CSC0056: Data Communication

Lecture 15:
Real-Time Data Communications

Instructor: Chao Wang 王超

Department of Computer Science and Information Engineering

NATIONAL TAIWAN NORMAL UNIVERSITY

Course information

- Instructor: Chao Wang 王超 (https://wangc86.github.io/)
  - Email: cw@ntnu.edu.tw
  - Office: Room 511, Applied Science Building, Gongguan Campus
  - Office hours: Wednesdays and Fridays, 10am-noon, or by appointment
- Course website: https://wangc86.github.io/csc0056/
  - Homework submission: via NTNU Moodle (https://moodle.ntnu.edu.tw/)
- Course meetings: Mondays 9:10-12:10 in C007, Gongguan Campus

Acknowledgement: Some slides' materials in this course are borrowed with permission from the 2014 edition of the course taught by Prof. Yao-Hua Ho 賀耀華

Figures are obtained from the textbook available at http://web.mit.edu/dimitrib/www/datanets.html

Outline of lecture15

- A review of flow control (Section 6.5.1 in particular)
- Timing aspects of data communications
- Real-time data communications
- Case study: Real-time event processing

References for lecture 15

- Section 6.5.1 of the course required textbook (https://wangc86.github.io/csc0056/#resource)
- Industrial Internet Reference Architecture v1.9 (https://www.iiconsortium.org/IIRA.htm)
- Gomaa, Hassan. *Real-Time Software Design for Embedded Systems.* Cambridge University Press, 2016.
- Chao Wang, Christopher Gill, and Chenyang Lu. *Real-Time Middleware for Cyber-Physical Event Processing*. ACM Transactions on Cyber-Physical Systems 3, 3, Article 29 (August 2019) (https://wangc86.github.io/pdf/tcps-cpep.pdf)
- Chao Wang, Christopher Gill, Chenyang Lu. *FRAME: Fault Tolerant and Real-Time Messaging for Edge Computing*. IEEE International Conference on Distributed Computing Systems (ICDCS), 2019. (https://wangc86.github.io/pdf/icdcs19-frame.pdf)

Review of flow control

- Following our discussion in lecture14, networked applications may have requirements in
  - data latency (e.g., online gaming)
  - data rate (e.g., video streaming)
- Flow control is needed to help the network meet those application requirements in various situations:
  - total amount of arriving flows > overall network capacity
  - retransmissions caused by transient congestion (e.g., the Aloha protocol)
  - IoT edge processing
  - internal network control (e.g., data replication for fault tolerance)

Review of flow control (cont.)

- In lecture14, we have discussed
  - window flow control
  - flow rate control (the classic leaky bucket scheme)
  - **flow rate adjustment** (problem transformation to optimal routing)

Flow rate adjustment

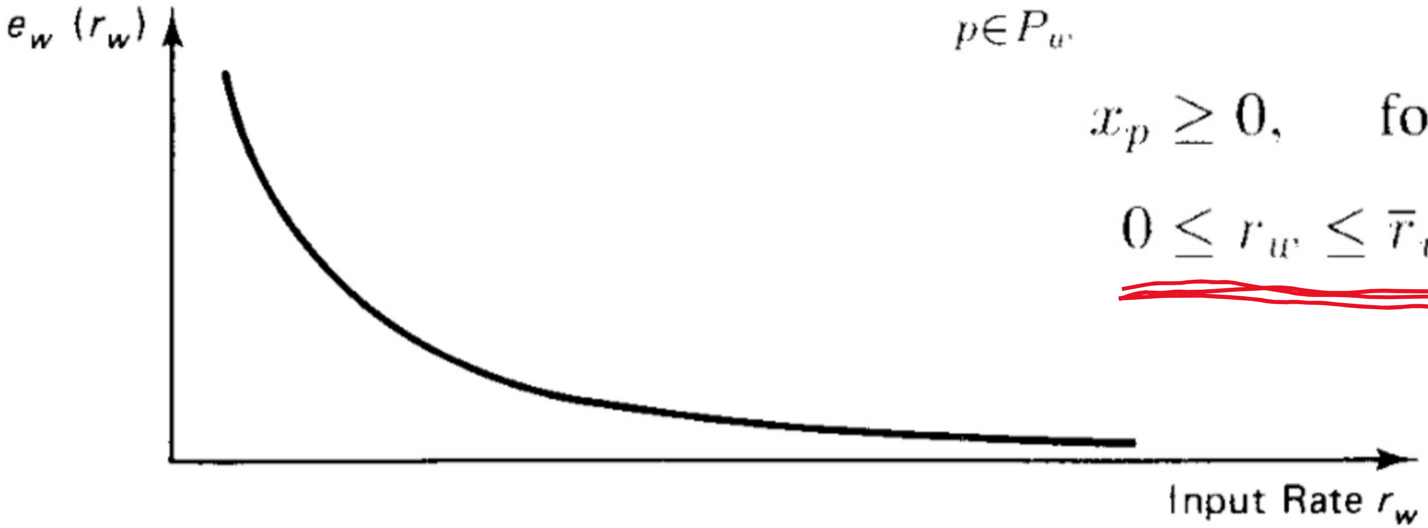
a penalty function to avoid selecting a solution of very small r_w

