Towards a Secure and Resilient Industrial Control System with Software-Defined Networking

Dong (Kevin) Jin
Department of Computer Science
Illinois Institute of Technology
TSS/SoS Seminar, March 15, 2016
Part of the SoS Lablet with

• David Nicol
• Bill Sanders
• Matthew Caesar
• Brighten Godfrey
Work with ...

Wenxuan Zhou     Jason Croft
Matthew Caesar    Brighten Godfrey

Christopher Hannon    Jiaqi Yan
Hui Lin     Chen Chen

Jianhui Wang     Junjian Qi
Zhiyi Li     Mohammad Shahidehpour
References to papers in this talk


• Dong Jin, Zhiyi Li, Christopher Hannon, Chen Chen, Jianhui Wang, Mohammad Shahidehpour. "Towards A Resilient and Secure Microgrid Using Software-Defined Networking." IEEE Transactions on Smart Grid, Special Issue on Smart Grid Cyber-Physical Security (submitted)
Industrial Control Systems (ICS)

• Control many critical infrastructures
  – e.g., power grids, gas and oil distribution networks, wastewater treatment, transportation systems ...

• Modern ICSes increasingly adopt Internet technology to boost control efficiency, e.g., smart grid
Cyber Threats in Power Grids

- 245 incidents, reported by ICS-CERT
- 32% in energy sector
- 80,000 residents in western Ukraine
- 6 hours, lost power on Dec 23, 2015

Protection of Industrial Control Systems

• Commercial of-the-shelf products
  – e.g., firewalls, antivirus software
  – fine-grained protection at single device only

• How to check system-wide requirements
  – Security policy (e.g., access control)
  – Performance requirement (e.g., end-to-end delay)

• How to safely incorporate existing networking technologies in control system infrastructures?
A Representative Smart Grid Control Network

Control Center
- Energy Management System
- Data Historian

Backbone
- Substation
- Substation

Modern Substation Network
- modem
- Remote Terminal Unit
- Remote Terminal Unit

Switched Network
- SCADA Data Aggregator

Real-Time Control/Monitoring
- Workstation Monitor

Challenges and Opportunities

Time-critical
- Control Updates
- Network Updates
Similarities
• black hole avoidance
• loop mitigation
• fast convergence speeds
• priority control
• multiple services on a single physical channel
• ...

Differences
• strictly defined forwarding paths
• end-to-end performance guarantee
• system-wide visualization
• real-time monitoring
• a deny-by-default security model
• ...

A Utility Control Network

An Enterprise Network
Problem Statement

• Minimize the gaps with an SDN-enabled communication architecture for ICS
• Create innovative applications for ICS security and resiliency
  – Real-time network verification
  – Self-healing network management
  – Context-aware intrusion detection
  – Many more ...

ICS – industrial control system
SDN – software-defined networking
SDN Architecture

Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation

Specialized Features
Specialized Control Plane
Specialized Hardware

Control Plane
Control Plane
Control Plane

Open Interface
Open Interface

Merchant Switching Chips

Source: Nick McKeown, Open Networking Summit 2012
SDN Architecture - Continue

Applications
- QoS
- Access Control
- VPN

Control Plane
- OpenFlow Controller

Data Plane
- OpenFlow Protocol
- OpenFlow Switches
- Net 1
- Net 2
- Net 3
- Net 4
- Net 5
- Net 6

Applications
- OpenFlow
- QoS
- Access Control
- VPN
An SDN-Enabled Power Grid

Impact
- Instability
- Loss of Load
- Synchronization Failure
- Contingency
- Loss of Economics

Power Control Applications
- Demand Response
- Frequency Control
- State Estimation
- Topology Control

Cyber Resources
- SCADA Servers
- Field Devices
- Communication Networks
- Routing

Cyber Attacks
- Denial of Service
- False Data Injection
- Malware
- Insider Attack

Current Power Grid: Potential Cyber Attacks and Their Implications

Future SDN-enabled Power Grid: A Cyber-Attack-Resilient Platform
Transition to an SDN-Enabled IIT Microgrid

