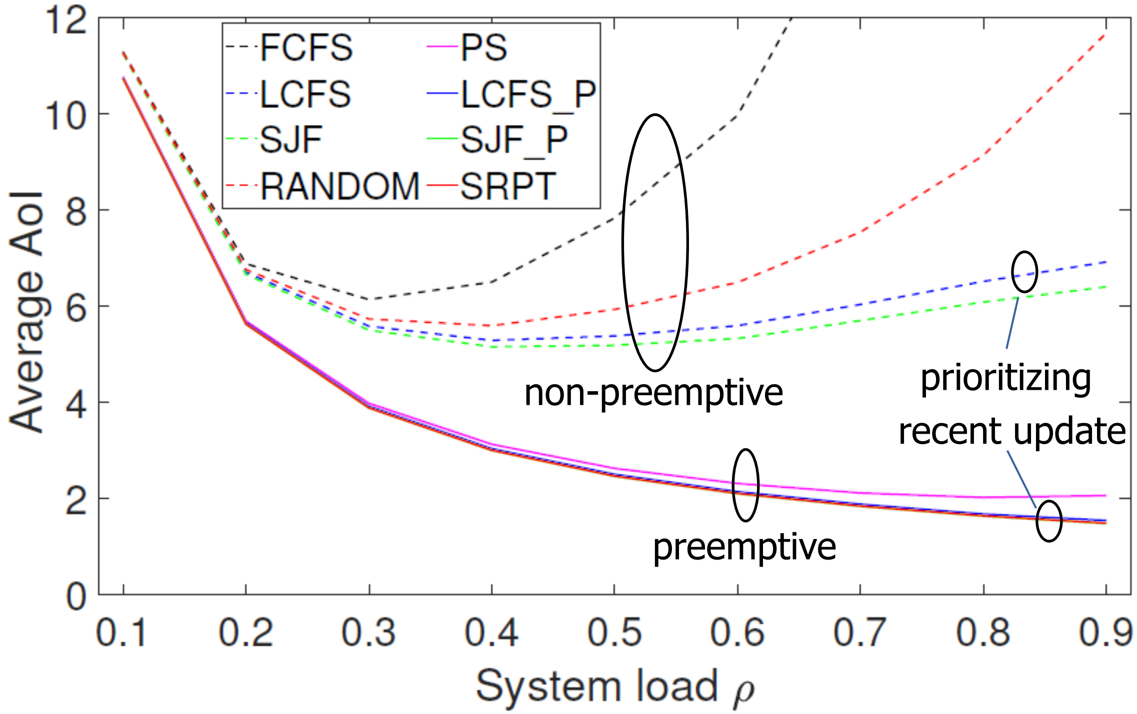


Weibull: $\mu = 1$ and $C^2 = 10$

Observation 2: LCFS>FCFS, RANDOM; LCFS_P>PS

Arrival-time-based Policies

- Average AoI performance (**without** update size info.)



