Coordinated City-Scale Traffic Management using Quartz “Time-as-a-Service”

Sandeep D’souza*, Heiko Koehler^, Akhilesh Joshi^ and Raj Rajkumar*

* Carnegie Mellon University, ^ Nutanix Inc.
Traffic-Control Demo Scenario

- **Sunnyvale Traffic Scenario**
  - 16 intersections
  - 40 vehicles per hour per lane-Km

- **TrafFlow: Distributed Traffic Control**
  - *Generate, Train, Deploy* and *Run*
    - Distributed Traffic Control in simulation
  - SUMO Traffic-Simulation Framework
  - *Deep RL-based* intersection controllers
Traffic-Control Demo Scenario

- Sunnyvale Traffic Scenario
  - 16 intersections
  - 40 vehicles per hour per lane-Km
- TrafFlow: Distributed Traffic Control
  - *Generate, Train, Deploy* and *Run* Distributed Traffic Control in simulation
  - SUMO Traffic-Simulation Framework
  - *Deep RL-based* intersection controllers

**Synchronized Clocks are useful**

- clocks help *order events*, build *distributed state*
- clocks enable *distributed-coordinated* scheduling
Enabling Coordination at Scale

● Quality of Time (QoT)
  ○ *end-to-end uncertainty* in the notion of time
  ○ each timestamp has bounds $t \in \{t - \varepsilon_l, t + \varepsilon_h\}$
  ○ helps applications *detect* clock-sync failure

*Anwar et al., RTSS '16*
Enabling Coordination at Scale

- **Quality of Time (QoT)**
  - *end-to-end* uncertainty in the notion of time
  - each timestamp has bounds $t \in \{t-e_l, t+e_h\}$
  - helps applications detect clock-sync failure

- **Timeline***: Virtual reference time base

- **Coordinated actions** on distributed components
  - all components bind to a common timeline
  - each specifying its required QoT

*Anwar et al., RTSS '16*
Enabling Coordination at Scale

- **Quality of Time (QoT)**
  - *end-to-end* uncertainty in the notion of time
  - each timestamp has bounds $t \in \{t - \varepsilon_i, t + \varepsilon_h\}$
  - helps applications detect clock-sync failure

- **Timeline**: Virtual reference time base

- **Coordinated actions** on distributed components
  - all components bind to a common timeline
  - each specifying its required QoT

*Anwar et al., RTSS ’16*
Enabling Coordination at Scale

- **Quality of Time (QoT)**
  - *end-to-end* uncertainty in the notion of time
  - each timestamp has bounds $t \in \{t-\varepsilon, t+\varepsilon\}$
  - helps applications *detect* clock-sync failure

- **Timeline***: *Virtual reference* time base

- **Coordinated actions** on distributed components
  - all components *bind* to a common timeline
  - each *specifying* its required QoT

Timelines *abstract* away clock synchronization
→ Applications *specify* QoT requirements, TaaS *orchestrates* the system

*Anwar et al., RTSS ’16*
Quartz Time-as-a-Service (TaaS)

- Adapts to Application QoT Demands
  - *tunable* clock synchronization
  - probabilistic QoT-estimation mechanisms
- Autonomous & Fault-Tolerant
  - *adapts* to clock-sync failures
  - *notifies* apps if QoT degrades beyond spec
- Built for Containerized Applications
  - user-space *micro-service* implementation
- Easy-to-use API
  - *TimelineBinding* class (C++ & Python bindings)
Quartz Time-as-a-Service (TaaS)

- Adapts to Application QoT Demands
  - *tunable* clock synchronization
  - probabilistic QoT-estimation mechanisms
- Autonomous & Fault-Tolerant
  - *adapts* to clock-sync failures
  - *notifies* apps if QoT degrades beyond spec
- Built for Containerized Applications
  - user-space *micro-service* implementation
- Easy-to-use API
  - *TimelineBinding* class (C++ & Python bindings)

Micro-service architecture → *meeting* QoT requirements + *maintaining* timelines over a wide area
Quartz: Wide-area Deployment

Cluster 1

Cluster Node 1

App 1
Timeline Service
QoT Clock-Sync Service

App 2

App 3

App 4
Timeline Service
QoT Clock-Sync Service

App 5

Cluster Node N

Cluster N

Cluster Node 1

App 1
Timeline Service
QoT Clock-Sync Service

App 2

App 3

App 4
Timeline Service
QoT Clock-Sync Service

App 5

Cluster Node N

Coordination Service

Distributed Configuration & Discovery Service