Automated Bootstrapping of A Fault-Resilient In-Band Control Plane

Ermin Sakic, Amaury Van Bemten, Mirza Avdic, Wolfgang Kellerer
Technical University Munich & Siemens Germany

ACM SOSR 2020
San Jose, March 3, 2020
INTRODUCTION
Industrial Networks Overview

- **Strict requirements:**
  - **QoS:** Sub-ms hard real-time E2E delays
  - **Dependability:** Control & data plane HA & reliability
  - **Topology dynamics:** factory cell / work-piece (de)-attachment

- **TSN group (802.1) standardizes industrial CP & DP:**
  - E.g., TAS (Qbv), Frame Pre-emption (Qbu), FRER (CB), Policing (Qci) etc.
Industrial Networks Overview

- **Strict requirements:**
  - **QoS:** Sub-ms hard real-time E2E delays
  - **Dependability:** Control & data plane HA & reliability
  - **Topology dynamics:** factory cell / work-piece (de)-attachment

- TSN group (802.1) standardizes **industrial CP & DP:**
  - E.g., TAS (Qbv), Frame Pre-emption (Qbu), FRER (CB), Policing (Qci) etc.
  - Centralized (CNC) and distributed stream reservation

- TSN requires a highly-available CNC w/ in-band, dynamically extensible CP & DP
Industrial Network Topologies
Industrial Network Topologies

VirtuWind – Virtual and programmable industrial network prototype deployed in operational wind park - https://5g-ppp.eu/virtuwind/
Control Plane Design

In-Band

- SDN Controller
  - Control Plane
  - Southbound API

- DATA/CONTROL Network
  - Network device
    - Southbound API
    - Data Plane
  - Network device
    - Southbound API
    - Data Plane

Out-of-Band

- SDN Controller
  - Control Plane
  - Southbound API

- CONTROL Network
  - Network device
    - Southbound API
    - Data Plane
  - Network device
    - Southbound API
    - Data Plane

- DATA Network
Control Plane Design

**In-Band**

**Goal of Bootstrapping:**
Automated establishment of a functional and resilient In-Band SDN control plane

**Required:**
- Initial C2S and C2C connections
- Control plane fault tolerance
- Full topology available (no blocked ports!)
- Network extensions
- Compliant with current implementations

**Constraints:**
- Switches know nothing about the controllers
- Controllers know whitelisted IP addresses of remote controllers (e.g., standardized)
- Switches and controllers exchange PKI certificates
Control Plane Design

High-level steps:

1. Controllers distribute IP addresses to switches from a common pool
2. Controllers provides each switch with controller lists (e.g., OF)
3. Controllers establish control channels to each switch (e.g., OF)
Resilience Requirements

**CP:** Must tolerate $F$ out of $2F+1$ Fail-Stop controller failures

**DP:** Must tolerate $k$ element failures
  - $k+1$ fully or maximally disjoint paths
Bootstrapping Co-Dependency

- DP requires appropriate table rules
- Rule configuration requires C2C
- In-Band C2C requires DP connectivity

→ Break bootstrapping procedure into sub-phases
**Design Overview**

**Contribution:** Two automated bootstrapping schemes for a reliable multi-controller in-band control plane

- **Hybrid Switch Approach (HSW):** Assumes (R)STP
- **Hop-By-Hop Approach (HHC):** No (R)STP

<table>
<thead>
<tr>
<th>Design</th>
<th>Auto. Switch Management IP Provisioning</th>
<th>Auto. Controller List Provisioning</th>
<th>Resilience of Control Flows</th>
<th>Multi-Controller Support</th>
<th>(R)STP Not Required</th>
<th>No Proprietary Switch Extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sharma et al. [39–41]</td>
<td>✓</td>
<td>✓</td>
<td>✓ / X (reactive)</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Schiff et al. I [36–38]</td>
<td>X</td>
<td>X</td>
<td>✓ / X (timeout)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Heise et al. [18]</td>
<td>X</td>
<td>X</td>
<td>✓ (proactive replication)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Canini et al. [7]</td>
<td>X</td>
<td>X</td>
<td>✓ (reactive)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Su et al. [45]</td>
<td>✓</td>
<td>✓</td>
<td>✓ (reactive)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rentel et al. [8]</td>
<td>✓</td>
<td>✓</td>
<td>✓ (proactive replication)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hybrid Switch (HSW)</td>
<td>✓</td>
<td>✓</td>
<td>✓ (proactive replication)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hop-by-Hop (HHC)</td>
<td>✓</td>
<td>✓</td>
<td>✓ (proactive replication)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Why regard (R)STP?

