Khronos: Middleware for Simplified Time Management in Cyber Physical Systems

Stefanos Peros, Stéphane Delbruel, Sam Michiels, Wouter Joosen and Danny Hughes
Fast-moving consumer goods company:

- Line 3: (200, 600) ms
- Line 4: (30, 100) ms
- Line 5: (250, 400) ms
- Line 6: (300, 500) ms
- Line 1: (150, 400) ms
- Line 2: (60, 200) ms

Gateway → Cloud → Display

Operator
Challenge

› Managing event arrival-time boundaries in CPS
  › **varying** network latency
    › wireless medium
    › packets propagate across different paths
  › **varying** packet inter-generation delay
    › clock drift
State-of-the-Art

› Rely on application developer
  › static timeouts @ compile time
  › e.g. leased signals[1]
Problem Description

› Predicting time-boundaries at compile time
  › **impractical** (if not impossible)
    › CPS application developer != infrastructure expert
    › non-deterministic event arrival times
Problem Description

› Predicting time-boundaries at compile time
  › impractical (if not impossible)
  › inefficient
    › waiting too long can fail to produce useful result
    › not waiting long enough may lead to faults
      › incomplete information
Application developers do not know how long to wait for sensor packet arrivals.
Problem Description

› Application developers do not know
  › how long to wait for sensor packet arrivals

› But do know
  › how important it is to wait for sensor packet arrivals
    › before proceeding with complex event computation
  › % completeness constraint
Timeliness vs Completeness

› **Trade-off**
  › Higher *completeness* constraint
    › larger timeouts
      › slower (re)actions (**timeliness**)  
  › Lower completeness constraint
    › smaller timeouts
      › faster (re)actions
Related Work

› ProbSlack[2]
  › adds **dynamic** offset to user-defined timeout
    › delay model
    › user tolerance $\delta$ for missed events (\sim completeness)
ProbSlack[2]

- Relies on **developer** to specify @ compile time
  - **timeout** (query frequency)
    - e.g. sampling periods can change at runtime
  - additional **configuration**
    - refresh period T for delay model(s)
Research Problem

- State-of-the-art **time management** solutions for **CPS** rely heavily on the **application developer**
  - timeout specification @ compile time
  - user-defined parameter configuration
Requirements for CPS Middleware

› **A.** Completeness constraint per device
› **B.** Not rely on developer
› **C.** Dynamism
› **D.** Heterogeneity
› **E.** Context
Approach
Khronos

- satisfy application completeness constraint(s)
- automatically determine timeout(s)
  - per sensor data stream
  - per completeness constraint
  - per packet arrival
Prediction Technique (1/3)

- Inspired by TCP’s Retransmission TimeOut (RTO)
  - **non-deterministic** ACK arrival times
    - varying network latency
  - **trade-off**: completeness vs timeliness
    - too long -> slow speed
    - too short -> unnecessary retransmissions
Prediction Technique (2/3)

› **Timeout**

\[ TO(t_i) = S(t_i) + K \times \nabla(t_i) + D_T \]

› **Smoothed Arrival Time**

\[ S(t_i) = \alpha S(t_{i-1}) + (1 - \alpha)R(t_i) \]

› **Smoothed Arrival Time Variance**

\[ \nabla(t_i) = \beta \nabla(t_{i-1}) + (1 - \beta)|S(t_{i-1}) - R(t_i)| \]
Prediction Technique (2/3)

- **Timeout**

\[ TO(t_i) = S(t_i) + K \cdot \mathbb{V}(t_i) + D_T \]

- **Smoothed Arrival Time**

\[ S(t_i) = \alpha S(t_{i-1}) + (1 - \alpha) R(t_i) \]

- **Smoothed Arrival Time Variance**

\[ \mathbb{V}(t_i) = \beta \mathbb{V}(t_{i-1}) + (1 - \beta) |S(t_{i-1}) - R(t_i)| \]
Prediction Technique (3/3)

› Lightweight
  › \( O(n) \), where \( n \) the number of completeness constraints
  › 10 operations to compute next timeout
    › 5 multiplications + 5 additions

› Simple
  › no configuration post deployment (req. B)
Sensitivity Factor K

- $K = f(\text{constraint})$
- offline mapping
  - $\sim 3$ weeks of network monitoring
- smallest $K$ that satisfies given constraint
  - overprovision $\times 2$

$$TO(t_i) = S(t_i) + K \times \nabla(t_i) + D_T$$
API(1/2)

register constraint *(req. A)*:

`registerCompleteness(device, constraint, on_next, on_timeout, on_violation)`

register (static) timeout:

`registerTimeout(device, timeout, on_next, on_timeout)`
Three **callback** methods *(req. E)*:

- **on_next(value, timeout, completeness)**
  - packet arrives before timeout
- **on_timeout(timeout, completeness)**
  - timeout occurs before packet arrival
- **onViolation(value, timeout, completeness)**
  - completeness < constraint
Architecture

› Three layers
Implementation
Network

› Wireless mesh
  › 33 devices (20 sensors)

› SmartMesh IP
  › broadly used in IIoT & CPS applications
  › TSCH(default), CSMA/CA
  › self-forming & self-maintaining
Middleware

- Raspberry Pi 3
- Python v3.6
  - flask (REST)
  - Pyro 4.6 (RMI)
- CoAP & websocket
  - gateway communication
Evaluation
Evaluation

› **Performance** of predicted time windows
  › network & application *dynamism* (req. C)
    › 4 experiments
  › network & application *heterogeneity* (req. D)
    › 4 experiments
Metrics (1/2)

- Prediction Error (PE)

\[ PE_{d, \rho} = \frac{1}{n} \sum_{k=1}^{n} distance(p_k, to_k), \quad distance(p_k, to_k) = \text{abs}(p_k - to_k) \]

- \( d \): device, \( \rho \): constraint, \( p_k \): \( k \)'th arrival time, \( to_k \): \( k \)'th timeout
- measured in seconds
- \( \downarrow \) PE \( \uparrow \) timeliness
Metrics (2/2)

- Constraint Violation % (CV%)

- $\rho$ satisfied when:
  - $\text{completeness} \geq \rho$, over 99.999% of the time
  - $\text{completeness}$: fraction of packets that arrive before timeout
    - measured as moving average
  - if $\rho = 1.0$, best-effort
Alternative Approaches

› Double Sampling Period (DSP)
  
  \[ TO(t_i) = 2 \times (Sampling\ Period) \]

› Sampling Period Network Delay (SPND)
  
  \[ TO(t_i) = (Sampling\ Period) + \text{avg}(\text{latency}) \]

› Static Timeout Oracle (STO)
  
  \[ TO(t_i, \rho) = \text{smallest\ timeout\ that\ satisfies}\ \rho \]

› **theoretical**, reference benchmark
Default Topology

Gateway in Floor 3

Table 1: Deployed peripherals and their settings.

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Peripheral Type</th>
<th>Quantity</th>
<th>Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>3302/5500</td>
<td>Sensor (Presence)</td>
<td>1</td>
<td>10s</td>
</tr>
<tr>
<td>9803/9805</td>
<td>Sensor (Light)</td>
<td>3</td>
<td>120s</td>
</tr>
<tr>
<td>3303/5702</td>
<td>Sensor (Temperature)</td>
<td>3</td>
<td>120s</td>
</tr>
<tr>
<td>8040/8042</td>
<td>Sensor (Pressure)</td>
<td>3</td>
<td>60s</td>
</tr>
<tr>
<td>9903/9904/2</td>
<td>Sensor (Thermocouple)</td>
<td>1</td>
<td>10s</td>
</tr>
<tr>
<td>1010/9000</td>
<td>Sensor (Battery)</td>
<td>10</td>
<td>900s</td>
</tr>
</tbody>
</table>
Dynamism

- Sampling Period
- Network Size
- Network Latency
Sampling Period

- 60s → 120s → 240s
- every ~24 hours
- $\rho = 0.8$
- default topology
Heterogeneity

- Range of Completeness Constraints
- Medium Access Control Protocol
- Sampling Period
- Network topology
Range of Completeness Constraints (1/3)

- \( \rho \in \langle 0.1, 0.2, \ldots, 1.0 \rangle \)
- default topology
- default sampling periods
Constraint Violation %

SPND violates $\rho \geq 0.6$

$\rho = 1.0$

Khronos $\sim 0.32\%$

3x less than DSP
Range of Completeness Constraints (3/3)

- **Prediction Error (s)**
  - $\text{PE(Khr)} < \text{PE(DSP)}$
  - $\text{PE(Khr)} \sim \text{SPND/STO}$
  - $\rho = 1.0$
    - $\text{PE(Khr)} < \text{PE(DSP)}$
    - $\text{CV(Khr)} < \text{CV(DSP)}$

---

![Graph showing comparison of prediction errors for different completeness constraint levels.](image)
Conclusion
Conclusion

› CPS integrated with critical physical processes
  › e.g. manufacturing, healthcare, smart grids
› reacting **timely** under **complete** information is **crucial**
  › **heterogeneity** and **dynamism**
  › platform, network and application
Conclusion

› Khronos

› trade-off **timeliness vs completeness** in CPS applications

› specification of completeness **constraints**

› **automatically** determine timeouts

› improve timeliness

› lift burden of manual timeouts from developer
Conclusion

- Extensive evaluation on physical testbed
  - dynamism
  - heterogeneity
- Khronos outperforms alternative approaches
  - *consistent* constraint satisfaction
  - *smaller* timeouts
    - up to two order(s) of magnitude
Thank you!