$$\text{minimize} \sum_{(i,j)} D_{ij}(F_{ij}) + \sum_{w \in W} e_w(r_w)$$

$$\text{subject to} \sum_{p \in P_w} x_p = r_w, \quad \text{for all } w \in W$$

$$x_p \geq 0, \quad \text{for all } p \in P_w,\ w \in W$$

$$0 \leq r_w \leq \bar{r}_w, \quad \text{for all } w \in W$$



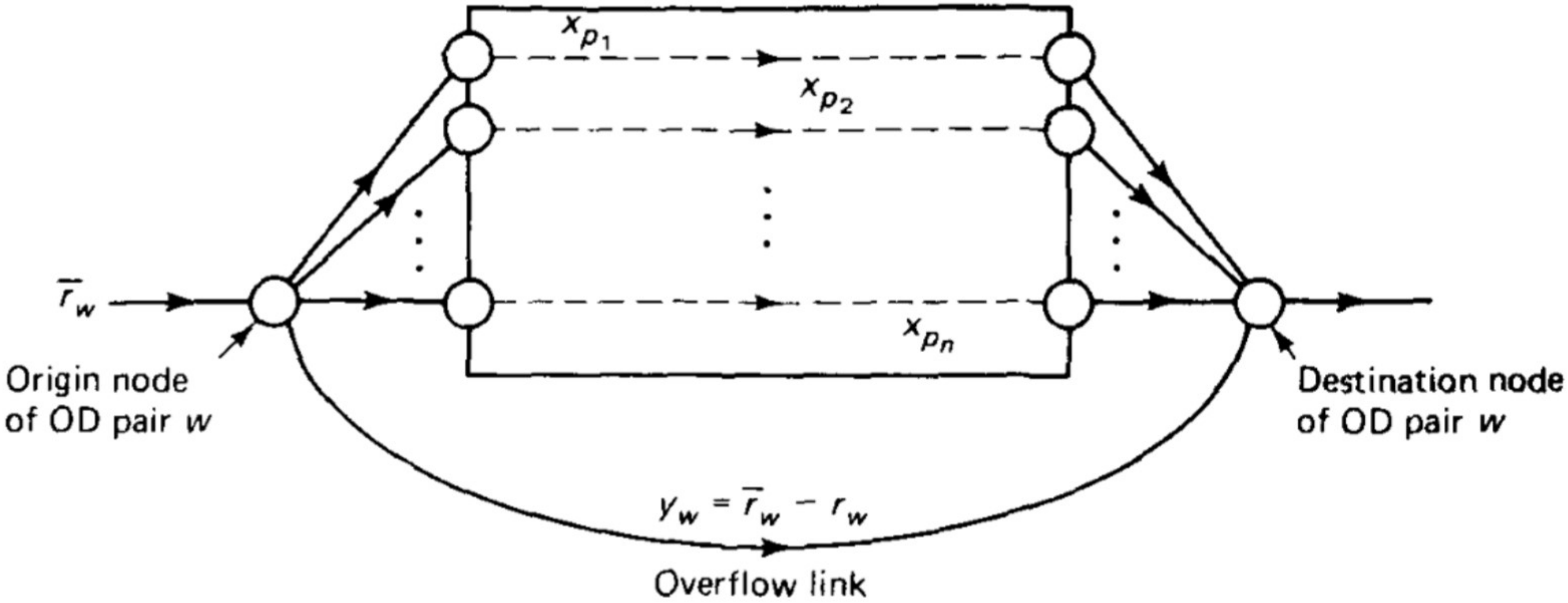
Flow rate adjustment (cont.)

- Use $y_w = \overline{r}_w - r_w$ to represent the the blocked portion of a flow

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)} D_{ij}(F_{ij}) + \sum_{w \in W} E_w(y_w) \\
\text{subject to} \quad & \sum_{p \in P_w} x_p + y_w = \overline{r}_w, && \text{for all } w \in W \\
& x_p \geq 0, && \text{for all } p \in P_w,\ w \in W \\
& y_w \geq 0, && \text{for all } w \in W
\end{aligned}
$$

where $E_w(y_w) = e_w(\overline{r}_w - y_w)$

Equivalence with the optimal routing



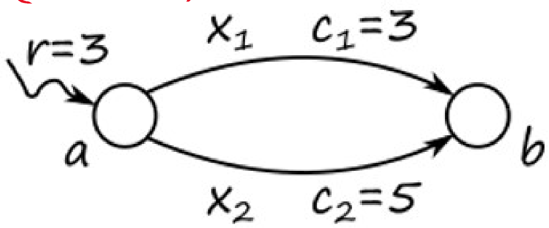
A quick review of optimal routing

$D_1'(x_1) = (c_1 - x_1)^{-2}$ $D_2'(x_2) = (c_2 - x_2)^{-2}$

$D(x) = D_1(x_1) + D_2(x_2)$

$D_1(x_1) = (c_1 - x_1)^{-1}$

$D_2(x_2) = (c_2 - x_2)^{-1}$



Suppose we exclusively use the second path. The optimal condition leads to

$$\frac{1}{(5-r')^2} \le \frac{1}{(3-0)^2} \Rightarrow r' \le 2$$

this means that for r=3 we should not exclusively use the second path.

