CSC0056: Data Communication

Lecture 16: Time Synchronization and Timely Data Replication

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

Outline of lecture16

- Time synchronization among communication hosts
  - NTP and PTP
- Fault tolerance via data replication
  - Active replication vs. passive replication
- Case study: Fault-tolerant and real-time messaging for edge computing

References for lecture 16

- Time synchronization
  - Mills, et al. Network Time Protocol Version 4: Protocol and Algorithms Specification. RFC 5905. (https://tools.ietf.org/html/rfc5905)
  - Mills, D.L., "Computer Network Time Synchronization - the Network Time Protocol", CRC Press, 304 pp, 2006.
  - IEEE. 2008. IEEE standard for a precision clock synchronization protocol for networked measurement and control systems. IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002) (July 2008), 1–300.
- Case study
  - 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)

The need for time synchronization

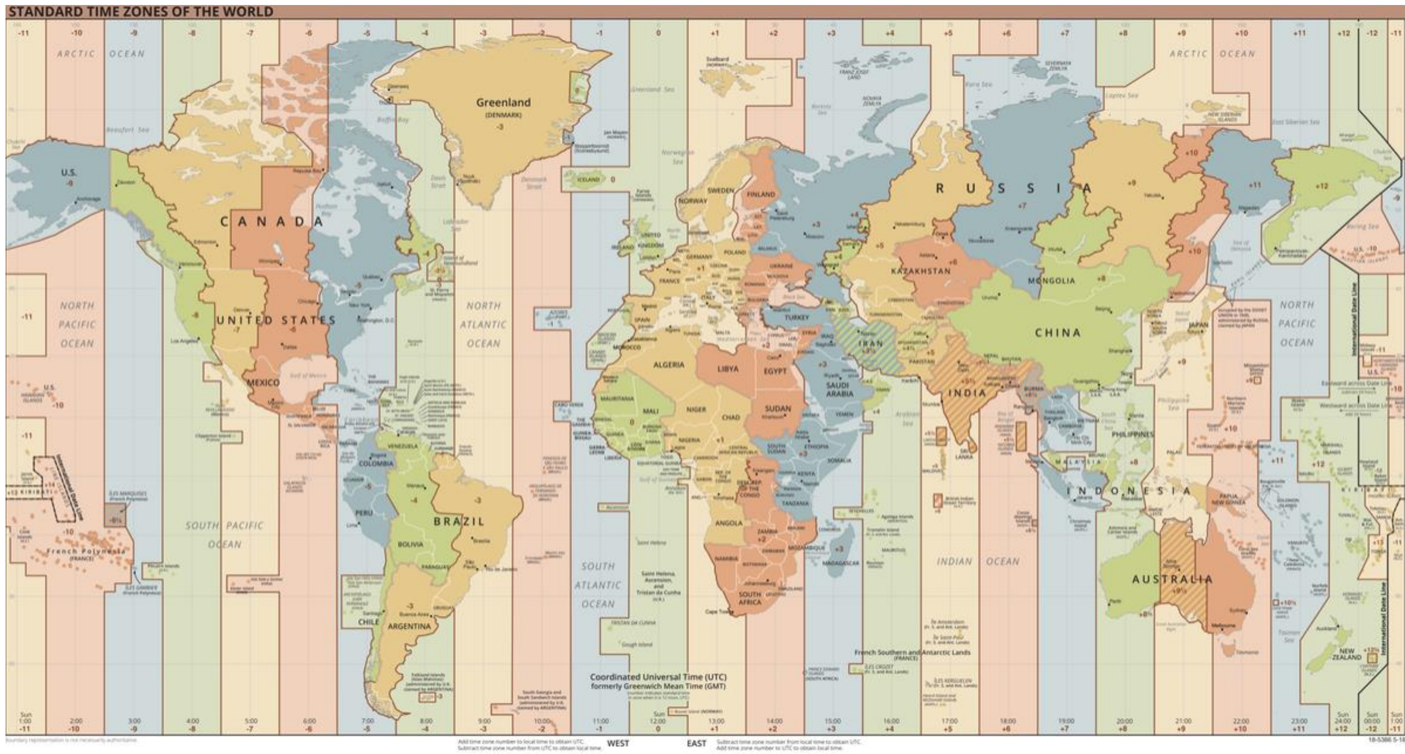
- Some embedded devices simply do not have battery powered hardware clock

- In many applications, to measure end-to-end latency performance, both ends must have synchronized clocks

NTP: network time protocol

- Goal
  - "minimize both the time difference and frequency difference between UTC and the system clock. When these differences have been reduced below nominal tolerances, the system clock is said to be synchronized to UTC."

  UTC:
    - Coordinated Universal Time
    - Temps Universel Coordonné



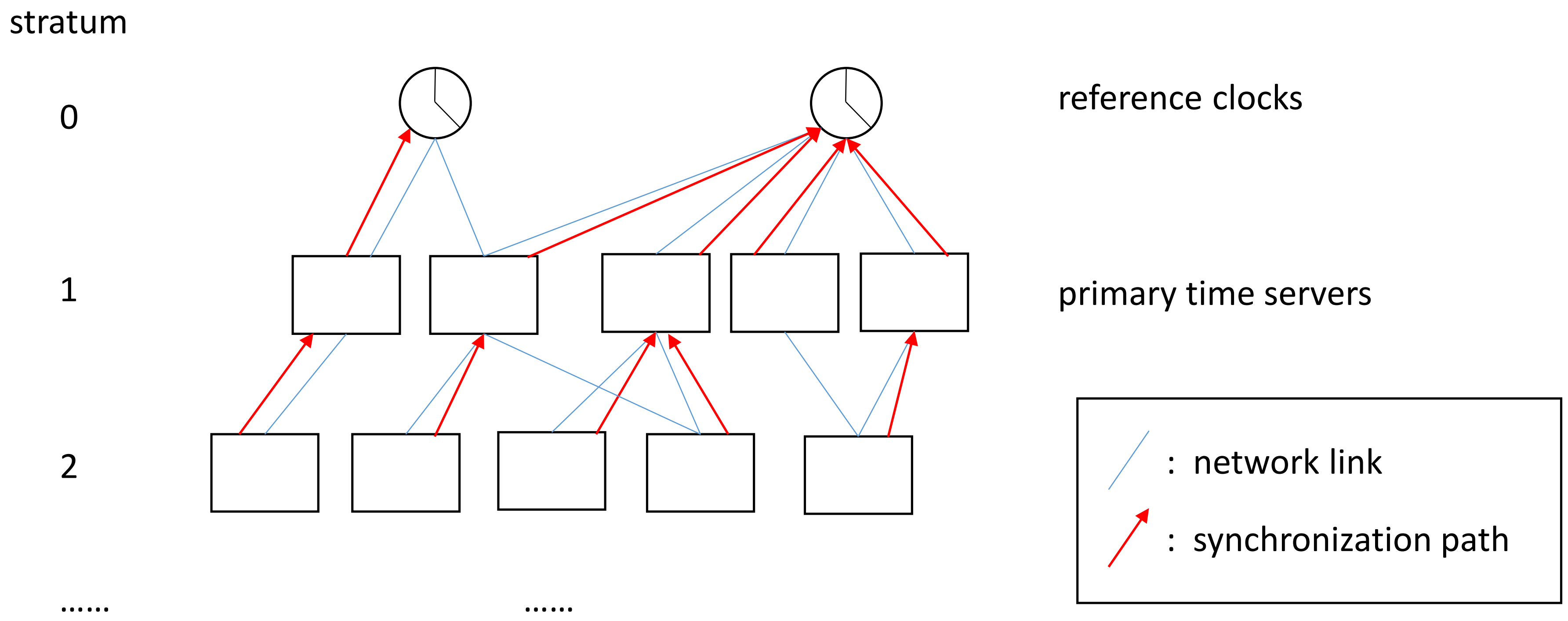
https://commons.wikimedia.org/wiki/File:World_Time_Zones_Map.png

NTP accuracy

- Conventionally, can achieve milliseconds accuracy
  - With improvement, may achieve an accuracy up to tens of microseconds
- Primary time servers
  - Synced to national standards by wire or radio
  - Accuracy: tens of microseconds
- Secondary time servers
  - Synced to primary time servers
  - Accuracy: a few hundred microseconds to a few tens of milliseconds
- Clock strata

