Real-Time Edge Computing

Chenyang Lu
Industrial Internet of Things (IIoT)

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

Condition monitoring, Emergency response, Predictive maintenance, ...

Applications

Wireless sensor network (e.g., in a wind farm)

Edge 1

IIoT services

Edge 2

Edge N

Cloud

Machine learning training

Database

Private cloud for training and storage
Research challenge #1: timeliness

- Timing constraints:
  - IIoT applications have latency requirements
  - Events carrying physical data have temporal semantics

Application example: condition monitoring

Image source: https://www.maintwiz.com/what-is-condition-monitoring/
Research challenge #1: timeliness

- **Timing constraints:**
  - IIoT applications have latency requirements
  - Events carrying physical data have temporal semantics

**Contribution #1: Cyber-Physical Event Processing Architecture**

- latency differentiation
- time consistency enforcement

Application example: condition monitoring

Image source: [https://www.maintwiz.com/what-is-condition-monitoring/](https://www.maintwiz.com/what-is-condition-monitoring/)
Research challenge #2: loss-tolerance

- An IIoT service must deliver messages reliably, but
  - fault-tolerant systems can be slow or costly
  - heterogeneous traffic and platforms can increase pessimism
Research challenge #2: loss-tolerance

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

Contribution #2: Fault-Tolerant Real-Time Messaging Architecture

- co-scheduling fault-tolerant real-time activities
- traffic/platform-aware service configuration
Research challenge #3: efficiency

- Efficiency atop loss-tolerance and timeliness:
  - costly to backup many in-band small computations
  - costly to recompute for fault recovery

Example of in-band computations: AWS Lambda function for IIoT inference

Image source: https://aws.amazon.com/lambda/
Research challenge #3: efficiency

- Efficiency atop loss-tolerance and timeliness:
  - costly to backup many in-band small computations
  - costly to recompute for fault recovery

**Contribution #3**: Adaptive Real-Time Reliable Edge Computing

- selective lazy data replication
- proactive cleanup of obsolete data

Example of in-band computations: AWS Lambda function for IIoT inference

Image source: [https://aws.amazon.com/lambda/](https://aws.amazon.com/lambda/)
Contributions

Three new IIoT middleware design and implementations:

- Real-time cyber-physical event processing (CPEP)
- Fault-tolerant real-time messaging (FRAME)
- Adaptive real-time reliable edge computing (ARREC)

All have been implemented and validated within the TAO real-time event service [1].

Outline

- **CPEP**: real-time cyber-physical event processing
- **FRAME**: fault-tolerant real-time messaging
- **ARREC**: adaptive real-time reliable edge computing

![Diagram showing Supplier Proxies, Subscription & Filtering, Event Correlation, Dispatching, Consumer Proxies, CPEP, loss-tolerance, timeliness, efficiency]
Cyber-physical event processing model

Temporal semantics

- **Absolute time consistency**
  - A bound on an event’s elapse time since its creation

- **Relative time consistency**
  - A bound on the difference between events’ creation times
Real-time event processing

- Processing in the order of priorities propagated from application:

- Temporal semantics enforcement and shedding:
  - Absolute time consistency
  - Relative time consistency
    - Track both the earliest and the latest event creations, per operator
Both workers and movers are further prioritized, enabling an appropriate activity ordering.
Enforcing Absolute Time Consistency

- Tracking the earliest end time of validity interval

- Responses to consistency violation
  - Marking: deferring the handling to consumers
  - Shedding: cancelling all subsequent processing (Improving efficiency)
Enforcing Relative Time Consistency

- Maintaining an ordered list of events’ timestamp
  - One timestamp per event type
  - Comparing the maximum time difference with validity interval

- Responses to consistency violation
  - Marking: deferring the handling to consumers
  - Shedding: cancelling all subsequent processing (Improving efficiency)
Experiment design

- **IIoT workload:**
  - Filtering
  - Data transform
  - Encryption

- **Test-bed configuration:**

- **Comparison baseline:**
  - Apache Flink streaming processing framework [1]