Now, suppose we use both paths. Then the optimal condition leads to

$$\begin{cases} \frac{1}{(5-r_2)^2} = \frac{1}{(3-r_1)^2} \\ r_1 + r_2 = 3 \end{cases} \Rightarrow \begin{cases} r_1 = 0.5 \\ r_2 = 2.5 \end{cases}$$

We may verify our solution by computing the total cost for each case:

$(x_1, x_2) = (0, 3)$

$\Rightarrow$ total cost $= \frac{1}{3-0} + \frac{1}{5-3}$

$= \frac{25}{30}$

$(x_1, x_2) = (0.5, 2.5)$

$\Rightarrow$ total cost

$= \frac{1}{2.5} + \frac{1}{2.5}$

$= \frac{24}{30} < \frac{25}{30}$ #

The optimality condition for flow control

$$x_p^* > 0 \quad \Rightarrow \quad d_p^* \le d_{p'}^*, \quad \text{for all } p' \in P_w, \text{ and } \boxed{d_p^* \le -e'_w(r_w^*)} \qquad (6.7a)$$

$$r_w^* < \overline{r}_w \quad \Rightarrow \quad -e'_w(r_w^*) \le d_p^*, \quad \text{for all } p \in P_w \qquad (6.7b)$$

where d_p^* is the first derivative length of path p [$d_p^* = \sum_{(i,j)\in p} D'_{ij}(F_{ij}^*)$, and F_{ij}^* is the total flow of link (i,j) corresponding to $\{x_p^*\}$].

That is derived from the following:

$$\underbrace{-e'_w(r_w^*)\cdot\delta}_{\text{the cost pertaining to the overflow link}} - \underbrace{d_p^*\cdot\delta}_{\text{the cost pertaining to path } p} \ge 0$$

this equation represents the change of cost by shifting some flow from path p to the overflow link. Be sure to read the presentation on P452 and P522 for detail account.

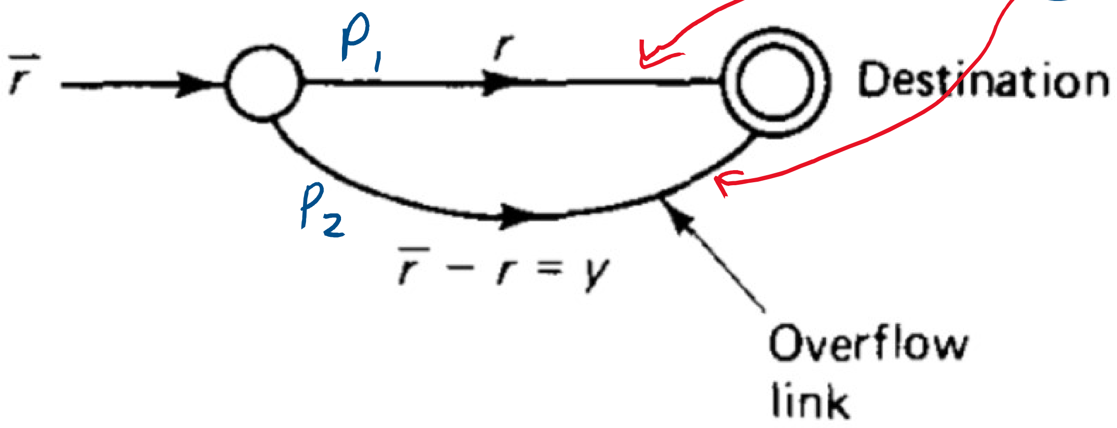
example 6.3 in the textbook.

