Towards a Resilient Information Architecture Platform for Smart Grid

Gabor Karsai, Abhishek Dubey (Vanderbilt)
Srdjan Lukic (NCSU)
Anurag Srivastava (WSU)

Supported by DOE ARPA-E under award DE-AR0000666

The Energy Revolution: Big Picture

From centralized to *decentralized* and *distributed* energy systems

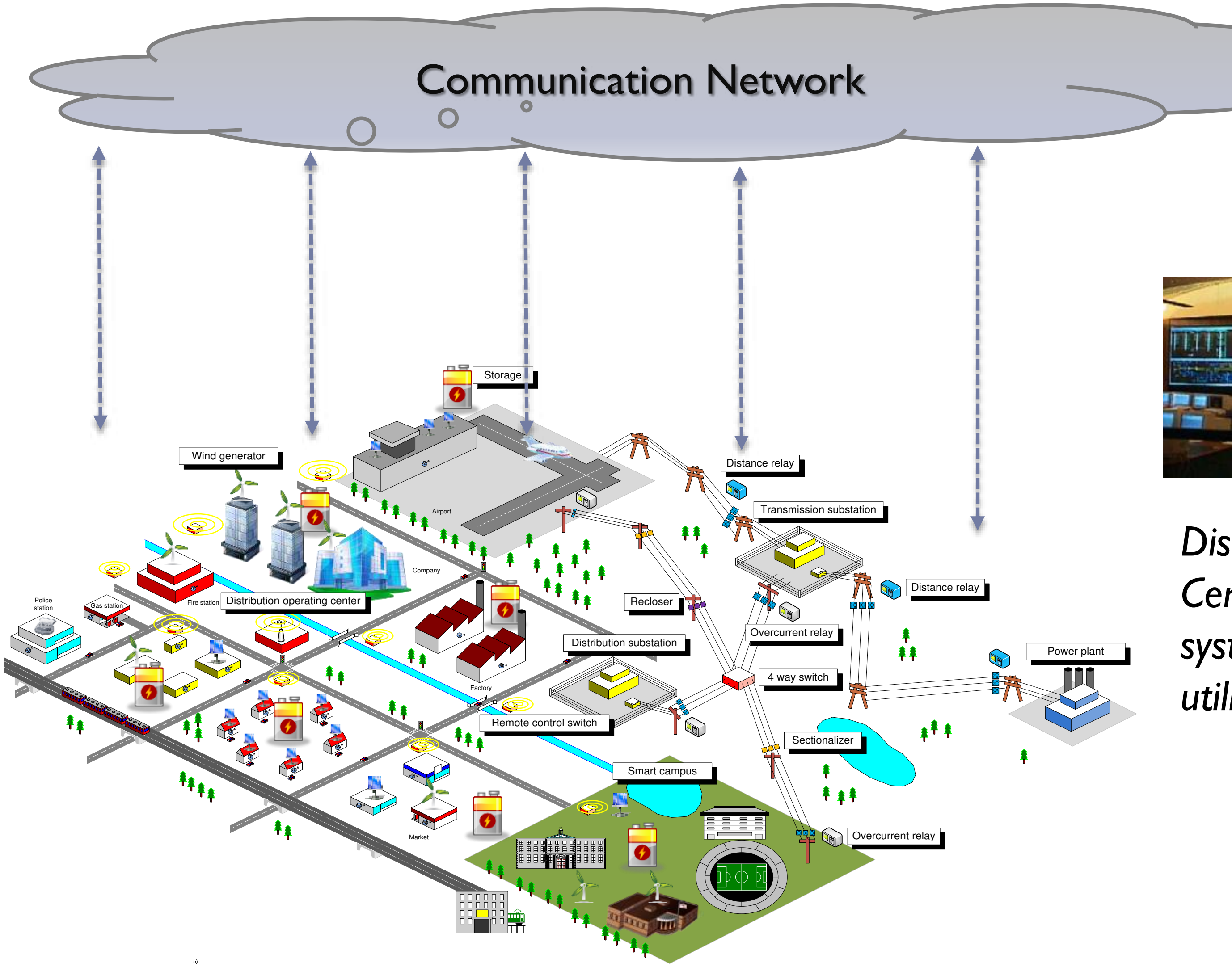
Changing Generation Mix

Electric Vehicles

Transactive Energy

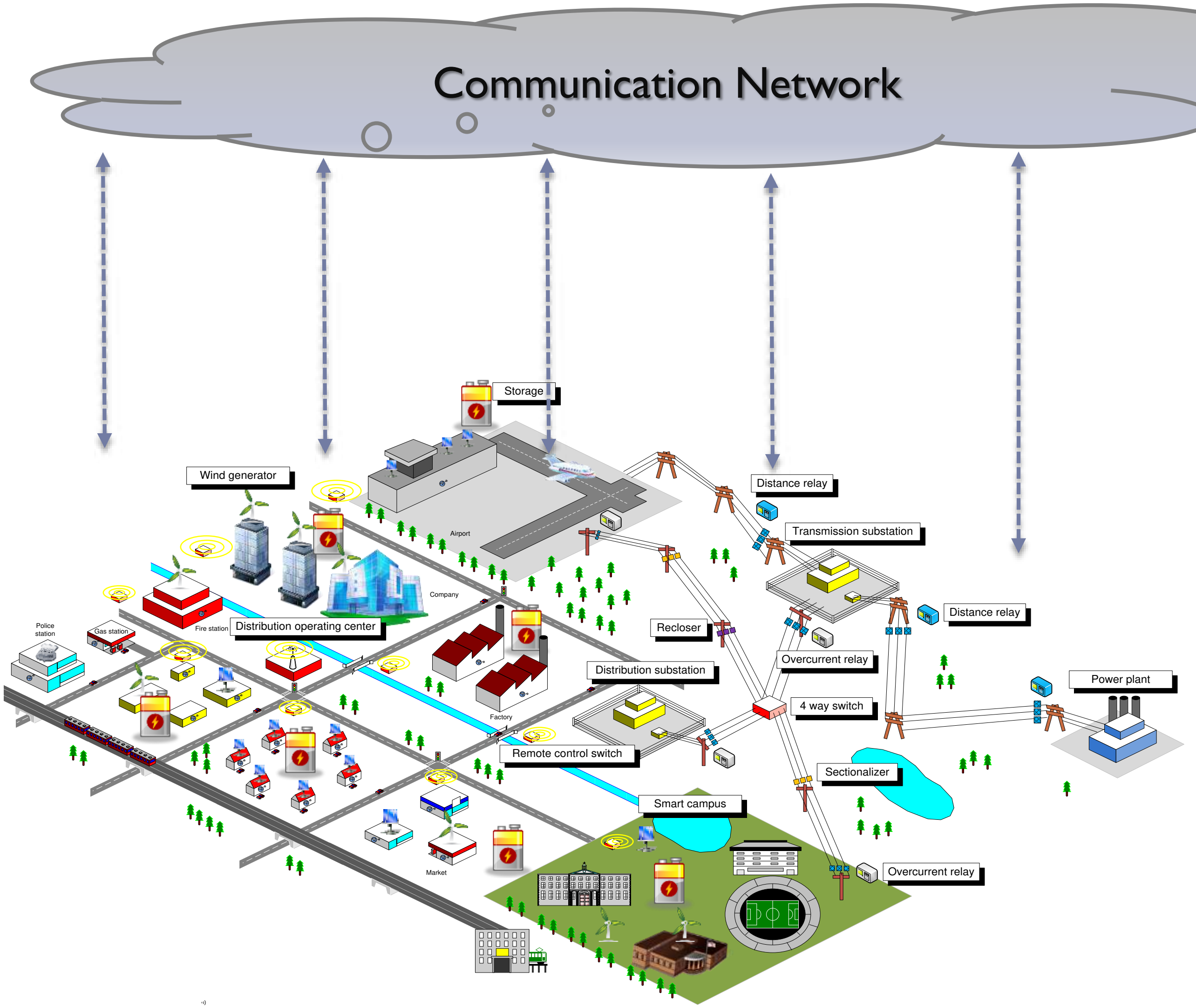
Decentralization

The control picture has not changed



Distribution: Centralized SCADA system managed by the utility company

The control picture has not changed

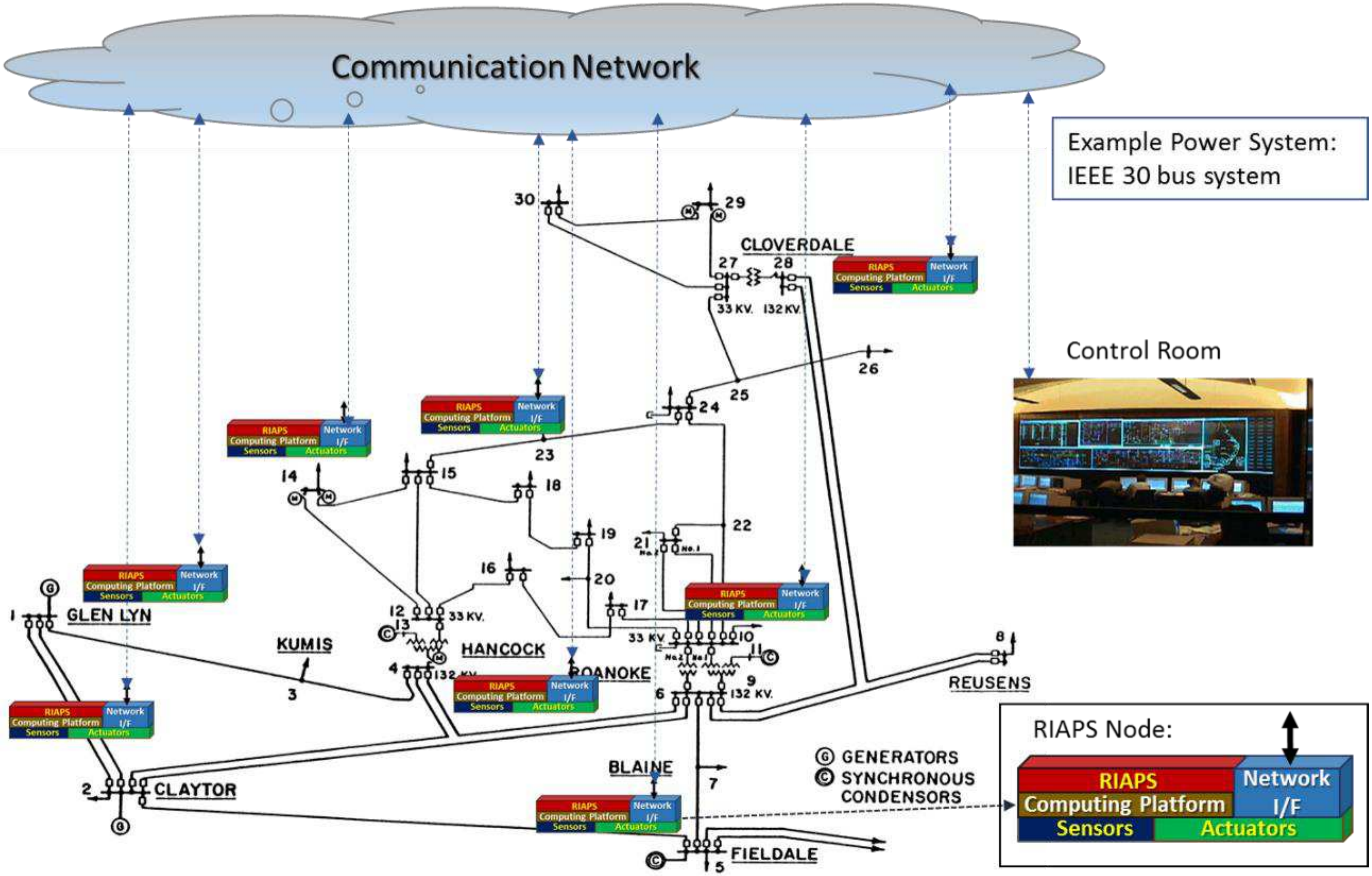


