Infrastructure for
Time Sensitive Networking (TSN)
in the INET Framework
Background

• Application Areas
  – Originated from large multi-media networks
  – Nowadays used in industrial and in-vehicle networks

• IEEE 802.1

• Goals
  – Bounded latency
  – Low packet delay variation
  – Low packet loss
  – Fault tolerance
Missing Pieces from INET

- Clock models
- Time synchronization protocols (not covered, e.g. gPTP)
- Aborting transmissions
- Cut-through transmissions
- Versatile queueing model
- Composable Ethernet model
- Synchronous intra-node packet streaming
Covered Topics

Clock Model
Queueing Model
Packet Streaming
Enhanced Transmission Support
Composable Ethernet Model
Clock Model

- Clock time vs. simulation time
- Clock drift (e.g. temperature dependence, thermal noise)
- Hardware precision vs clock accuracy
- Separate oscillator models
- Avoid periodic events
Available Models

- Clock models
  - IdealClock
  - OscillatorBasedClock\{oscillator, initialTime\}
  - SettableClock extends OscillatorBasedClock

- Oscillator models
  - IdealOscillator\{tickLength\}
  - ConstantDriftOscillator\{nominalTickLength, tickOffset, driftRate\}
  - RandomDriftOscillator\{changeInterval, driftRateChange\}
Using Clocks

- Optional submodules in network interfaces and network nodes
- **Subclass from** `ClockUserModuleBase or ClockUserModuleModuleMixin<T>`
- **Use inherited methods or** `IClock` **and** `IOscillator` **C++ interfaces**
- **Usage is similar to standard event scheduling mechanism**
  - `simtime_t vs clocktime_t`
  - `getClockTime()`
  - `scheduleClockEventAt(time, event)`
  - `scheduleClockEventAfter(delay, event)`
  - `cancelClockEvent(event)`
Covered Topics

Clock Model
Queueing Model
Packet Streaming
Enhanced Transmission Support
Composable Ethernet Model
Queueing and Traffic Conditioning

- Traffic shaping
  - Delaying
  - Reordering
  - Metering
  - Classification

- Traffic policing
  - Marking
  - Dropping

- Queueing
  - Prioritizing
  - Shared buffering
  - Gating
  - etc.
Transmission Selection

- Select next transmitted frame based on
  - Priority, frame length, packet data, meta data
  - Timing constraints and guard intervals
  - Credits for channel use
  - etc.
Gating Mechanism

- Gates can be open or closed based on
  - Predefined periodic scheme
  - Available credits
  - etc.

Nth cycle

1st period

2nd period

VLAN priority 3

VLAN priorities 7,6,5,4,2,1,0

Time [s]
Simple Queueing Model Elements

- Sources and sinks
- Queues, buffers, and servers
- Classifiers and schedulers
- Filters and gates
- Meters and markers
- Multiplexer, demultiplexer, and delay
- etc.
Compound Queueing Model Elements

- Priority queues
- Shared buffer queues
- Queues with gating mechanism
- Traffic shapers and traffic conditioners
- Traffic sources for applications
- Request-response traffic generator applications
Protocol Support Model Elements

- Transmitters and receivers
- Inter-packet gap inserter
- Fragmenters and defragmenters
- Aggregators and deaggregators
- Padding and CRC inserters and checkers
Communication between Modules

- Intra-node or inter-node
- Asynchronous (message sending)
- Synchronous (C++ function call)
Packet Processing and Queueing API

● Sources and sinks
  - IActivePacketSource, IPassivePacketSink
  - IActivePacketSink, IPassivePacketSource

● Queues and buffers
  - IPacketCollection, IPacketQueue, IPacketBuffer

● Classifiers, schedulers, filters, gates, etc.
  - IPacketClassifier, IPacketScheduler, IPacketFilter, IPacketGatee, etc.
Active Source and Passive Sink Interface

The **IPassivePacketSink** C++ interface

- `pushPacket(packet, gate)`
- `canPushPacket(packet, gate)`
- `pushPacketStart(packet, gate, datarate)`
- `pushPacketEnd(packet, gate)`
- `pushPacketProgress(packet, gate, datarate, position)`

The **IActivePacketSource** C++ interface

- `handleCanPushPacketChanged(gate)`
- `handlePushPacketProcessed(packet, gate, bool)`
Synchronous Packet Flow

- Pushing a packet

- Pulling a packet

- Supports backpressure
Covered Topics

Clock Model
Queueing Model
Packet Streaming
Enhanced Transmission Support
Composable Ethernet Model
Passing vs Streaming a Packet

- **Passing a packet**
  - Active Module \(\rightarrow\) Passive Module
  - asynchronous send passes whole packet

- **Streaming a packet**
  - start
  - progress (conceptually at every bit)
  - end
  - Remaining part — **may change**
  - position \([b] \text{ and } [s]\)
  - datarate \([bps]\)
  - length \([b] \text{ and } [s]\)
Streaming a Packet over Time

- Transferred data [b]:
  - 1000 B
  - 600 B
  - 200 B

- Simulation time [s]:
  - 10us
  - 22us
  - 28us

- Functions:
  - pushPacketStart()
  - pushPacketEnd()
  - pushPacketProgress()
Covered Topics

- Clock Model
- Queueing Model
- Packet Streaming
- Enhanced Transmission Support
- Composable Ethernet Model
Ethernet Frame Preemption

- Long frame delays high priority frame

- Reduce latency by aborting transmission
Ethernet Cut-through Switching

- Store and forward

- Start forwarding as soon as MAC header is received
Covered Topics

Clock Model
Queueing Model
Packet Streaming
Enhanced Transmission Support
Composable Ethernet Model
Ethernet Standards

