Dampers with Forward Traffic Isolation

Johannes Specht, University of Duisburg-Essen
Introduction

ATS (P802.1Qcr)
- Bounded delay, robust, integrated policing

Related work
- Concept known: DJ-Regulators/Dampers
- Bounded delay and bounded jitter without global synchronization/[g]PTP
- Challenge: Integrity, Traffic Isolation

This Slidedeck
- How it works: Rate-based Shaping (ATS) vs. Damping
- Pros and Cons
- Forward Traffic Isolation (new)

No Goal: Let’s do this in P802.1Qcr
Dampers
Initial Assumptions and Simplifications

1. **Perfect cables:** No propagation delays
2. **Simple Bridges:** No delays in relays & MACs and cables, no oscillator variations, no numeric imprecision, no gates, no preemption, etc.
3. **Two-level queuing model:** FIFO $\rightarrow$ shaper $\rightarrow$ FIFO
4. **Single hop:** Bridge A $\rightarrow$ Bridge B
5. **Two traffic classes:** Shaped class (High), Best Effort (Low)
6. **Simple traffic:** Periodic small frames, sporadic large best effort frames
Rate-based Shaping (e.g., P802.1Qcr)

- **Per-hop delay**: \( d_{A,i} \)
- **Symbols**
  - \( s_k \): Shaper with associated with Bridge \( k \)
  - \( q_{TX/RX,k} \): FIFO queues associated with Bridge \( k \)
  - \( d_{A,i} \): Delay of the \( i \)th frame from A (\( s_z \) to \( s_A \))

For Illustration
Bad FCS before Bridge A, talker didn’t send, etc.

Rate-based Shaping
“forced spacing”, smoothens bursts

Delay variation, a.k.a. Jitter
1. A pre-configured per-hop delay bound $d_{\text{max},k}$
   - Trust me ... again – not too complicated, cmp. ATS
   - Similar to CQF cycle duration – though it can differ per hop

2. Define $d_{\text{TX},A,i}$ and $d_{\text{RX},A,i}$
   - $d_{\text{TX},k,i}$: post-shaper residence time in the upstream Bridge/Station
   - $d_{\text{RX},k,i}$: pre-shaper residence time in the downstream Bridge

3. Transfer $d_{\text{TX},k,i}$ per frame $\rightarrow$ Dynamic Packet State
   - Encoding is not the main point here (this is not a Standard!)
   - Data integrity addressed later

4. Shape differently $\rightarrow$ Force $d_{\text{RX},k,i} = d_{\text{max},k} - d_{\text{TX},k,i}$
   - I know, S&F, ..., would just add more symbols to my slides (this is not a Standard!)

Symbols:
- $s_k$: Shaper with associated with Bridge $k$
- $q_{\text{TX/RX},k}$: FIFO queues associated with Bridge $k$
- $d_{A,i}$: Delay of the $i$th frame from A ($s_z$ to $s_A$)
- $d_{\text{max},A}$: Per-hop delay bound for A
- $d_{\text{TX},A,i}$: Residence time in $q_{\text{TX},A}$
- $d_{\text{RX},A,i}$: Residence time in $q_{\text{RX},A}$ and S&F
Damping Illustrated

Per-hop delay $d_{A,i} \leq d_{\text{max},A}$

Symbols
- $s_k$: Shaper with associated with Bridge $k$
- $q_{\text{TX/RX},k}$: FIFO queues associated with Bridge $k$
- $d_{A,i}$: Delay of the $i$th frame from A ($s_z$ to $s_A$)
- $d_{\text{max},A}$: Per-hop delay bound for A
- $d_{\text{TX},A,i}$: Residence time in $q_{\text{TX},A}$
- $d_{\text{RX},A,i}$: Residence time in $q_{\text{RX},A}$ and S&F

No delay variation, a.k.a. Jitter!
Example Use-Case: FRER Path-Delay Balancing

Description
• Post-merging burstiness nearly identical to the pre-duplication burstiness
• MatchRecoveryAlgorithm sufficient
• More discussion: https://opus.bibliothek.uni-wuerzburg.de/frontdoor/index/index/docId/20582

Symbols
\( d_{\text{max},k} \): Per-hop delay bound for \( k \)

![Diagram showing network nodes A, D, B, M with delays labeled: \( d_{\text{max},k} = 5\ \mu s, 10\ \mu s, 15\ \mu s, 30\ \mu s \). Duplicates arrive simultaneously.
Pros and Cons
Pros and Cons

Pros

• **Low/no Jitter**

• **No state** (Shaper FSMs): All information in Dynamic Packet State

• Should work with simplified ATS queuing (**“interleaved shaping”**), i.e. no FIFO queue per flow needed.

