FRAME: Fault-tolerant ReAl-time Messaging architecture

Paper By: Chao Wang, Christopher Gill, Chenyang Lu

Presented By: Gregor Peach

What is Frame?

Basic Problem

- We have IoT devices
- They need to communicate
- Message passing!!!
- Goals
  - Don’t lose messages
  - Messages go
    - zoom
    - Zoom

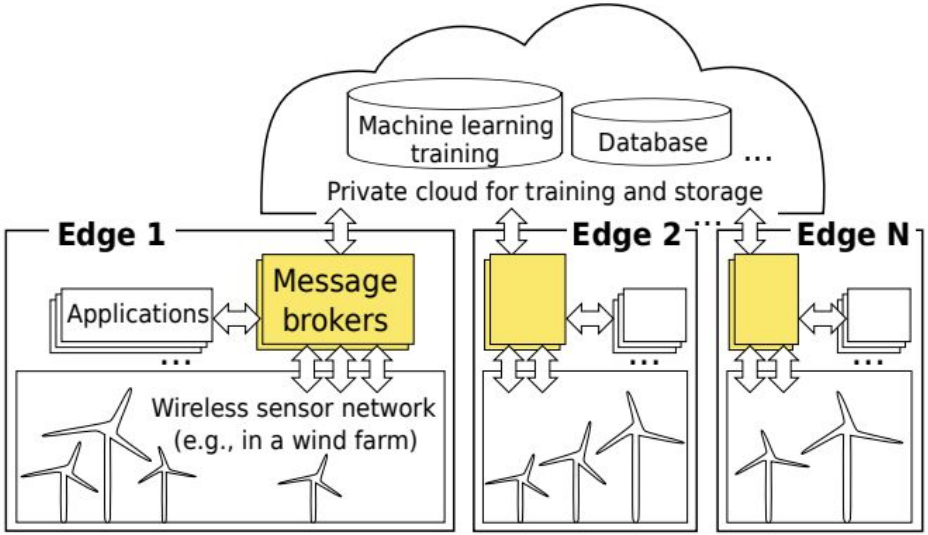


Fig. 1. An Illustration of IIoT Edge Computing.

A System Has Many Parts

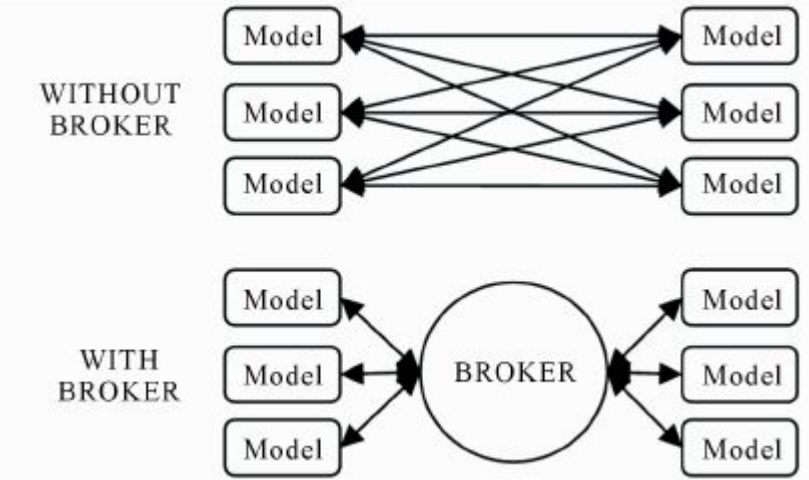
Heterogeneous requirements

- Emergency Stop Mechanism
  - No message loss
  - 10ms latency needed
- Logger
  - No message loss
  - 1sec latency okay
- Dashboard Updater
  - Ocasional message loss
  - 100ms latency desired