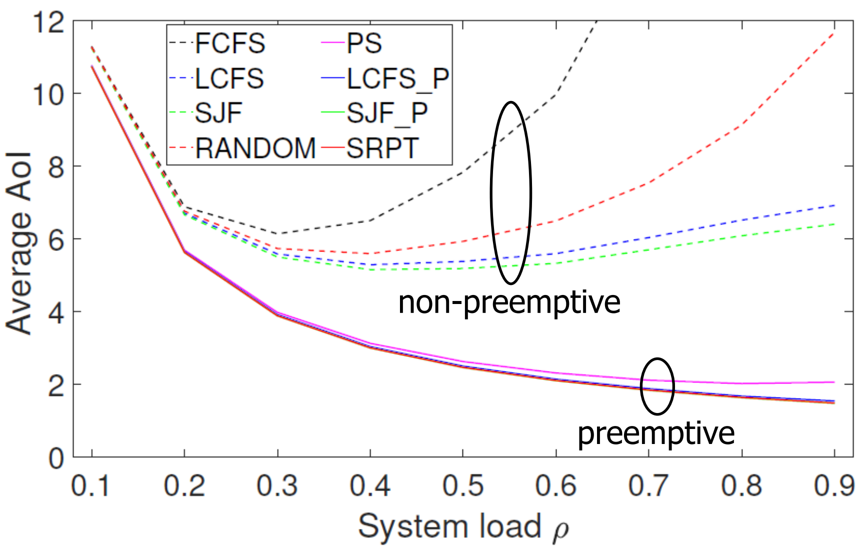
Weibull: $\mu = 1$ and $C^2 = 10$

Guideline 2: Prioritizing recent updates

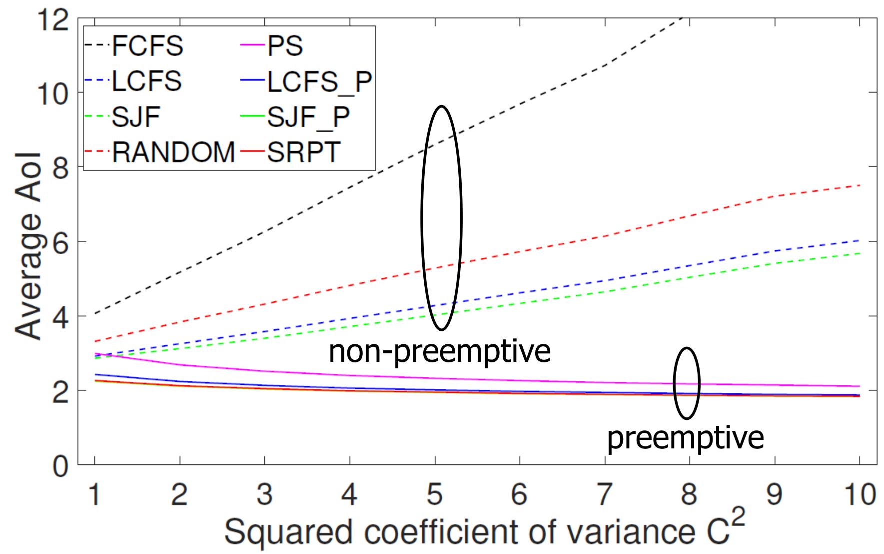
[Kaul et al., 2012; Costa et al., 2016]

Preemptive Policies

- Average AoI performance (with service preemption)



Weibull: $\mu = 1$ and $C^2 = 10$



Weibull: $\mu = 1$ and $\rho = 0.7$

Observation 3:

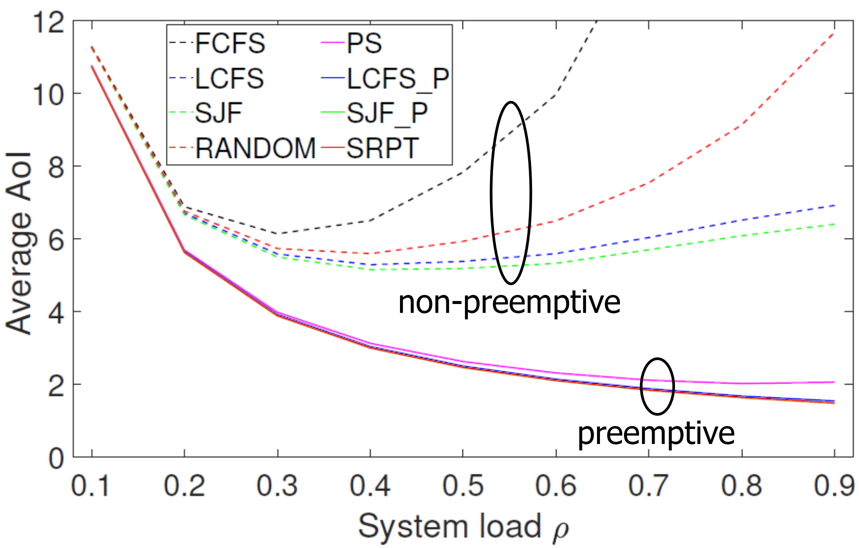
Preemptive>Non-preemption

Observation 4:

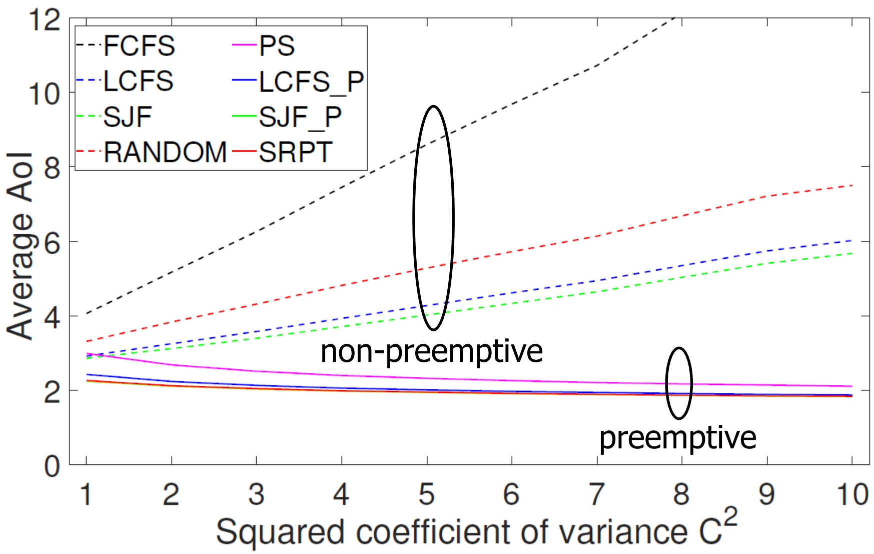
Preemptive policies are less **sensitive** to update size variability

Preemptive Policies

- Average AoI performance (with service preemption)



Weibull: $\mu = 1$ and $C^2 = 10$



Weibull: $\mu = 1$ and $\rho = 0.7$

Guideline 3: Allowing service preemption

[Kaul et al., 2012; Bedeway et al., 2016; Najm et al,. 2018]

Current Guidelines

- Summary of guidelines

	Non-preemptive	Preemptive
Blind to Size	**Guideline 1: Prioritizing updates with small size**	
	Guideline 2: Prioritizing recent updates	
Uses Size	**Guideline 3: Allowing service preemption**	

Current Guidelines

- Summary of guidelines

	Non-preemptive	Preemptive
Blind to Size		
Uses Size		

AoI-based Policies

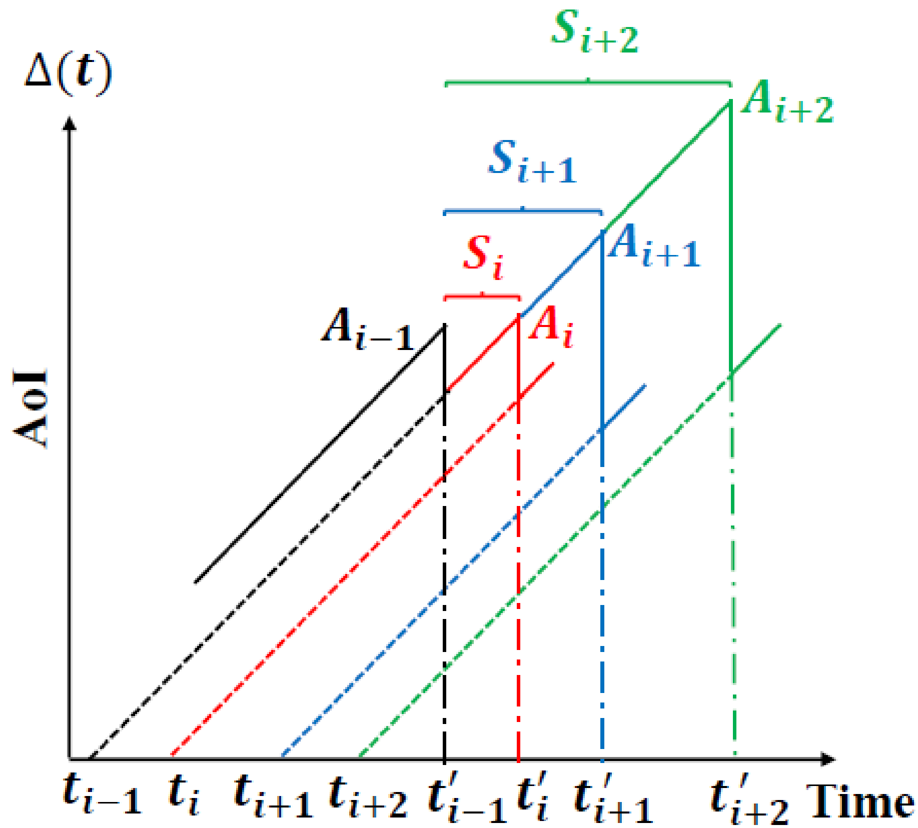
Guideline 1: Prioritizing updates with small size

