Resilient Information Architecture Platform for the Smart Grid

Gabor Karsai, Vanderbilt University

Network Optimized Distributed Energy Systems (NODES) Annual Review Meeting

Mar 26-28, 2018
Project Summary

¬ **Goal**: To create an open source *software platform* to run Smart Grid applications and demonstrate it through *selected applications*. A software platform defines:
  ¬ Programming model (for distributed real-time software)
  ¬ Services (for application management, fault tolerance, security, time synchronization, coordination, etc.)
  ¬ Development toolkit (for building and deploying apps)

¬ **Uniqueness**:
  ¬ Focus on distributed *applications* - not only on networking
  ¬ Focus on *resilience* – services for fault recovery
  ¬ Focus on *security* – maintain confidentiality, integrity, availability
Project Summary

Communication Network

Example Power System:
IEEE 30 bus system

Control Room

RIAPS Node:

RIAPS Computing Platform
Sensors
Actuators

Network I/F
Project Summary

› Challenges:
  – How do we build *distributed fault tolerant smart grid applications* in a real-time context? – *It is more than a middleware or networking problem.*
  – How do we *manage* accidental complexities in the development process? – *Developers need tools to be productive.*

› Deliverables:
  – Software platform run-time: middleware and other libraries + services used by apps
  – Development toolkit for building, deploying, and managing apps
  – Example applications for the Smart Grid
  – Design documentation
Key outcomes:

- The *open source platform* will enable developers – sanctioned by utilities - to build reusable components and applications

- The platform *specification* and its *prototype implementation* is open source, but for-profit entities will provide software development services for it

- A new *open standard* that will change how software for the smart grid is developed
Team

- Organizations: Vanderbilt University, North Carolina State University, Washington State University
  - Vanderbilt University (Prof. Gabor Karsai & Abhishek Dubey) Institute for Software-Integrated Systems has decades of experience in researching and developing middleware, model-driven development tools, real-time fault diagnostics, software platforms.
  - North Carolina State University (Prof. Srdjan Lukic, David Lubkeman) is home to the NSF Future Renewable Electric Energy Delivery and Management (FREEDM) ERC, have expertise in power grid based on power electronics, high bandwidth digital communication, and distributed control, including testing of experimental and commercial microgrid controllers.
  - Washington State University (Prof. Anurag Srivastava, Chen-Ching Liu, Dave Bakken) has expertise in power system operation and control, hosts the Smart Grid Testbed (SGDRIL), does research on power systems operation in extreme scenarios, Smart City Testbed, and on fault tolerant computing and middleware for power systems.
Team

- While all team members have electrical engineering background they specialize in complementary fields:
  - Distributed Real Time Embedded systems (Karsai and Dubey)
  - Fault Tolerant Computing (Bakken and Karsai)
  - Electrical Power Engineering (Srivastava, Liu and Lukic)
  - Cyber-Physical Testbeds (Srivastava, Liu and Lukic)
  - Control Engineering (Srivastava, and Liu)

- The team members have solid industry connections that will enable technology transition:
  - Help from industry advisory board to target the technology for the market
  - Ability to do hardware in the loop testing allows having product ready for field installation
## Project Progress

<table>
<thead>
<tr>
<th>PE</th>
<th>Year1</th>
<th>Year2</th>
<th>Year3</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="Green" alt="Green" /></td>
<td><img src="Green" alt="Green" /></td>
<td><img src="Green" alt="Green" /></td>
<td>On track</td>
</tr>
<tr>
<td>2</td>
<td><img src="Green" alt="Green" /></td>
<td><img src="Green" alt="Green" /></td>
<td><img src="Green" alt="Green" /></td>
<td>On track</td>
</tr>
<tr>
<td>3</td>
<td><img src="Green" alt="Green" /></td>
<td><img src="Green" alt="Green" /></td>
<td><img src="Green" alt="Green" /></td>
<td>On track</td>
</tr>
<tr>
<td>4</td>
<td><img src="Green" alt="Green" /></td>
<td><img src="Green" alt="Green" /></td>
<td><img src="Green" alt="Green" /></td>
<td>On track</td>
</tr>
<tr>
<td>5</td>
<td><img src="Green" alt="Green" /></td>
<td><img src="Green" alt="Green" /></td>
<td><img src="Green" alt="Green" /></td>
<td>On track</td>
</tr>
<tr>
<td>6</td>
<td><img src="Green" alt="Green" /></td>
<td><img src="Green" alt="Green" /></td>
<td><img src="Green" alt="Green" /></td>
<td>On track</td>
</tr>
</tbody>
</table>