Messaging Design

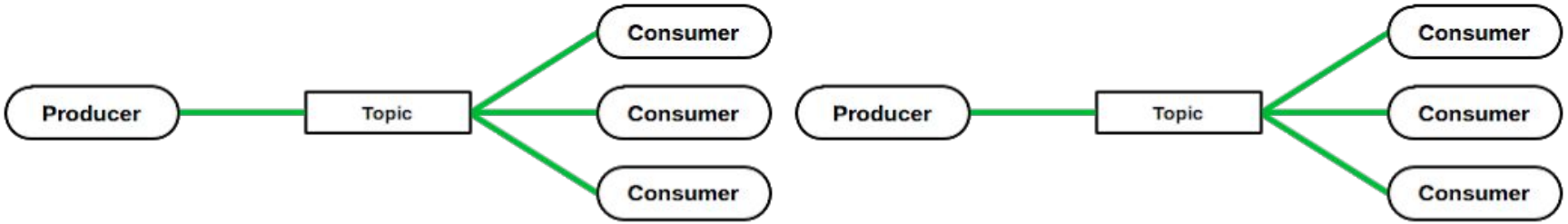
- Two Basic Designs of Messaging
  - Brokered
  - Unbrokered
  - Q: What are the pros/cons of each?

- Broker Placement
  - Cloud
  - Edge
  - Q: Which is better for IoT?



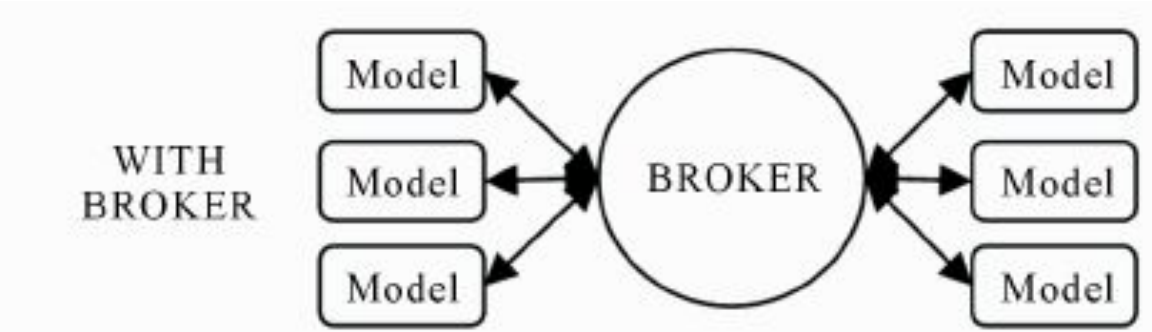
Topics

- Broker handles many message streams
- A Topic is like a email-list for messages
- Topic operations
  - Send/Publish on topic
  - Receive/Subscribe to topic
- Q: How is this different from a set of queues?



Frame Is Middleware

- Software for sending messages to topics
- Software for subscribing to topics
  - Also has to handle asynchronously receiving messages
- A bunch of servers serving as message brokers



Frame (The Novel Parts)

- Each Topic has metadata
  - What many messages in a row can be lost?
  - What is the deadline for messages on this channel? (Real time!!!)
- FRAME Guarantees
  - Always meet loss prevention promises
  - If there are no faults
    - Always meet message deadlines
  - If there are faults
    - Recovery is fast so deadlines aren’t too badly compromised

Contribution Breakdown

Model For Fault Tolerant Real Time messaging

“A new fault-tolerant real-time messaging model. We describe timing semantics for message delivery, identify under what conditions a message may be lost, prove timing bounds for real-time fault-tolerant actions in terms of traffic/service parameters, and demonstrate how the timing bounds can support efficient and appropriate message differentiation to meet each requirement.” (FRAME pg 2)