Guideline 2: Prioritizing recent updates

Guideline 3: Allowing service preemption

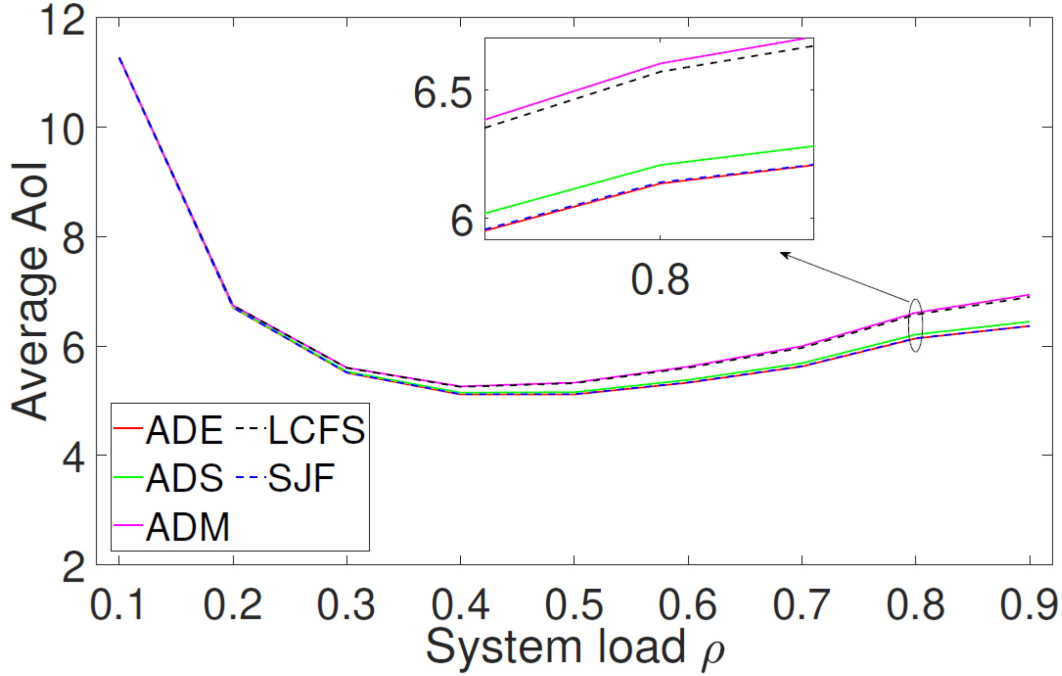
AoI-based Policies

- AoI-based policies
  - AoI-Drop-Earliest (**ADE**): make next AoI drop the **earliest**
  - AoI-Drop-to-Smallest (**ADS**): making the next AoI drop to the **smallest**
  - AoI-Drop-Most (**ADM**): making the next AoI drop **most**



