Crash Farming and Gang Programming over Ethernet: Old Concepts Meet New Challenges

Tools and Techniques in Managing Address Conflicts Associated with Parallel Testing and Flashing of Identical Ethernet Devices

IEEE-SA Ethernet & IP @ Automotive Technology Day
September 24th, 2019

John Simon,
Product & Applications Manager

Agenda

- Background
- What is a Crash Farm?
- Gang Programming and its similarities
- Ideal vs. Practical Solutions
- Summary & Conclusions

Background

Despite all the lessons learned, the most talented engineers, disciplined processes, and today's incredible development tools……

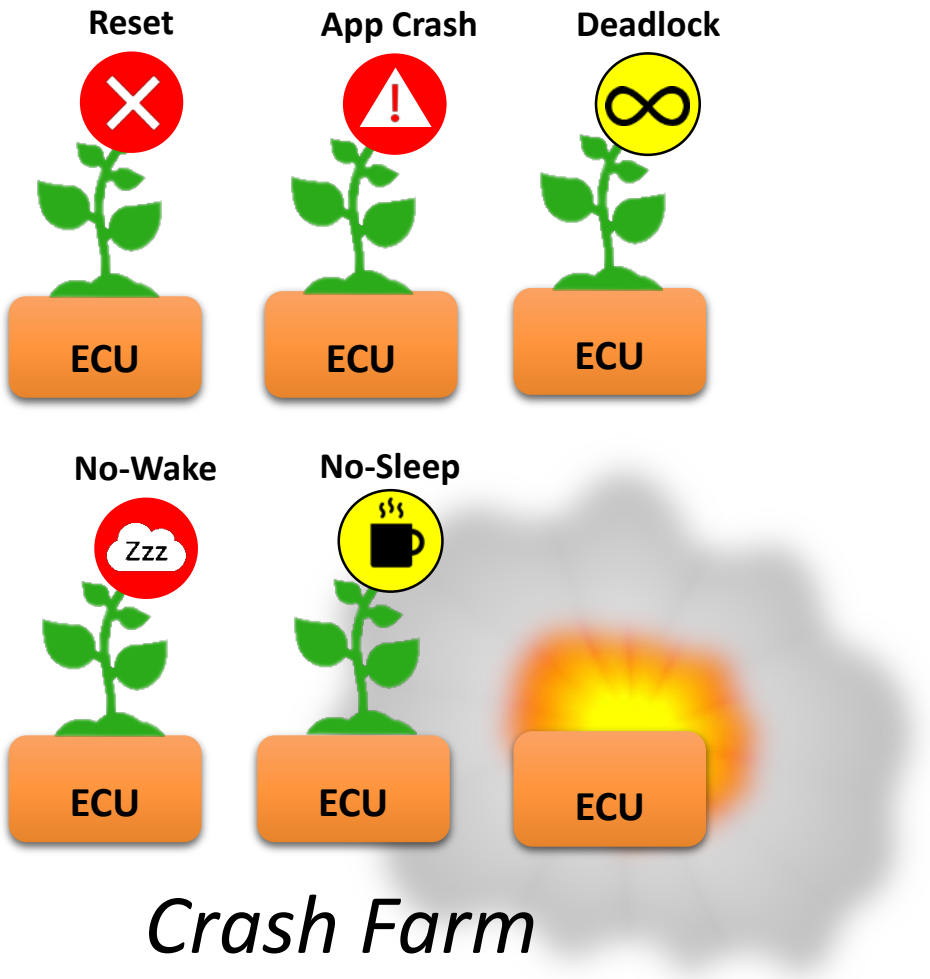
The most elusive defects will likely be found late in Product Development.

- All the easy bugs have been fixed!
- Growing Sample Size
- System Integration
- Feature Creep
- Unintended Use Cases
- Inherent Compulsion to Push Boundaries

- Reproducing the fault can be incredibly time consuming
  - Resets/Reboots
  - Application Crashes
  - Memory Leaks/Corruption
  - Deadlocks
  - Power Moding Issues
- Limited quantity and access to complete vehicle systems

“Crash Farm”?

Server Farm



Crash Farm

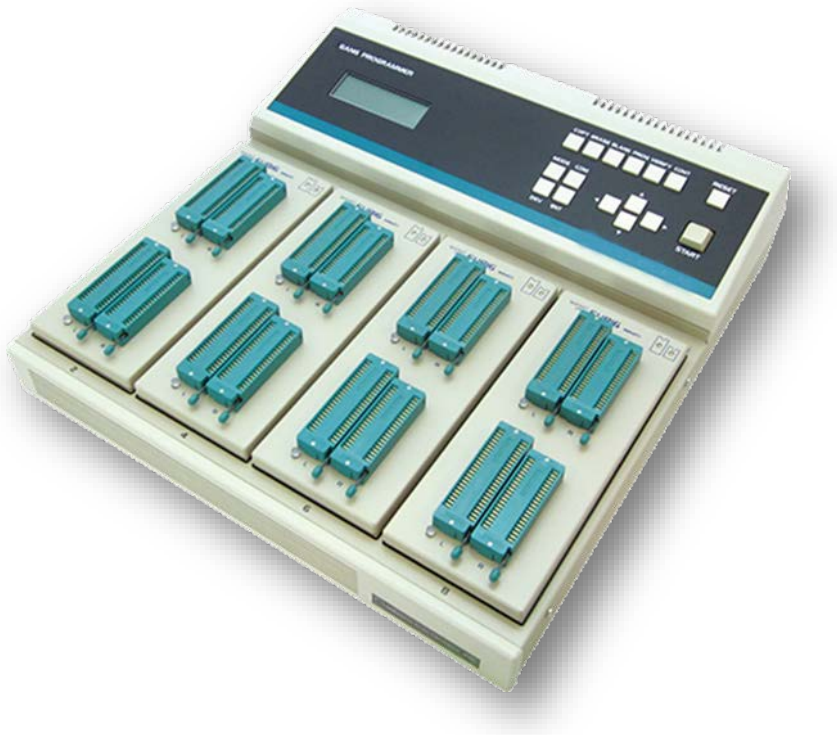
What is a Crash Farm?

A Crash Farm does not necessarily emulate an entire system

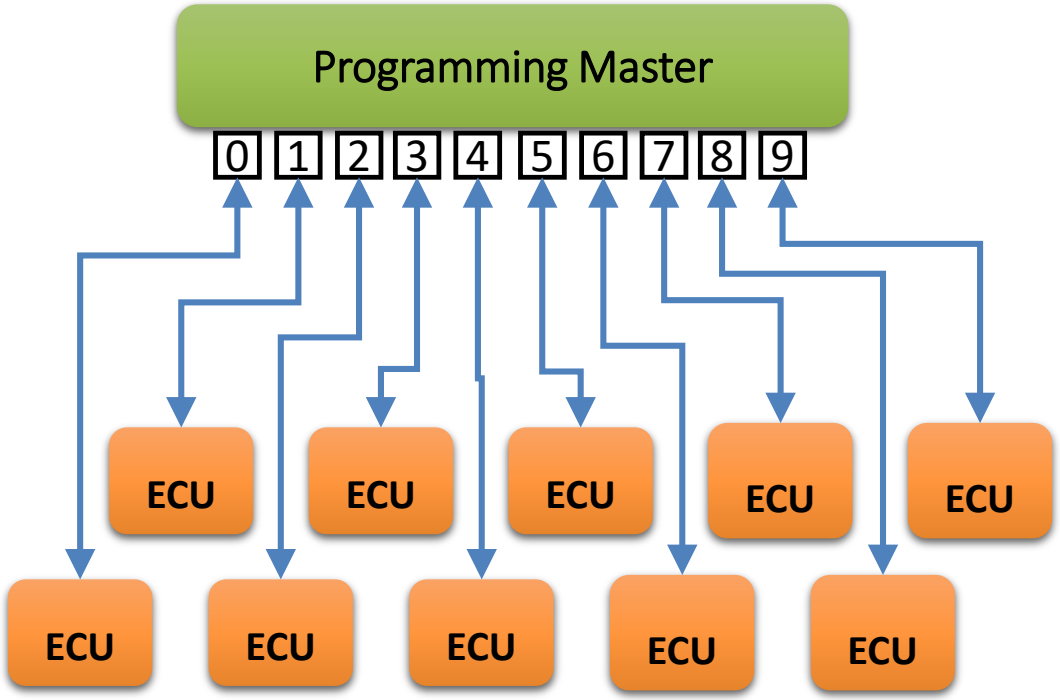
Target specific sequences of events or use cases