NTP topology example

- Synchronizing clocks along the synchronization paths

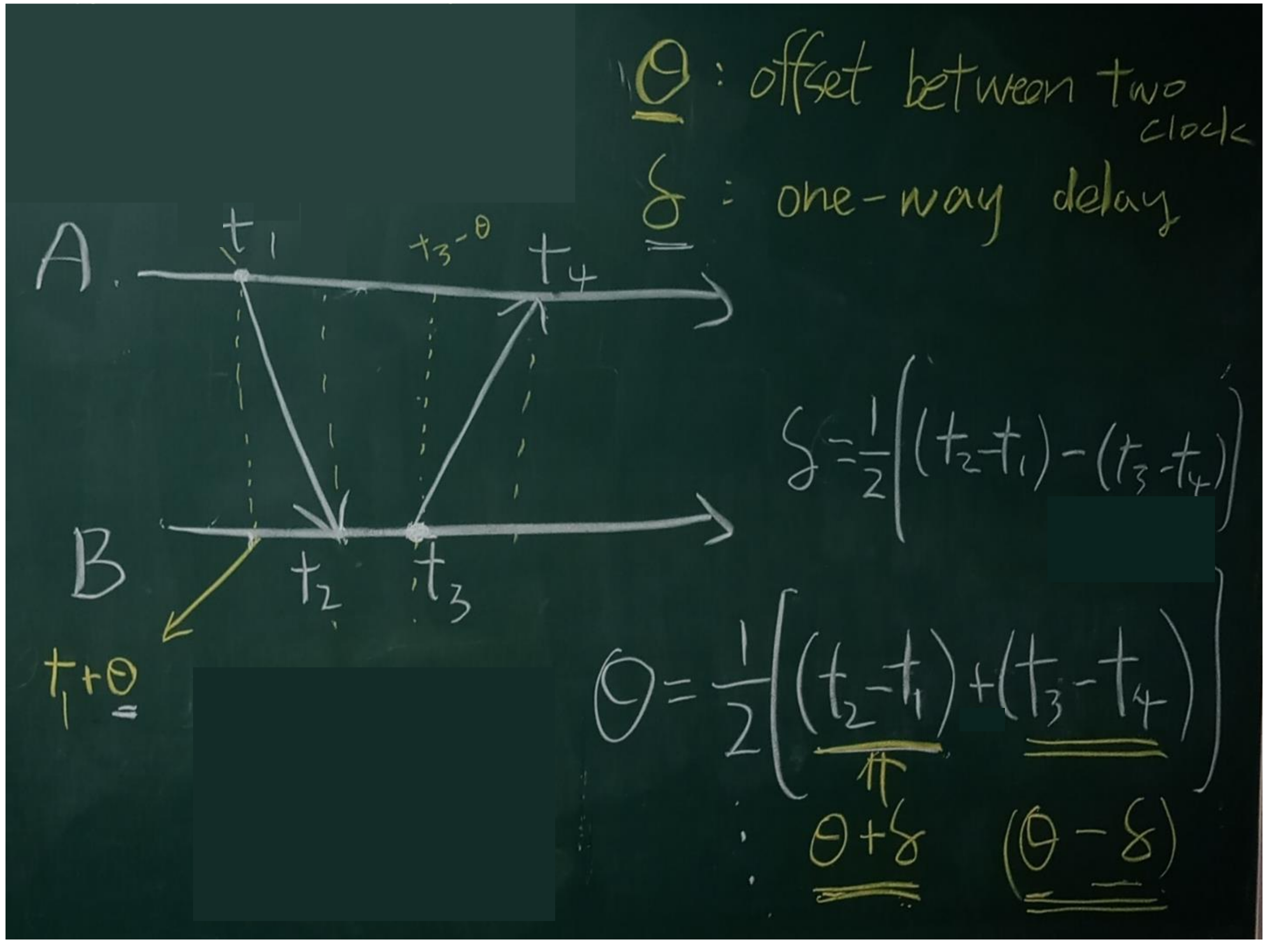


NTP topology (cont.)

- Master-slave subnetwork with synchronization paths determined by a minimum spanning tree (lecture 12)
- “As a standard practice, timing network topology should be organized to avoid timing loops and minimize the synchronization distance. In NTP, the subnet topology is determined using a variant of the Bellman-Ford distributed routing algorithm, which computes the shortest-path spanning tree rooted on the primary servers. As a result of this design, the algorithm automatically reorganizes the subnet, so as to produce the most accurate and reliable time, even when there are failures in the timing network.” - RFC 5905

NTP synchronization basics

- Suppose that host A is going to synchronize its clock to that of host B
- The mean offset between two clocks is $[(t_2-t_1)+(t_3-t_4)]/2$
- Synchronization is performed by gradually reducing the mean offset
  - Here we omit the details



θ : offset between two clock
δ : one-way delay
A
B
t_1
$t_3-\theta$
t_4
t_2
t_3
$t_1+\theta$
$\delta=\frac{1}{2}\left((t_2-t_1)-(t_3-t_4)\right)$
$\theta=\frac{1}{2}\left((t_2-t_1)+(t_3-t_4)\right)$
$\theta+\delta$
$(\theta-\delta)$

NTP: network time protocol

- "Reliable message delivery such as TCP can actually make the delivered NTP packet less reliable since retries would increase the delay value and other errors." - RFC 5905
- If the network is very busy, or the server's CPU is very busy, would that affect the performance of NTP?
  - Ans: Potentially yes, because synchronization is based on message-exchange, and software timestamping may be delayed due to preemption
  - To address those problems:
    - Network: out-of-band synchronization
    - CPU: pinned tasks to certain core(s)

PTP: precision time protocol

- Designed to achieve microsecond to sub-microsecond accuracy and precision
- Spatially localized
- Administration free
- Accessible for both high-end devices and low-cost, low-end devices

PTP topology

- Synchronization is performed by syncing each slave clock to its corresponding master clock (master-slave relationship is defined per link)

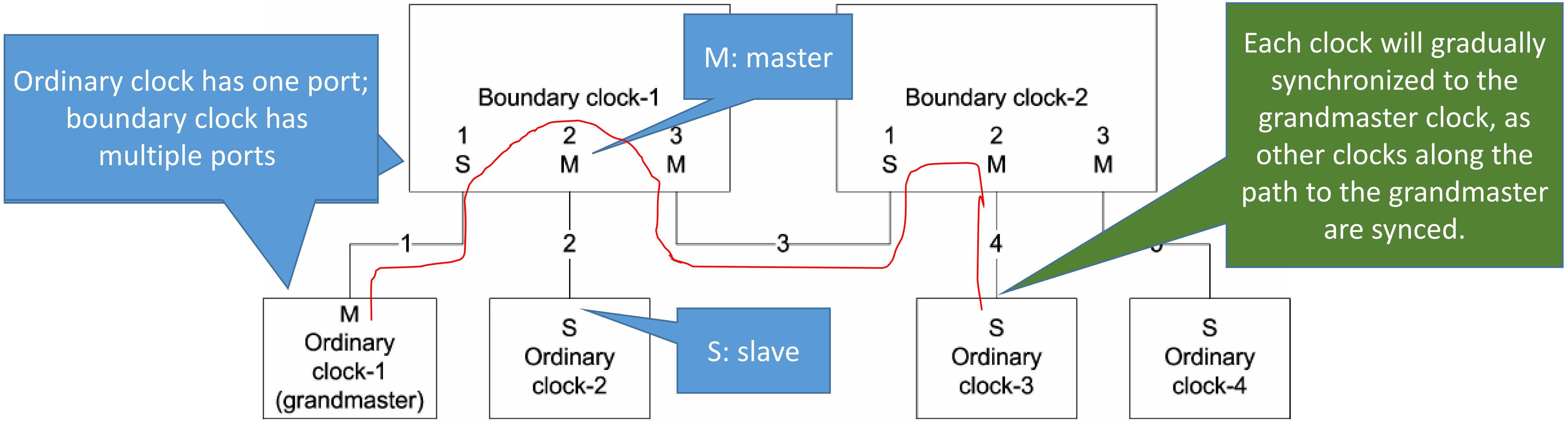
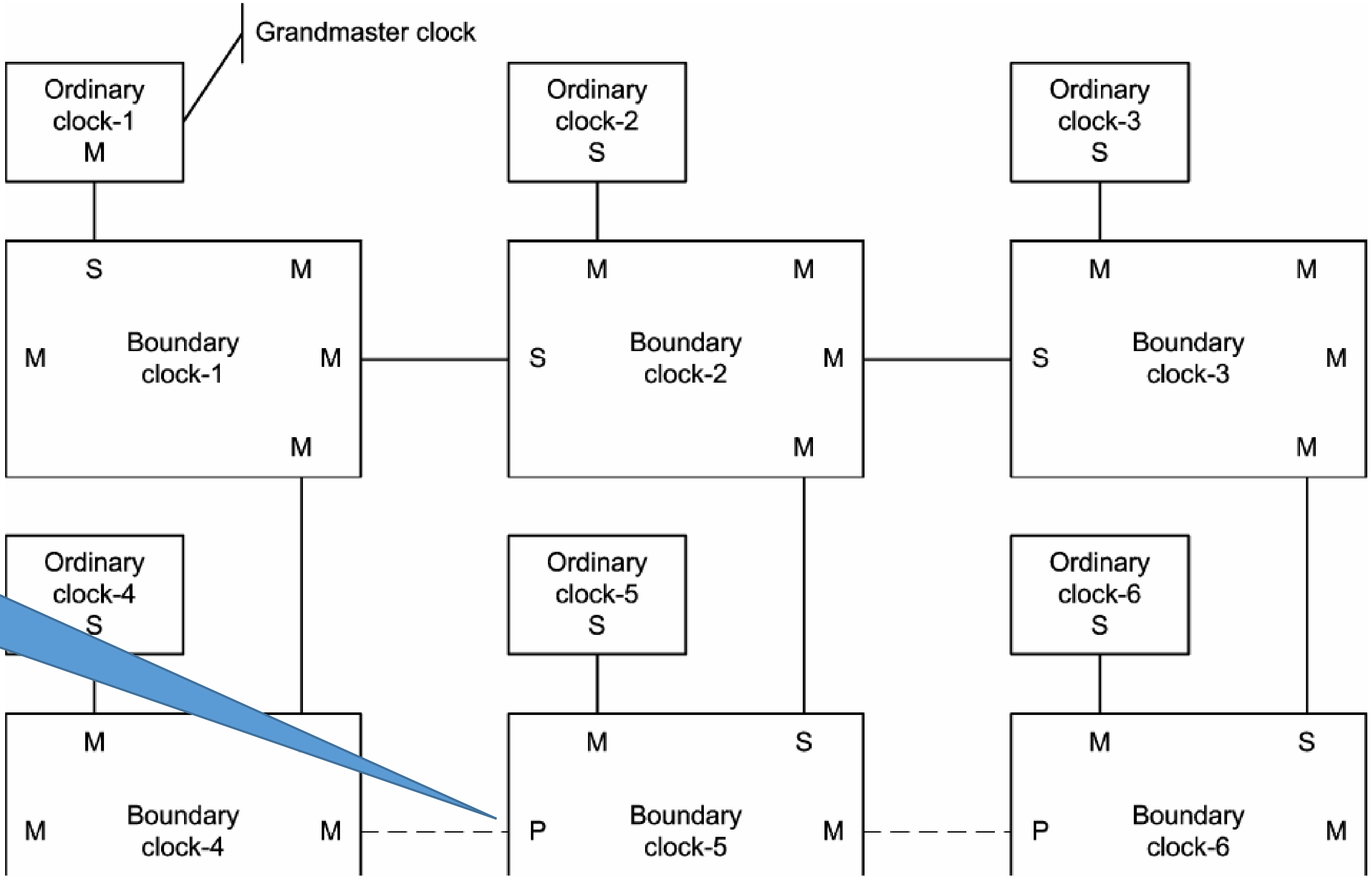

Ordinary clock has one port; boundary clock has multiple ports
M: master
Each clock will gradually synchronized to the grandmaster clock, as other clocks along the path to the grandmaster are synced.
Boundary clock-1
1 S
2 M
3 M
Boundary clock-2
1 S
2 M
3 M
1
2
3
4
5
M Ordinary clock-1 (grandmaster)
S Ordinary clock-2
S: slave
S Ordinary clock-3
S Ordinary clock-4

Figure 10—Simple master–slave clock hierarchy

PTP topology (cont.)



