NeSTiNg
A Network Simulator for Time-sensitive Networking
David Hellmanns
Agenda

- Our Motivation
- Discrete Event Simulation (DES): A primer
- Implementation
- Evaluation of Simulation Results
- Conclusion & Future Work
Our Motivation

• TSN **evaluation** for our research
  • NeSTiNg started as an internal tool for TSN evaluation
    → Publication of code as a contribution to the research community
  • Simulation of functional behavior, not emulation of real systems

• Additional results of the development
  • Deeper understanding of standards
  • Awareness of edge cases and corner cases
Discrete Event Simulation (DES): A primer

- Global simulation clock
Discrete Event Simulation (DES): A primer

- Global simulation clock: Time domain is discrete
Discrete Event Simulation (DES): A primer

- Global simulation clock: Time domain is discrete
- Simulation framework maintains global event queue
Discrete Event Simulation (DES): A primer

- Global simulation clock: Time domain is discrete
- Simulation framework maintains global event queue
Discrete Event Simulation (DES): A primer

- Global simulation clock: Time domain is discrete
- Simulation framework maintains global event queue
- Simulation framework skips idle periods
Discrete Event Simulation (DES): A primer

- Global simulation clock: Time domain is discrete
- Simulation framework maintains global event queue
- Simulation framework skips idle periods
Discrete Event Simulation (DES): A primer

- Global simulation clock: Time domain is discrete
- Simulation framework maintains global event queue
- Simulation framework skips idle periods

Global Event Queue

\[ T = 4 \]
SendMessage(h0, h3)
Discrete Event Simulation (DES): A primer

- Global simulation clock: Time domain is discrete
- Simulation framework maintains global event queue
- Simulation framework skips idle periods
Discrete Event Simulation (DES): A primer

- Global simulation clock: Time domain is discrete
- Simulation framework maintains global event queue
- Simulation framework skips idle periods
- Processing an event can create new events
Discrete Event Simulation (DES): A primer

- Global simulation clock: Time domain is discrete
- Simulation framework maintains global event queue
- Simulation framework skips idle periods
- Processing an event can create new events

![Global Event Queue diagram]
Building the Model

- A simulation model:
  - Abstraction of the real system
  - Reduced to critical parts

- Which TSN components do we want to simulate?

- Which abstractions can we make for our TSN model?

- Which auxiliary components are necessary to simulate our TSN model?
TSN Components

- Synchronization
- Reliability
- Latency
- Resource Mgmt
TSN Components

- Clock synchronization

- Latency
  - Scheduled Traffic (Qbv)
  - Credit Based Shaper (Qav)
  - Frame-Preemption (Qbu/\[.3br\])
Time-aware Shaping

- Affected Systems:
  - Bridges
  - End stations
- Required capabilities:
  - VLAN $\rightarrow$ PCP
  - Queuing
  - Gate Control
  - Scheduling
- Prerequisite: Time synchronization of network
Building Block: INET Framework

• “An open-source OMNeT++ model suite for wired, wireless and mobile networks.”

• We do not want to reinvent the wheel

• Already implements
  • Ethernet channels and Ethernet components
  • Higher Layer protocols

• Compatibility to INET allows simulation of converged networks by utilizing its higher layer components
Global simulation clock could be used to synchronize clocks of Bridges and End systems

- Global simulation clock cannot be used to simulate drift, jitter or PTP

→ Interface for a user-defined clock model
  - Currently only ideal clock is implemented

- User-defined clock may not simulate every tick
  → only interesting time points are being simulated (DES)

- Interested components can subscribe to ticks
  - Priority queue
  - Clocks notifies subscribed components if a tick occurs
Implementation

- generated.s4
- 0 interfaces
- interfaceTable
- clock
- filteringDatabase
- scheduleSwap
- relayUnit
  - processingDelay[0]
  - queuing[0]
  - lowerLayer[0]
  - D
  - processingDelay[1]
  - queuing[1]
  - lowerLayer[1]
  - D
  - processingDelay[2]
  - queuing[2]
  - lowerLayer[2]
  - D
  - processingDelay[3]
  - queuing[3]
  - lowerLayer[3]
  - D
  - processingDelay[4]
  - queuing[4]
  - lowerLayer[4]
  - D
  - processingDelay[5]
  - queuing[5]
  - lowerLayer[5]
Implementation
Implementation

Incoming frames
Implementation: Frame Preemption (Inbound)
Implementation: Frame Preemption (Inbound)

Distinguish Preemptable Frames and Express Frames
Implementation: Frame Preemption (Inbound)
Implementation: Frame Preemption (Inbound)
Implementation: Frame Preemption (Inbound)

Preemption of frame is unforeseeable
Implementation: Frame Preemption (Inbound)
Implementation: Frame Preemption (Inbound)
Implementation: Frame Preemption (Inbound)
Implementation: Processing Delay

Adding bridge-specific delay
Implementation: Relay Unit

Selecting destination port
Implementation: Queuing
Implementation: Queuing

Assign frame to queue
Implementation: Transmission Selection Algorithm

Transmission Selection Algorithm:
- Strict-priority
- Credit-based
Implementation: Transmission Selection Algorithm

Transmission Selection Algorithm:
- Strict-priority
- Credit-based
Implementation: Transmission Selection Algorithm

Interaction between **Gating**, **Frame Preemption**, and **Credit-based Shaper** require complex event handling strategies.

→ Increasing credit at each tick is not desired

Transmission Selection Algorithm:
- Strict-priority
- Credit-based
Implementation: Gating mechanism
Implementation: Transmission Selection

Selecting next frame for transmission (incl. Length-aware Scheduling)
Order of events in Global Event Queue does not necessarily correspond to priority of frames

→ All events at t needs to be processed (incl. consequential events) before decision can be made

Selecting next frame for transmission (incl. Length-aware Scheduling)
Implementation: Transmission Selection

q0  q1  q2

Global Event Queue

Transmission Selection

* Assumption: Gates of all three queues are open
Implementation: Transmission Selection

Assumption: Gates of all three queues are open
Implementation: Transmission Selection

* Assumption: Gates of all three queues are open
Implementation: Transmission Selection

* Assumption: Gates of all three queues are open
Implementation: Frame Preemption (Outbound)
Implementation: Frame Preemption (Outbound)
Implementation: Frame Preemption (Outbound)
Analysis of Simulation Results

- OMNeT++ logs general data automatically
  - For special interests (e.g. deadlines) additional code is mandatory

- scavetool: An internal result analysis tool of OMNeT++
  - Can be used to have a casual view of the data
  - Is capable of filtering, sorting, and joining data to prepare it for export

- Toolchain for extensive evaluation of results
  - scavetool for data export
    - (Excel)
    - Python SciPy Stack / R
Analysis of Simulation Results
Conclusion & Future Work

- NeSTiNg implements a basic set of TSN synchronous shaping features
- Standard compliance is a key driver
- Publication of code as contribution to the research community
- Compatibility with INET allows simulation of converged networks

Future Work
- Migration to INET 4.0
- Further extension of the model (e.g. Ingress Policing and Filtering)
- Extensive evaluation
- Validation

- https://1.ieee802.org/protocol-simulations/