**Demonstrations**

On track
Validation Plan - Summary

- **HIL system**
  - For RIAPS (software platform):
    - Development platform: Linux
    - Target platform: Beaglebone Black (embedded ARM)
  - For RAS (WSU):
    - Simulation: RTDS
    - Target platform (BBB)
  - For Microgrid control (NCSU):
    - Simulation: Opal-RT
    - Target platform (BBB) + custom DSP

- **Managed DERs:**
  - RAS: Wind farm
  - Microgrid: PV, Batteries (via inverter)

- **Test plan**
  - Platform: M1.2.1, 1.5.1, 1.7.1, 1.9.1., 2.1.2, 2.1.4, 2.2.1, 2.2.3, 3.1.2, 3.1.5, 3.2.4, 4.1.3, 4.2.3
  - RAS: 5.9.2,5.10.1,5.11.1.,5.11.2,
  - Microgrid: 5.2.1

- **Field validation test sites:**
  - Discussions with IAB members (Duke Energy/OpenFMB)

- **Large scale simulation plan:** RTDS + Opal-RT
Vision:
Distributed Computing for the Grid
RIAPS: The Software Platform

<table>
<thead>
<tr>
<th>Applications</th>
<th>Component Framework</th>
<th>Platform Managers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remedial Action Scheme</td>
<td><strong>Component Interactions</strong> Component Messaging</td>
<td><strong>Application Manager</strong> Application Management and Deployment Service</td>
</tr>
<tr>
<td>Microgrid Management</td>
<td><strong>Component Scheduling</strong> Event/Time-triggered</td>
<td><strong>Coordination Manager</strong> Distributed Coordination Service</td>
</tr>
<tr>
<td>State Estimation</td>
<td><strong>Lifecycle Management</strong> Initialize, Start, Stop, Checkpoint, Destroy</td>
<td><strong>Discovery Manager</strong> Broker Service</td>
</tr>
<tr>
<td>Data Analytics</td>
<td><strong>Language Run-time</strong> C/C++, etc.</td>
<td><strong>Time Manager</strong> Time Synchronization Service</td>
</tr>
<tr>
<td>Distributed SCADA</td>
<td><strong>Fault Management</strong> Fault Management Service</td>
<td><strong>Device Manager</strong> Device Interface Service</td>
</tr>
<tr>
<td>****</td>
<td><strong>Logging</strong> Logging Service</td>
<td><strong>Security Manager</strong> Security Management Service</td>
</tr>
<tr>
<td>****</td>
<td><strong>Persistence</strong> Persistence Service</td>
<td><strong>Persistence Manager</strong> Persistence Service</td>
</tr>
</tbody>
</table>

**OS Kernel**

**Hardware Platform**

Device Interfaces (Sensors/Actuators/Communications/GPS/...)

Network Interface(s)

Storage
RIAPS apps: Components + Actors

**Components** are the building blocks: defined interfaces (ports) + execution semantics – simple code, may encapsulate complex applications (e.g. numerical solvers)

**Actors** are built from components that interact solely via messages and are deployed on computing nodes in a network. All applications are built as a fabric of interacting components.
RIAPS services: Deployment

- RIAPS nodes and apps
  - are managed by a system operator (control room)
  - can join and leave the network at any time
RIAPS services: Discovery

- RIAPS components form a peer-to-peer network, organized and configured via the Discovery service
  - Service provider – service client match-up
RIAPS services: Resource management

- Resource: memory, CPU cycles, file space, network bandwidth, I/O devices

- Goal: to protect the ‘system’ from the over-utilization of resources by faulty (or malevolent) applications