Part of **Figure 11—Pruned mesh topology**

Some ports are passive (denoted by S), to prevent loops in the PTP topology

Basic synchronization method

- termed "delay request-response"
- *Mean* path delay
  $= [(t_2-t_1)+(t_4-t_3)]/2$
- *Mean* offset from master
  $= [(t_2-t_1)+(t_3-t_4)]/2$

used for synchronization

Offset from Master time
= time on the slave clock - time on the master clock
= t_2 - t_1 - mean path delay (- correction)

Optional, by the use of transparent clock (next slides)

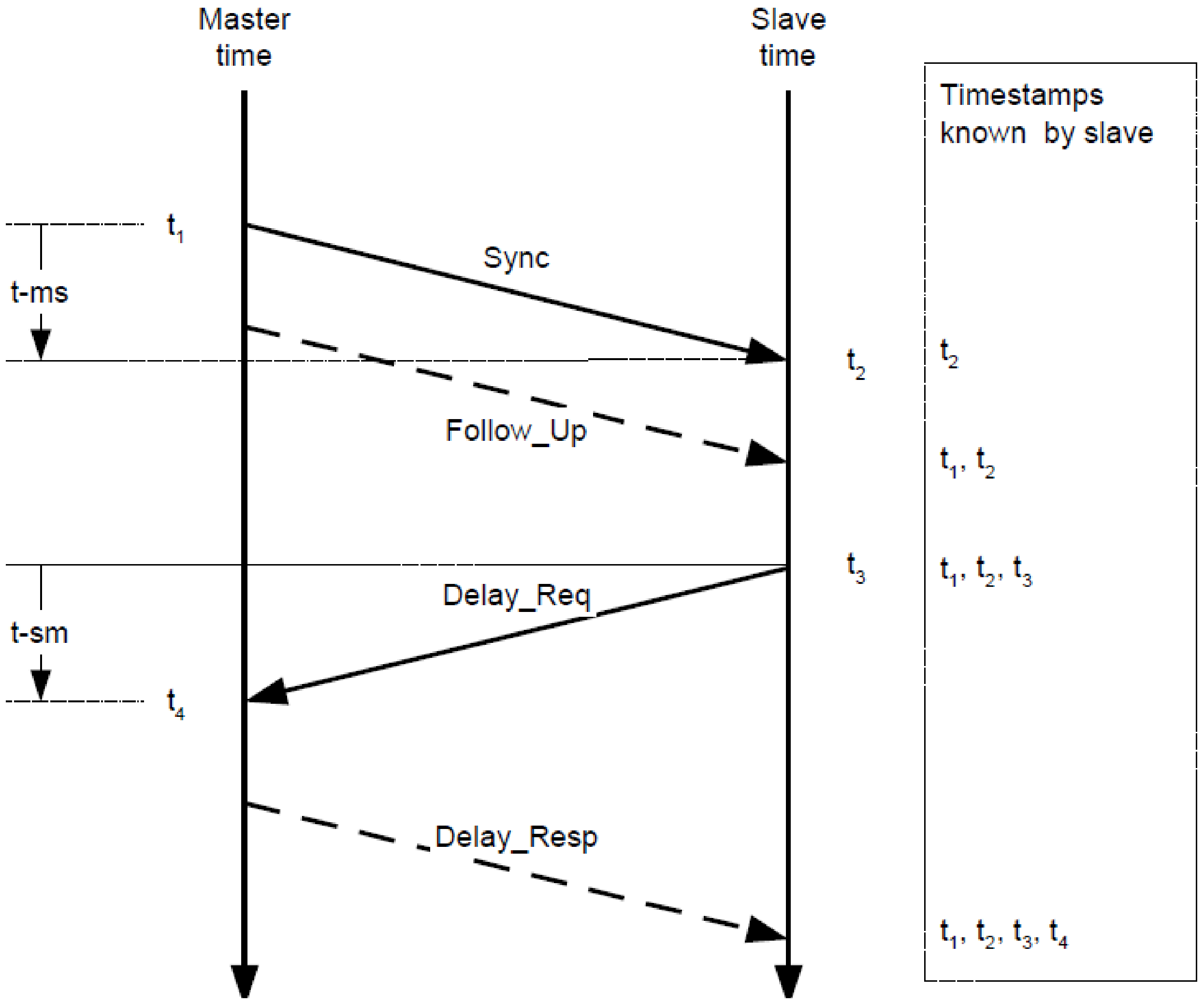
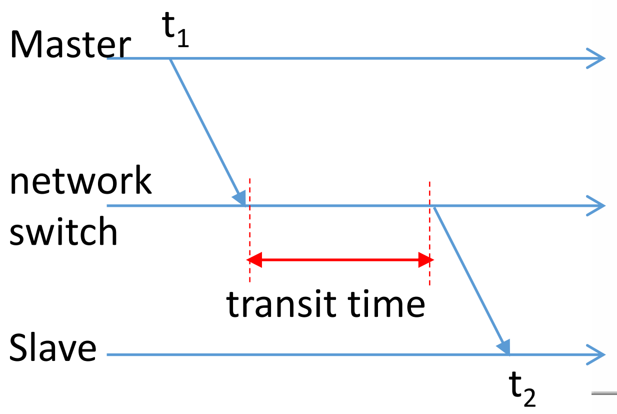


Figure 12—Basic synchronization message exchange

End-to-end transparent clock

for intermediate device, e.g., a network switch

- A transparent clock records the transit time (time interval from ingress to egress), which may be subtracted later to correct the path delay



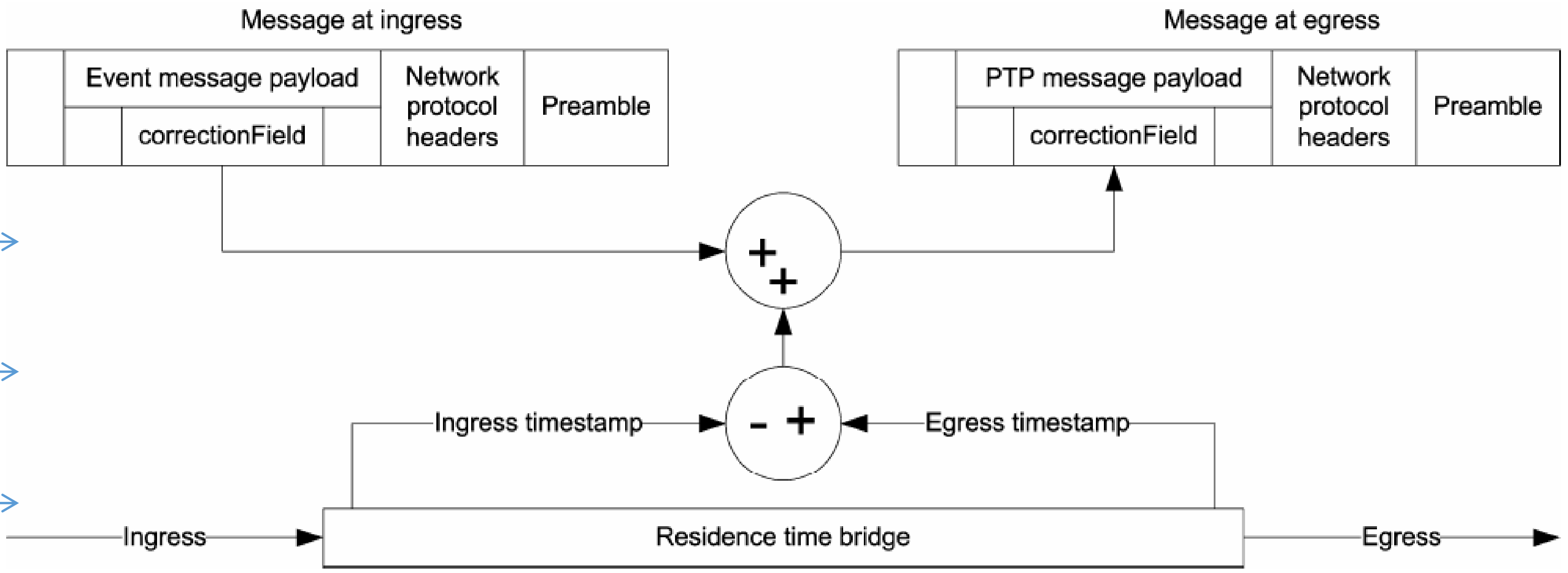
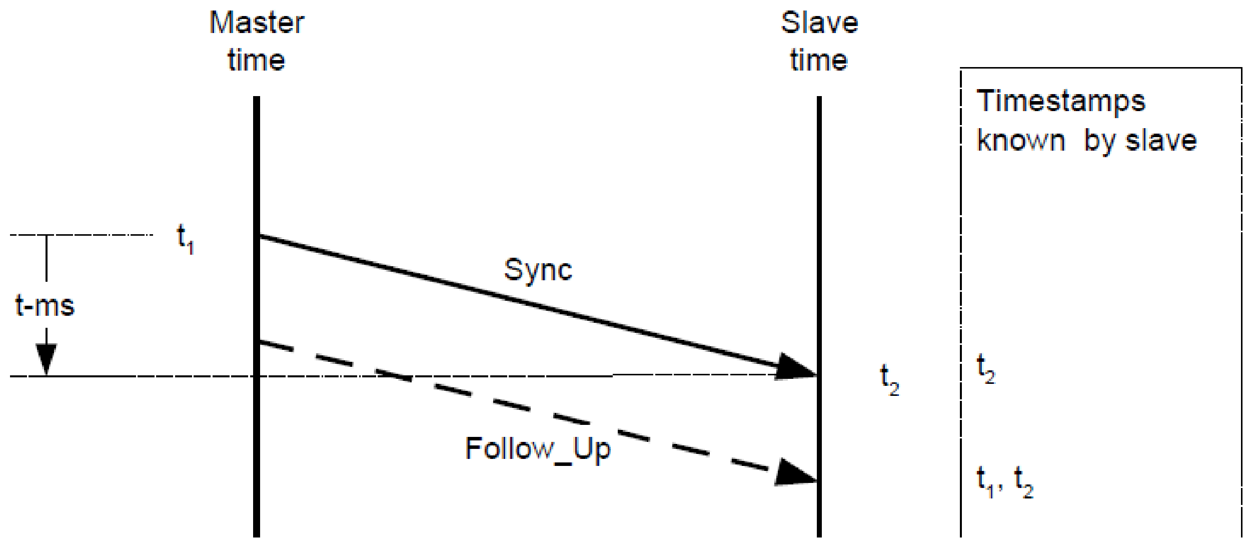


Figure 5—End-to-end residence time correction model

Alternative to delay request-response: Peer delay link measurement

- Idea:
  - Following slides 15, if we can determine the mean path delay, then there is no need for Delay_Req/Delay_Resp message exchange.
  - This may both speed up synchronization process and save some processing load.