Example: optimal flow control

- Consider the cost function $\boxed{\frac{r}{C-r} + \frac{a}{r}}$ call it D

$$\frac{\partial}{\partial p_1}(D) = \frac{C}{(C-r)^2}$$

$$\frac{\partial}{\partial p_2}(D) = \frac{-a}{r^2}$$



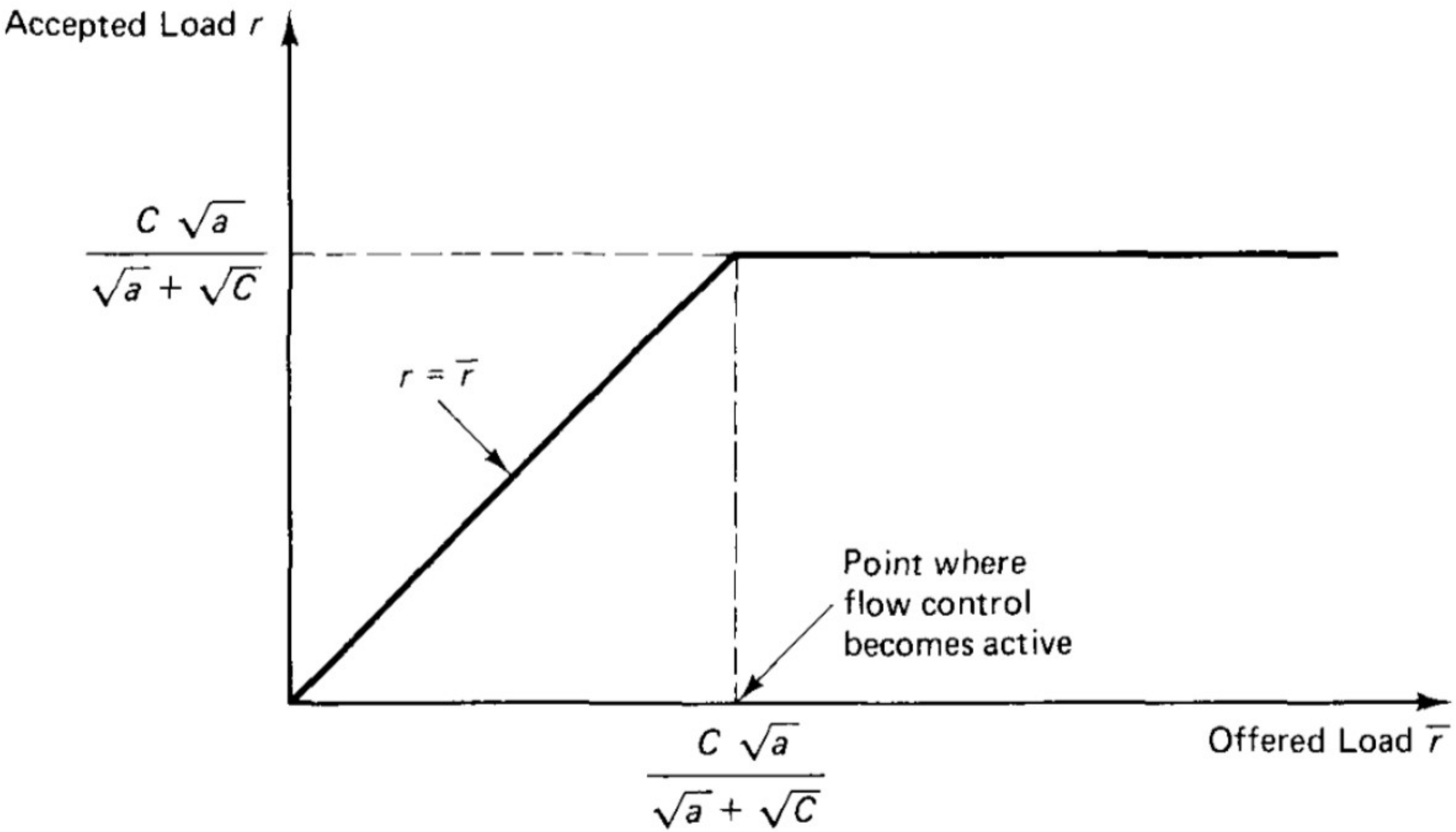
Now, suppose $y = 0$

$\Rightarrow r = \bar{r}$ and according to the optimal condition we have:

$$\frac{C}{(C-\bar{r})^2} \le -\left(\frac{-a}{\bar{r}^2}\right)$$

$$\Rightarrow \bar{r} \le C\,\frac{\sqrt{a}}{\sqrt{a}+\sqrt{C}}$$

Example: optimal flow control (cont.)



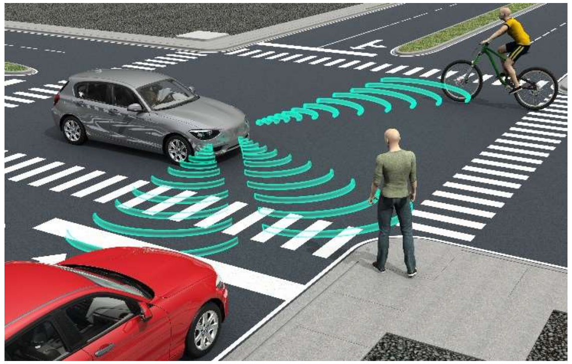
Outline of lecture15

- A review of flow control (Section 6.5.1 in particular)
- **Timing aspects of data communications**
- **Real-time data communications**
- Case study: Real-time event processing

Practical considerations in data communication: timing

- Timing is important!
  - Example 1: stock trading AI
    - Learn from data and make timely trading decisions
  - Example 2: autonomous vehicles
    - Detect pedestrians and prevent accidents

- Low-latency communication
- Real-time communication
  - Given multiple data flows, which one to process first?



https://www.tradingacademy.com/financial-education-center/high-frequency-trading.aspx

https://www.com-magazin.de/news/internet-dinge/neues-gesetz-autonomes-fahren-in-deutschland-1220648.html

Low-latency data communications

- Low latency is an essential feature in many networked applications
- Example: emergency notification
  - Fire
  - Flood
  - ______________

- Example: acute weather prediction and notification
  - Earthquake and tsunami
  - Volcanic eruption
  - Tornado

Low-latency → Real-time

- Conceptually, in data communication, fast enough is good enough
- *Deadline:* a way to specify what we meant by fast enough
- Soft deadlines vs. hard deadlines
  - Missing a soft deadline is not desirable but may be acceptable
  - Missing a hard deadline will lead to disastrous consequences
- Soft/hard real-time system: a system that meets soft/hard deadlines

Examples of soft/hard deadlines:

Soft deadlines	Hard deadlines

Real-time data communications

- For networked applications, people often specify *end-to-end* deadline for data communications
  - From "data created by a sensor" (one end)
    to "data received by an application" (the other end)

- From one end of the system to the other end:
  - Sender
  - Link(s) between sender and intermediary
  - Intermediary (messaging broker/event service/edge computing)
  - Link(s) between intermediary and receiver
  - Receiver