A System BUILT Around That Model

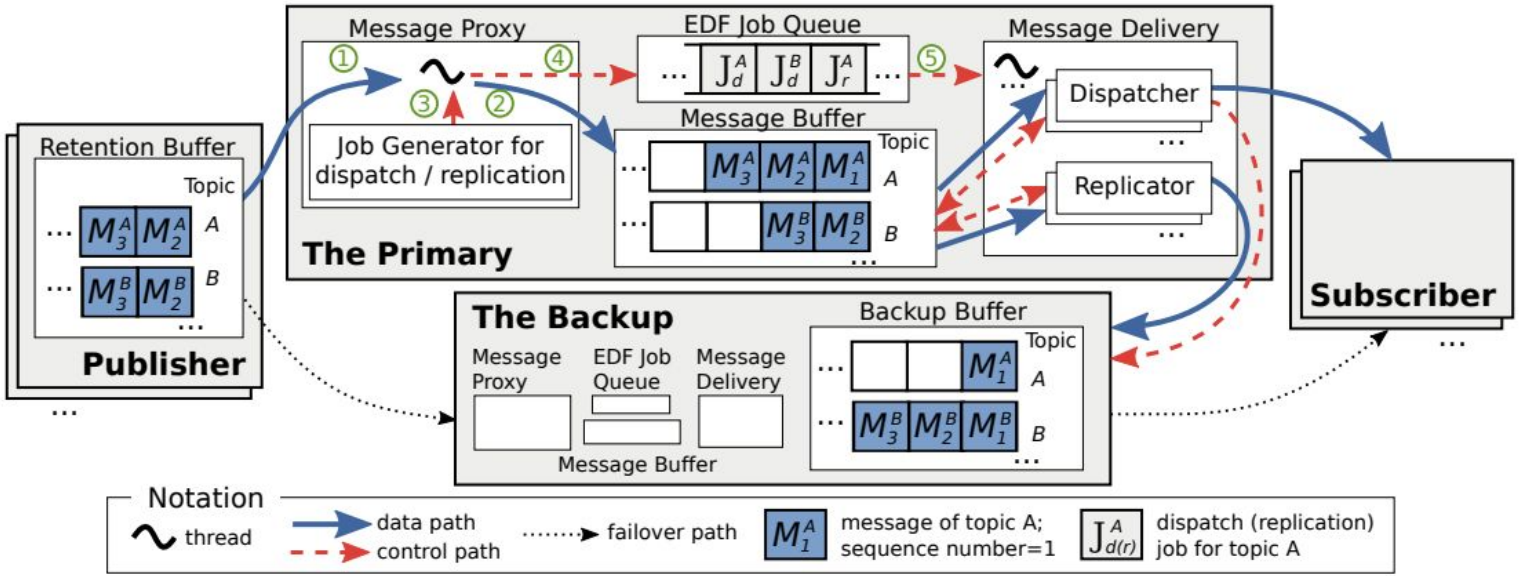


Fig. 4. The FRAME Architecture.

A System BUILT Around That Model

- Avoiding Data Loss
  - There are two servers, a primary and a backup
  - The backup becomes the primary (and visa versa) if the primary crashes
    - Messages are only replicated if needed for guarantees
  - Q: Can messages be delivered twice?

TABLE 3
ALGORITHM FOR DISPATCH-REPLICATE COORDINATION.

Type of Operation	Procedure
Dispatch	1. dispatch the message to the subscriber 2. set *Dispatched* to `True` 3. if *Replicated* is `True`, request the Backup to set *Discard* to `True`
Replicate	1. if *Dispatched* is `True`, abort 2. replicate the message to the Backup 3. set *Replicated* to `True`
Recovery (in the Backup)	1. if *Discard* is `True`, skip the message 2. create a dispatching job for the message 3. push the job into the EDF Job Queue

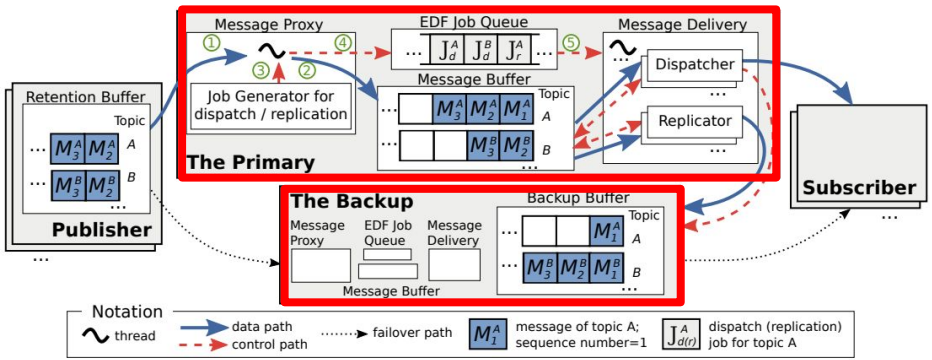


Fig. 4. The FRAME Architecture.

A System BUILT Around That Model

- Fast Message Delivery
  - Broker lives in the edge (even though it supports cloud clients)
  - Earliest deadline first message transmission
  - Q: Why EDF? What are the pros and cons?