+ Beneficial for effortless initial C2C connectivity

- Dimensioning the (R)STP-disable timer non-trivial
  → Delays in bootstrapping convergence

- Added complexity in the data plane:
  → Prone to additional failure vectors (YMMV)
DESIGN OF
THE TWO SCHEMES
System Initialization

HSW - (R)STP enabled:

- standalone mode
  - Heavy use of NORMAL port

- in-band mode enabled

HHC - (R)STP unavailable:

- secure mode

- in-band mode disabled
  - „generic“ OF rules
System Initialization

HHC: How to fight initial broadcast storms without (R)STP?

Police problematic C2C traffic
(ARP, TCP SYN, TCP SYN ACK)

priority=10, in_port=local, eth_src=SW_MAC, tcp, tcp_dst=6633, actions=all
priority=10, eth_dst=SW_MAC, tcp, tcp_src=6633, actions=local
priority=10, udp, udp_src=67, udp_dst=68, actions=local
priority=10, in_port=local, eth_src=SW_MAC, udp, udp_src=68, udp_dst=67, actions=all
priority=10, in_port=local, eth_src=SW_MAC, tcp, tcp_src=22, actions=all
priority=9, eth_src=SW_MAC, arp, arp_op=2, actions=all
priority=8, arp, arp_op=2, actions=all
priority=7, tcp, tp_dst=2550, actions=normal
priority=7, tcp, tp_src=2550, actions=normal
HSW Phases 0 and 1

Inter-controller synchronization and leader election

- LEADER
- FOLLOWER
- FOLLOWER

Inter-controller synchronization is done concurrently with the bootstrapping procedure

DHCP: Switch IP

SSH: Controllers’ IP:port
HHC Phases 0 and 1

Inter-controller synchronization and leader election

Inter-controller synchronization is done concurrently with the bootstrapping procedure
Output: Phases 0 and 1

**HSW (with (R)STP)**

- C1, C2, C3 IPs
- 10.10.0.31
- 10.10.0.34

**HHC (no (R)STP)**

- C1, C2, C3 IPs
- 10.10.0.31
Phase 2: Resilience Embedding

HSW (with (R)STP):
Step 2a: - Establish OF sessions **FCFS**, install initial rules, disable *in-band* rules
Step 2b: - Disable R(STP)
  - Install resilient flow rules

HHC (no (R)STP):
Step 2a: - Establish OF sessions **Hop-By-Hop**, install tree flow rules
Step 2b: - Install resilient flow rules whenever possible
HSW Phase 2a

- **Phase 0**
  - DHCP: Switch IP
  - SSH: Controllers' IP:port

- **Phase 1**
  - OF handshake
  - OF: Initial rules

- **Phase 2a**
  - SSH: Disable In-band mode

- **Inter-controller synchronization and leader election**
  - Leader
  - Follower

Inter-controller synchronization is done concurrently with the bootstrapping procedure.
<table>
<thead>
<tr>
<th>Service/Process</th>
<th>Protocol</th>
<th>Parameters</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic switch IP config (DHCP)</td>
<td>udp, udp_src=68</td>
<td>actions=normal</td>
<td></td>
</tr>
<tr>
<td>Dynamic switch IP config (DHCP)</td>
<td>udp, udp_src=67</td>
<td>actions=normal</td>
<td></td>
</tr>
<tr>
<td>Remote switch config (SSH)</td>
<td>tcp, tcp_src=22</td>
<td>actions=normal</td>
<td></td>
</tr>
<tr>
<td>Remote switch config (SSH)</td>
<td>tcp, tcp_dst=22</td>
<td>actions=normal</td>
<td></td>
</tr>
<tr>
<td>OpenFlow interaction (OF)</td>
<td>tcp, tcp_src=6633</td>
<td>actions=normal</td>
<td></td>
</tr>
<tr>
<td>OpenFlow interaction (OF)</td>
<td>tcp, tcp_src=6633</td>
<td>actions=normal</td>
<td></td>
</tr>
<tr>
<td>Switch IP resolution (ARP)</td>
<td>arp, arp_tpa=CP IP prefix</td>
<td>actions=normal</td>
<td></td>
</tr>
<tr>
<td>Switch IP resolution (ARP)</td>
<td>arp, arp_sp=CP IP prefix</td>
<td>actions=normal</td>
<td></td>
</tr>
<tr>
<td>Controller IP resolution (ARP)</td>
<td>arp, arp_tpa=cX IP,</td>
<td>actions=normal</td>
<td></td>
</tr>
<tr>
<td>Controller IP resolution (ARP)</td>
<td>arp, arp_sp=cX IP,</td>
<td>actions=normal</td>
<td></td>
</tr>
<tr>
<td>Inter-controller sync (C2C)</td>
<td>ip, ip_src=cX IP, ip_dst=cY IP</td>
<td>actions=normal</td>
<td></td>
</tr>
<tr>
<td>Inter-controller sync (C2C)</td>
<td>ip, ip_src=cX IP, ip_dst=cY IP</td>
<td>actions=normal</td>
<td></td>
</tr>
<tr>
<td>Topology discovery (LLDP)</td>
<td>eth_type=0x88cc,</td>
<td>actions=controller</td>
<td></td>
</tr>
<tr>
<td>Topology discovery (ARP)</td>
<td>arp, arp_tpa=arbitrary IP,</td>
<td>actions=controller</td>
<td></td>
</tr>
</tbody>
</table>
HSW Phase 2b