- Use case:
  - Runaway, less important application monopolizes the CPU and prevents critical applications from doing their work

- Solution: model-based quota system, enforced by framework
  - Quota for application file space, CPU, network, and memory + access rights to I/O devices + response to quota violation – captured in the application model.
  - Run-time framework sets and enforces the quotas (relying on Linux capabilities)
  - When quota violation is detected, application actor can (1) ignore it, (2) restart, (3) shutdown.
RIAPS services: Fault management

- **Assumption**
  - Faults can happen anywhere: application, software framework, hardware, network

- **Goal**
  - RIAPS developers shall be able to develop apps that can recover from faults anywhere in the system.

- **Use case**
  - An application component hosted on a remote host stops permanently, the rest of the application detects this and ‘fails over’ to another, healthy component instead.

- **Philosophy:**
  - The platform provides the mechanics, but app-specific behavior must be supplied by the app.
RIAPS services: Distributed Coordination

- **Group membership:**
  - An app component can dynamically create/join/leave a *group* that facilitates fast communication among members.

- **Leader election:**
  - A group can ‘elect’ a *leader*: a component that makes global decisions. Election is automatic and fault tolerant, group members directly interact with the leader.

- **Consensus:**
  - Group members can ‘vote’ in a *consensus* process that reaches agreement over a value.

- **Time-coordinated control action:**
  - Group members use a combination of the above three features to agree on a *control action* that is executed at a scheduled point in time in the future.

- **Application example – Microgrid control**
  - Group Membership and Leader Election: ‘microgrid’ groups for sharing information for better control
  - Consensus: on voltage and frequency values
  - Time-coordinated control action: microgrid to islanded mode
RIAPS Development Tools

Eclipse

Development C++/Python

App model

Application package

App Deployment

App Execution
Objective: To minimize wind curtailment while keeping the system reliability

Constraints: Wind farm operational limits, line limits

Distributed Implementation: Distributed Simplex Method in Linear Programming

Offline simulation with N-1 computational block failure

Verification with decentralized algorithm implemented in real time using CISCO Fog and Beaglebone Black

<table>
<thead>
<tr>
<th>Mode of Operation</th>
<th>Wind Generation Curtailment</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>All nodes operational</td>
<td>100.0 MW -&gt; 84.14 MW</td>
<td>0.956s</td>
</tr>
<tr>
<td>Node failure</td>
<td>100.0MW -&gt; 81.60 MW</td>
<td>1.035s</td>
</tr>
</tbody>
</table>

- Without RAS, one of the transmission lines is overloaded by 16.6%.
- RAS algorithm curtail wind farm outputs and totally eliminated line overflows violation without any load shedding.
Testing and Validation Using Testbed

Substation #1

Substation #2

Substation #3

GPS → PMU → Measurements

Actuator

BBB

PMU

Measurements

Control Commands

GPS

Actuator

BBB

Substation #2

GPS → PMU → Measurements

Actuator

BBB
Testing and Validation Using Testbed

RAS for Minimum Wind Spilling
Contingency Scenario
- The power grid splits into two islands due to the tripping of three lines and two transformers;
- The South Island is with an active power deficit.

Simulation Results (RTDS and 3 RIAPS Nodes)
- Without RAS, frequency drops quickly leading to the system collapse;
- With RAS-2, the frequency decline is stopped at $t = 3.04$ s.

<table>
<thead>
<tr>
<th>UFLS Schemes</th>
<th>With RAS-2</th>
<th>No RAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVE LOAD SHEDDING (MW)</td>
<td>19.37</td>
<td>66</td>
</tr>
<tr>
<td>REACTIVE LOAD SHEDDING (MVAR)</td>
<td>9.58</td>
<td>32</td>
</tr>
<tr>
<td>LOWEST FREQUENCY (Hz)</td>
<td>58.2</td>
<td>SYSTEM COLLAPSE</td>
</tr>
<tr>
<td>STABLE FREQUENCY (Hz)</td>
<td>58.89</td>
<td>SYSTEM COLLAPSE</td>
</tr>
<tr>
<td>TIME WHEN FREQUENCY DECLINE STOPS (SEC)</td>
<td>3.02</td>
<td>SYSTEM COLLAPSE</td>
</tr>
<tr>
<td>MW REDUCTION IN LOAD SHEDDING COMPARED WITH &quot;No RAS&quot;</td>
<td>70.65%</td>
<td>N/A</td>
</tr>
<tr>
<td>MVAR REDUCTION IN LOAD SHEDDING COMPARED WITH &quot;No RAS&quot;</td>
<td>70.06%</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Testing and Validation Using Testbed
**Project Goal:** Implement *Distributed* Microgrid controller using RIAPS