- Several related IEEE 802.1 and 802.3 standards
  - br, AS, Qav, Qat, Qcc, Qch, Qci, Qbv, Qbu, Qcr, Qca, CB, CS, Qdd, ABdh
  - Many combinations
  - Different interpretations
  - Complex behavior
- Need composable Ethernet model
Ethernet Protocol Layer

Separate protocol layer in network nodes

Network node fragment
Some submodules are optional, not used by default

No-op submodules disappear at runtime
Ethernet MAC Layer

Uses several generic modules

- queue
- server
- streamer
- fcSinserter
- outboundEmitter

- fcChecker
- inboundEmitter
Ethernet Preempting MAC Layer

Combines two Ethernet MAC Layers with several generic modules.
Backpressure from transmitter goes up to the server in the MAC layer.
Examples

Queueing Model
Ethernet Frame Preemption
Ethernet Cut-through Switching
Telnet Application Example

Telnet client

characterProducer → enterProducer → emptyProducer
consumer → scheduler

"Network"

client → server

Markov chain-based scheduler
Token-based server

Telnet server

responseProvider
responseServer
cloner
classifier
characterConsumer
enterTokenGenerator
multiplexer
Examples

Queueing Model
Ethernet Frame Preemption
Ethernet Cut-through Switching
Ethernet Frame Preemption Example

- Legend
  - Ongoing packet streaming
  - Direction of information flow

![Diagram](image.png)
Ethernet Frame Preemption Example
Ethernet Frame Preemption Example
Ethernet Frame Preemption Example
Ethernet Frame Preemption Moment

packet log

<table>
<thead>
<tr>
<th>#</th>
<th>Time</th>
<th>Host 1</th>
<th>Host 2</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>0.00001162</td>
<td>host1</td>
<td>host2</td>
<td>background:0</td>
</tr>
<tr>
<td>31</td>
<td>0.00001162</td>
<td>host1</td>
<td>host2</td>
<td>background-0-frag0:start</td>
</tr>
<tr>
<td>35</td>
<td>0.00001738</td>
<td>host1</td>
<td>host2</td>
<td>background-0-frag0:end</td>
</tr>
<tr>
<td>37</td>
<td>0.00001834</td>
<td>host1</td>
<td>host2</td>
<td>ts-0</td>
</tr>
<tr>
<td>40</td>
<td>0.00002466</td>
<td>host1</td>
<td>host2</td>
<td>ts-0:end</td>
</tr>
<tr>
<td>45</td>
<td>0.00002562</td>
<td>host1</td>
<td>host2</td>
<td>background-0-frag1</td>
</tr>
<tr>
<td>53</td>
<td>0.00012114</td>
<td>host1</td>
<td>host2</td>
<td>background-0-frag1:end</td>
</tr>
</tbody>
</table>
**Ethernet Frame Preemption in Wireshark**

Analyzing a recorded PCAPng trace

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>216</td>
<td>0.018455</td>
<td>0a:aa:00:00:00:01</td>
<td>0a:aa:00:00:00:02</td>
<td>FPP</td>
<td>845</td>
<td>[Initial fragment: SMD-S1]</td>
</tr>
<tr>
<td>217</td>
<td>0.018462</td>
<td>10.0.0.1</td>
<td>10.0.0.2</td>
<td>UDP</td>
<td>79</td>
<td>1026 → 1001 Len=25</td>
</tr>
<tr>
<td>218</td>
<td>0.018497</td>
<td>10.0.0.1</td>
<td>10.0.0.2</td>
<td>UDP</td>
<td>421</td>
<td>1025 → 1000 Len=1200</td>
</tr>
</tbody>
</table>

- Frame 218: 421 bytes on wire (3368 bits), 421 bytes captured (3368 bits) on interface eth0
- IEEE 802.3br Frame Preemption Protocol
  - Preamble: 0x5555555555555
  - mData: 3f3f3f3f3f3f3f3f3f3f3f3f3f3f3f3f3f3f3f3f3f3f3f3f3f3f3f3f
  - [2 Message fragments (1242 bytes): #216(833), #218(409)]
    - [Frame: 216, payload: 0-832 (833 bytes)]
    - [Frame: 218, payload: 833-1241 (409 bytes)]
    - [Message fragment count: 2]
    - [Reassembled fpp length: 1242]
  - CRC: 0x948e1777 [correct]
- Ethernet II, Src: 0a:aa:00:00:00:01 (0a:aa:00:00:00:02), Dst: 0a:aa:00:00:00:02 (0a:aa:00:00:00:02)
  - Destination: 0a:aa:00:00:00:02 (0a:aa:00:00:00:02)
  - Source: 0a:aa:00:00:00:01 (0a:aa:00:00:00:01)
  - Type: IPv4 (0x8000)
- Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.2
- User Datagram Protocol, Src Port: 1025, Dst Port: 1000
- Data (1200 bytes)
Examples

Queueing Model
Ethernet Frame Preemption
Ethernet Cut-through Switching
Ethernet Cut-through Example
Ethernet Cut-through Example
Ethernet Cut-through Example
Ethernet Cut-through in the Sequence Chart

<table>
<thead>
<tr>
<th>20us</th>
<th>27us</th>
<th>28us 240ns</th>
<th>30us</th>
<th>80us</th>
<th>128us</th>
<th>129us</th>
<th>130s 130us 850ns</th>
</tr>
</thead>
</table>

- **host1**
- **switch1**
- **switch2**
- **host2**

Source-0

Source-0: end

Source-0

Source-0: end

Source-0

Source-0: end

Source-0

Source-0: end
Closing

- Applicability points beyond TSN
  - Queueing model
  - Packet streaming
- Standalone teaching material
  - Queueing model
  - Protocol support
Thank you for your attention!