• [g]PTP **Hardware re-use**

Cons

• **Increased Overhead** for Dynamic Packet State

• FCS re-calculation per Hop required → **Decreased data integrity**

• No state (Shaper FSMs): → **No protection and isolation against malicious traffic/babbling idiots!**
Protection & Isolation
Babbling Idiot Impact (e.g., Frame Repetition)

Symbols
- $s_k$: Shaper with associated with Bridge $k$
- $q_{TX/RX,k}$: FIFO queues associated with Bridge $k$
- $d_{max,k}$: Per-hop delay bound for Bridge $k$
- $d_{TX,k,i}$: Residence time in $q_{TX,k}$
- $d_{RX,A,i}$: Residence time in $q_{RX,k}$

“Blindly” executes

\[ d_{RX,A2,i} = d_{max,A2} - d_{TX,A2,i} \ldots \]

... grows towards $\infty$ over time!

Note: No BE frames and S&F delays shown (unnecessary for illustration).

6/15/2020 Johannes Specht, University of Duisburg-Essen
**Is this an Issue? – Depends on the Network**

<table>
<thead>
<tr>
<th></th>
<th>Low Protection</th>
<th>High Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional</td>
<td>Low State Req.</td>
<td>High State Req.</td>
</tr>
<tr>
<td>Dependable</td>
<td>High State Req.</td>
<td>Low State Req.</td>
</tr>
</tbody>
</table>

### Case 1: Conventional Networks

The edge (=Station) is considered problematic, the core (=Bridges) is considered to never fail (or if it does, only fail silent is considered).

- **Protection:** Edge Bridge Ports only  
  (i.e., Bridge ports connected to Stations)
- **State:** Edge Bridge Ports only  
  (# of Streams from a single Station is limited)

### Case 2: Dependable Networks

It doesn’t matter whether Station or Bridge. Devices can fail arbitrarily according to their failure rate (MTBF, etc.). And we don’t know how (i.e., babbling idiot behavior) ...

- **Protection:** Every Bridge Port  
  (no matter whether it’s a Station or a Bridge upstream)
- **State:** Every Bridge Port in every Bridge  
  (“Per-stream Filtering and Policing” in every Port)
Case 1: Conventional Networks

- The edge (=Station) is considered problematic, the core (=Bridges) is considered to never fail (or if it does, only fail silent).
- Protection: Edge Bridge Ports only (i.e., Bridge ports connected to Stations)
- State: Edge Bridge Ports only (# of Streams from a single Station is limited)

Case 2: Dependable Networks

- It doesn't matter whether Station or Bridge. Devices can fail arbitrarily according to their failure rate (MTBF, etc.). And we don't know how (i.e., babbling idiot behavior).
- Protection: Every Bridge Port (no matter whether it's a Station or a Bridge upstream)
- State: Every Bridge Port in every Bridge ("Per-stream Filtering and Policing" in every Port)
Forward Traffic Isolation (FTI)
Forward Traffic Isolation (FTI)

– Key Concepts

1. Enhanced PSFP on edges only
   • Enhanced Flow meters ("PSFP+")
   • Max. SDU size filtering

2. Additional Validation Data in Frames
   • Part of Dynamic Packet State (DPS)

3. Exploit Redundant HW on Paths
   • Example: One bridge with $10^{-6}$ failure/h → two nodes with $\sim10^{-12}$ failure/h
   • FTI interleaves along the path – validation data tunneled through the next (potentially faulty) Bridge downstream

4. Validation Data is Signed
   • Asymmetric: Read/verify with public key, modification requires private key
   • Important notes:
     • Signature algorithms against HW faults, not necessarily against intelligent/human attacks
       → less computation, several literature on this topic
       (e.g., K. Echtle and T. Kimmeskamp, Fault-Tolerant and Fail-Safe Control Systems Using Remote Redundancy, 22th International Conference on Architecture of Computing Systems 2009)
     • Symmetric signatures (e.g., CRCs) are possible, but with more DPS and “clever” key distribution
       → subsequent slides stick to asymmetric concepts
Failure Assumptions

1. One “Box” fails at a time
   
   We can support more, but this one is simple and enough for illustration, plus system failure probability already goes notably lower.