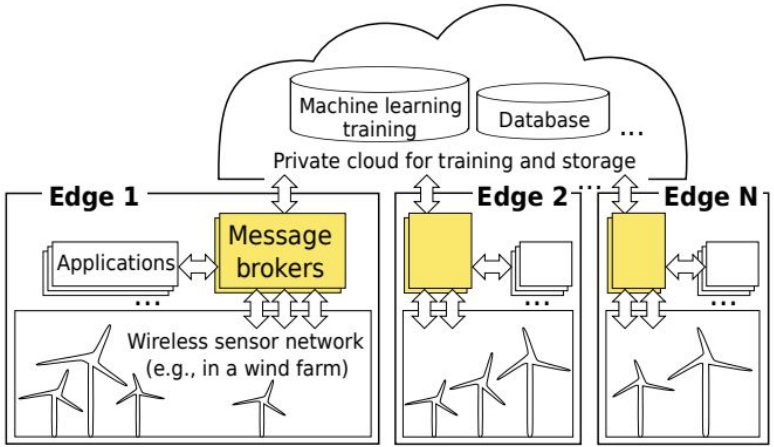
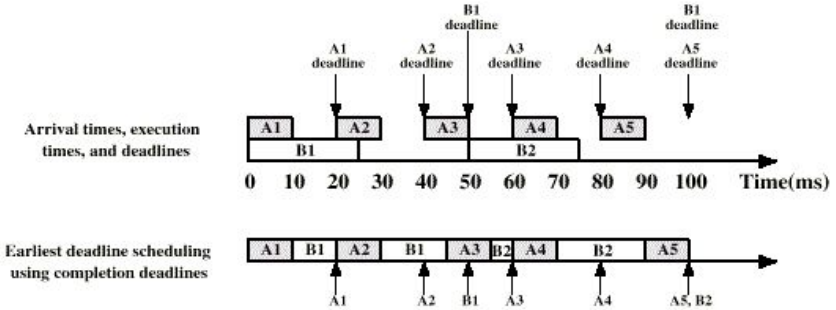


Fig. 1. An Illustration of IIoT Edge Computing.



Evaluation

Loss Prevention Success Rates

- The Candidates
  - FRAME
    - I just tried to explain this
  - FRAME+
    - FRAME + publisher message retention
  - FCFS (first-come first-serve)
    - Delivers messages as they come
    - Coordinates with backup whenever a message is sent (Running on same basic framework as FRAME)
  - FCFS- (first-come first-serve-)
    - Very limited coordination with backup
    - Only occasionally lets backup know what it has sent

Loss Tolerance Success Rate

D_i = deadline (in ms)

L_i = max allowable

Losses in a row

TABLE 4
SUCCESS RATE FOR LOSS-TOLERANCE REQUIREMENT (%).

D_i	L_i	FRAME+	FRAME	FCFS	FCFS-
		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

Note: Test includes fault injection

Loss Prevention Success Rates: Analysis

- FCFS with backup coordination sucks
  - Spends too much time backing stuff up
- FCFS- saves some of that time
  - Doesn't coordinate with backup
  - Backup will save messages already sent
  - So recovering from the backup is slow (even though there is little data loss)
- FRAME
  - Is smart about coordinating with backup
    - Doesn't backup every message
- FRAME+
  - Shows the power of publisher retention

TABLE 4
SUCCESS RATE FOR LOSS-TOLERANCE REQUIREMENT (%).

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 $\pm$ 30.1	0.0	100.0
50	3	100.0	80.0 $\pm$ 30.1	0.0	100.0
100	0	100.0	73.2 $\pm$ 30.7	0.0	78.4 $\pm$ 13.3
100	3	100.0	79.3 $\pm$ 29.9	0.0	99.3 $\pm$ 0.5
100	∞	100.0	100.0	100.0	100.0
500	0	100.0	80.0 $\pm$ 30.1	0.0	100.0