Latency performance

<table>
<thead>
<tr>
<th>Priority</th>
<th>Service</th>
<th>Number of middle-priority streams</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>High</td>
<td>Flink</td>
<td>3.8 ± 0.1</td>
</tr>
<tr>
<td></td>
<td>CPEP</td>
<td>0.8 ± 0.0</td>
</tr>
<tr>
<td>Middle</td>
<td>Flink</td>
<td>4.5 ± 0.1</td>
</tr>
<tr>
<td></td>
<td>CPEP</td>
<td>1.6 ± 0.0</td>
</tr>
<tr>
<td>Low</td>
<td>Flink</td>
<td>5.2 ± 0.3</td>
</tr>
<tr>
<td></td>
<td>CPEP</td>
<td>3.7 ± 0.3</td>
</tr>
</tbody>
</table>

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.
Effectiveness of CPEP Sharing

- **Experiment setup**

- **Results of sharing vs. non-sharing**

CPEP sharing helped reduce latency.
Effectiveness of Sharing

Results of sharing vs. non-sharing

Latency of low-priority processing

CPU utilization
Outline

- CPEP: Real-time cyber-physical event processing
- **FRAME**: Fault-tolerant real-time messaging
- ARREC: Adaptive real-time reliable edge computing

![Diagram showing the components of CPEP, FRAME, and ARREC]
Message loss-tolerance requirement

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

<table>
<thead>
<tr>
<th>Value of $L_i$</th>
<th>Application examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>emergency response; predictive maintenance</td>
</tr>
<tr>
<td>$k &gt; 0$</td>
<td>condition monitoring</td>
</tr>
</tbody>
</table>

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

Fault-tolerance model

- A crash failure may happen to an IIoT service host (fail-stop)
- Lost messages may be recovered
  1. via retransmissions from message publishers
  2. via a backup service

Fault-tolerant real-time processing

- Specify provable deadlines for message replication and dispatch
- Co-schedule replication and dispatch using, e.g., earliest-deadline-first (EDF)
Necessary condition for a message loss

- A message may loss only if both
  1. publisher has deleted its copy
  2. a copy of message has not been replicated to the Backup

Events between message creation and its delivery:
Deadlines for dispatch and replication

---

**Deadline for dispatch:**

\[ D_{id}^d \leq D_i - \Delta_{PB} - \Delta_{BS} \]

The deadline specifications help in configuring IIoT traffic/platform parameters.

**Deadline for replication:**

\[ D_{ir}^r \leq (N_i + L_i)T_i - \Delta_{PB} - \Delta_{BB} - x \]

---

- \( T_i \): topic’s sending period
- \( N_i \): # of most-recent messages a publisher can retransmit
- \( D_i \): latency requirement
- \( L_i \): loss-tolerance requirement
The FRAME messaging architecture

- EDF scheduling to dispatch/replicate a message
- Suppress replication if the dispatch deadline is smaller
- Prune dispatched messages
Experiment design

- **IIoT topic configuration:**

<table>
<thead>
<tr>
<th>Topic Category</th>
<th>( T_i )</th>
<th>( D_i )</th>
<th>( L_i )</th>
<th>( N_i )</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>50</td>
<td>50</td>
<td>0</td>
<td>2</td>
<td>Edge</td>
</tr>
<tr>
<td>1</td>
<td>50</td>
<td>50</td>
<td>3</td>
<td>0</td>
<td>Edge</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>100</td>
<td>0</td>
<td>1</td>
<td>Edge</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>100</td>
<td>3</td>
<td>0</td>
<td>Edge</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>100</td>
<td>( \infty )</td>
<td>0</td>
<td>Edge</td>
</tr>
<tr>
<td>5</td>
<td>500</td>
<td>500</td>
<td>0</td>
<td>1</td>
<td>Cloud</td>
</tr>
</tbody>
</table>

- **Test-bed deployment:**

- **Service configurations:**
  - FRAME+; FRAME; FCFS; FCFS-
Loss tolerance performance

FRAME succeeded in assuring loss-tolerance.