2. A faulty box cannot find out the private key another fault free box
   
   A faulty box has a private key, but this is different than the private keys of its upstream neighbors 1 and 2 hops upwards. It cannot “find out” the other boxes’ private key by e.g. random hardware faults.
FTI - Keys, Roles, Dynamic Packet State (DPS)

Talker T
- Pub. Key(s): T
- PSFP+ Check; FTI FSM: Check/Update
- Eligibility time
  - Of \( s_X, s_T, \ldots \) and “sibling” shapers in \( T, A, \ldots \)
  - Can overflow (later slides)

Bridge A
- Pub. Key(s): T
- FTI FSM: Check/Update
- \( d_{TX,T,i} \rightarrow v_{T,i} \)
- \( v_{T,i} \rightarrow v_{A,i} \)
- \( v_{A,i} \rightarrow v_{B,i} \)
- \( v_{B,i} \rightarrow v_{C,i} \)
- As-is (earlier slides)

Bridge B
- Pub. Key(s): T, A
- FTI FSM: Check/Update
- \( d_{TX,T,i} \rightarrow v_{T,i} \)
- \( v_{T,i} \rightarrow v_{A,i} \)
- \( v_{A,i} \rightarrow v_{B,i} \)
- \( v_{B,i} \rightarrow v_{C,i} \)

Bridge C
- Pub. Key(s): A, B
- FTI FSM: Check/Update
- \( d_{TX,A,i} \rightarrow v_{A,i} \)
- \( v_{A,i} \rightarrow v_{B,i} \)
- \( v_{B,i} \rightarrow v_{C,i} \)

DPS
- length (T’s view)
- etime (T’s view)
- signature (over the former, T’s private key)

Eligibility time
- Of \( s_X, s_T, \ldots \) and “sibling” shapers in \( T, A, \ldots \)
- Can overflow (later slides)
First Bridge: Enhanced Flow Meters for Dampers (“PSFP+”)

Summary

- Flow meters algorithm executed after computation $d_{RX,k,i} = d_{max,k} - d_{TX,k,i}$
- Eliminates jitter after shaper output
- Flattens the arrival curve (i.e., lower CommittedBurstSize values possible)
FTI: Simplified FSM in Bridges

Note: Preliminary/untested

6/15/2020 Johannes Specht, University of Duisburg-Essen

Simplified FSM in Bridges

**ALIGNED**

```
rFrameEna = FALSE;
```  

**UNALIGNED**

```
rFrameEna = FALSE;
```  

/* Check the signatures */

if (!isSigValid(frame.v[0])) break;

if (frame.v[1] != null && isSigValid(frame.v[1])) break;

/* Check the frame lengths */

if (!isLengthValid(frame.length, frame.v[0].length)) break;

if (frame.v[1] != null && !isLengthValid(frame.length, frame.v[1].length)) break;

etime = d_max - frame.d;

/* Align, if unaligned */

if (!aligned) {
  tLastFrame = now;
  delta0 = frame.v[0].etime - etime;
  if (frame.v[1] != null) delta1 = frame.v[1].etime - etime;
  aligned = TRUE;
  proceed(frame, etime);
  break;
}

/* Check eligibility time consistency */

if (delta0 != frame.v[0].etime - etime)
  break;

if (frame.v[1] != null) && (delta1 != frame.v[1].etime - etime) break;

/* Update time, forward the frame (i.e., not discarded) */

if (delta0 != frame.d || delta1 != frame.d) break;

/* Transient */

Frame frame;
Time etime;
Time now;

/* Scope: Per-Rx-Port */

Boolean rFrameEna;

/* Scope: Per-RX-Port-per-Class */

Boolean aligned;
Time tLastFrame;
Time delta0;