Offset from Master time
= time on the slave clock - time on the master clock
= t_2 - t_1 - mean path delay (- correction)

Peer delay link measurement (cont.)

- Mean path delay may be calculate peer-to-peer at each pair of ports
- Mean path delay $= [(t_2-t_1)+(t_4-t_3)]/2$

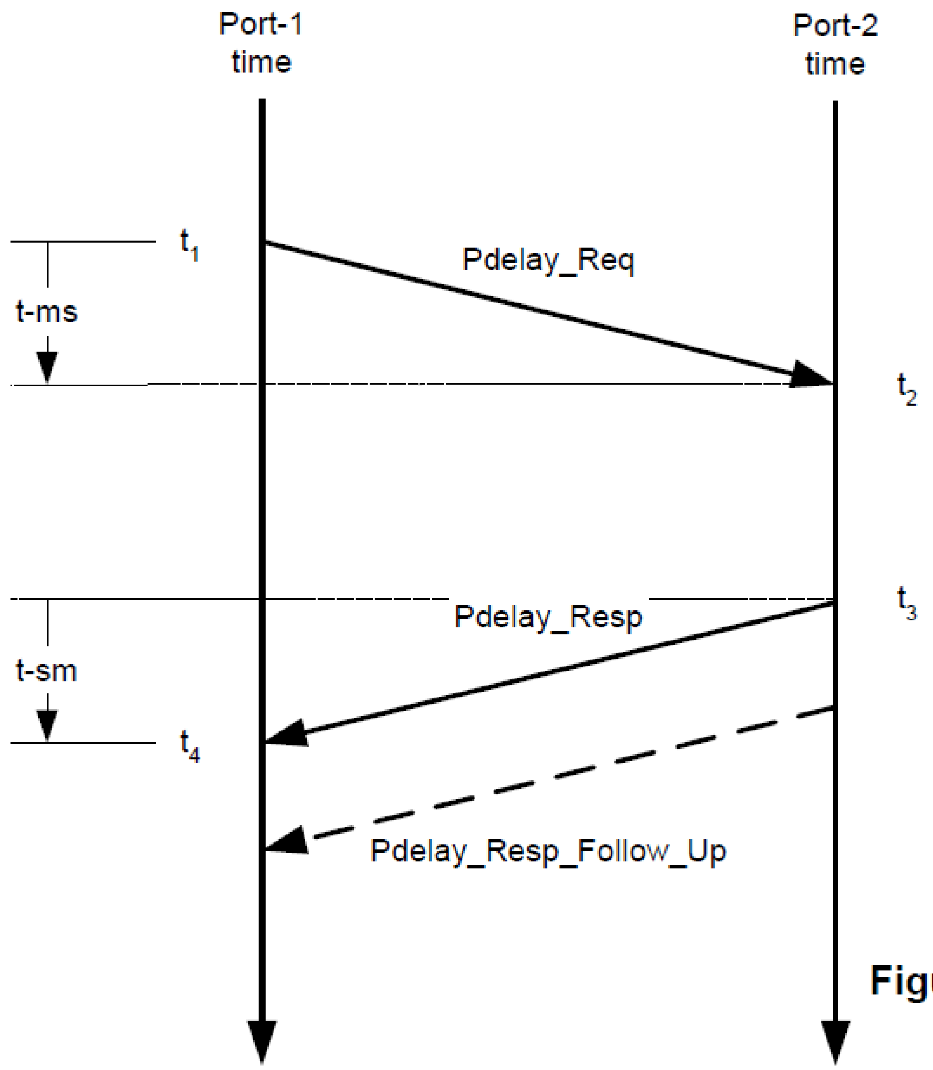
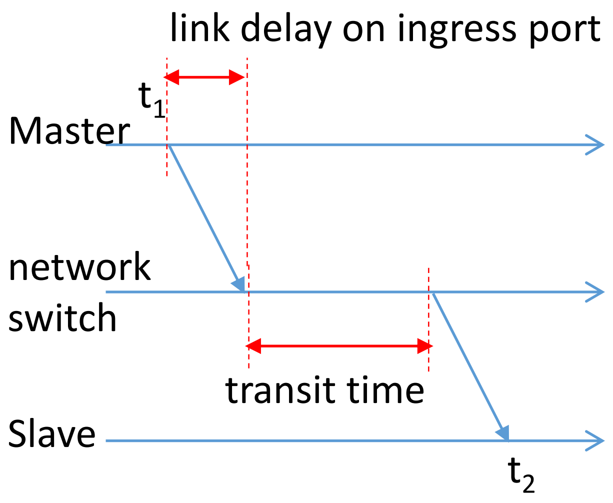