- Power Mode Cycles (Full/Interrupted)
  - No Boot
  - No Sleep
  - Loss of Function
  - Time Variance
- Robustness & Stability Tests
  - High CPU Loading
  - High Memory Usage
  - Any Abusive Test Imaginable
- Use Case Specific
  - Memory Leaks
  - Rogue Driver/DMA
  - Application Crash
  - System Reset
- Proactive Use
  - Bootloader/Kernel Changes
  - Part Substitutions
  - Regression Testing

Gang Programming



Gang Programmer

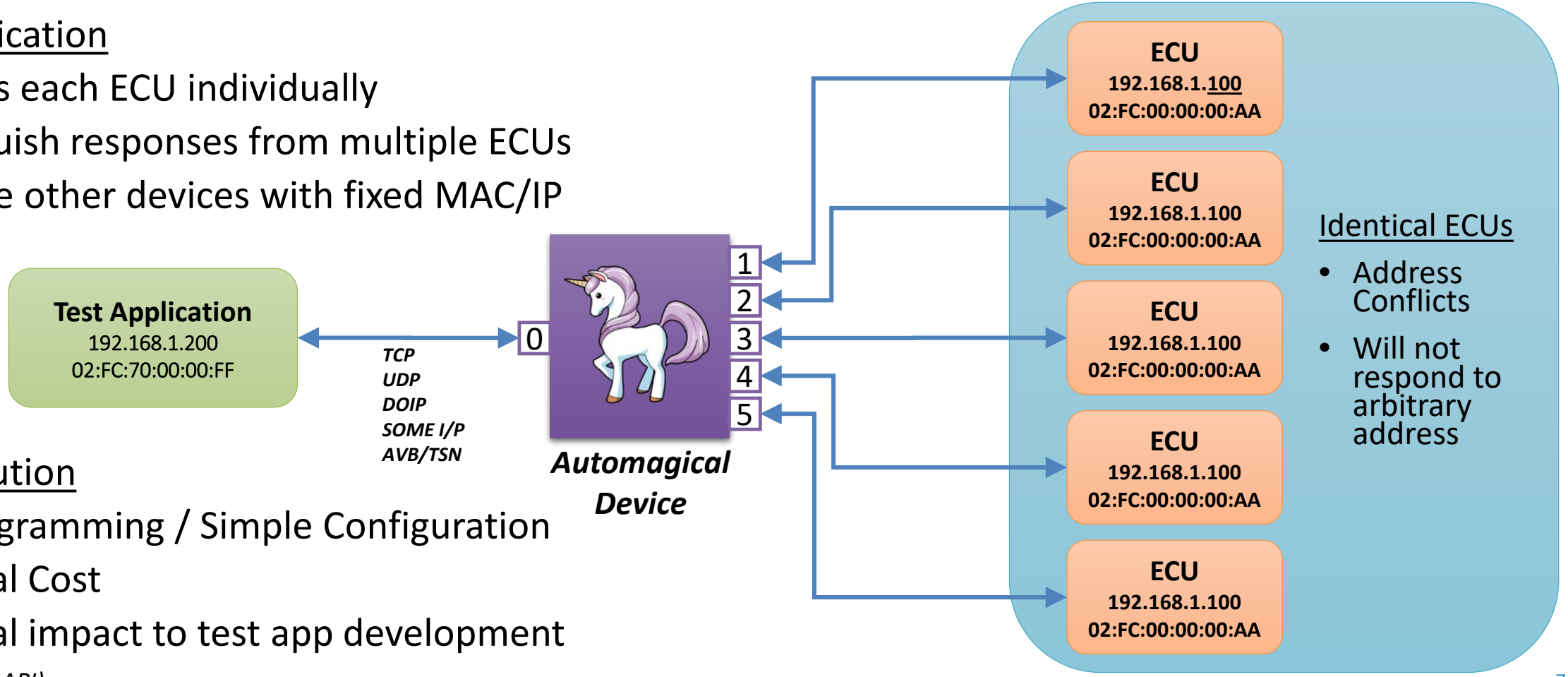


Gang Programming

Crash Farm / Gang Programming Topology

Test Application

- Address each ECU individually
- Distinguish responses from multiple ECUs
- Emulate other devices with fixed MAC/IP