/* Scope: Per-Upstream-RX-Port-per-Class */

Time delta1;

/* Helpers */

void forward(Frame frame); // Forwards the frame
Signature sign(Frame frame); // Generates a local signature
Boolean isLengthValid(int l1,l2); // Length consistency check
Boolean isSigValid(Validation v); // Validates a signature

void proceed(Time etime, Frame frame) {
  frame.v[1] = frame.v[0];
  frame.v[0].etime = etime;
  frame.v[0].length = frame.length;
  frame.v[0].signature = sign(frame.v[0].etime, frame.v[0].length);
  forward(frame);
}

Limited Topology Dependency
Like for public keys, the table size here depends on the number of dual-hop upstream devices (see later slides)

Simplified
Lookups/tables not explicitly shown

Local Device Size
Table size depends on the local device’s dimensioning

Simplified
Tolerances for imprecisions (oscillators, limited numeric resolutions, etc.) omitted
FTI – Illustration and FSMs

Symbols

- $s_k$: Shaper with associated with k
- $q_{TX/RX,k}$: FIFO queues associated with k
- $d_{\text{max},k}$: Per-hop delay bound for k
- $d_{TX,k,i}$: Residence time in $q_{TX,k}$
- $v_{k,i}$: FTI information in k’s $i$th frame
- $\Delta_{k,m}$: Diff. between k’s and m’s local time (k’s view)

Check (and handle)

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$ ??

Update (if passed)

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

Align

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

Check (and handle)

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

$\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$
FTI – Illustration and FSMs

Symbols
- $s_k$: Shaper with associated with k
- $q_{TX/RX,k}$: FIFO queues associated with k
- $d_{max,k}$: Per-hop delay bound for k
- $d_{TX,RX,i}$: Residence time in $q_{TX,k}$
- $v_{k,i}$: FTI information in k’s ith frame
- $\Delta_{k,m}$: Diff. between k’s and m’s local time (k’s view)

Check (and handle)
- $v_{t,i}$ length vs. received length
- $\Delta_{A,T} := (v_{A,i}.etime - v_{T,i}.etime) \text{ ?}
- Update (if passed)
  - $\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$

Check (and handle)
- $v_{t,i}$ length vs. received length
- $\Delta_{B,T} := (v_{B,i}.etime - v_{T,i}.etime) \text{ &&}
- $\Delta_{B,A} := (v_{B,i}.etime - v_{A,i}.etime) \text{ ?}
- Update (if passed)
  - $\Delta_{B,T} := v_{B,i}.etime - v_{T,i}.etime$
  - $\Delta_{B,A} := v_{B,i}.etime - v_{A,i}.etime$

Align
- $\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$
- Check (and handle)
  - $v_{t,i}$ length vs. received length
  - $\Delta_{B,T} := v_{B,i}.etime - v_{T,i}.etime$

Check (and handle)
- $v_{t,i}$ length vs. received length
- $\Delta_{A,i} := v_{A,i}.etime - v_{T,i}.etime$

Align
- $\Delta_{B,T} := v_{B,i}.etime - v_{T,i}.etime$$

Differences
- $\Delta_{A,T} := v_{A,i}.etime - v_{T,i}.etime$
- $\Delta_{B,T} := v_{B,i}.etime - v_{T,i}.etime$
- $\Delta_{B,A} := v_{B,i}.etime - v_{A,i}.etime$

Public Key(s): T

Public Key(s): T, A

6/15/2020 Johannes Specht, University of Duisburg-Essen
Some Failure Scenarios for Illustration

**Goal:** Capture malicious traffic immediately after the faulty device. Merge point not shown subsequently, though capturing immediately after the faulty is enough.

**Note:** Compared to earlier slides, the blue path contains the faulty node.
FTI – Faulty T, excessive burst

- Talker T
- Bridge A
  - $q_{RX,A}$
  - Pub. Key(s): T
  - $q_{TX,A}$
- Bridge B
  - $q_{RX,B}$
  - Pub. Key(s): T, A
  - $q_{TX,B}$

Talker T

PSFP+ Check; FTI Check/Update

For $T$, $d_{max,T}$

- $s_x$ out
- $d_{TX,T,1}$ -- $v_{T,1}$
- $d_{TX,T,2}$ -- $v_{T,2}$
- $d_{TX,T,3}$ -- $v_{T,3}$

FTI Insert

$2^{nd}$ frame to close to $1^{st}$ frame

Caught by PSFP+, Committed Burst Size exceeded!