Figure 13—Link delay measurement

Peer-to-peer transparent clock

- Used along with peer delay link measurement



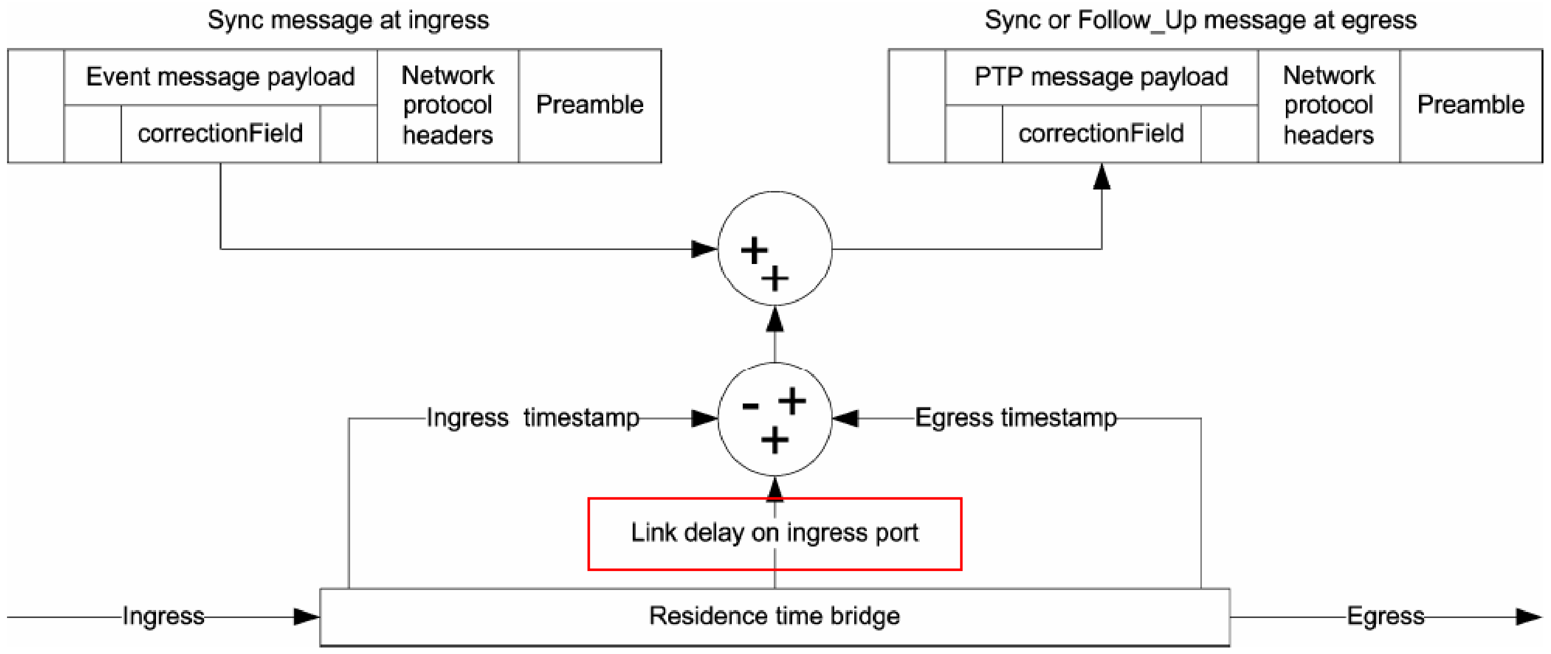


Figure 8—Peer-to-peer residence time and link delay correction model

Timestamp generation

- A timestamp may be taken at point A, B, or C

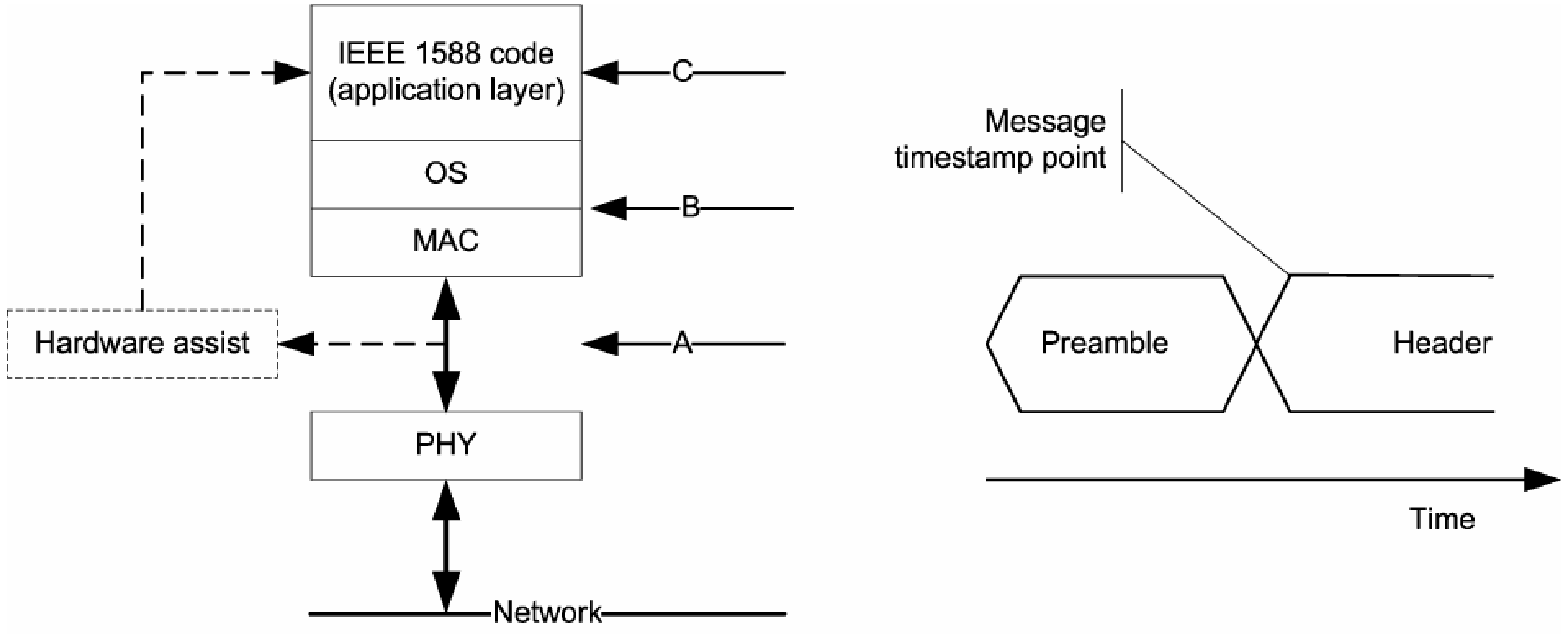


Figure 14—Timestamp generation model

Choosing NTP or PTP?

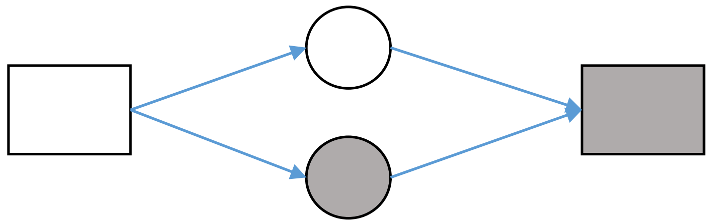
- It depends.
- With support of hardware timestamping, PTP may provide sub-microsecond accuracy
- There are many public NTP servers and built-in NTP clients for easy time synchronization
  - Amazon EC2 offers one for its VMs
- Example scenario (combine both PTP and NTP):
  - Use PTP to synchronize local devices, and use NTP to synchronize both local master clock and remote devices to a global NTP server

Outline of lecture16

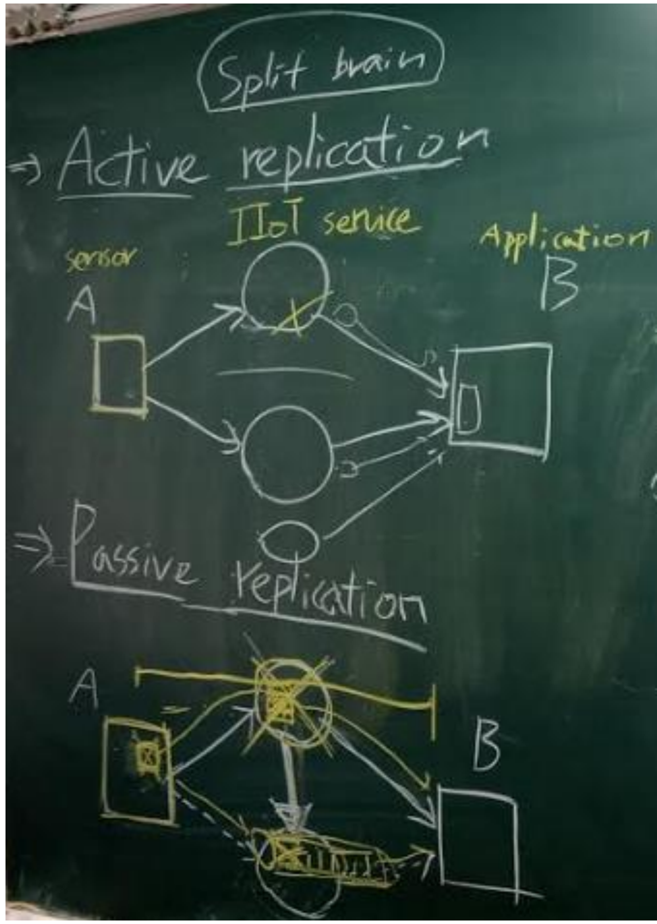
- Time synchronization among communication hosts
  - NTP and PTP
- **Fault-tolerant via data replication**
  - **Active replication vs. passive replication**
- Case study: Fault-tolerant and real-time messaging for edge computing

Fault tolerance via data replication

- Replicating data to a backup server help prevents data loss in case that the primary server crashed
- Active replication
  - Data sender sends data copies to both primary and backup servers; both servers process and deliver the data; data receiver filters out the duplicated result



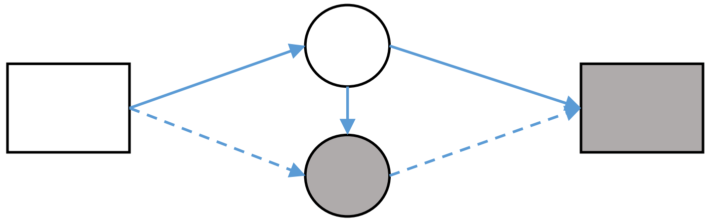
□	: Data sender
○	: Primary server (broker)
●	: Backer server (broker)
■	: Data receiver



  - Pros: almost no latency penalty during fault recovery
  - Cons: double resource cost; potential inconsistency (split-brain)

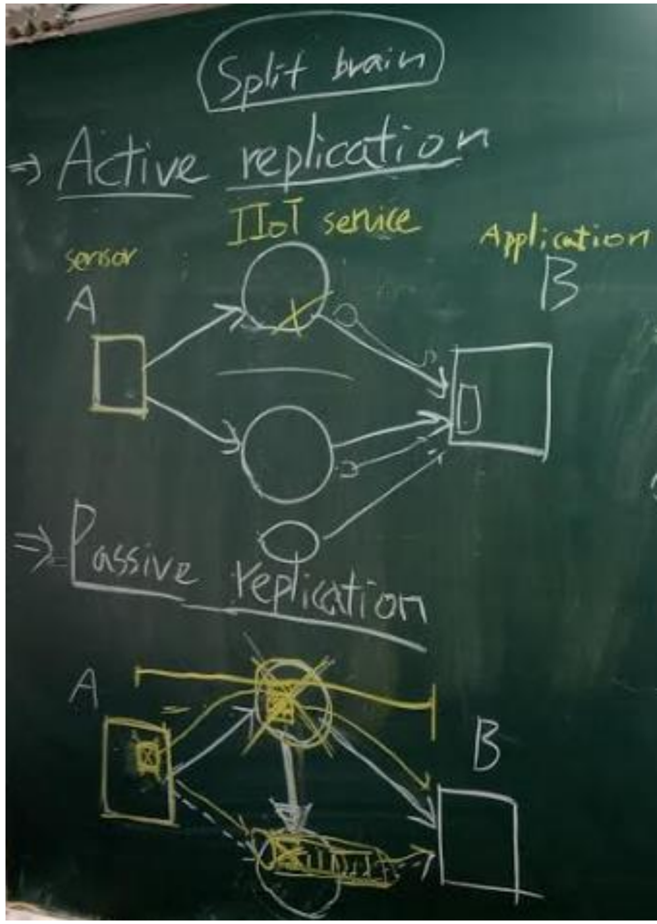
Fault tolerance via data replication (cont.)

- Passive replication
  - Data sender sends data only to the primary server; the primary server send a copy of the data to the backup server; the backup server will promote to become the new primary should the original primary server crash, and the new data will flow through the backup path (dotted arrows)



□	: Data sender
○	: Primary server (broker)
●	: Backer server (broker)
■	: Data receiver

  - Pros: smaller resource cost (compared to active replication)
  - Cons: latency penalty during and after fault recovery



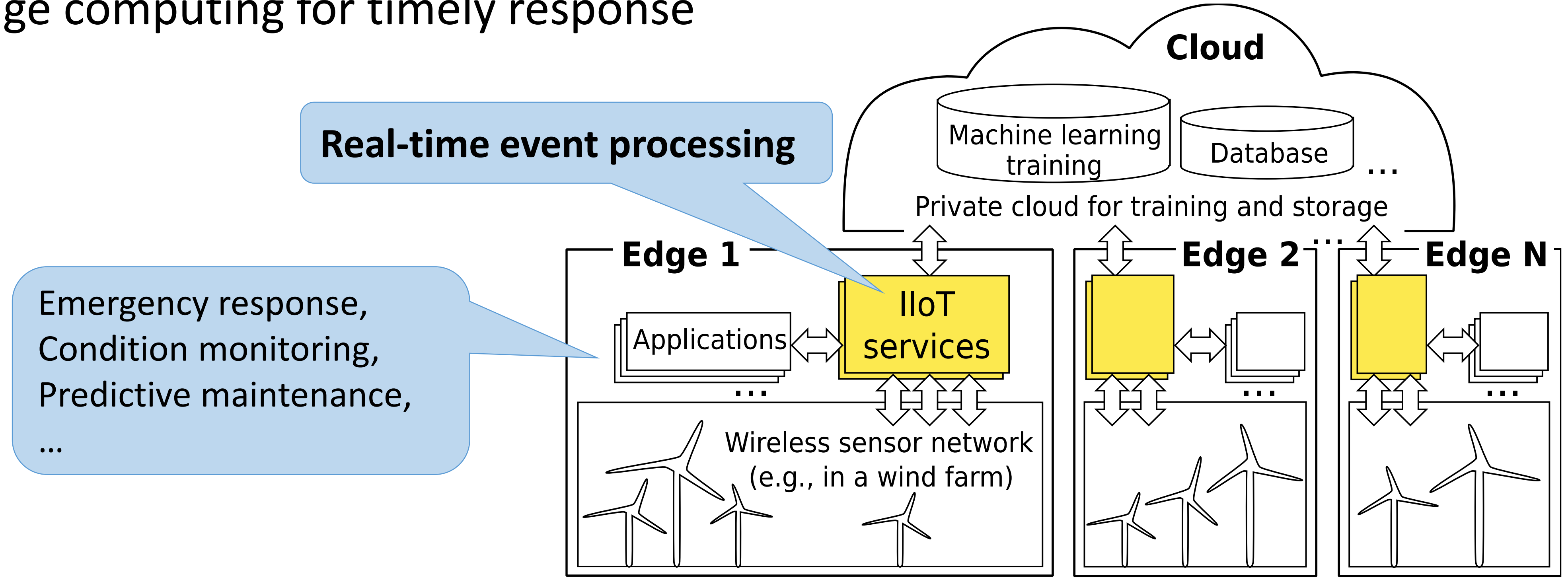
In the following case study, we assume the use of passive replication ⇩