Real-time data communications

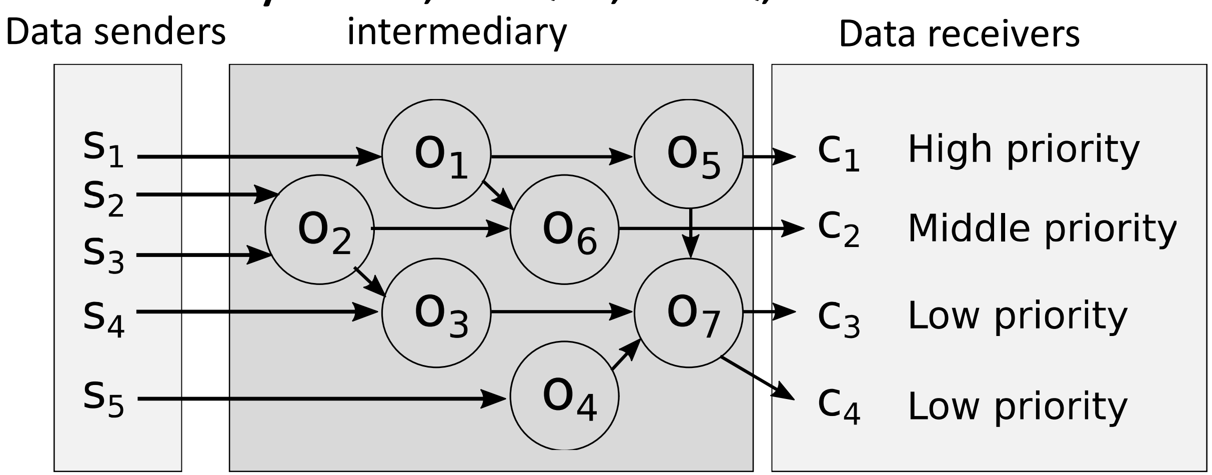
- For networked applications, people often specify *end-to-end* deadline for data communications
  - From “data created by a sensor” (one end) to “data received by an application” (the other end)

- From one end of the system to the other end:
  - ~~Sender~~
  - ~~Link(s) between sender and intermedi~~
  - **Intermediary**
  - ~~Link(s) between intermediary and receiv~~
  - ~~Receiver~~

We may assume that links are reliable and have bounded latency (using what we have learned in this course, for example).

Data communication intermediaries

- Purposes:
  - Decoupling senders and receivers
  - Simplifying senders and receivers
- Example intermediary: TAO, MQTT, NSQ, …



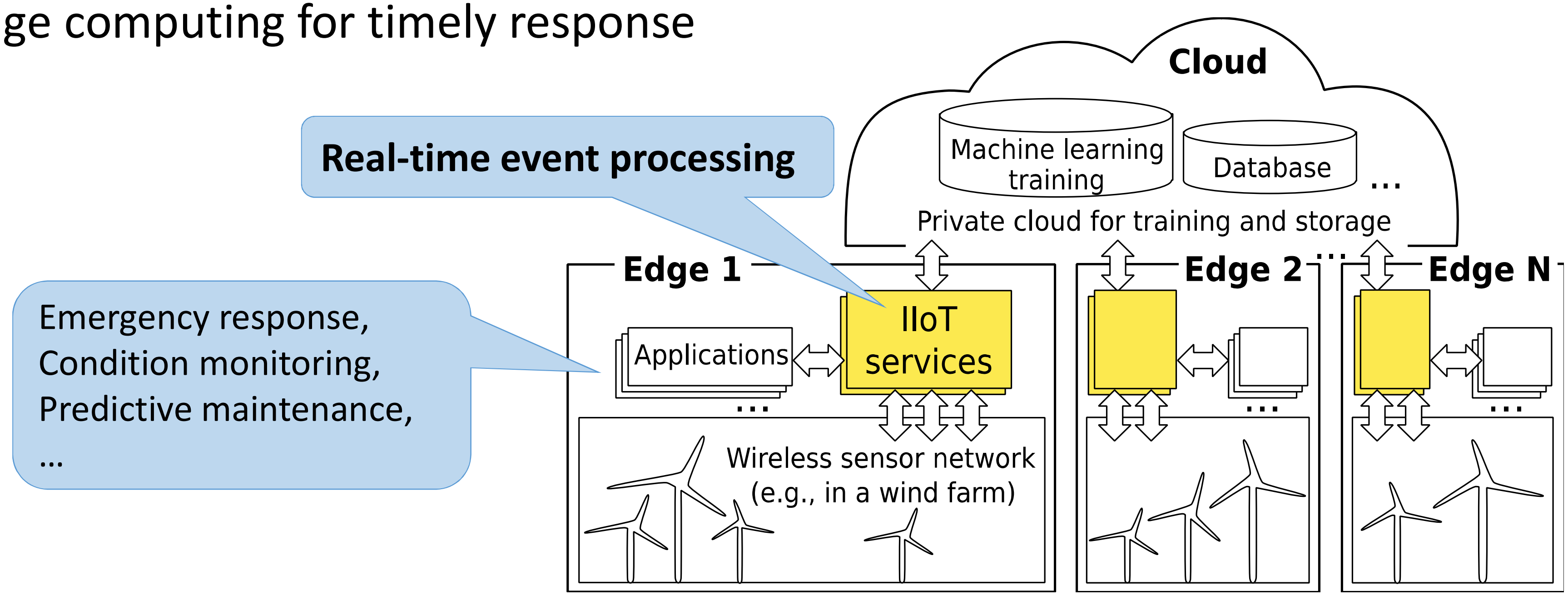
O_i: operations (filtering, transformation, encryption, …)

Outline of lecture15

- A review of flow control (Section 6.5.1 in particular)
- Timing aspects of data communications
- Real-time data communications
- **Case study: Real-time event processing**