Symbols
- $s_k$: Shaper with associated with $k$
- $q_{TX/RX,k}$: FIFO queues associated with $k$
- $d_{max,k}$: Per-hop delay bound for $k$
- $d_{TX,k,i}$: Residence time in $q_{TX,k}$
- $v_{k,i}$: FTI information in $k$'s $i^{th}$ frame
FTI – Faulty T, bad etime in $v_{T,i}$

**Symbols**
- $s_k$: Shaper with associated with $k$
- $q_{TX/RX,k}$: FIFO queues associated with $k$
- $d_{max,k}$: Per-hop delay bound for $k$
- $d_{TX,k,i}$: Residence time in $q_{TX,k}$
- $v_{k,i}$: FTI information in $k$'s $i$th frame
- $\Delta_{k,i}$: Diff. between $k$'s and $m$'s local time ($k$'s view)

**Diagram Description**
- Talker $T$
- Bridge $A$
- Bridge $B$
- Pub. Key(s): $T$
- Pub. Key(s): $T, A$

**Steps**
1. $s_x$ out
2. $s_T$ int.
3. FSM: $v_{T,i}$
4. $d_{max,T}$
5. $d_{TX,T,1} -- v_{T,1}$
6. $d_{TX,T,2} -- v_{T,2}$
7. $d_{TX,T,3} -- v_{T,3}$
8. $v_{T,3}$'s etime inconsistent with the remaining information
9. Caught by FTI Check, $T$'s offset $\Delta_{A,T}$ known by $A$!
FTI – Faulty A, excessive burst

FTI Insert

PSFP+ Check; FTI Check/Update

FTI Check/Update

3rd frame to close to 2nd frame, $d_{TX,A,i}$ and $v_{A,3}$ consistently wrong

Caught by FTI Check:
→ T’s offset $\Delta_{B,T}$ known by B!
→ T’s $v_{T,i}$ values cannot be “faked” by A!
FTI – Faulty A, excessive burst

Talker T

Bridge A

Pub. Key(s): T

Bridge B

Pub. Key(s): T, A

FTI Insert

PSFP+ Check; FTI Check/Update

FTI Check/Update

s_x out

d_{TX,T,1} \rightarrow v_{T,1}

d_{TX,T,2} \rightarrow v_{T,2}

d_{TX,T,3} \rightarrow v_{T,3}

s_T int.

FSM: v_{T,i}

A out

\[ \text{Align} \]

\[ \text{Check} \]

\[ \text{Align} \]

\[ \text{Check} \]

\[ \text{Align} \]

\[ \text{Check} \]

\[ \text{Align} \]

\[ \text{Check} \]

\[ \text{Align} \]

\[ \text{Check} \]

3rd frame to close to 2nd frame, \( d_{TX,A,i} \) and \( v_{A,3} \) consistently wrong

Caught by FTI Check:

\[ \rightarrow \] T’s offset \( \Delta_{B,T} \) known by B!

\[ \rightarrow \] T’s \( v_{T,i} \) values cannot be “faked” by A!

In fact, it doesn’t matter whether A’s frame is too early or too late. A cannot “fake” T’s \( v_{T,i} \) information, \( v_{T,i}.etime \) in particular. A does not know T’s private key. Same for \( v_{T,i}.length \) (not illustrated).

Symbols

\( s_k \): Shaper with associated with k

\( q_{TX/RX,k} \): FIFO queues associated with k

\( d_{\text{max},k} \): Per-hop delay bound for k

\( d_{TX,k,i} \): Residence time in \( q_{TX,k} \)

\( v_{k,i} \): FTI information in k’s \( i \)th frame

\( \Delta_{k,m} \): Diff. between k’s and m’s local time (k’s view)
FTI – Faulty A, bad etime in $v_{A,i}$

Caught by FTI Check:

$\rightarrow A$'s offset $\Delta_{B,T}$ known by $B$!