A small increase in $N_i$ can greatly improve performance (FRAME+).

$$D_i^r \leq (N_i + L_i)T_i - \Delta_{PB} - \Delta_{BB} - x$$

**Success rate of meeting loss-tolerance requirements**

<table>
<thead>
<tr>
<th>$D_i$</th>
<th>$L_i$</th>
<th>FRAME+</th>
<th>FRAME</th>
<th>FCFS</th>
<th>FCFS-</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>0</td>
<td>100.0</td>
<td>100.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>50</td>
<td>3</td>
<td>100.0</td>
<td>100.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>100.0</td>
<td>100.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td>100.0</td>
<td>100.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>100</td>
<td>$\infty$</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>500</td>
<td>0</td>
<td>100.0</td>
<td>100.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>

**Workload = 7525 Topics**

<table>
<thead>
<tr>
<th>$D_i$</th>
<th>$L_i$</th>
<th>FRAME+</th>
<th>FRAME</th>
<th>FCFS</th>
<th>FCFS-</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>0</td>
<td>100.0</td>
<td>100.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>50</td>
<td>3</td>
<td>100.0</td>
<td>100.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>100.0</td>
<td>100.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td>100.0</td>
<td>100.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>100</td>
<td>$\infty$</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>500</td>
<td>0</td>
<td>100.0</td>
<td>100.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>

**Workload = 10525 Topics**

<table>
<thead>
<tr>
<th>$D_i$</th>
<th>$L_i$</th>
<th>FRAME+</th>
<th>FRAME</th>
<th>FCFS</th>
<th>FCFS-</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>0</td>
<td>100.0</td>
<td>100.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>50</td>
<td>3</td>
<td>100.0</td>
<td>100.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>100.0</td>
<td>100.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td>100.0</td>
<td>100.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>100</td>
<td>$\infty$</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>500</td>
<td>0</td>
<td>100.0</td>
<td>100.0</td>
<td>0.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>

**Workload = 13525 Topics**

<table>
<thead>
<tr>
<th>$D_i$</th>
<th>$L_i$</th>
<th>FRAME+</th>
<th>FRAME</th>
<th>FCFS</th>
<th>FCFS-</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>0</td>
<td>100.0</td>
<td>80.0 ± 30.1</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>50</td>
<td>3</td>
<td>100.0</td>
<td>80.0 ± 30.1</td>
<td>0.0</td>
<td>100.0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>100.0</td>
<td>73.2 ± 30.7</td>
<td>0.0</td>
<td>78.4 ± 13.3</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td>100.0</td>
<td>79.3 ± 29.9</td>
<td>0.0</td>
<td>99.3 ± 0.5</td>
</tr>
<tr>
<td>100</td>
<td>$\infty$</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>500</td>
<td>0</td>
<td>100.0</td>
<td>80.0 ± 30.1</td>
<td>0.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>
Latency performance for recovery

Without suppressing replication, pruning could overload the system.

No pruning, however, could cause latency penalty by resending obsolete messages.

FRAME can mitigate latency penalty.
Latency during fault-free operations

Observation:
Both replication and pruning could delay message dispatching...

All configurations gave similar latency performance.

Success rate of meeting soft latency requirements

<table>
<thead>
<tr>
<th>$D_i$</th>
<th>$L_i$</th>
<th>FRAME+</th>
<th>FRAME</th>
<th>FCFS</th>
<th>FCFS-</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>0</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>50</td>
<td>3</td>
<td>100.0</td>
<td>99.9 ± 3.0E-2</td>
<td>99.9 ± 4.1E-2</td>
<td>100.0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td>100.0</td>
<td>100.0</td>
<td>99.9 ± 1.1E-3</td>
<td>100.0</td>
</tr>
<tr>
<td>100</td>
<td>∞</td>
<td>100.0</td>
<td>100.0</td>
<td>99.9 ± 1.9E-3</td>
<td>100.0</td>
</tr>
<tr>
<td>500</td>
<td>0</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>

Workload = 4525 Topics

How to improve efficiency?
Outline

- CPEP: Real-time cyber-physical event processing
- FRAME: Fault-tolerant real-time messaging
- ARREC: Adaptive real-time reliable edge computing

<table>
<thead>
<tr>
<th>Supplier Proxies</th>
<th>Subscription &amp; Filtering</th>
<th>Event Correlation</th>
<th>Dispatching</th>
<th>Consumer Proxies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supplier Proxies</td>
<td>ARREC</td>
<td></td>
<td></td>
<td>Consumer Proxies</td>
</tr>
</tbody>
</table>

- Efficiency
- Timeliness
- Loss-tolerance
Edge Computing for IIoT

- Timely, reliable, and efficient IIoT edge computing
  - CPEP: Real-time cyber-physical event processing
  - FRAME: Fault-tolerant real-time messaging
  - ARREC: Adaptive real-time reliable edge computing

![Diagram of edge computing for IIoT](image)
References