Identical ECUs

- Address Conflicts
- Will not respond to arbitrary address

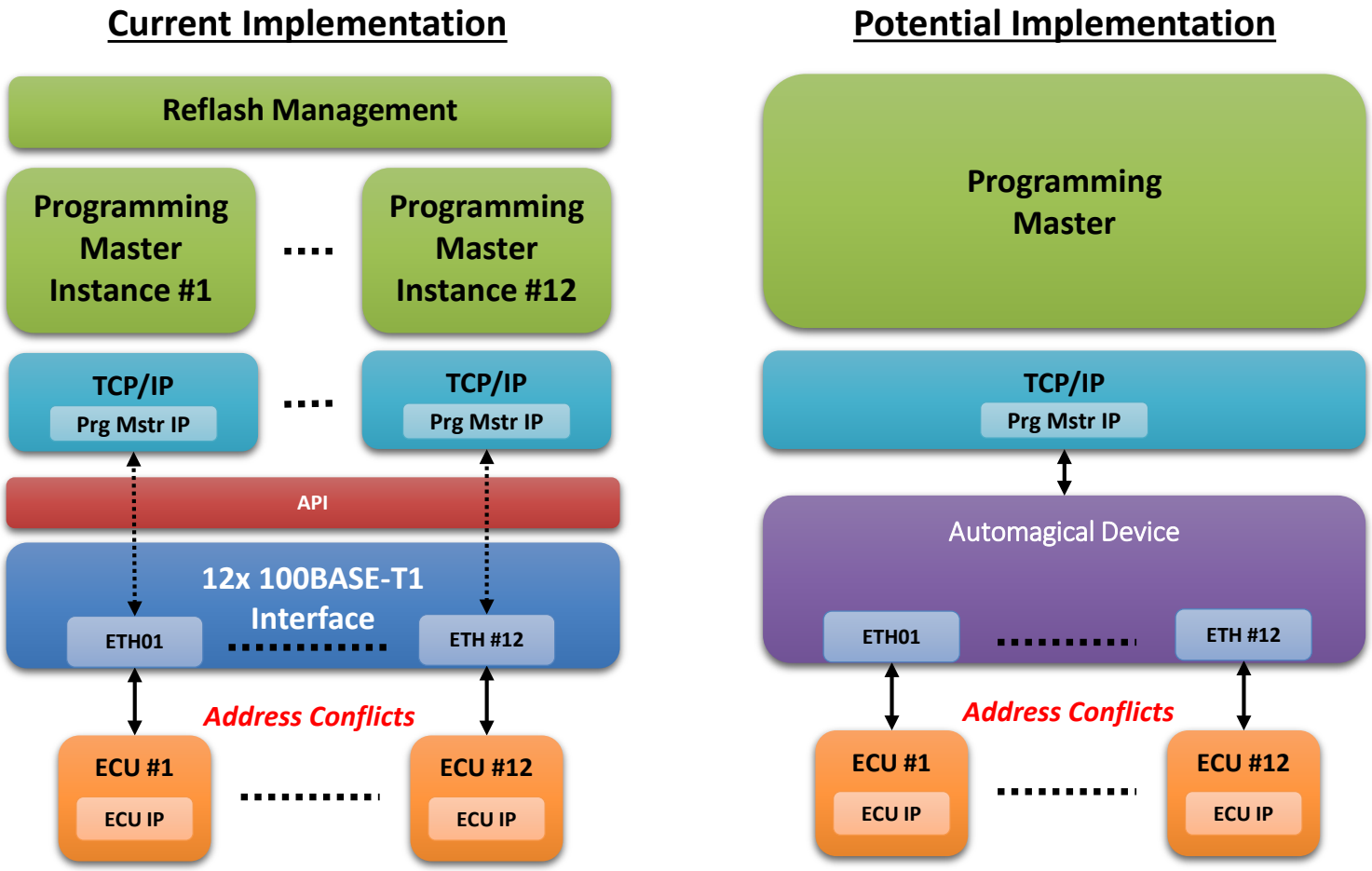
Ideal Solution

- No Programming / Simple Configuration
- Minimal Cost
- Minimal impact to test app development

(no special API)

Gang Programming
Current Implementation vs. Potential

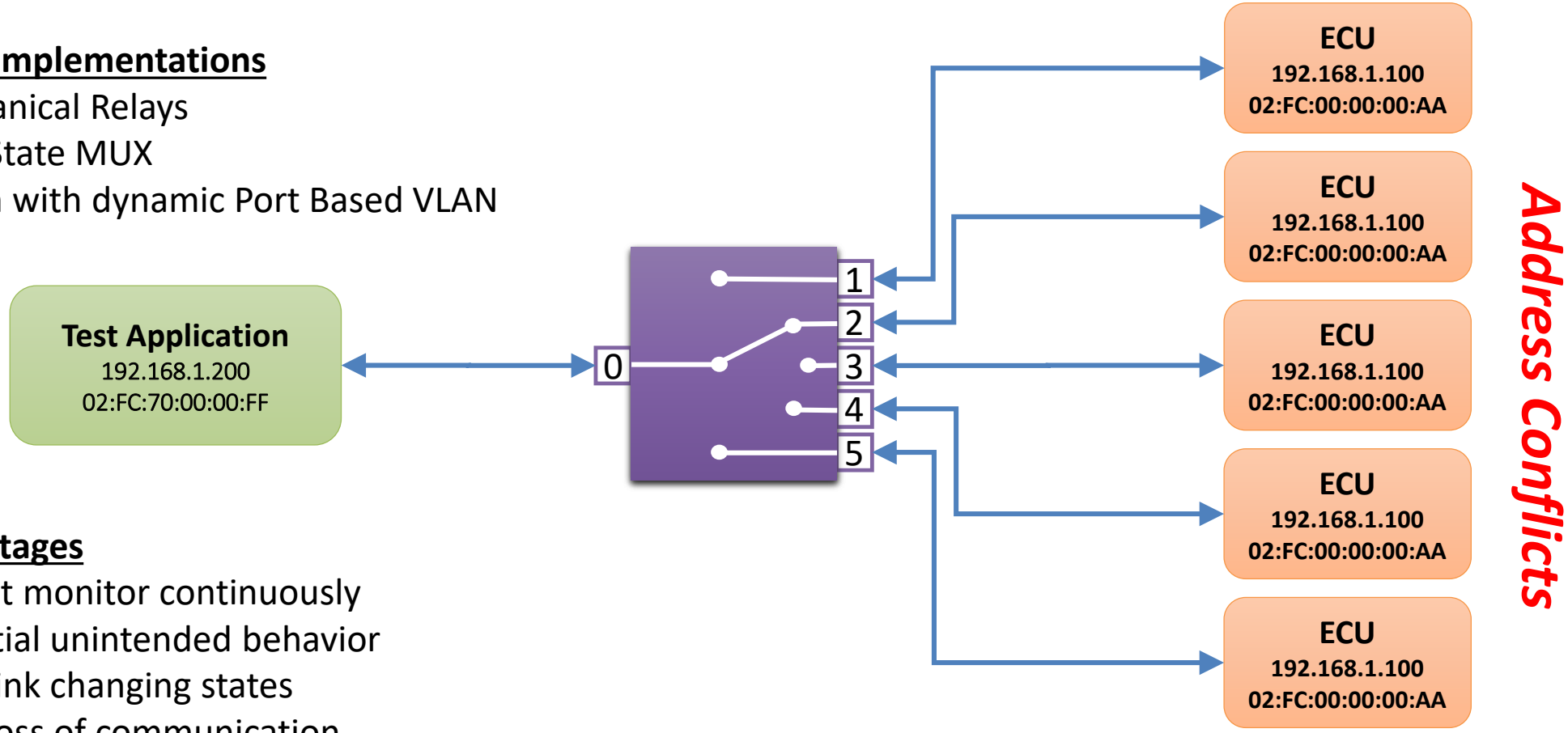
- Fixed IP address for Programming Master
- Multiple NICs / Single IP Address
- Drives independent application and stack for each NIC



Low Tech: MUX

Possible Implementations

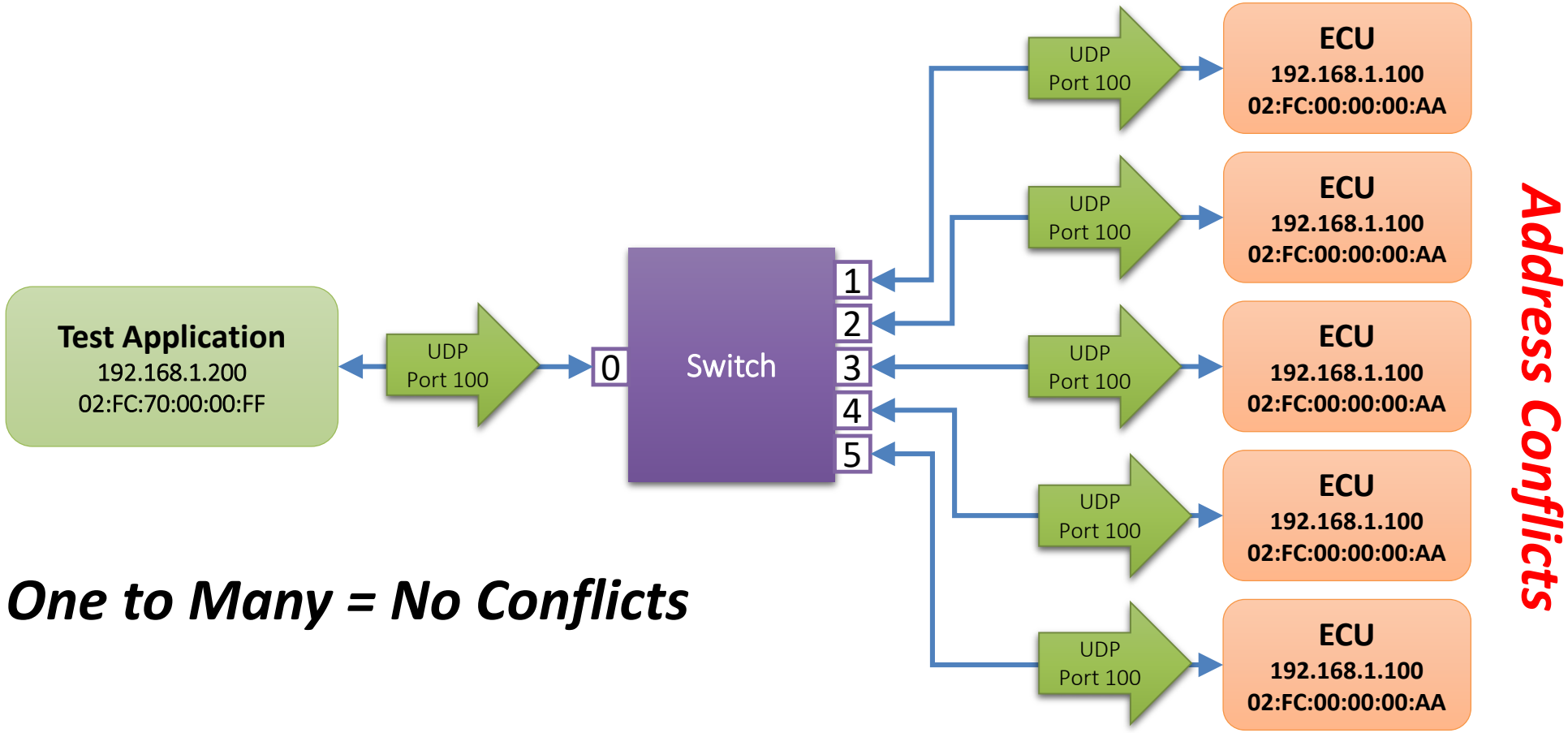
- Mechanical Relays
- Solid State MUX
- Switch with dynamic Port Based VLAN