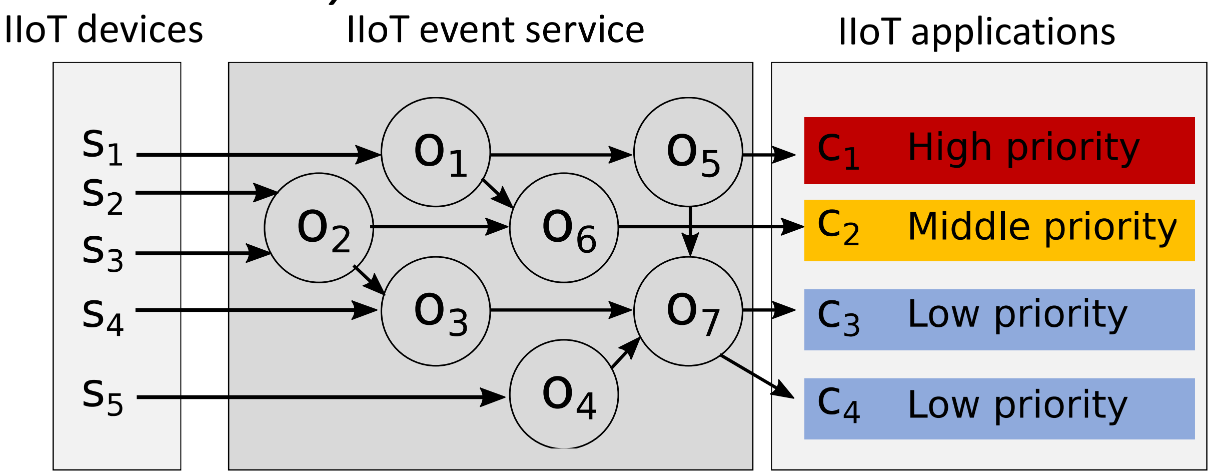
Industrial Internet of Things (IIoT)

- Synergizing sensing, analytics, and control
  - Cloud computing for high capacity
  - Edge computing for timely response



A model for event processing

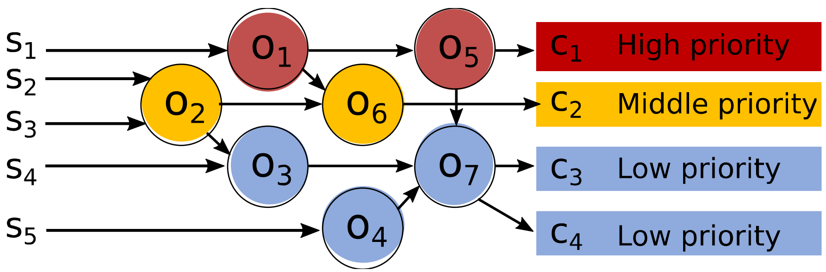
- Latency requirements
- Temporal semantics:
  - *Absolute time consistency* on an event's elapse time since creation
  - *Relative time consistency* on the difference between event's creation time



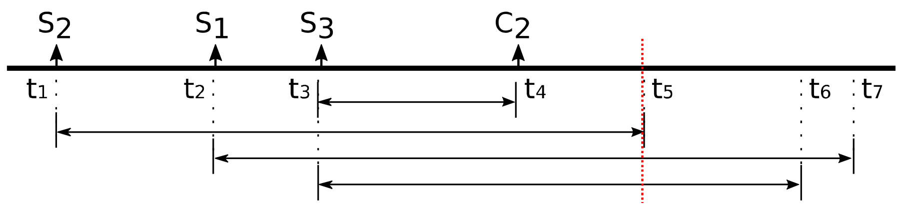
O_i: operations (filtering, transformation, encryption, …)

Real-time cyber-physical event processing

- Processing in the order of priorities propagated from application:

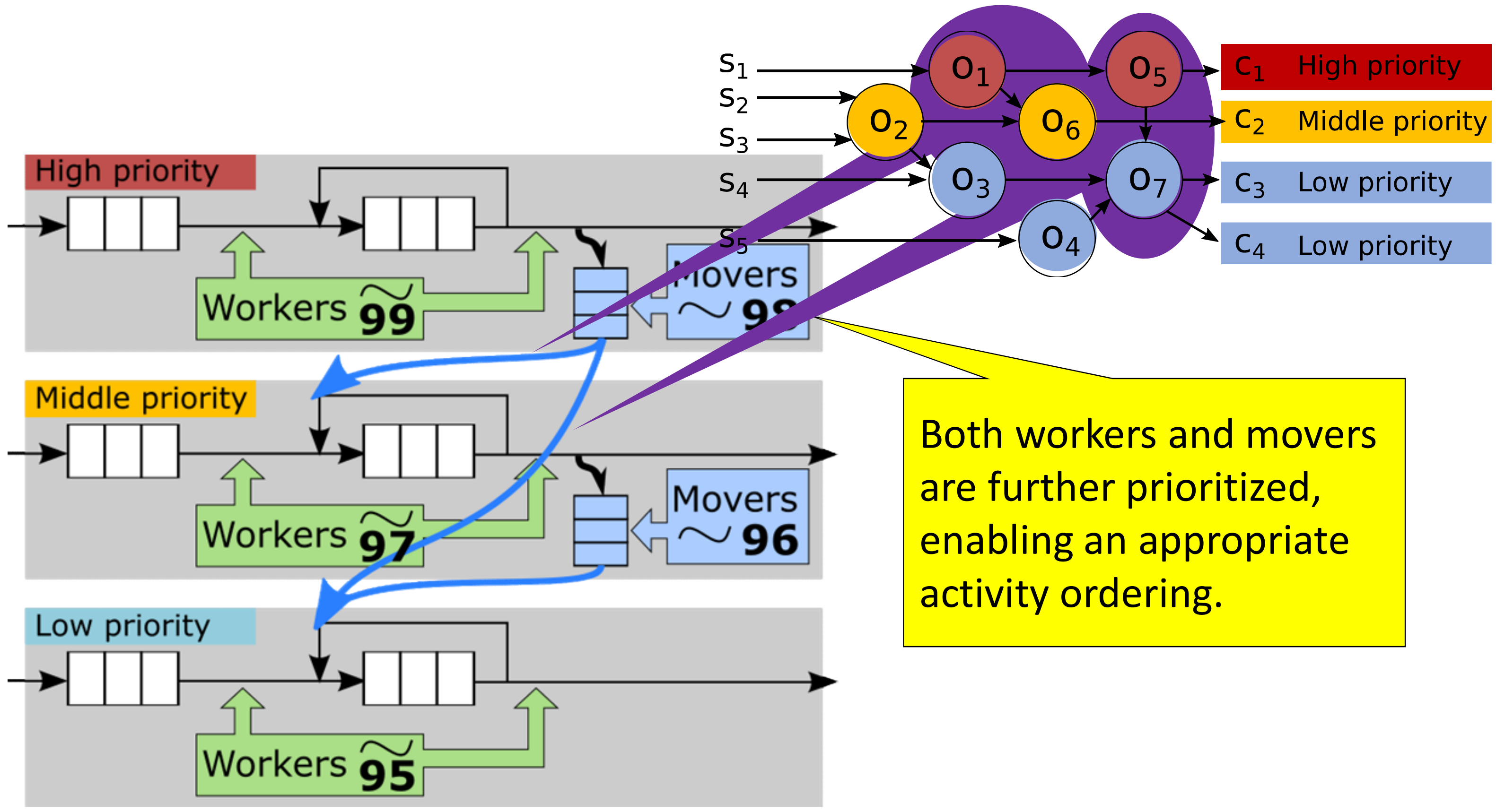


- Temporal semantics enforcement
  - Absolute time consistency



  - Relative time consistency: track both the earliest and the latest event creations per operator

Processing architecture



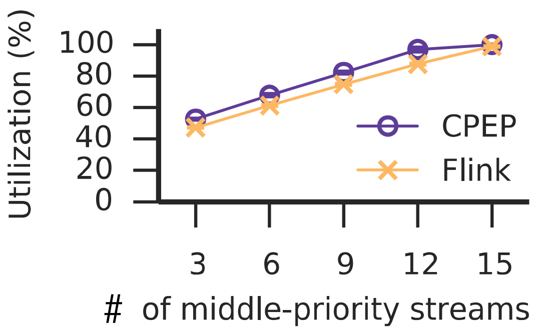
Both workers and movers are further prioritized, enabling an appropriate activity ordering.

Latency performance

99th percentile latency (unit: ms)

Priority	Service	Number of middle-priority streams				
		3	6	9	12	15
High	Flink	3.8 ± 0.1	5.9 ± 0.2	12.6 ± 0.4	52.6 ± 4.1	448.9 ± 171.7
	CPEP	0.8 ± 0.0	0.7 ± 0.0	0.7 ± 0.0	0.7 ± 0.0	0.7 ± 0.0
Middle	Flink	4.5 ± 0.1	6.4 ± 0.2	11.3 ± 0.4	28.9 ± 0.5	107.9 ± 18.1
	CPEP	1.6 ± 0.0	1.8 ± 0.0	2.2 ± 0.0	2.5 ± 0.0	3.0 ± 0.1
Low	Flink	5.2 ± 0.3	7.4 ± 0.2	15.5 ± 0.6	43.3 ± 1.3	679.8 ± 274.0
	CPEP	3.7 ± 0.3	4.8 ± 0.2	6.8 ± 0.6	10.6 ± 1.0	33.4 ± 0.2

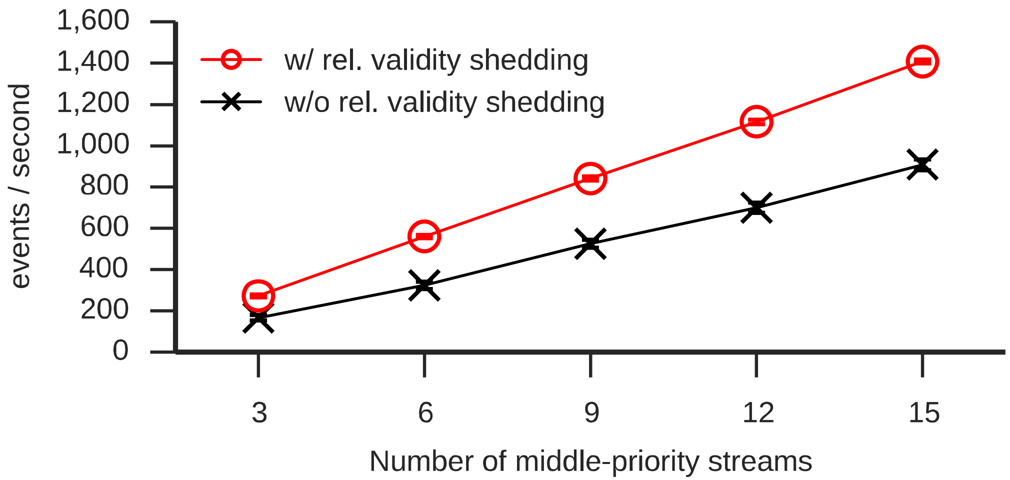
CPEP maintained high-priority latency performance as workload increased.