Outline of lecture16

- Time synchronization among communication hosts
  - NTP and PTP
- Fault tolerance via data replication
  - Active replication vs. passive replication
- **Case study: Fault-tolerant and real-time messaging for edge computing**

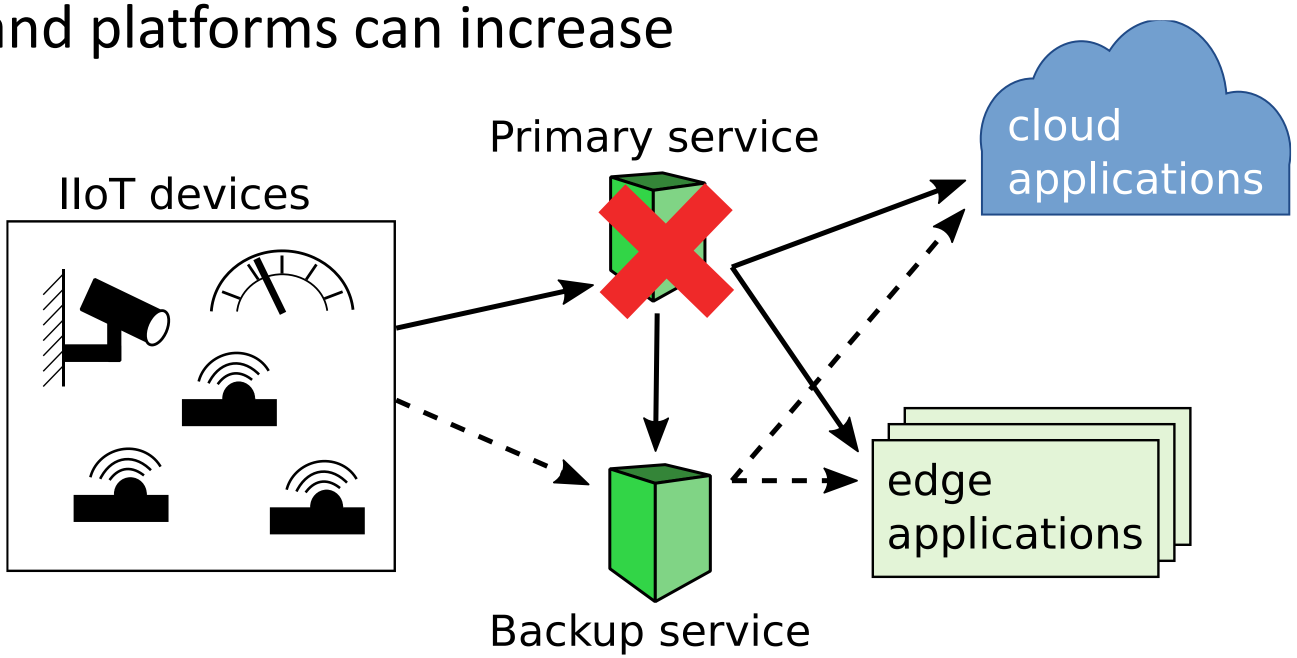
Industrial Internet of Things (IIoT)

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


Cloud
Machine learning training
Database
…
Private cloud for training and storage
Real-time event processing
Emergency response, Condition monitoring, Predictive maintenance, …
Edge 1
Edge 2
Edge N
Applications
IIoT services
Wireless sensor network (e.g., in a wind farm)

IIoT messaging

- An IoT service must deliver messages reliably, but
  - supporting fault tolerance can be costly (active replication) or slow (passive replication)
  - heterogeneous demands and platforms can increase pessimism

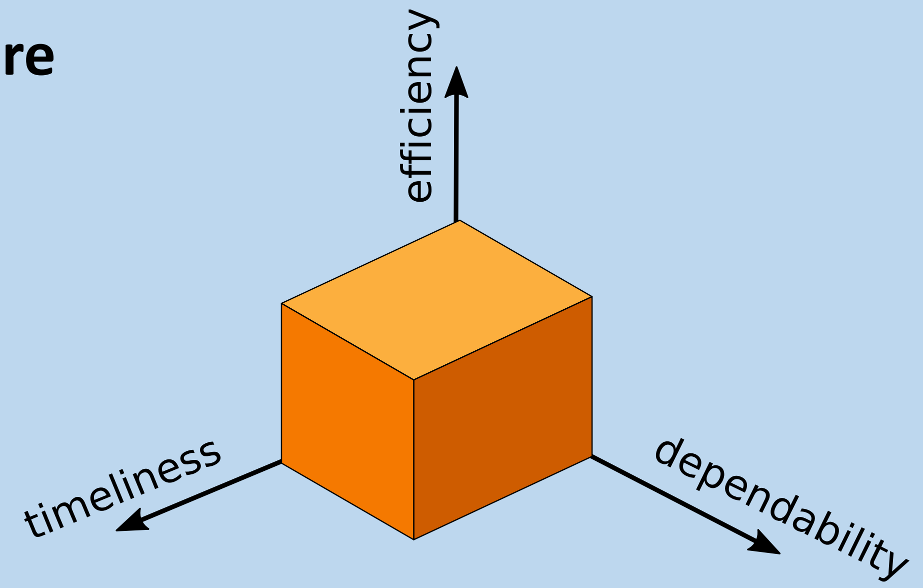


IIoT messaging

- An IoT service must deliver messages reliably, but
  - supporting fault tolerance can be costly or slow
  - heterogeneous demands and platforms can increase pessimism

Fault-Tolerant Real-Time Messaging Architecture

➡ Demand/platform-aware service differentiation.



IIoT requirements and constraints

- IIoT application requirements
  - Message loss: cannot accept more than L consecutive losses
  - Latency: receive a message within D time units since its creation

Latency \ Loss tolerance	*L = 0*	*L = k > 0*
D = 50 ms	emergency response	…
D = 100 ms	…	condition monitoring
D > 500 ms	predictive maintenance	

Handle which one first?

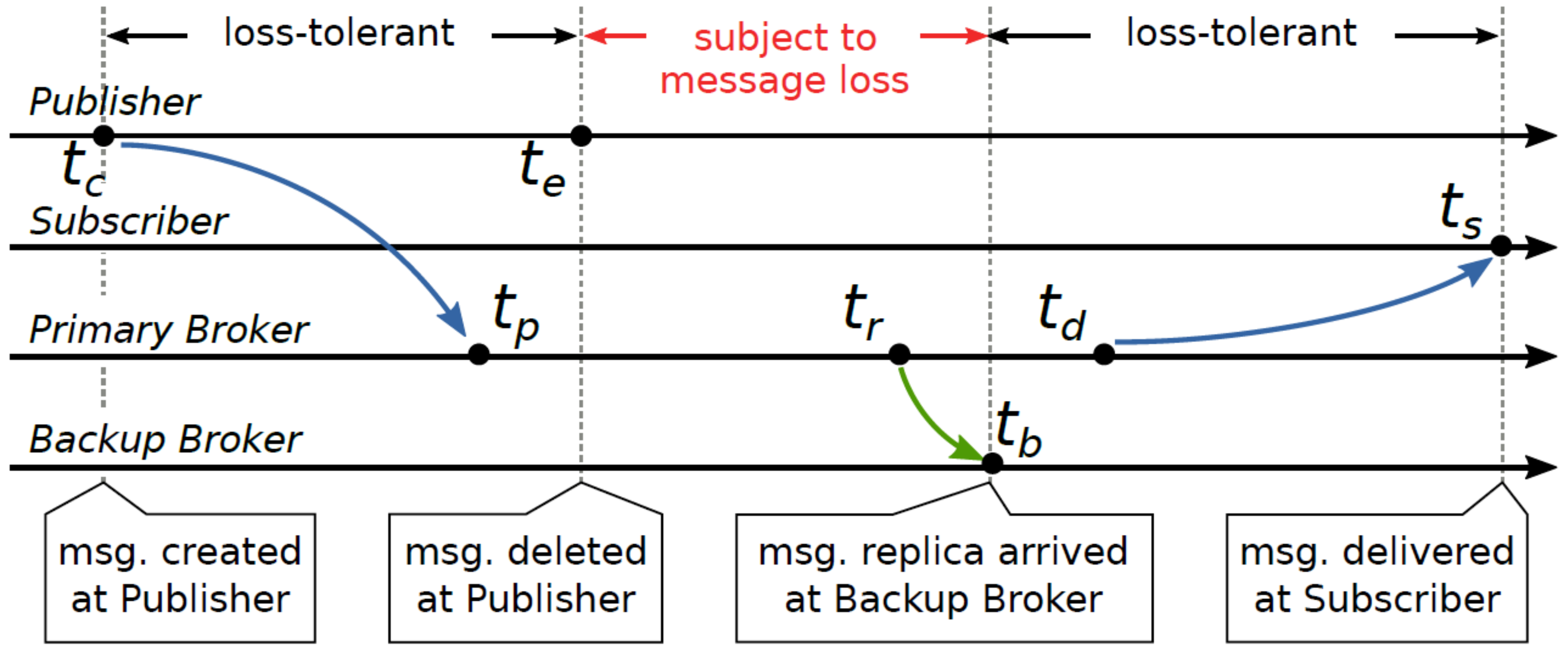
- IIoT platform constraints
  - IIoT devices only do limited message re-transmissions
  - Network latencies differ by orders of magnitude

    0.5 ms (edge) vs. 44 ms (cloud)