Disadvantages

- Cannot monitor continuously
- Potential unintended behavior
  - Link changing states
  - Loss of communication

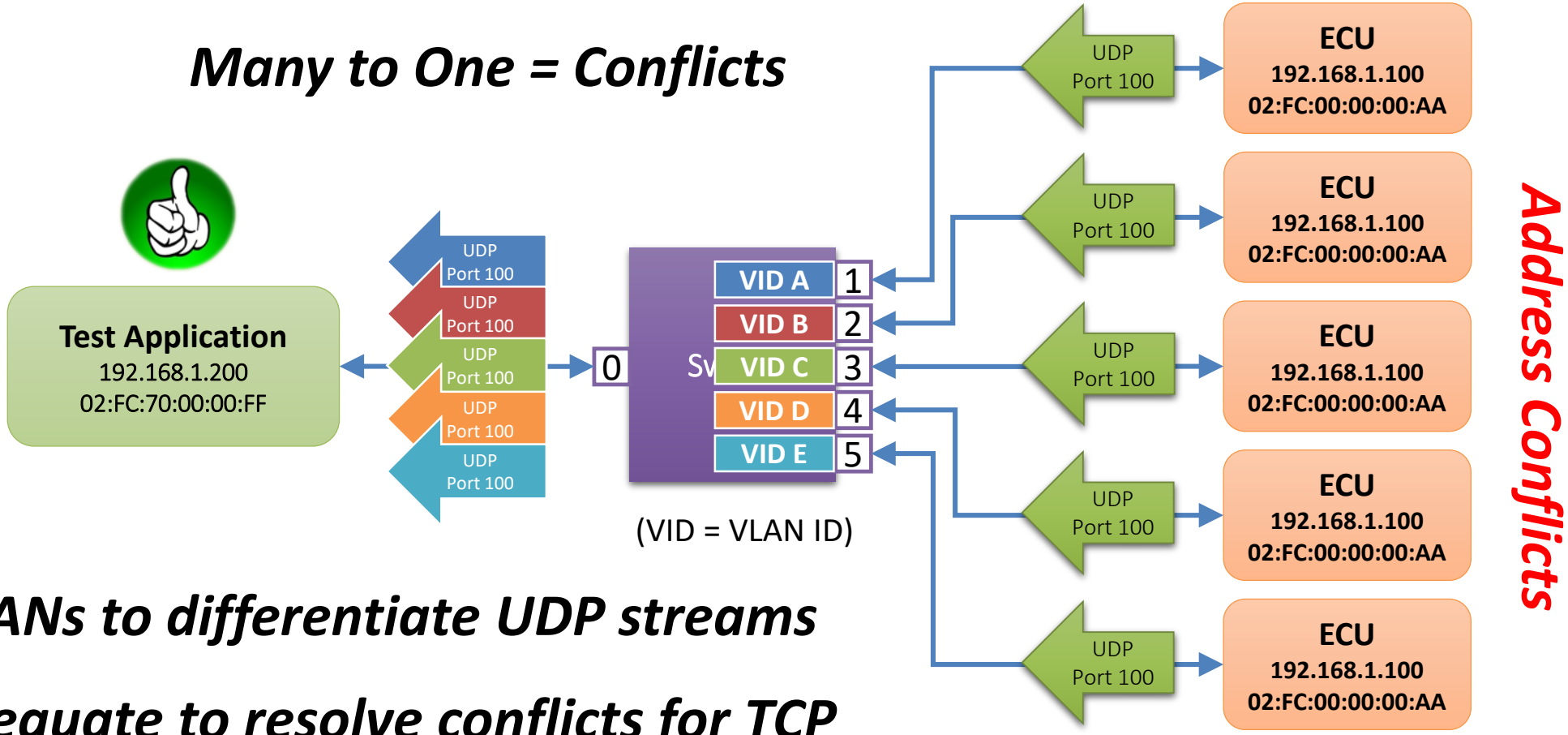
Switch: UDP Inbound to ECU



One to Many = No Conflicts

Switch: UDP Inbound to Test Application

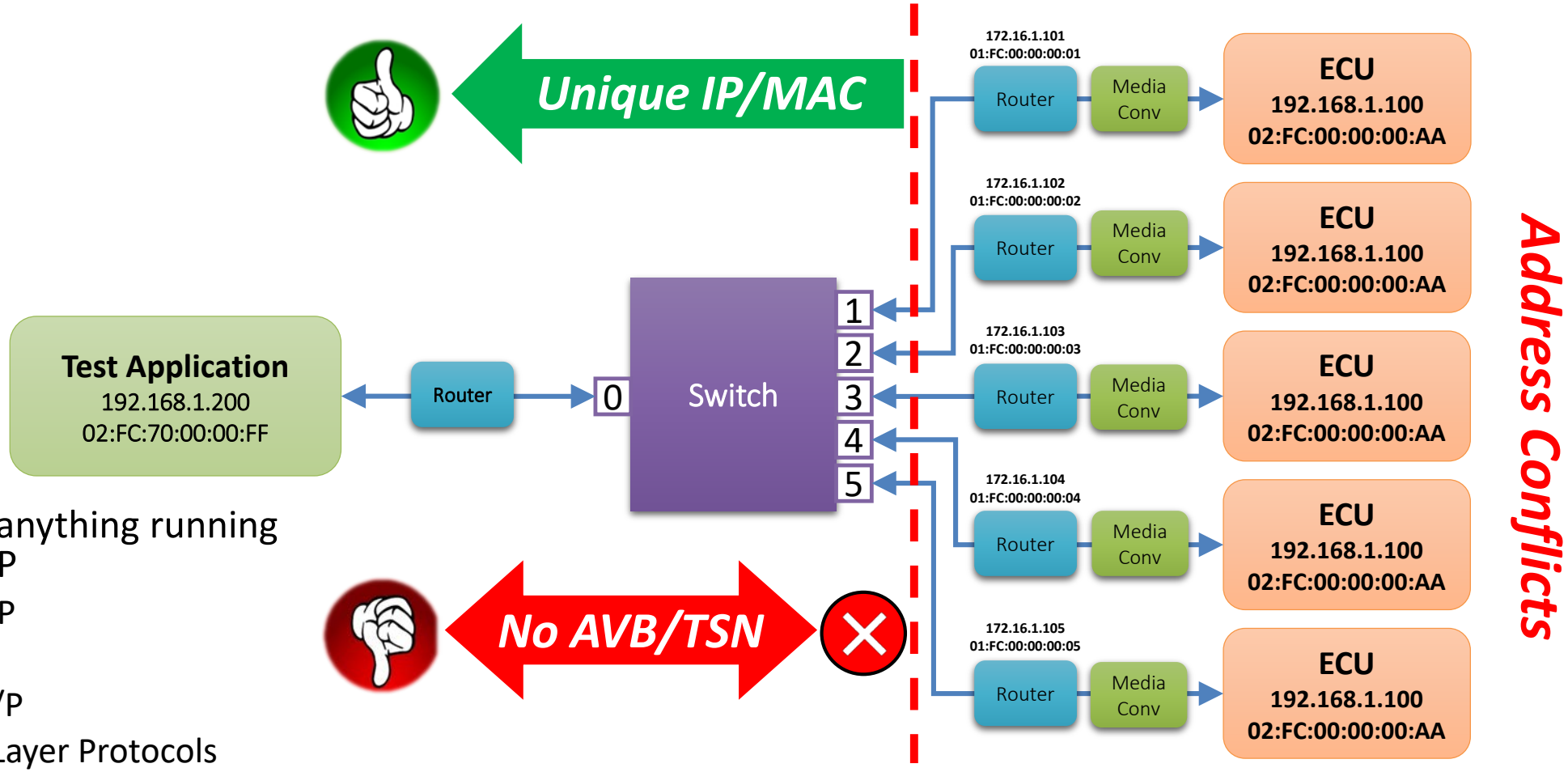
Many to One = Conflicts



Use VLANs to differentiate UDP streams

Not adequate to resolve conflicts for TCP