AoI-based Policies

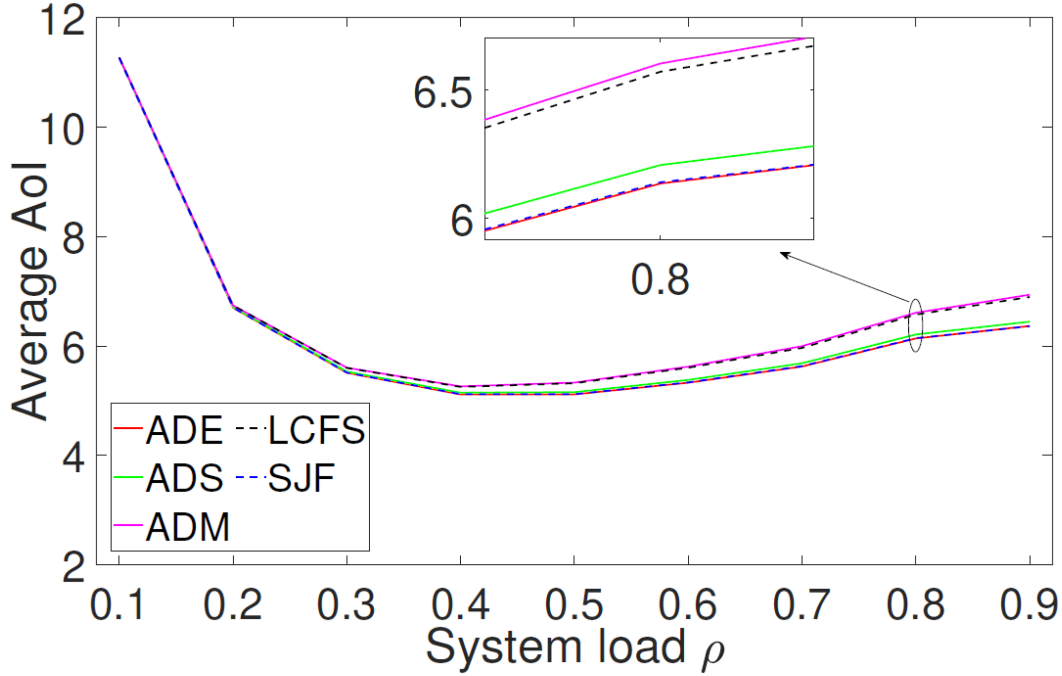
- AoI-based policies vs. non-AoI-based policies



Weibull: $\mu = 1$ and $C^2 = 10$

AoI-based Policies

- AoI-based policies vs. non-AoI-based policies



Weibull: $\mu = 1$ and $C^2 = 10$

Guideline 4: AoI-based policies can further improve the AoI performance

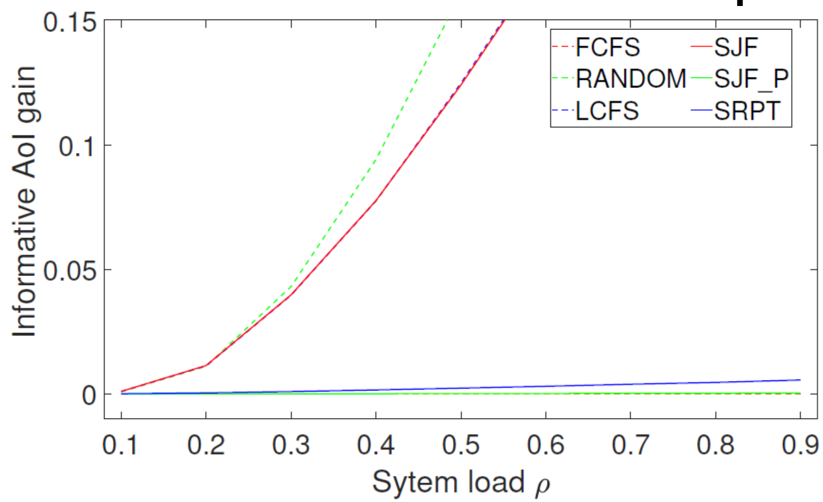
Informative Policies

- Informative policies
  - **Informative policies** only serve informative updates (can make AoI drop)
  - Almost all introduced policies have “informative” version

Informative Policies

- Informative policies
  - **Informative policies** only serve informative updates (can make AoI drop)
  - Almost all introduced policies have “informative” version
- Informative policies vs. Non-informative policies

$$\frac{\Delta_{\text{non-info.}} - \Delta_{\text{info.}}}{\Delta_{\text{non-info.}}} =$$