Inter-controller synchronization and leader election

Phase 0
- DHCP: Switch IP

Phase 1
- SSH: Controllers’ IP:port
- OF handshake
- OF: Initial rules
- SSH: Disable In-band mode

Phase 2a

Inter-controller synchronization is done concurrently with the bootstrapping procedure
HHC Phase 2a

Inter-controller synchronization and leader election

Inter-controller synchronization is done concurrently with the bootstrapping procedure

Phase 0

Phase 1

Phase 2a
DHCP example:

udp, in_port=4, tp_dst=68, actions=output:3, output:2, output:1
udp, in_port=4, tp_dst=67, actions=output:3, output:2, output:1
udp, in_port=3, tp_dst=68, actions=output:4, output:2, output:1
udp, in_port=3, tp_dst=67, actions=output:4, output:2, output:1
udp, in_port=2, tp_dst=68, actions=output:4, output:3, output:1
udp, in_port=2, tp_dst=67, actions=output:4, output:3, output:1
udp, in_port=1, tp_dst=68, actions=output:4, output:3, output:2
udp, in_port=1, tp_dst=67, actions=output:4, output:3, output:2

Also for OF, SSH, ARP

Additionally topology discovery LLDP, ARP rules
Inter-controller synchronization and leader election

Inter-controller synchronization is done concurrently with the bootstrapping procedure.
HHC Phase 2b

Inter-controller synchronization and leader election

- HOP 1
  - S
- HOP 0
  - S

C1, C2, C3 IPs
10.10.0.34
10.10.0.31
10.10.0.32

C1, C2, C3 IPs
C1, C2, C3 IPs
C1, C2, C3 IPs

Phase 0
Phase 1
Phase 2a

SSH: Initial tree rules
Phase 2: Outcome both schemes

$k+1$ max. disjoint paths for C2S pairs \textbf{(here only S4)}

$k+1$ max. disjoint paths for C2C pairs
Dynamic network extensions

- Allow new traffic to reach the leader via tree
  - **HSW**: Prim’s algorithm
  - **HHC**: Custom Hop-By-Hop Algorithm
- Special rule: `in_port=inactive port, udp, udp_src=68, actions=controller`
- Extend tree by parsing DHCP DISCOVERY message
Data Plane Failures

- Proactively compute alternative trees
- Embed an alternative tree in case a DP element fails
EVALUATION
Evaluation - KPIs

- Global Bootstrapping Convergence Time (GBCT)
- Network Extension Time (TEXT)
- Flow Table Occupancy (FTO)
Global Bootstrapping Convergence Time

Single Controller

* normalized by minimum mean ~13.5s
Global Bootstrapping Convergence Time

Multiple Controllers

*normalized by minimum mean ~33.9s
Network Extension Time

Single Controller

* normalized by minimum mean ~6.5s
Network Extension Time

Multiple Controllers

*normalized by minimum mean ~33.5s
Flow Table Occupation

Ratios of per-switch FTOs, normalized respective to the FTO in 1-controller case
SUMMARY
Summary - Pros and Cons

HSW - (R)STP
+ Straightforward; easier to implement
- Dependency on legacy protocols (and implementation)
- Worse performance due to (R)STP Timer

HHC - No (R)STP
+ Less legacy protocol dependencies
+ Faster on average
- Slightly more complex implementation
Artifacts and Future Updates

Source code for both approaches and Docker-based OpenFlow emulator available!

https://github.com/ermin-sakic/sdn-automated-bootstrapping
Artifacts and Future Updates

Source code for both approaches and Docker-based OpenFlow emulator available!

https://github.com/ermin-sakic/sdn-automated-bootstrapping
Artifacts and Future Updates

Potential optimizations:

- Automated rule compression for lower FTO
- Tree merging instead of swapping
- Support for concurrent multi-controller bootstrapping? (RAFT membership issues?)

Source code for both approaches and Docker-based OpenFlow emulator available!

https://github.com/ermin-sakic/sdn-automated-bootstrapping