- Real-time reconfiguration of power distribution assets
- Real-time islanding of critical loads
- Real-time optimization of power supply resources
Transition to an SDN-Enabled Microgrid

• SDN-based Applications
  – Real-time Verification
  – Self-healing PMU

• Hybrid Testbed
  – SDN emulation + Power Distribution System Simulation
Application 1: Network Verification – Motivation

89% of operators never sure that config changes are bug-free\(^1\)
82% concerned that changes would cause problems with existing functionality\(^1\)

- Unauthorized access
- Unavailable critical services
- System performance drop
  - Instability
  - Loss of load
  - Synchronization Failure
- ...

1. Survey of network operators: [Kim, Reich, Gupta, Shahbaz, Feamster, Clark, USENIX NSDI 2015]
Verification System Design

Policy Engine

ICS Application Models

System Framework

Diagnosis
- Vulnerabilities
- Errors

Network Models
- topology
- network-layer states (e.g., forwarding tables)

Dynamic Model Update/Selection

Verification

Verified System Updates

Dynamic Network Data (topology, forwarding tables ... )
Dynamic Application Data (control updates ... )
User-specified Policy (security, performance ... )
Network-Layer Verification

VeriFlow

Generate equivalence classes → Generate forwarding graphs → Run queries

Network Controller

New rules

Good rules

Rules violating network invariant(s)

Diagnosis report
- Type of invariant violation
- Affected set of packets

Prior Work
- FlowChecker [Al-Shaer et al., SafeConfig2010]
- HeaderSpaceAnalysis [Kazemian et al., NSDI2012]
- Anteater [Mai et al., SIGCOMM2011]
- VeriFlow [Khurshid et al., NSDI2012]

Pictures borrowed from VeriFlow slides [Khurshid, Zou, Zhou, Caesar, Godfrey NSDI 2013]
Challenges — Timing Uncertainty

Network devices are asynchronous and distributed in nature

Diagram:
- Switch A
- Switch B
- Controller
- Remove rule 1
- Install rule 2
- Rule 1
- Rule 2
Challenges — Timing Uncertainty

Remove rule 1 (delayed)

Install rule 2

Possible network states:

Packet

Switch A

rule 1

Controller

Install rule 2

Switch B

rule 2

Loop-freedom Violation
Uncertainty-aware Modeling

- Naively, represent every possible network state $O(2^n)$
- Uncertain graph: represent all possible combinations
Update synthesis via verification

Verifier

Verification Engine

Network Model

Controller

Stream of Updates

Update queue

Safe?

Enforcing dynamic correctness with heuristically maximized parallelism

1. mod A->C to A->F
2. add F->G
3. add G->H
4. add H->B

A should reach B

Slide borrowed from Wenxuan Zhou, “CCG” NSDI 2015
OK, but...

Can the system “deadlock”?  

- Proved classes of networks that never deadlock  
- Experimentally rare in practice!  
- Last resort: heavyweight “fallback” like consistent updates  

[Reitblatt et al, SIGCOMM 2012]

Is it fast?

<table>
<thead>
<tr>
<th>Time</th>
<th>Number of Rules in the Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/22/2014</td>
<td>10000</td>
</tr>
<tr>
<td>7/23/2014</td>
<td>15000</td>
</tr>
<tr>
<td>7/23/2014</td>
<td>20000</td>
</tr>
<tr>
<td>7/23/2014</td>
<td>25000</td>
</tr>
</tbody>
</table>

Immediate Update  
Consistent Updates  
CCG

Completion Time

Slide borrowed from Brighten Godfrey, TSS Seminar, Sep 2015
Application 2: Self-Healing Phasor Measurement Unit (PMU) Networks

Integration of A Communication Network and A PMU Network
Self-Healing PMU Networks

• Isolate compromised devices

• “Self-heal” the network by quickly re-establishing routes
  – To restore power system observability
  – Using an integer linear program model

Video Demo
Figure 2: DSSnet system architecture diagram. Note that the power simulator runs on a Windows machine and the network emulator runs on a Linux machine.