Problems

- Distributed Control
- Network latency
- Lack of interoperability
- Robust/resilient software
- Cyber-security
- Integration challenges
- …

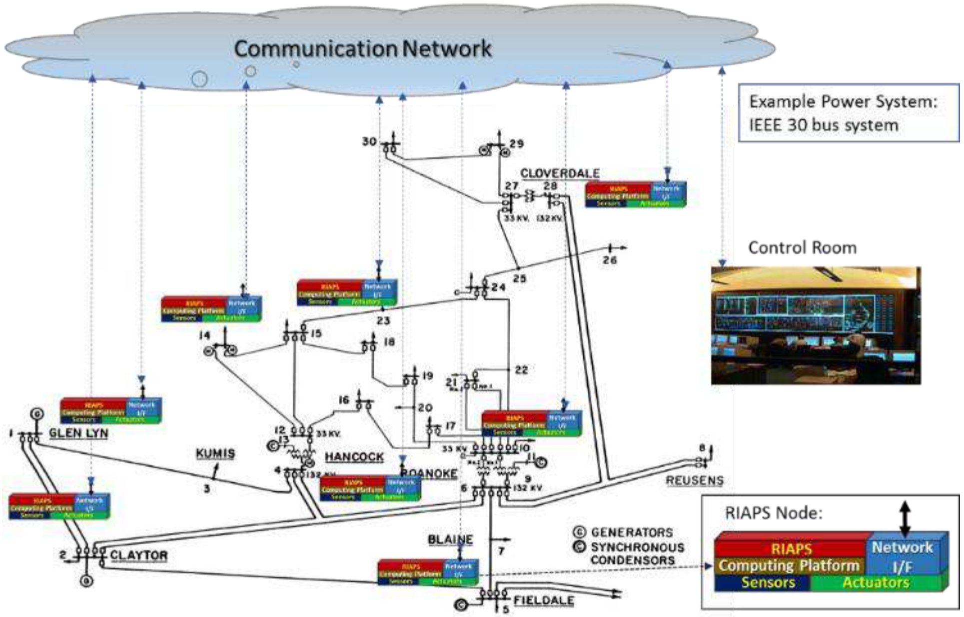
Q: IS THERE A BETTER WAY TO WRITE SOFTWARE FOR THIS?
A: YES, BUT WE NEED BETTER SOFTWARE INFRASTRUCTURE AND TOOLS.