Meeting the requirements of message loss and latency

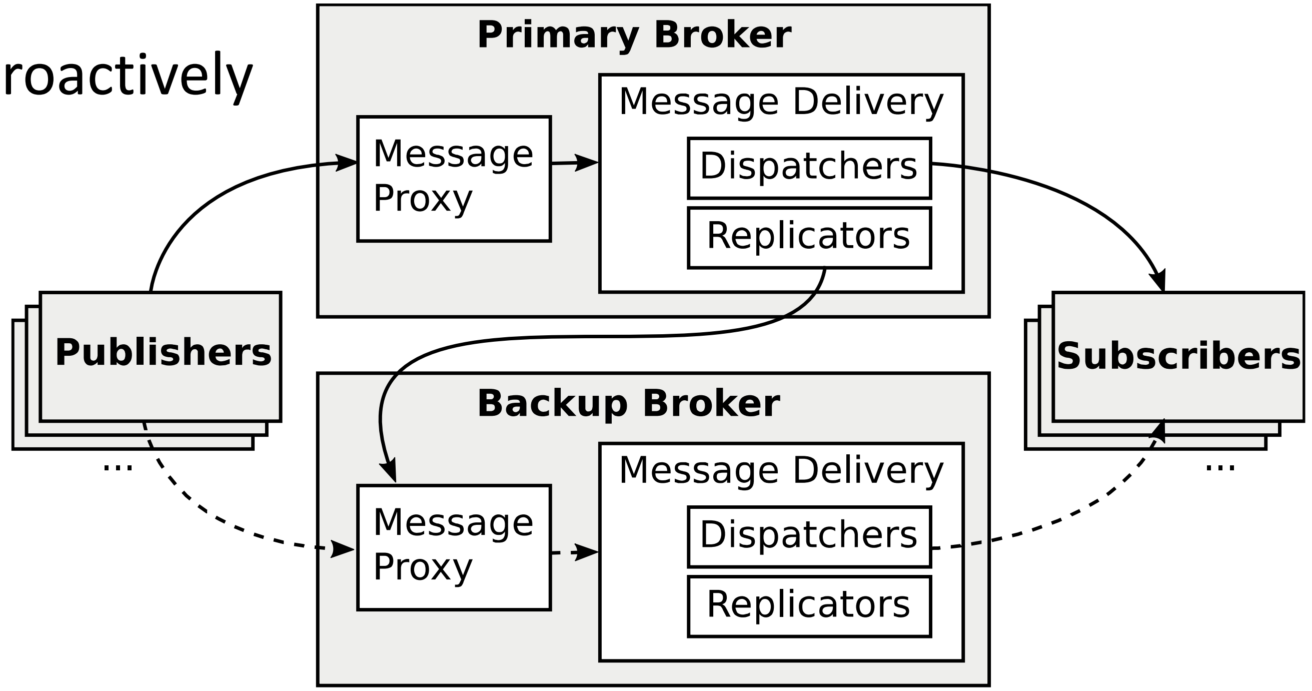
- Key observation:

  IIoT platform may incur timing constraints for loss tolerance!



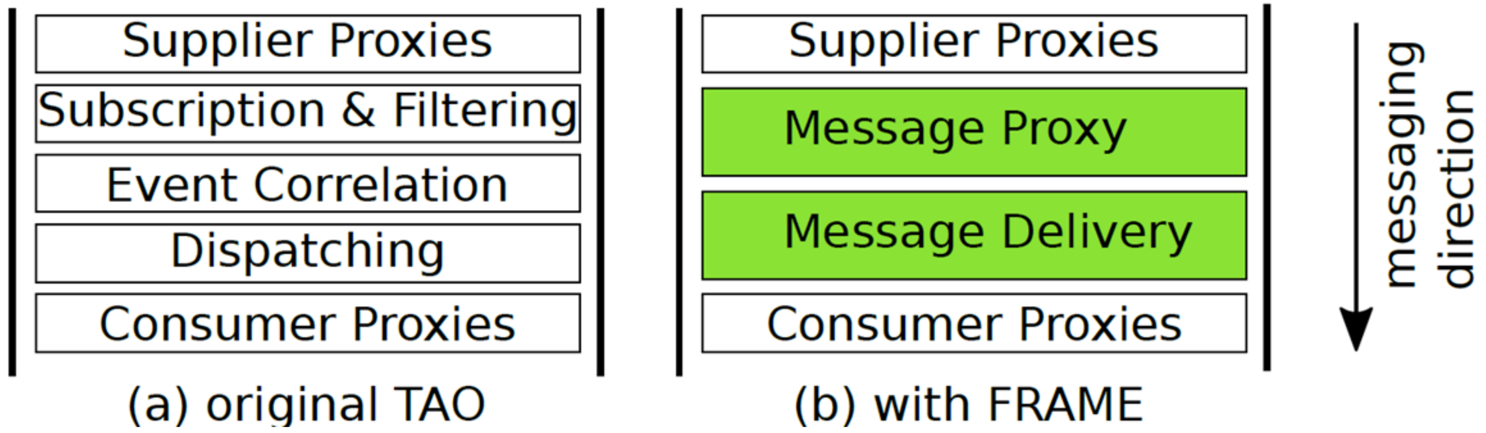
The FRAME messaging architecture

- Co-schedule message dispatching/replication
  - Schedule using an earliest-deadline-first (EDF) policy
  - Prune dispatched messages
  - Suppress message replication proactively



FRAME implementation

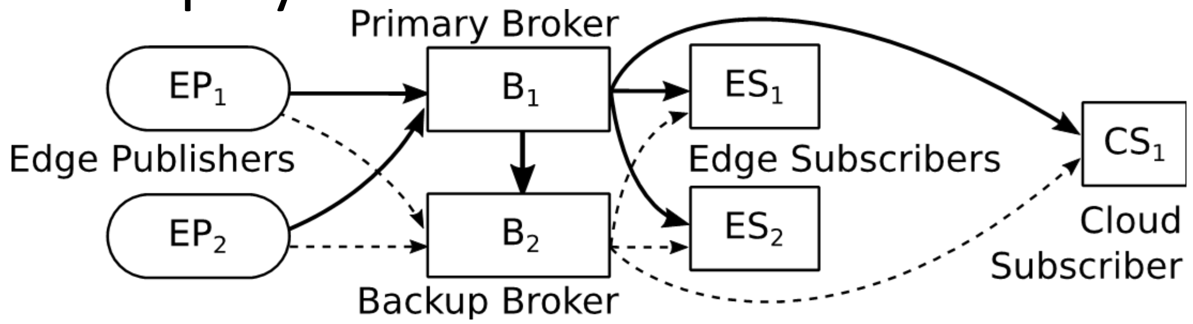
- Implemented within the TAO real-time event service [1]
  - TAO - a mature and widely-used middleware
  - Preserve both TAO's interfaces to publishers (suppliers) and subscribers (consumers)



[1] Harrison, T.H., Levine, D.L. and Schmidt, D.C., 1997. The design and performance of a real-time CORBA event service. *ACM SIGPLAN Notices*, *32*(10), pp.184-200.

FRAME performance validation

- Edge-cloud test-bed deployment:



(Amazon Virtual Private Cloud)

- Example IIoT message topics:

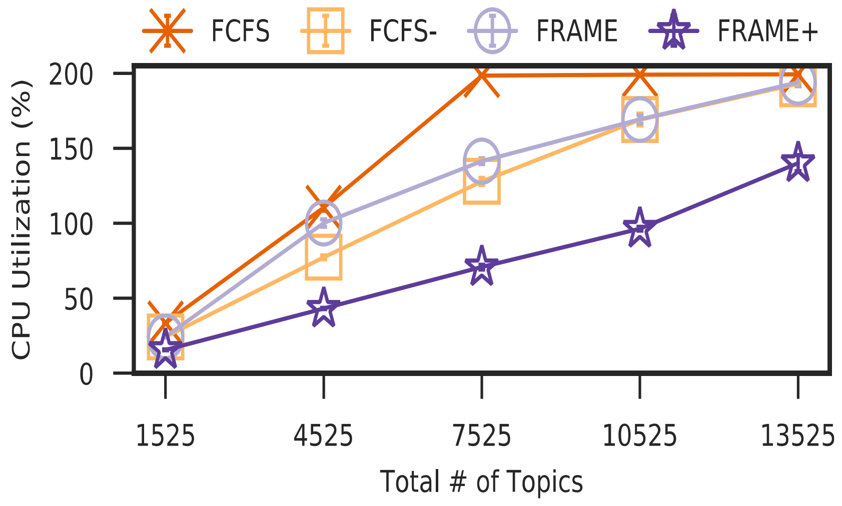
Topic Category	T_i	D_i	L_i	N_i	Destination
0	50	50	0	2	Edge
1	50	50	3	0	Edge
2	100	100	0	1	Edge
3	100	100	3	0	Edge
4	100	100	∞	0	Edge
5	500	500	0	1	Cloud