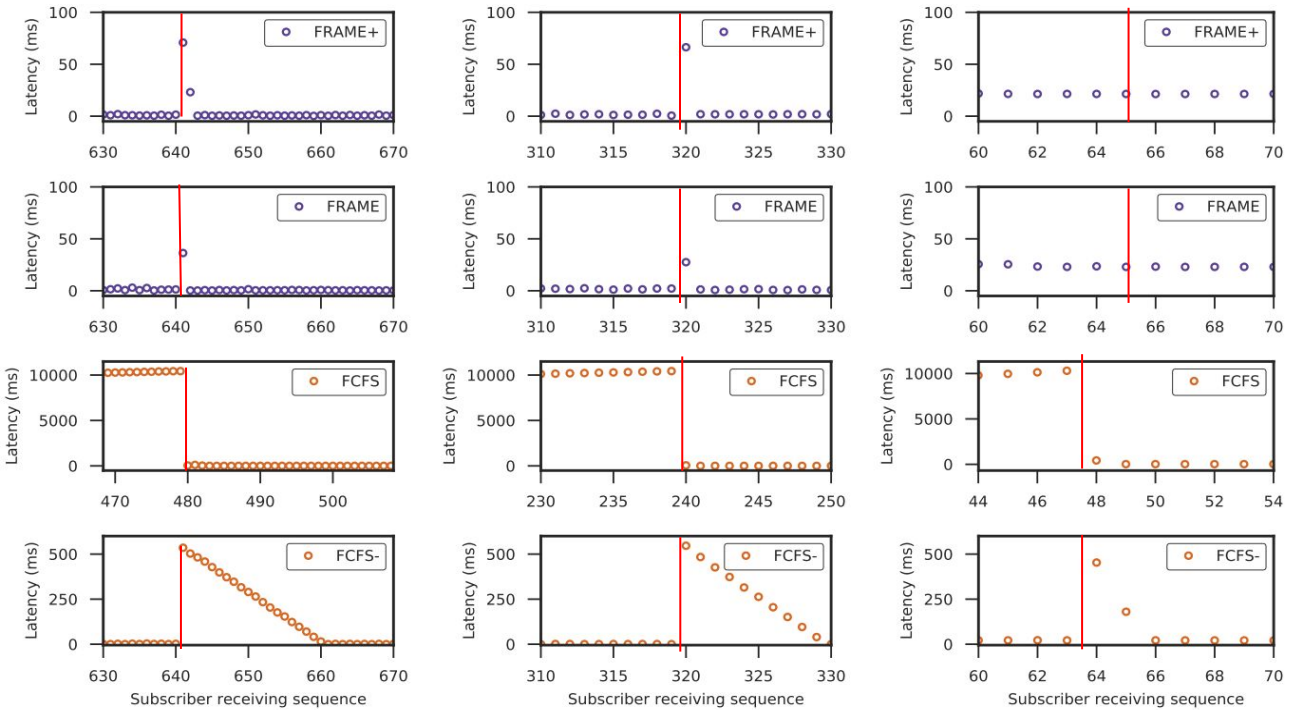
Fault Recovery Performance

FRAME+ =>

FRAME =>

FCFS =>

FCFS- =>



(a) Category 0, $T_i = 50$, $D_i = 50$.

(b) Category 2, $T_i = 100$, $D_i = 100$.

(c) Category 5, $T_i = 500$, $D_i = 500$.

HEAVY LOAD

MEDIUM LOAD

LIGHT LOAD

Real-Time Performance

TABLE 5
SUCCESS RATE FOR LATENCY REQUIREMENT (%).

		Workload = 13525 Topics			
50	0	98.4 ± 2.9	85.4 ± 21.7	0.1 ± 0.1	99.4 ± 3.6E-1
50	3	98.4 ± 2.9	85.3 ± 21.7	0.2 ± 0.2	99.5 ± 2.3E-1
100	0	97.6 ± 4.4	83.7 ± 21.9	2.6E-4 ± 6.0E-4	98.3 ± 1.0
100	3	97.6 ± 4.4	83.8 ± 21.9	9.9E-4 ± 2.2E-3	98.3 ± 1.1
100	∞	97.6 ± 4.4	83.8 ± 21.9	6.6E-4 ± 1.5E-3	98.3 ± 1.1
500	0	98.6 ± 2.8	86.1 ± 21.8	0.0	100.0

Critiques

From The Github Issue...

@hjaensch7: Some figures seemed inefficient?

@RyanFisk2: What happens if the backup fails? Or if both fail?

@tuhinadasgupta: What are the security concerns of FRAME?