Securing the Smart Grid with Software Defined Networking

Christopher Hannon

Monday 08-28-2017
Outline

Securing the Smart Grid with Software Defined Networking
  Introduction
  Background
  Combining Simulation and Emulation Systems for Smart Grid Planning and Evaluation
  Conclusion and Future Work
Christopher Hannon

- 3rd year Direct Ph.D Student
- BS CS from IIT in 2015
- Advisor: Kevin Jin
- Collaborators: Jiaqi Yan, Zhiyi Li, M. Shahidepour, J. Wang, C. Chen
Background - Smart Grid

- Utilization of communication network to improve power network
  - Reliability of power
  - Resiliency to failure and attack
  - Flexibility in managing the power grid
  - Efficiency of power usage
- Generation
- Transmission
- Distribution
Background - What Led to SDN

- Software Defined Networking (SDN)
  - What is Software Defined Networking?
  - Why did SDN emerge?
  - Why can SDN be useful for Smart Grid?
Background - First Networks

- Telephone networks
- Circuit switching
- OSI Layer 1
- Time-division multiplexing (virtual circuit switching)
Background - Networking

- Internet emerges
- Subnets
- Routing
- OSI Layer 3
Background - Growth of Networking

- Local Area Networks (LANs)
- Star shaped networks
  - Too complicated
  - Buses and broadcast domains
- Mac addresses (precursor)
  - 8 bit manually set
- OSI Layer 2
Background - Networking

- Large networks
  - Big companies and universities
  - LANs get bigger
- Multiple interconnected buses
  - Single bus is bottleneck
  - Switching and bridging is needed
Background - Why SDN?

- Switching gets really complicated!
  - Spanning tree protocol
  - Layer 2 can not scale to Layer 3
  - DHCP, ARP, layer 2 to layer 3 interfaces+
- Middleboxes
  - Intrusion detection
  - load balancing
  - etc.
- Data Centers!
Background - Software Defined Networking (SDN)

- Enables Easier Control of Communication Network
- Separates the:
  - Data plane - performs packet forwarding according to rules
  - Control plane - makes decisions on how to forward packets
- Provides
  - Centralized management
  - Global view of the network
  - Easier deployment
  - Greater flexibility in network
Background - Communication in Power Grids

Different applications have different requirements

- **Power Grid Protection**
  - Very low latency
  - High priority traffic
- **SCADA**
  - Millisecond latency
  - Medium priority
- **Metering Infrastructure**
  - Low priority
- **PLC**
- **Wireless (radio/ cellular)**
- **Ethernet**
Example
Motivation - SDN in Smart Grid

- SDN can be used to create a reliable and resilient power system
- Enable Smart Grid applications to interact with networking and power systems
Motivation

- Real system has high overhead cost and time
- Simulation based tool that combines power system with communication network
- DSSnet - Distribution System Solver network
- Features
  - Power flow analysis
  - Software Defined Network communication network modeling
  - Smart Grid control applications
  - Cyber security evaluation
Combining Simulation and Emulation Systems for Smart Grid Planning and Evaluation


Related Work

- Existing tools
  - FNCS - (Distribution + Transmission + Communication)
  - GECO - Global Event Driven
  - EPOCHS - Agent Based (commercial software)
  - PSLF + ns-2
  - OpenDSS + HiL Testbed
- DSSnet (Distribution System Electric Power Simulation + Communication Network Emulation)
Design - Architecture

- Power Simulation
  - Simulates power flow

- Communication Network Emulation
  - Emulates power devices
    - Communication elements
    - Processing elements

- Emulation Coordinator

- Power Coordinator

- Virtual Time Module
  - Supports synchronization
Background - Simulation & Emulation

- Emulation
  - Virtual machine/ container based emulation
  - Processes execute instructions to advance the clock
  - Inherently continuous
  - Mininet network emulator
    - Hosts & switches

- Simulation
  - Mathematical model of system
  - Solved at timesteps
  - Event queue
    - Clock advances to next event in queue
    - Runs as fast as possible
  - OpenDSS power simulator
Design - Architecture

DSSnet

Windows

Power Coordinator

IED Configuration

Named Pipe

Input or Import

Named Pipe

TCP Socket

Windows COM Port

Legend

Processes/Elements

Components

Linux

Network Coordinator

Synchronization Events

Network Coordinator

Virtual Time System

Mininet

ONOS 1

ONOS 2

ONOS 3

Open vSwitch

HOSTS

OpenDSS

Elements

Settings

Monitors

Controls
Challenges

- How can the power Simulator be combined with communication network emulator?
  - Emulation is continuous
  - Simulation is solved in discrete time steps (not real time)
- Synchronization is a challenge
Design - Synchronization