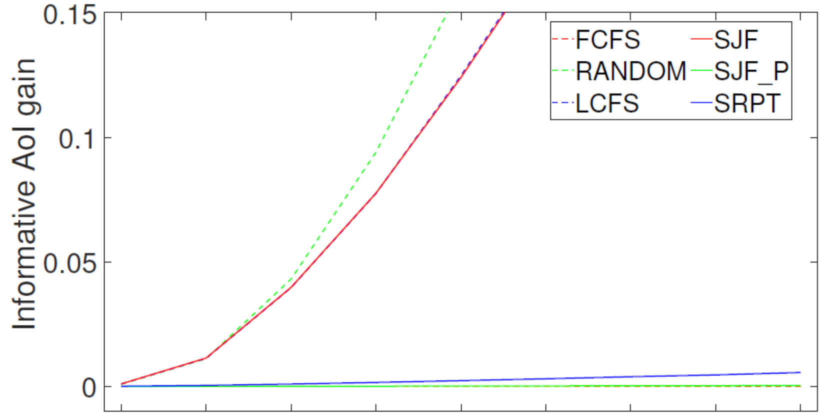


Weibull: $\mu = 1$ and $C^2 = 10$

Informative Policies

- Informative policies
  - **Informative policies** only serve informative updates (can make AoI drop)
  - Almost all introduced policies have “informative” version
- Informative policies vs. Non-informative policies

$$\frac{\Delta_{\text{non-info.}} - \Delta_{\text{info.}}}{\Delta_{\text{non-info.}}} =$$



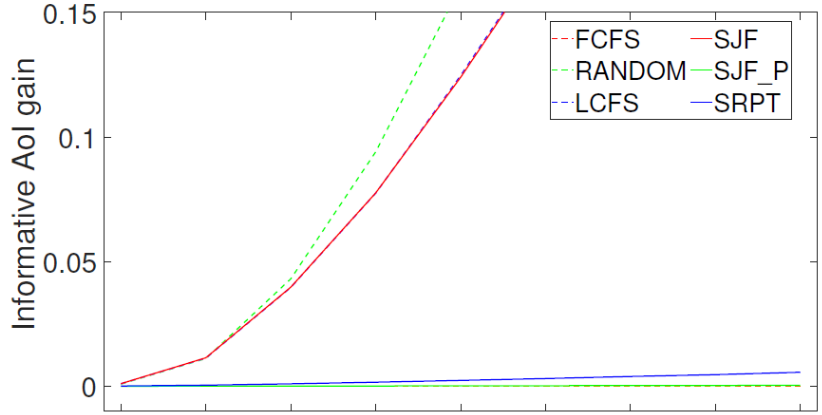
Guideline 5: Prioritizing informative updates

[Costa et al., 2014; Pappas et al., 2015]

Informative Policies

- Informative policies
  - **Informative policies** only serve informative updates (can make AoI drop)
  - Almost all introduced policies have "informative" version
- Informative policies vs. Non-informative policies

$$\frac{\Delta_{\text{non-info.}} - \Delta_{\text{info.}}}{\Delta_{\text{non-info.}}} =$$



Guideline 5: Prioritizing informative updates

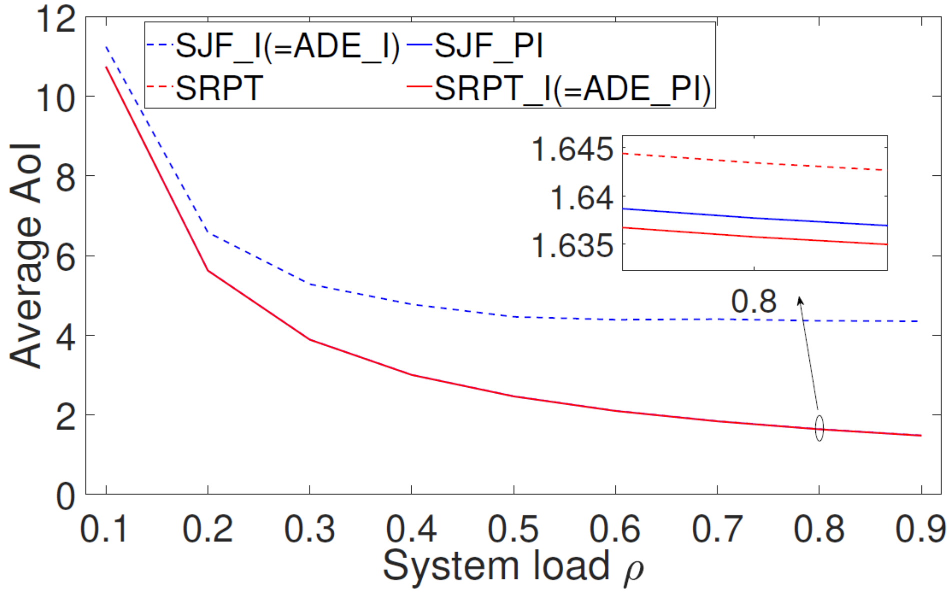
[Costa et al., 2014; Pappas et al., 2015]