3.1.5 Virtual Time System

Unlike simulation, the emulation clock elapses with the real wall clock. Therefore, pausing the emulation requires more than just stopping the execution of the emulated entities, but also pausing their clocks. Virtual time can be used to achieve this goal [9, 19]. We choose to extend the work of [9], in which Mininet is patched with virtual time support. However, their motivation is different from ours.

In general, virtual time has at least two categories of application. The first one is to slow down emulation so that it appears to emulated entities that they have sufficient virtual resources. Slowing down execution also alleviates the problems caused by resource multiplexing. Another usage of virtual time is for emulation-simulation synchronization. In DSSnet, we assign every container a private clock, instead of using the global time provided by the Linux OS. The containers now have the flexibility to slow down, speed up or stop its own clock when synchronizing with the simulator. However, the emulator needs to manage the consistency across all containers. This is achieved by a centralized time-keeper in [19], and by a two-layer consistency mechanism [9]. In practice, the emulator configuration guarantees that all containers are running with one shared virtual clock; Similarly, the container leverages the Linux process hierarchy to guarantee that all the applications inside the container are using the same virtual clock. The two-layer consistency approach is well-suited to this work for pausing and resuming because:

1. All hosts should be paused or resumed when we stop or restart the emulation.
2. All processes inside a container should be paused or resumed when we stop or restart the emulation.

The first task is done by the network coordinator. The second task is implemented based on the fact that processes inside a container belong to the same process group.

3.2 Synchronization

A key challenge in DSSnet is the synchronization between connecting the emulated communication network and the simulated power system. The root cause is that two different clock systems are used to advance experiments. Ordinary virtual-machine-based network emulators use the system clock, and a simulator often uses its own virtual clock. This difference would lead to causality errors as shown in the following example.

In Figure 3, there are three cross-system events ($E_i$), each with a response ($R_i$). $E_1$ occurs before $E_2$, however, $E_2$ may require information from $R_1$. Since the response occurs after the second event, the global causality is violated, and thus reduces experiment fidelity. An example of $E_1$ is a request.
A Hybrid Testing Platform

• Challenges
  – Temporal fidelity in network emulation
  – Synchronization between two sub-systems
    • Emulation – executing “native” software to produce behavior in wall-clock time
    • Simulation – executing model software to produce behavior in virtual time
Our approach: Virtual Time

• Key idea: trade execution time with fidelity
• Time dilation factor (TDF) [Gupta, 2011]

\[
\text{TDF} = \frac{\text{time passing rate in the physical world}}{\text{time passing rate in a VM's perception of time}}
\]

• TDF = 10
  – 10 seconds in real time <=> 1 second in a time-dilated emulated host
  – a 100 Mbps link is scaled to a 1 Gbps link

Virtual Time System Architecture for a Container-based Network Emulator

Source code: https://github.com/littlepretty/VirtualTimeForMininet
Virtual Time is Useful

1. Emulation Fidelity Enhancement

2. Simulation/Emulation Synchronization

![Diagram showing network topology and timing synchronization]
Future Work

• More applications
  – e.g., Specification-based Intrusion Detection
• Network layer → Application layer and Cross-layer verification
• In-house research idea → Real system deployment
  – IIT Microgrid
  – First Cluster of Microgrids in US (12MW IIT + 10MW Bronzeville)
Specification-based Intrusion Detection

- **Virtualized Utility Network 1**: Frequency Control
- **Virtualized Utility Network 2**: Demand Response
- **Virtualized Utility Network 3**: State Estimation
- **Virtualized Utility Network 4**: Topology Control

**Cross-Layer Verification**

**Intrusion Detection**

**Control Center**
Cross-layer Verification

Power Control Application layer

Communication Network layer

A network environment with desired properties (performance, security...)

Correct app behaviors
Thank you