- Service configurations: FRAME+; FRAME; FCFS; FCFS-

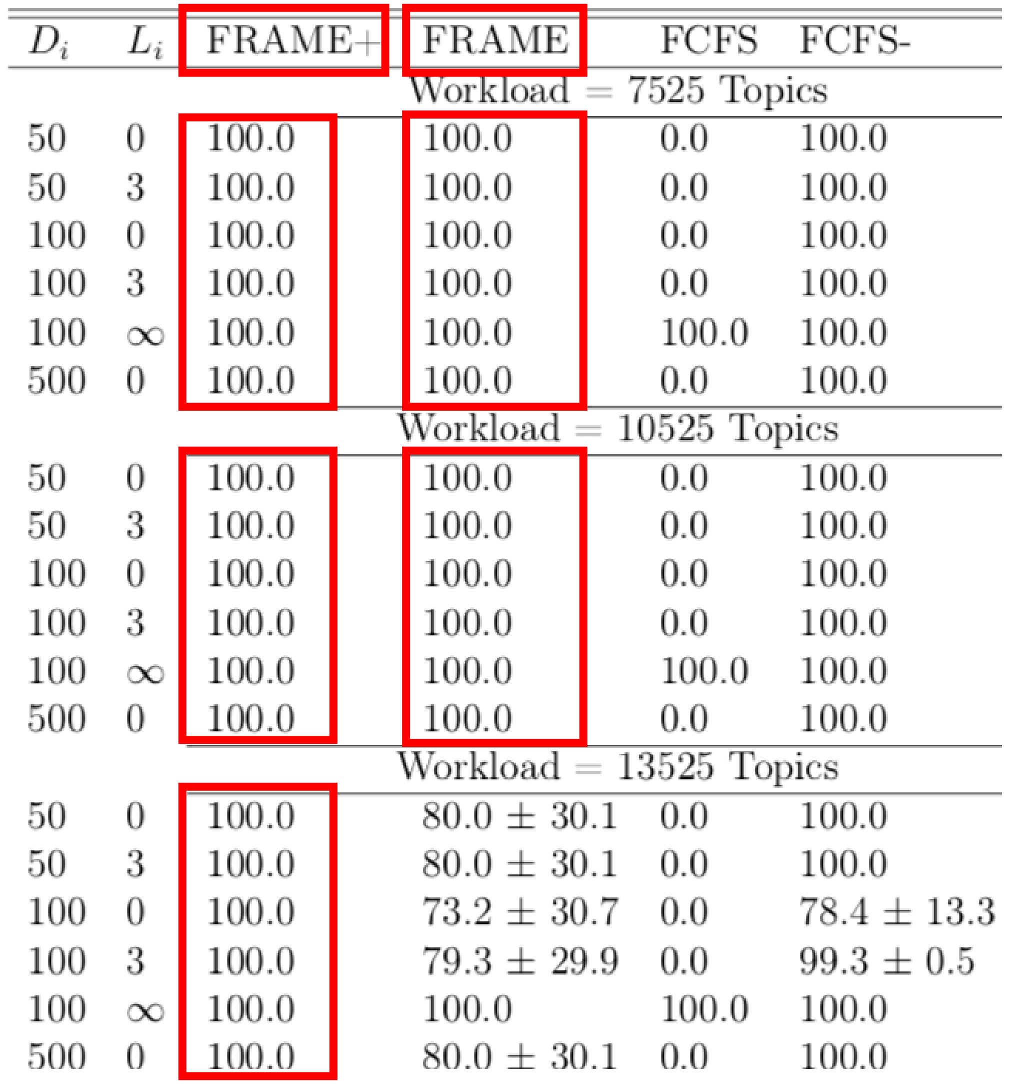
Loss-tolerance performance

FRAME succeeded in assuring loss-tolerance.

By increasing N_i by one, a system can both
(1) accommodate more topics,
(2) save CPU utilization.

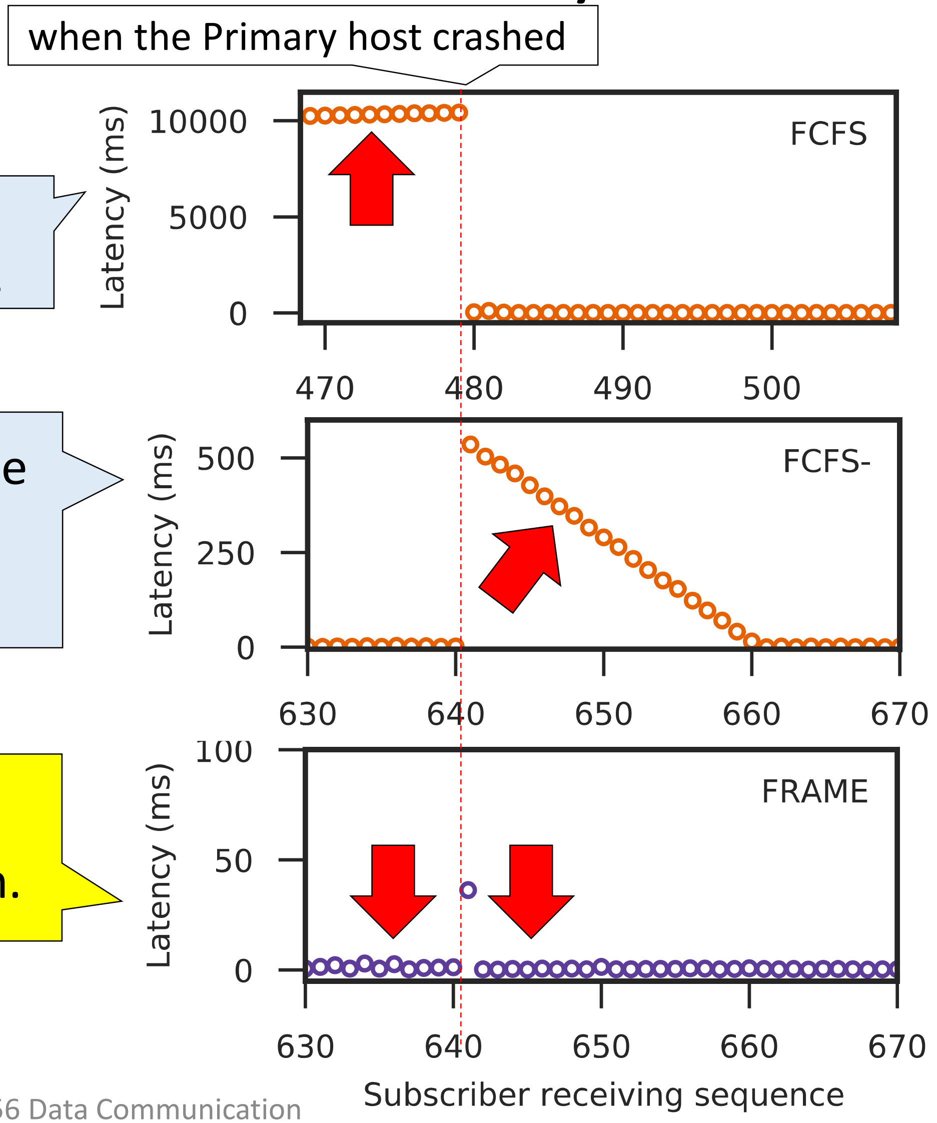


Success rate of meeting loss-tolerance requirements



D_i	L_i	FRAME+	FRAME	FCFS	FCFS-
			Workload = 7525 Topics		
50	0	100.0	100.0	0.0	100.0
50	3	100.0	100.0	0.0	100.0
100	0	100.0	100.0	0.0	100.0
100	3	100.0	100.0	0.0	100.0
100	∞	100.0	100.0	100.0	100.0
500	0	100.0	100.0	0.0	100.0
			Workload = 10525 Topics		
50	0	100.0	100.0	0.0	100.0
50	3	100.0	100.0	0.0	100.0
100	0	100.0	100.0	0.0	100.0
100	3	100.0	100.0	0.0	100.0
100	∞	100.0	100.0	100.0	100.0
500	0	100.0	100.0	0.0	100.0
			Workload = 13525 Topics		
50	0	100.0	80.0 ± 30.1	0.0	100.0
50	3	100.0	80.0 ± 30.1	0.0	100.0
100	0	100.0	73.2 ± 30.7	0.0	78.4 ± 13.3
100	3	100.0	79.3 ± 29.9	0.0	99.3 ± 0.5
100	∞	100.0	100.0	100.0	100.0
500	0	100.0	80.0 ± 30.1	0.0	100.0

Latency performance for recovery



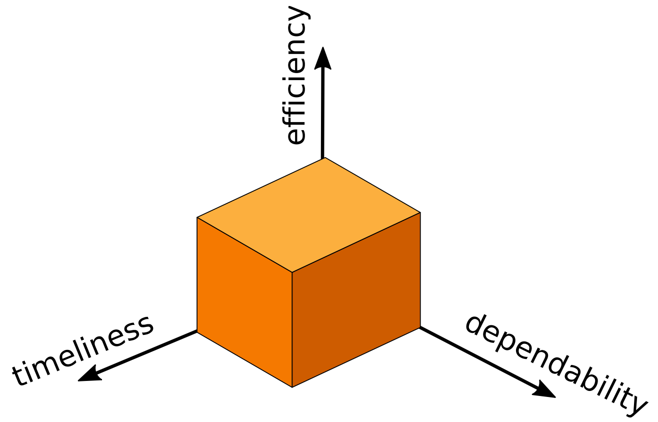
Without suppressing replication, pruning may overload the system.

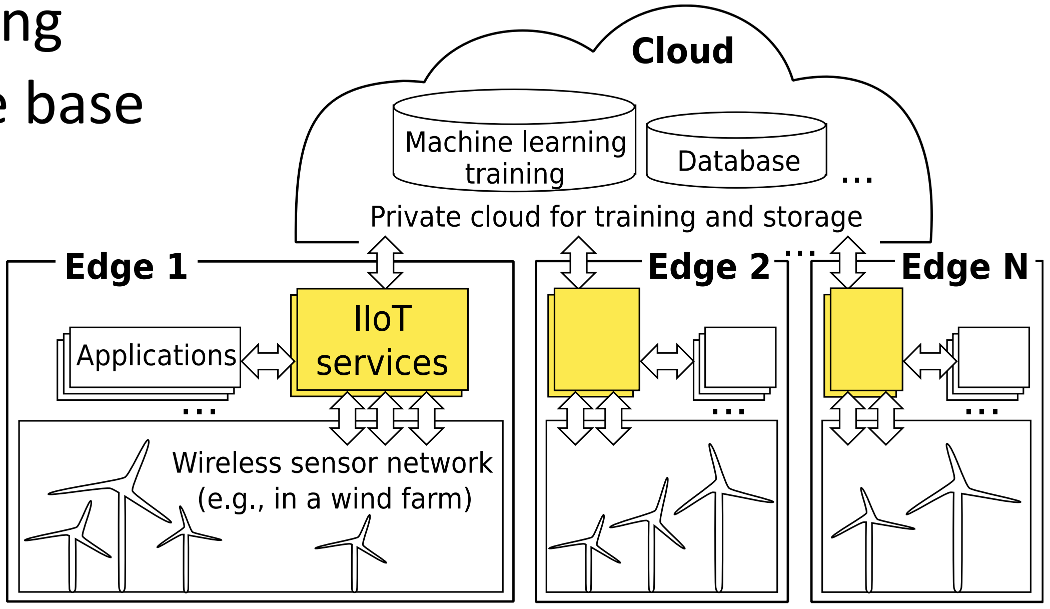
Without pruning, message re-sending may cause latency penalty.

FRAME can reduce latency both before and after crash.

Conclusions for FRAME

- IIoT brings a new challenge to fault-tolerant real-time systems
  - dependable and timely messaging on a heterogeneous platform
- FRAME can meet the challenge:
  - leveraging platform constraints in timing
  - implemented on the mature TAO code base
  - validated for empirical performance




Cloud
Machine learning training
Database
...
Private cloud for training and storage
Edge 1
Edge 2
Edge N
Applications
IIoT services
Wireless sensor network (e.g., in a wind farm)