Focus on time sensitive applications: islanded mode operation and transitions between grid connected and islanded operation

**Islanded Mode Applications:**
- F & V Restoration
- Load Shedding

**Fault Applications:**
- Protection
- Islanding Detection
- µ-grid Membership
- FTS Execution

**Grid-Connected Mode Applications:**
- Fault Transition Scheme (FTS)

**Scheduled Outage Applications:**
- µ-grid Membership
- Int. Islanding

**Grid Restored Applications:**
- Resynchronization

**Diagram:**
- Islanded Mode
- Grid-Connected Mode
- Fault
- Scheduled Outage
Implemented features

- Voltage and frequency restoration in islanded mode
- Distributed Microgrid Synchronization
- Islanding Detection
- Dynamic virtual Impedance control for improved reactive power sharing
- Private And Decentralized Energy Transactions
- Adaptive Interleaving to eliminate ac harmonics
- Time synchronized event logger
Hardware in the Loop Implementation

- **Opal-RT real time simulator**
  - Inverter-switching model in FPGA solver
  - Power system model in CPU solver
- **Texas Instruments F28377S MCU**
  - Inverter control
  - Modbus communication with Beaglebone Black
- **Beaglebone Black**
  - RIAPS node hardware
  - Communication with DSP via MODBUS
- **Linux machine**
  - Graphana Display
App: Private and Decentralized Energy Transactions

- Vanderbilt used real-world energy production / consumption data from a German microgrid provided by Siemens, CT

- Vanderbilt deployed system on a private Ethereum network; 90% of trades were closed within 23 seconds or less

- NCSU focus: Implement private and decentralized energy transactions on NCSU RIAPS HIL Testbed
FREEDM Hardware Testbed

- Demonstrate RIAPS in the FREEDM Green Energy Hub
- Testbed consists of 3 solid-state transformers (SST), 3 battery energy storage systems (ESS), smart house, house loads, programmable loads, etc.
- In the process of integrating RIAPS with 3 SSTs and 3 ESS
- SST can be configured to behave like a conventional inverter to emulate more conventional microgrids
**Project Challenges**

- **Past challenges:**
  - Many milestones required production of significant documentation → Focus on the essentials in documents
  - Complexity of software code based required for the platform is considerable → Use existing open-source packages (after careful analysis and testing)

- **Possible project challenges going forward:**
  - Testing and validation of platform will be a significant effort → Drive testing and validation with apps
List of Achievements

https://riaps.isis.vanderbilt.edu/papers.html

RIAPS

- P. Volgyesi, A. Dubey, T. Krentz, I. Madari, M. Metelko, G. Karsai, "Time Synchronization Services for Low-Cost Fog Computing Applications", International Symposium on Rapid System Prototyping (RSP), Seoul, South Korea, 10/2017
List of Achievements
https://riaps.isis.vanderbilt.edu/papers.html

RAS

List of Achievements

https://riapsisis.isis.vanderbilt.edu/papers.html

Microgrid

Objective:
- Open-source platform, supported by a spin-off

Market segment:
- Power system software developers and users

Commercial partners/advisors:
- ABB, Cisco, Duke Energy, National Instruments, RTE France, National Grid, OSISoft, Siemens, South California Edison, TVA
Future Plan

Next steps:
- Completion and verification of platform services
- Implementing the security features of the platform
- More distributed applications (improved RAS, microgrid, transactive energy, distributed SCADA, etc.)
- Tutorials for app developers

Interactions:
- Started discussions with OpenFMB about potential collaboration, integration, extensions and technology exchange