Simulation Time

Emulation Time

Wall Clock

$t_1$, $t_2$, $t_3$, $t_4$

$E_i$ -- event

$R_i$ -- response

Synchronization Event
Design - Synchronization

Simulation Time

Emulation Time

Wall Clock

$t_1$, $t_2$, $t_3$, $t_4$

$E_i$ -- event
$R_i$ -- response

Synchronization Event
Design - Synchronization

Simulation Time

Emulation Time

Wall Clock

$E_i$ -- event

$R_i$ -- response

Synchronization Event

$t_1$, $t_2$, $t_3$, $t_4$
Design - Synchronization

Simulation Time

Emulation Time

Wall Clock

$E_i$ -- event
$R_i$ -- response

Synchronization Event

$t_1$, $t_2$, $t_3$, $t_4$
Design - Synchronization

Simulation Time

Emulation Time

Wall Clock

E_i -- event
R_i -- response

Synchronization Event

E_1
R_1

t_1 t_2 t_3 t_4
Design - Synchronization

![Diagram showing the relationship between Simulation Time, Emulation Time, and Wall Clock, with events and responses marked by E_i and R_i, respectively. The diagram includes timestamps t_1, t_2, t_3, and t_4. The synchronization event is indicated by an upward arrow.]
Design - Synchronization

- Simulation Time
- Emulation Time
- Wall Clock

Events:
- $E_i$ -- event
- $R_i$ -- response

Synchronization Event
Design - Synchronization

![Diagram showing synchronization events and responses over time]

- $E_i$: event
- $R_i$: response

Simulation Time

Emulation Time

Wall Clock

$t_1$, $t_2$, $t_3$, $t_4$
Design - Synchronization

- $E_i$: event
- $R_i$: response
- Synchronization Event

Simulation Time

Emulation Time

Wall Clock

$t_1$, $t_2$, $t_3$, $t_4$
Virtual Time

- Each process in the emulation has its own clock
- Virtual time module has the following abilities
  - add processes to virtual time
  - pause processes
  - resume processes
Design - Synchronization

- Simulation Execution Time
- Emulation Execution Time
- DSSnet Perceived Time
- Wall Clock

\[ E_i \quad -- \quad \text{event} \]
\[ R_i \quad -- \quad \text{response} \]

Synchronization Event
Design - Synchronization

- \( E_i \) -- event
- \( R_i \) -- response

Simulation Execution Time
Emulation Execution Time
DSSnet Perceived Time
Wall Clock

Synchronization Event
Design - Synchronization

- $E_i$ -- event
- $R_i$ -- response

- Simulation Execution Time
- Emulation Execution Time
- DSSnet Perceived Time
- Wall Clock

Synchronization Event
Design - Synchronization

Simulation
Execution Time

Emulation
Execution Time

DSSnet Perceived
Time

Wall Clock

E_i -- event
R_i -- response
Synchronization Event
Design - Synchronization

- Simulation Execution Time
- Emulation Execution Time
- DSSnet Perceived Time
- Wall Clock

- Event $E_i$
- Response $R_i$

Synchronization Event
Design - Synchronization

- Simulation Execution Time
- Emulation Execution Time
- DSSnet Perceived Time
- Wall Clock
Design - Synchronization

Simulation Execution Time

Emulation Execution Time

DSSnet Perceived Time

Wall Clock

E_i -- event
R_i -- response

Synchronization Event
Design - Synchronization

- Simulation Execution Time
- Emulation Execution Time
- DSSnet Perceived Time
- Wall Clock

$E_i$ -- event
$R_i$ -- response

Synchronization Event
Design - Synchronization

- Simulation Execution Time
- Emulation Execution Time
- DSSnet Perceived Time
- Wall Clock

\[ E_i \rightarrow \text{event} \]
\[ R_i \rightarrow \text{response} \]

Synchronization Event
Design - Synchronization

- Simulation Execution Time
- Emulation Execution Time
- DSSnet Perceived Time
- Wall Clock

E$_i$ -- event
R$_i$ -- response

Synchronization Event
Design - Synchronization

\[ Time_{wall\_clock} = \sum t_{E\_i} + \sum t_{S\_i} \]

\[ Time_{DSSnet} = \sum t_{E\_i} + \sum t_{S'\_i} \]

\[ ret = \frac{t_{S'\_i}}{t_{S\_i}} \]

\[ (1, \infty) \]
\[ (0, 1] \]
\[ 0 \]
Design - Synchronization

Diagram showing synchronization event with timelines for:
- Simulation Execution Time
- Emulation Execution Time
- DSSnet Perceived Time
- Wall Clock
Evaluation