Network Address Translation



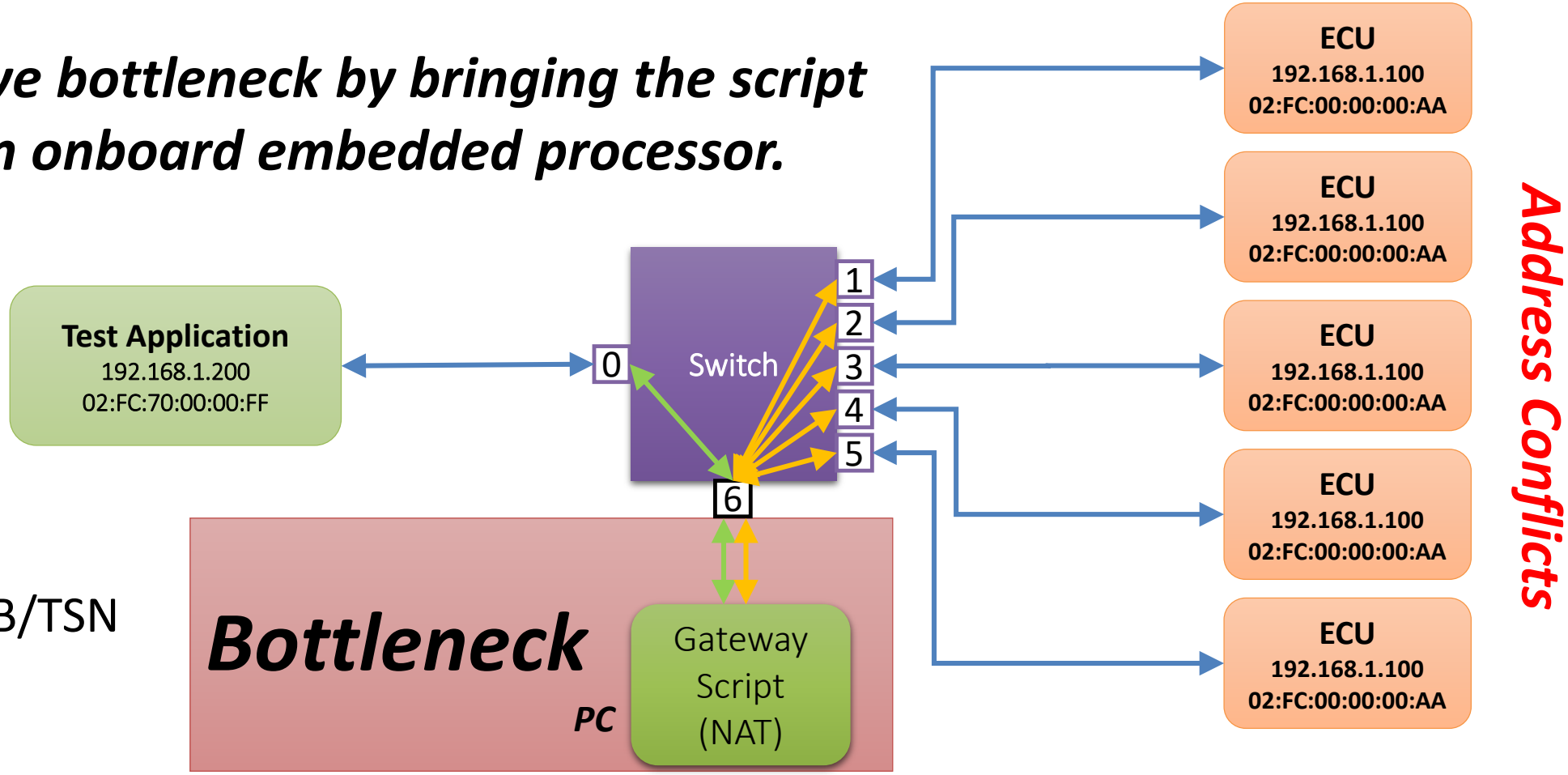
Works for anything running on top of IP

- TCP/UDP
- DOIP
- SOME I/P
- Higher Layer Protocols

(Router + Media Conv) x # ECUs = $$$

Switch with Gateway

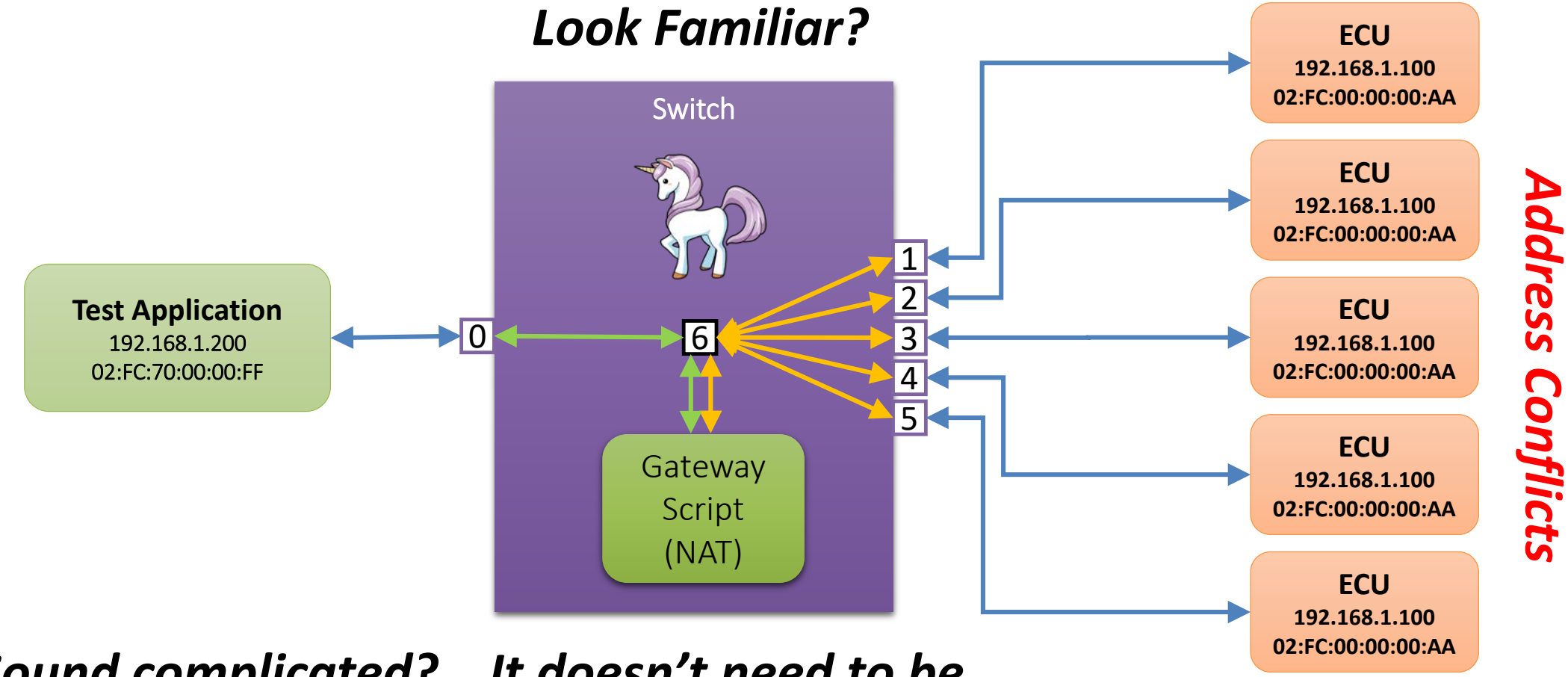
Remove bottleneck by bringing the script into an onboard embedded processor.



AVB/TSN

Switch with Embedded Gateway

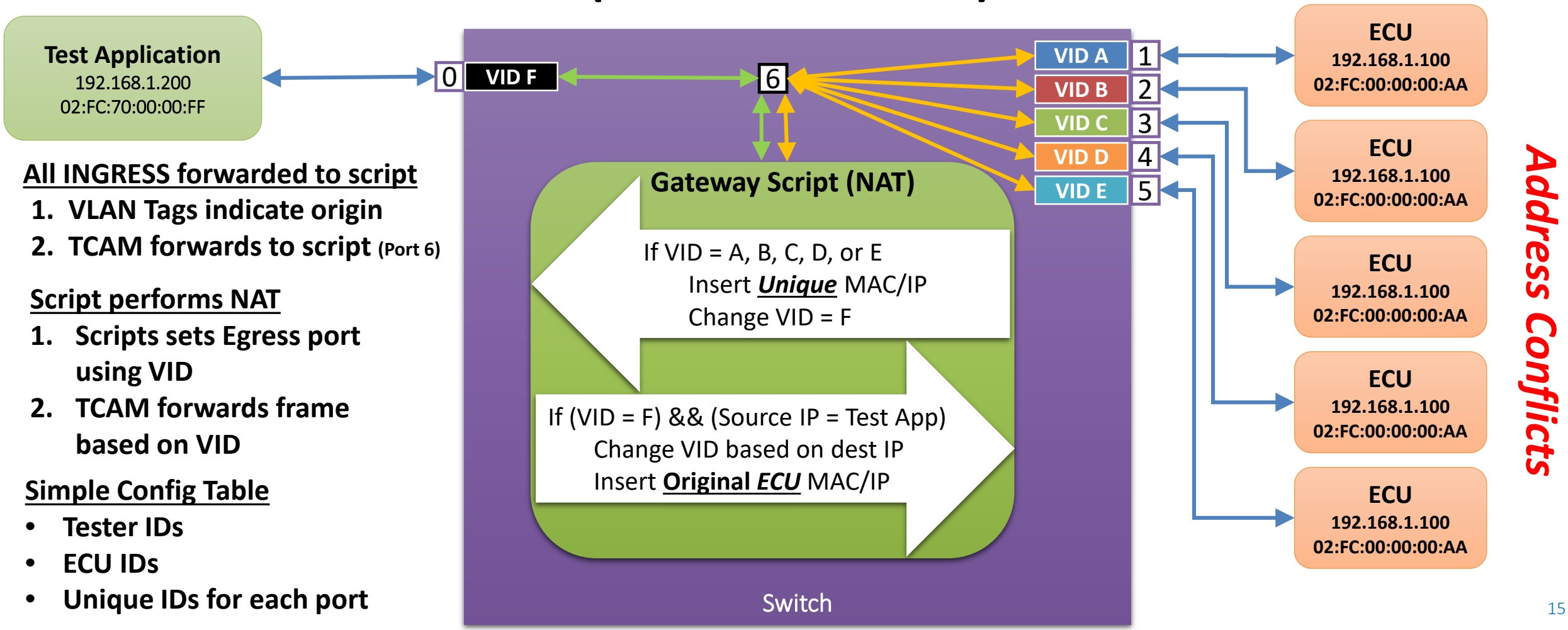
Look Familiar?