- Theoretical result

Proposition 1: In the G/M/1 queueing system, the AoI under LCFS_I is stochastically smaller than that under LCFS.

Preemptive, Informative, AoI-based Policies

- Simulation results



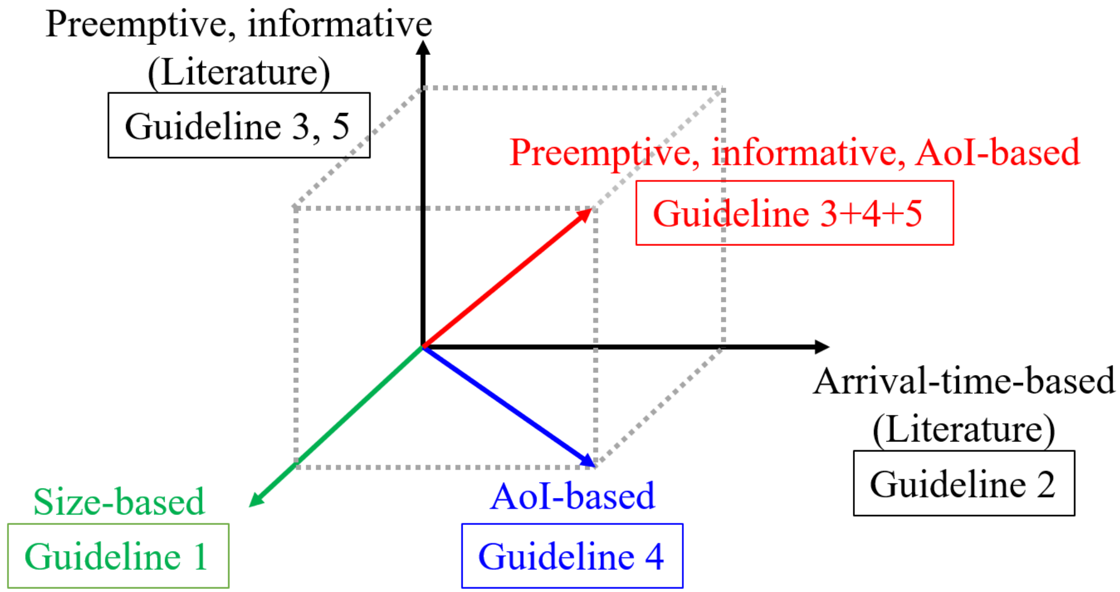
Weibull: $\mu = 1$ and $C^2 = 10$

- Sample path equivalence results

Proposition 2: SRPT_I is equivalent to ADE_PI.

Proposition 3: SJF_I is equivalent to ADE _I.

Conclusion



- Design AoI-efficient scheduling policies
  - Prioritizing small updates, allowing service preemption, prioritizing informative updates
- Equivalence between some size-based and AoI-based policies
  - SRPT_I = ADE_PI; SJF_I=ADE_I
- Delay-efficient → AoI-efficient (for exogenous source)
  - High load: *delay* dominates   Low load: *interarrival* dominates

Future Work

- Pursue more theoretical results
  - Does any informative policy always outperform its non-informative counterpart?
  - Can we derive the closed-form formulas of the average AoI/PAoI for the AoI-efficient scheduling policies (such as SRPT)?

- Apply to more complex network
  - Does our guidelines hold for multi-server queues?
  - What’s the performance of our policies/guidelines in a more complex network (e.g., multi-hop networks)?


Thank you