The graph illustrates the Emulation Overhead against the number of hosts. The y-axis represents the Average Overhead in milliseconds, while the x-axis shows the Number of Hosts. As the number of hosts increases, the average overhead also increases, indicating a linear relationship. The graph includes error bars to indicate the variability in overhead measurements.
Design - Parallel Execution

- Minimize pause and unpause
- Multiple event queue design
  - Blocking (forces pause)
  - Non-blocking (enables the systems to run in parallel)
Design - Parallel Execution

- **E_i**: event
- **R_i**: response
- **E_1**: non-blocking
- **E_2**: blocking

Simulation Time

Emulation Time

Wall Clock

t_1  t_2  t_3  t_4
Design - Parallel Execution

- $E_i$ -- event
- $R_i$ -- response
- $E_1$ -- non-blocking
- $E_2$ -- blocking

Simulation Time

Emulation Time

Wall Clock

$t_1$, $t_2$, $t_3$, $t_4$
Design - Parallel Execution

Synchronization Event

$E_i$ -- event
$R_i$ -- response

$E_1$ -- non-blocking
$E_2$ -- blocking

Simulation Time
Emulation Time
Wall Clock

$t_1$, $t_2$, $t_3$, $t_4$
Design - Parallel Execution

Simulation Time

Emulation Time

Wall Clock

$t_1$ $t_2$ $t_3$ $t_4$

$E_1$ -- event
$R_i$ -- response
Synchronization Event

$E_1$ -- non-blocking
$E_2$ -- blocking
Design - Parallel Execution

Simulation Time

Emulation Time

Wall Clock

$E_i$ -- event
$R_i$ -- response

Synchronization Event

$E_1$ -- non-blocking
$E_2$ -- blocking
Design - Parallel Execution

![Diagram showing parallel execution with timelines and events](image)

- $E_i$ -- event
- $R_i$ -- response
- $E_1$ -- non-blocking
- $E_2$ -- blocking

Simulation Time

Emulation Time

Wall Clock

$t_1$, $t_2$, $t_3$, $t_4$
Design - Parallel Execution

- \( E_i \) -- event
- \( R_i \) -- response

Synchronization Event

- \( E_1 \) -- non-blocking
- \( E_2 \) -- blocking

Simulation Time

Emulation Time

Wall Clock

\( t_1 \) \( t_2 \) \( t_3 \) \( t_4 \)
Design - Parallel Execution

Synchronization Event

$E_i$ -- event
$R_i$ -- response

$E_1$ -- non-blocking
$E_2$ -- blocking

Simulation Time

Emulation Time

Wall Clock

$t_1$  $t_2$  $t_3$  $t_4$
Design - Parallel Execution

- \( E_i \) -- event
- \( R_i \) -- response

Synchronization Event

- \( E_1 \) -- non-blocking
- \( E_2 \) -- blocking

Simulation Time

Emulation Time

Wall Clock
Evaluation

The graph shows the evaluation of the iperf throughput (Mbps) over time (Seconds). The x-axis represents time in seconds, ranging from 0 to 800, and the y-axis represents iperf throughput in Mbps, ranging from 0 to 800.

Two conditions are compared:
- **No Freeze**
- **Freeze Duration=1 s, Interval=1 s**

The graph indicates that the freeze condition slightly decreases the throughput compared to the no freeze condition.
Evaluation

Cumulative Distribution

PING RTT (Milliseconds)

No Freeze
Freeze, Duration=0.1 s, Interval=0.1 s
Use Case

- Demand Response Application
  - Dynamic Wind Generation
  - Energy Storage Device
- Sensor Monitors Generation and Sends to Control Center
- Control Center Sends Commands to the Energy Storage Device
Use Case
Use Case

- Using default SDN reactive forwarding application
- Link failure on primary path at \( t=3.1 \)
- Network self-heals automatically to take alternative path
Use Case

**Bus 652 Voltages**

**Bus 675 Voltages Phase 2**

**Bus 675 Voltages Phase 1**

**Bus 675 Voltage Phase 3**
Future Work

- Hardware-in-the-Loop testbed
  - using embedded Linux hardware
  - modify the scheduler
- Create a distributed emulator environment for scalability
- Develop SDN Smart Grid Applications
  - Context aware intrusion detection and mitigation
  - Application based self-healing
Conclusion

- DSSnet
  - Electric Power Simulation
  - Communication Network Emulation
- For Smart Grid Planning and Evaluation