Address Conflicts

Sound complicated? It doesn't need to be….

Gateway Script for the Switch
(NAT with a Twist)



All INGRESS forwarded to script

1. VLAN Tags indicate origin
2. TCAM forwards to script (Port 6)

Script performs NAT

1. Scripts sets Egress port using VID
2. TCAM forwards frame based on VID

Simple Config Table

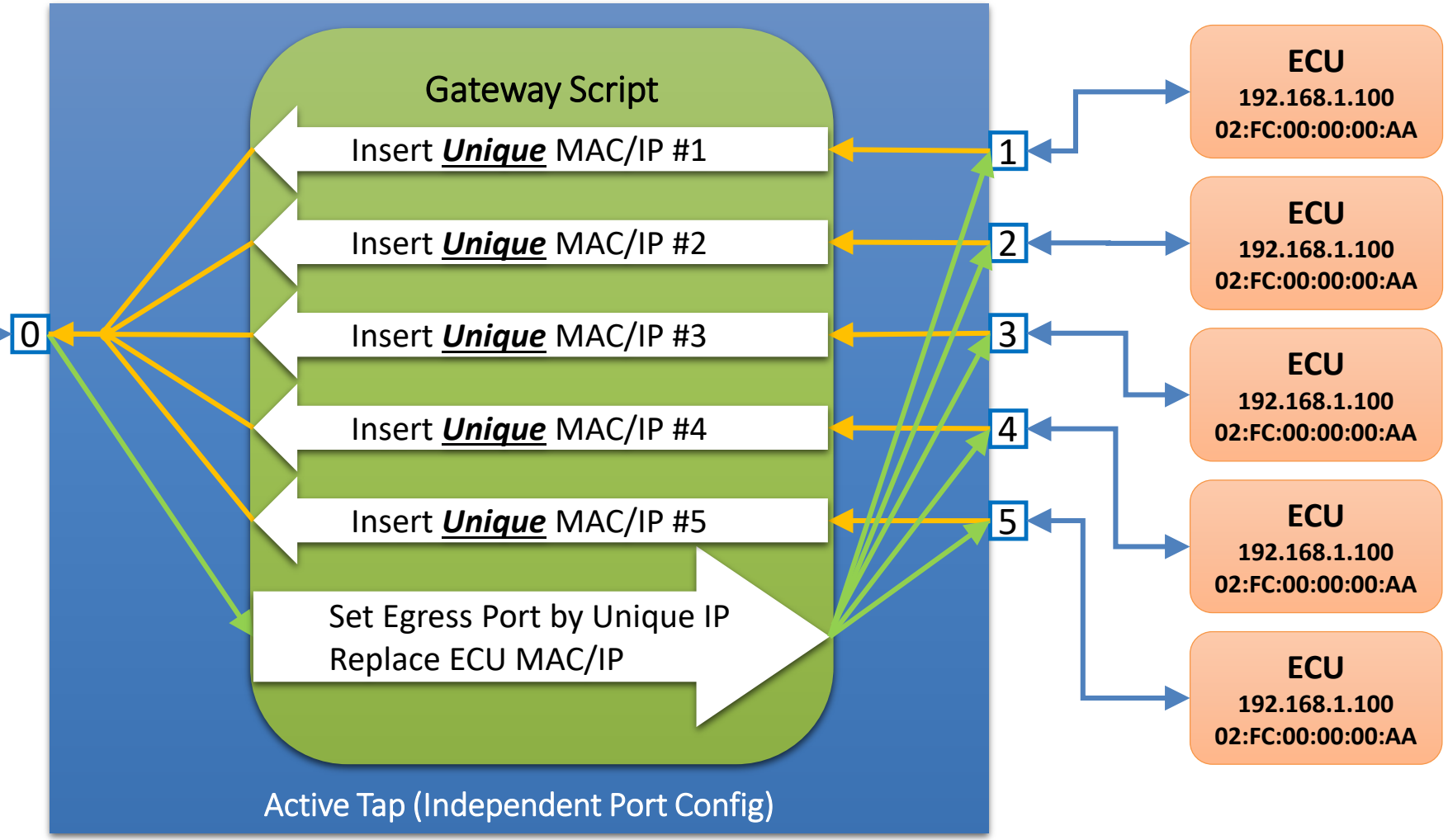
- Tester IDs
- ECU IDs
- Unique IDs for each port