Email: stefanos.peros@cs.keluven.be
Repository: https://github.com/mazerius/khronos

Future Work
Future Work

› Online training for sensitivity factor K
  › smaller deployment overhead
  › e.g. incremental learning, control theory, …

› Reactive Programming
  › suitable for CPS application development[3,4]
  › integrate Khronos API with ReactiveX framework(s)
Motivation

› why RTO?
  › durable solution
  › on top of wide, heterogeneous, dynamic infrastructure
  › lightweight
    › 2x EWMA (SRTT and SAT)
register constraint

```javascript
1 // register 25% completeness constraint for device 'LightSensor1'
2 // update average light value when packet arrives within timeout
3 // create pop-up on screen when timeout occurs
4 // write error message to log file when constraint is violated
5 registerCompleteness('LightSensor1', 0.25, updateAverage(data),
6     alert('Timeout!'), logger.write('Constraint Violation!'))
```

register (static) timeout

```javascript
8 // register static timeout of 40 seconds for device 'LightSensor1'
9 // update average light value when packet arrives within timeout
10 // create pop-up on screen when timeout occurs
11 registerTimeout('LightSensor1', '40s', updateAverage(data),
12     alert('Timeout!'))
```
Network

› Real-life SMIP testbed

› 33 devices

› 1x VersaSense Gateway (M01)

› 10x VersaSense wireless devices (P02)
  › 20x peripherals (sensors)

› 22x SMIP motes (DC9003A-B)
  › forward sensor data
resulting $K$ based on TSCH same values used for CSMA/CA

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K$</td>
<td>0</td>
<td>0.1</td>
<td>0.6</td>
<td>1</td>
<td>1.2</td>
<td>1.4</td>
<td>2</td>
<td>2.8</td>
<td>4.6</td>
<td>300</td>
</tr>
</tbody>
</table>
Network Size

- reduced up to 66.67%
- turn off devices
- $\rho = 0.8$
- default topology
- sampling period = 10s
Sampling Period(2/2)

› 240s → 120s → 60s
› every ~24 hours
› $\rho = 0.8$
› default topology
Network Latency

- basebw, bwmult
- requires network reset
- $\rho = 0.8$
- default topology
- sampling period 60s
Medium Access Control (1/3)

- TSCCH
- CSMA/CA
- ~ 72 hours per MAC protocol
  - ~ 2 million packets @ gateway
- all devices within 1 meter of gateway
Medium Access Control (2/3)

- Constraint Violation %
  - $\rho = 0.8$
  - only SPND fails constraint

<table>
<thead>
<tr>
<th>Approach</th>
<th>TSCH</th>
<th>CSMA/CA</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>SPND</td>
<td>27.8%</td>
<td>40%</td>
</tr>
<tr>
<td>STO</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Khronos</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>
Medium Access Control (3/3)

- **Prediction Error (s)**
- **PE(Khr) < PE(DSP)**
- **PE(Khr) ~ SPND, STO**
Sampling Period (1/2)

- Constraint Violation %
- $\rho = 0.8$
- default deployment
- sampling periods: 10s, 60s, 120s, 900s
- SPND always fails constraint

<table>
<thead>
<tr>
<th>Approach</th>
<th>10s</th>
<th>60s</th>
<th>120s</th>
<th>900s</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>SPND</td>
<td>21.5%</td>
<td>20.3%</td>
<td>25.16%</td>
<td>16.18%</td>
</tr>
<tr>
<td>STO</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Khronos</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>
Sampling Period (2/2)

- Prediction Error (s)
- PE(DSP) > PE(Khr)
  - \( \propto \) sampling period
- PE(Khr) \( \sim \) SPND, STO
Network Topology (1/3)

- Two topologies
  - topology **A**: within 1 meter of the gateway
  - topology **B**: up to two floors away from gateway
- ~ 72 hours of data per topology
  - ~ 2 million packets @ gateway
Network Topology (2/3)

- Constraint Violation %
  - $\rho = 0.8$
- default sampling rates
- SPND & DSP violate the constraint

<table>
<thead>
<tr>
<th>Approach</th>
<th>Topology A</th>
<th>Topology B</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP</td>
<td>0%</td>
<td>0.045%</td>
</tr>
<tr>
<td>SPND</td>
<td>27.8%</td>
<td>42.8%</td>
</tr>
<tr>
<td>STO</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Khronos</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>
Network Topology (3/3)

- Prediction Error (s)
- $\rho = 0.8$
- $\text{PE}(\text{DSP}) > \text{PE}(\text{Khr})$
- $\text{PE}(\text{Khr}) \sim \text{SPND}, \text{STO}$