**Symbols**
- $s_k$: Shaper with associated with $k$
- $q_{TX/RX,k}$: FIFO queues associated with $k$
- $d_{max,k}$: Per-hop delay bound for $k$
- $d_{TX,k,i}$: Residence time in $q_{TX,k}$
- $v_{k,i}$: FTI information in $k$'s $i$th frame
- $\Delta_{k,m}$: Diff. between $k$'s and $m$'s local time ($k$'s view)
FTI – Faulty B, excessive burst

Symbols:
- \( s_k \): Shaper with associated with \( k \)
- \( q_{TX/RX,k} \): FIFO queues associated with \( k \)
- \( d_{\text{max},k} \): Per-hop delay bound for \( k \)
- \( d_{TX,k,i} \): Residence time in \( q_{TX,k} \)
- \( v_{k,i} \): FTI information in \( k \)'s \( i \)th frame
- \( \Delta_{k,m} \): Diff. between \( k \)'s and \( m \)'s local time (\( k \)'s view)

Note: Case just to simplify illustration how FTI operates along the path
Further Aspects

Not shown in earlier slides
Public Key Distribution

Either static, or via a protocol. A protocol has not been presented, though this is not so critical, given it is the slow, not so critical, path (control plane).

Public Key Identification/Lookup

On frame reception, the associated public key for $v_{k,i}$ values must be identified. This aspect wasn’t covered, though it can be an extra field of $v_{k,i}$ not covered by the signature (think of the following: If a faulty node in the middle “fakes” this field, a wrong public key is selected and signature check fails).
Reducing State Requirements

**Dual-hop Upstream State**

Consider Bridge A has 1000 ports, connected 999 Talkers, and to Bridge B downstream, which is a small 3 Port Bridge. B would require $1000 \Delta_{k,m}$ state variables just to serve these 1000 talkers. However, Bridge A will comprise multiple Chips, ASICs, etc. which can reasonably independent from each other in terms of reliability. There can be multiple FTI check and update points with associated FSMs in Bridge A (e.g., one per ASIC), thus massively reducing the required $\Delta_{k,m}$ state variables in Bridge B (i.e., think of every ASIC in Bridge A is a Bridge itself).
Various

\( v_{k,i}.etime \) Overflows and Timeouts
Each FSMs times out if the time range of \( v_{k,i}.etime \) is exceeded. The FSMs then fall back to unaligned state. A faulty node can exploit this, however, it can at most send one bad frame per time range. The resulting maximum “noise” caused by such a node consumes considerable low bandwidth - appears ok for worst-case consideration.

Missing Frames
Due to FCS errors, different routing, etc. a frame sequence upstream can be incomplete at the next two hops downstream. This is no issue, the exact sequence can contain gaps. It’s just \( \Delta_{k,m} \) state variables that are updated less frequently.

FTI in other Areas
Though dampers provide higher delay-performance, there is e.g. a DPS-based asynchronous Cyclic Queueing and Forwarding derivate (https://datatracker.ietf.org/doc/draft-qiang-detnet-large-scale-detnet/). FTI can be applied here, too, just think of eligibility times with “low resolution” (i.e., cycle numbers).

6/15/2020 Johannes Specht, University of Duisburg-Essen
Summary

Dampers
- Low jitter asynchronous traffic shaping
- Stateless in Bridges
- Dynamic Packet State is used → Integrity is an Issue

Forward Traffic Isolation
- New concept for traffic isolating against babbling idiots
- No 100% solution - residual errors hard to quantify – but qualitatively high degree of protection from an engineers point of view
- Moderate state requirements (i.e., topology dependent, limited to two hops) – typically significantly lower than per flow state
- Scheme applicable in other Areas
Questions, Opinions, Ideas?

Johannes Specht  
Dipl.-Inform. (FH)

Dependability of Computing Systems
Institute for Computer Science and Business Information Systems (ICB)
Faculty of Economics and Business Administration
University of Duisburg-Essen

Schuetzenbahn 70
Room SH 502
45127 Essen
GERMANY
T +49 (0)201 183-3914
F +49 (0)201 183-4573

Johannes.Specht@uni-due.de
http://dc.uni-due.de