Gateway with Active Tap

Logging precision with timestamps

AVB/TSN

Test Application
192.168.1.200
02:FC:70:00:00:FF



Address Conflicts

Simple Config Table

- **Tester IDs**
- **ECU IDs**
- **Unique IDs for each port**

Other Scripting Details

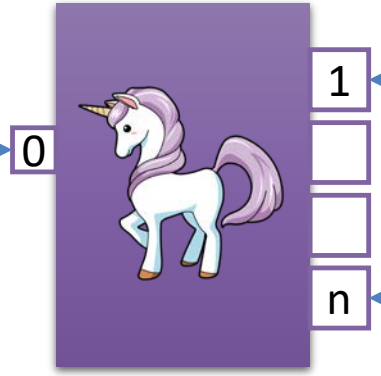
- Checksums
  - Layer 2: Done in Hardware
  - Layer 3: IP Checksum
    - Only covers IP header
    - Surgical change like this is as simple as adding an offset
    - Could be done by gateway stack
  - Layer 4: TCP Checksum
    - Dependent on some content of the IP Header (Pseudo Header)
    - Changing IP addresses can be as simple as adding an offset
    - Obvious performance cost for having to recalculate the TCP checksum. (includes payload)
- ARP: Easy to script proper responses (if needed)
- Other Protocols
  - Not difficult in layers 3 and below (TCP checksum is for entire payload)
  - Layer 2 protocols may require manipulating multiple addresses and IDs in header.

Test Setup

Test/Programming Application

- neoECU AVB/TSN
  - i.MX6Q
  - Linux 4.1.15
  - 1000BASE-T
- iPerf 3 Clients
  - 1x TCP Socket/ECU
  - IP: 192.168.1.200
  - MAC: 02:FC:00:00:00:FF