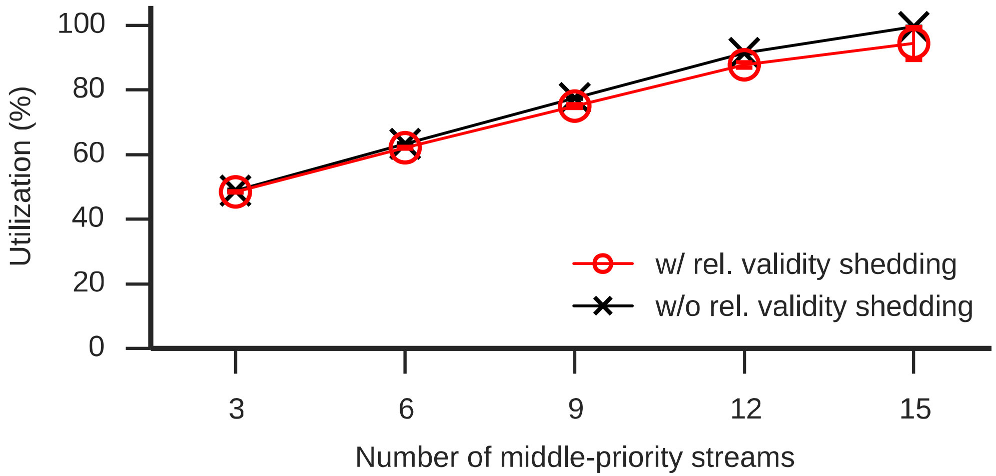
CPEP differentiated latency according to priority level.

Benefits of shedding inconsistent events

Improve the throughput of consistent events.



Save CPU utilization.

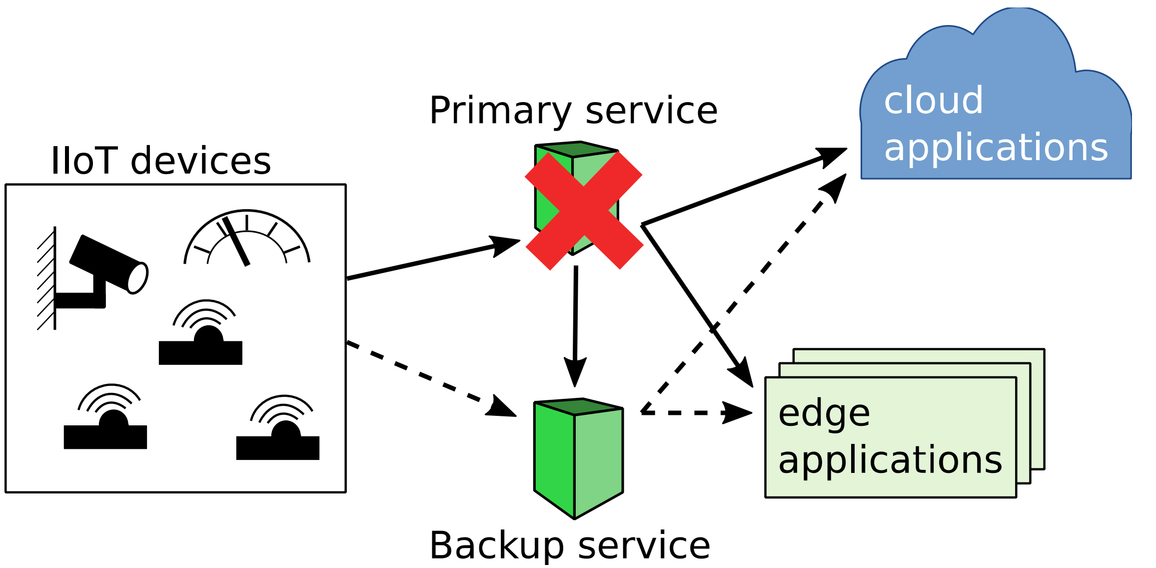


Dependability issues in computing systems

- What if some of the network components may fail to work?
- How to keep applications running properly while fixing the underlying network problems?
- Will fault-tolerance affect the performance of a data network? In which ways? How to amend it?

Loss tolerance

- An IoT service must deliver messages reliably, but
  - fault-tolerant systems can be slow or costly
  - heterogeneous traffic and platforms can increase pessimism



Message loss-tolerance requirement

- Application-specific requirements to an IoT service
  - L_i : the tolerable number of consecutive losses for topic *i*

Value of L_i	**Application examples**
0	emergency response; predictive maintenance
$k > 0$	condition monitoring

(Within the tolerable number, applications may use estimates for the missing data.)

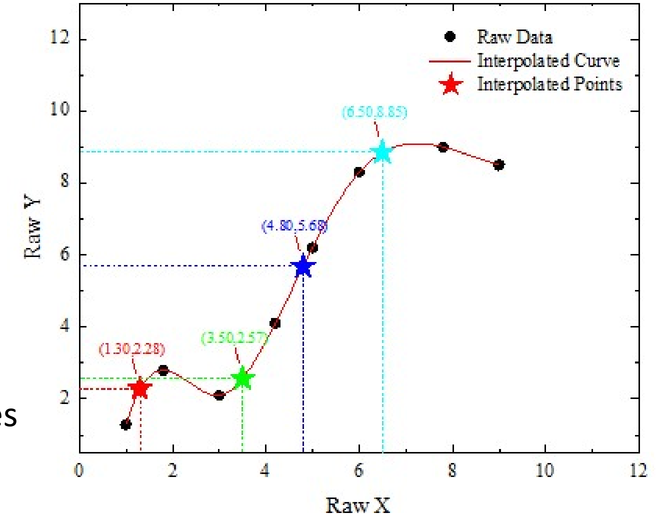


Image source: https://www.originlab.com/doc/Origin-Help/Math-Inter-Extrapoltate-YfromX