1000BASE-T



NAT/Gateway

100BASE-T1

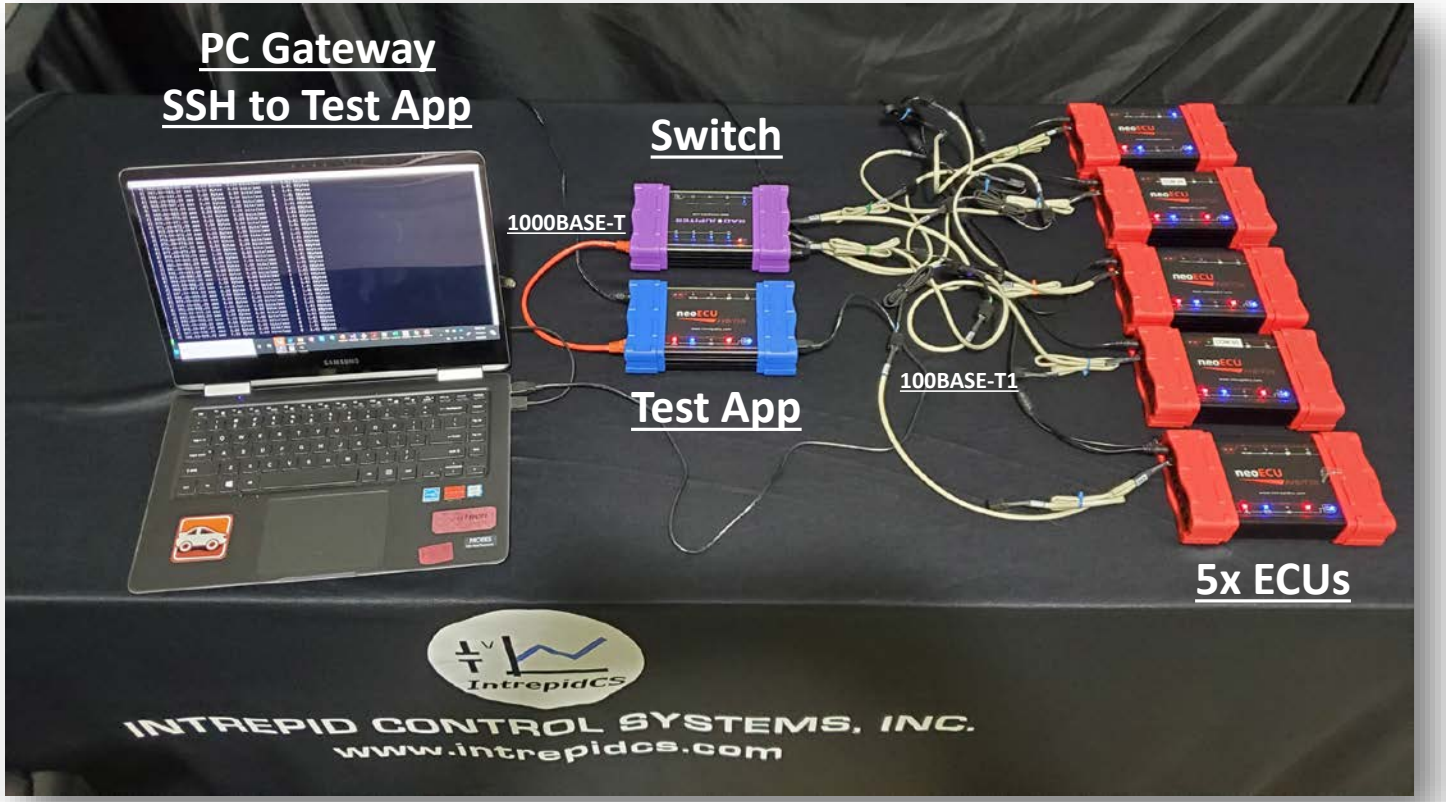
ECUs

ECU #1

ECU #n

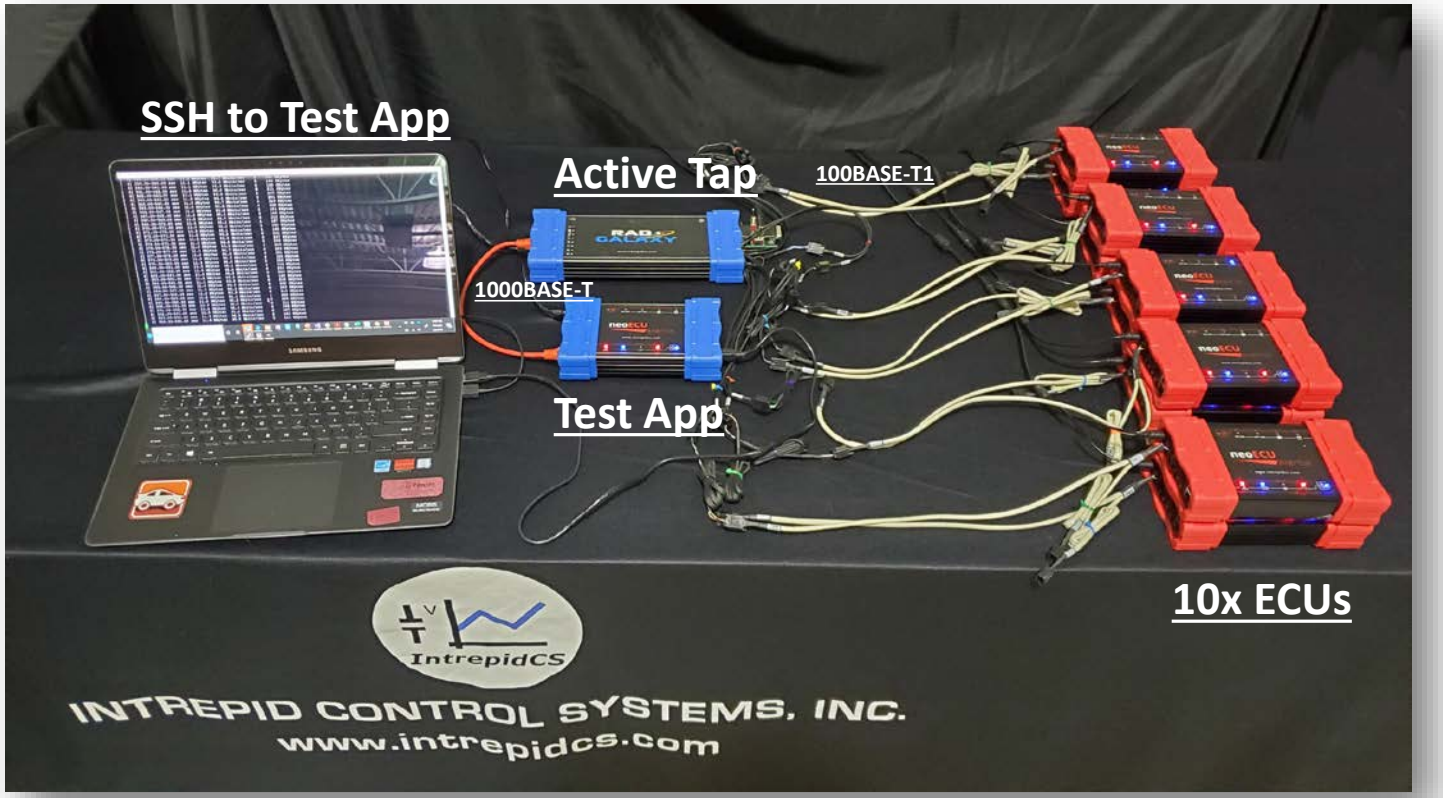
- neoECU AVB/TSN
  - i.MX6Q
  - Linux 4.1.15
  - 100BASE-T1
- iPerf 3 Server
  - **IP: 192.168.1.100**
  - **MAC: 02:FC:00:00:00:AA**

Test Configurations 1 & 2



1. **Switch w/ PC Gateway**
   - Samsung Notebook 9
     - Intel i7-7500U
     - Windows 10
   - 5-Port 100BASE-T1 Switch (RAD-Jupiter)
   - Vehicle Spy 3.9.1 / Gateway Builder (Beta)
   - 5x ECUs
2. **Switch w/ Embedded Gateway**
   - 5-Port 100BASE-T1 Switch (RAD-Pluto)
   - Vehicle Spy 3.9.1 / Gateway Builder (Beta)
   - 5x ECUs

Test Configuration 3



3. **Active Tap w/ Embedded Gateway**
   - Vehicle Spy 3.9.1 / Gateway Builder (Beta)
   - RAD-Galaxy (12 Independent Ports)
   - 10x ECUs

Performance Summary

	Gateway Latency	TCP Max Bandwidth (Mbps)									
		1x ECU	2x ECU	3x ECU	4x ECU	5x ECU	6x ECU	7x ECU	8x ECU	9x ECU	10x ECU
Switch + PC Gateway (RAD-Jupiter)	1.5-2 mS	3.2	1.3	Unstable	--	--	--	--	--	--	--
Switch + Embedded Gateway (RAD-Jupiter)	~350 µS	8	3.8	2.3	1.8	1.4	--	--	--	--	--
Active Tap + Embedded Gateway (RAD-Galaxy)	~60 µS	94 (Note 1)	67 (Note 2)	44	33	26	22	18	16	14	13

Note 1 : Point to point testing (no gateway) demonstrated ~94 Mbps

Note 2: A memory bottleneck was identified in the FPGA the Active Tap of ~130 Mbps aggregate bandwidth across all channels. This bottleneck was resolved, but not merged with beta release in time for revised testing for this presentation.

Topology Summary

	UDP	TCP	AVB /TSN	gPTP	Prec. Tstamp	Latency	Agg BW (Mbps)	Cost	Complexity	Notes
MUX	Y	Y	N	N	N	Low		$	Low	Not continuous Drops link and/or com
Switch	Y	N	N	Y (note 1)	N	Low	Switch Dependent	$$	Low	Limited to UDP
Router/NAT	Y	Y	N	N	N	Low	Switch Dependent	$$	Med	No AVB/TSN Media Converters Add Up $$
Switch + PC Gateway (RAD-Jupiter)	Y	Y	Y	Y (note 1)	N	Too High	~3	$$	Low-High (note 2)	Script Bottleneck PC Performance Dependent
Switch + Embedded Gateway (RAD-Jupiter)	Y	Y	Y	Y (note 1)	N	High	~8	$$	Low-High (note 2)	Less compromises
Active Tap + Embedded Gateway (RAD-Galaxy)	Y	Y	Y	Y	Y	Med	Today ~130 Q4-19 >600	$$$$	Low-High (note 2)	Least compromises (at a cost)

Note 1: A switch supporting gPTP can only be connected to a single Grand Master

Note 2: Depends on scripting environment (Configuration Table vs. Coding)

Conclusions and Next Steps

- There are a variety of methods to manage conflicting ECU addresses
  - In Crash Farm testing, define the scope of testing and evaluate tradeoffs to choose the best topology.
  - Gang Programming,
    - Using a Router/Media Converter on each ECU is not the most elegant solution but has best performance.
    - Switch or Active Tap with Gateway may have sufficient performance, but additional testing would be required.
- The initial plan for a SOME/IP Crash Farm as a proof of concept was too ambitious.
- iPerf helped characterize performance to reach the following conclusions.
  - A non-embedded gateway adds too much latency to be considered.
  - An embedded gateway with a switch may be suitable for some applications of limited bandwidth.
- Next Steps
  - Retest Active Tap with memory bottleneck removed
  - Interested in finding more realistic use cases

Thank You for Your Interest and Attention

Discover more at www.intrepidcs.com

Or contact us at moreinfo@intrepidcs.com

John Simon,
Product & Applications Manager
jsimon@intrepidcs.com