RIAPS Vision



- *Showing a transmission system, but it applies to distribution systems, microgrids, etc.*

RIAPS Vision



- Push computation to the *edge*
- Enable *common* technology stack – a platform - across the ecosystem
- Provide core services to enable the *rapid* development of *smart* apps

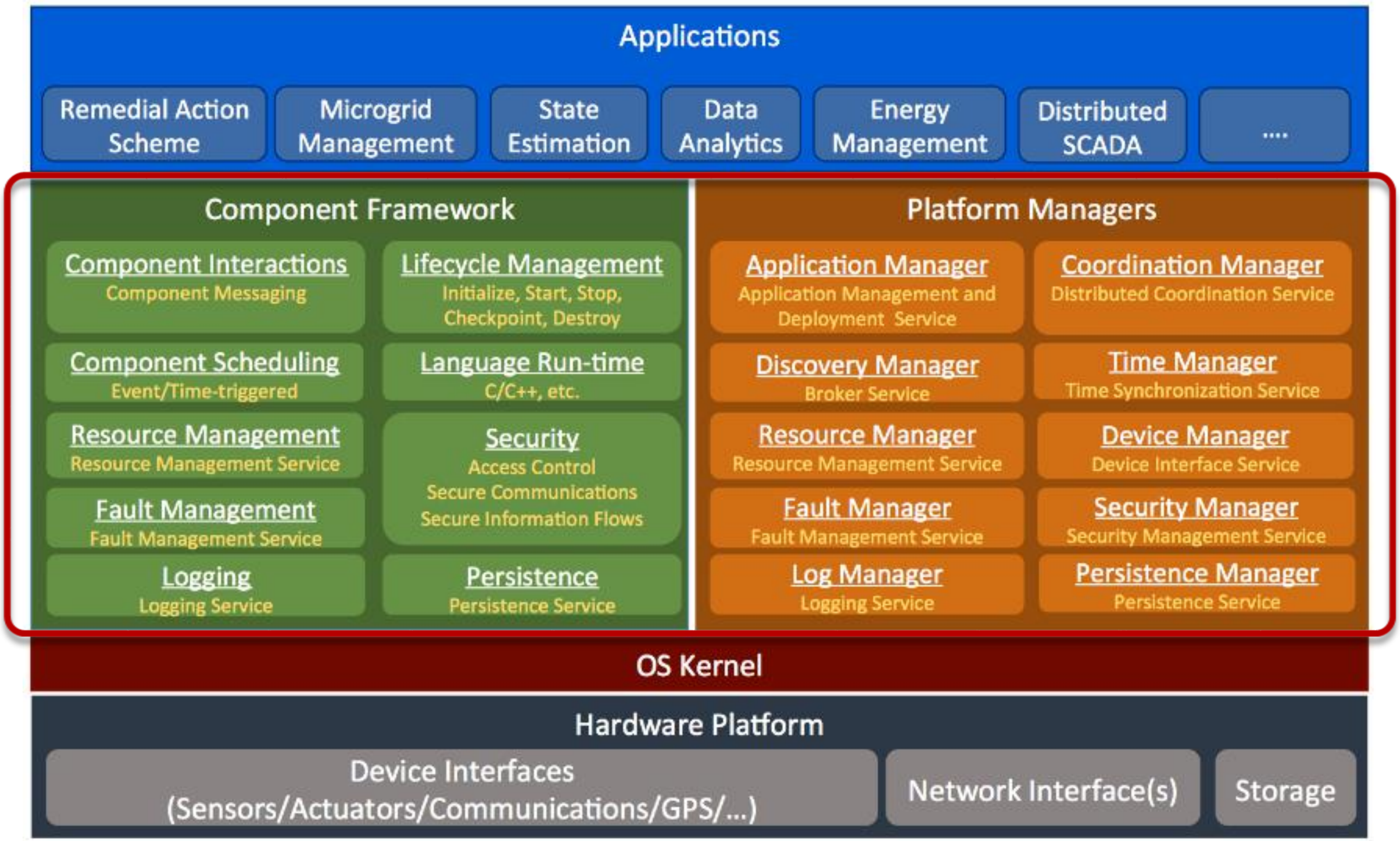
RIAPS Software Platform

- At the core of the RIAPS vision is **a reusable technology** stack to run **Smart Grid applications.**
- The software platform defines:
  - Programming model (for distributed real-time software) on embedded nodes **dispersed throughout the power grid**
  - 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

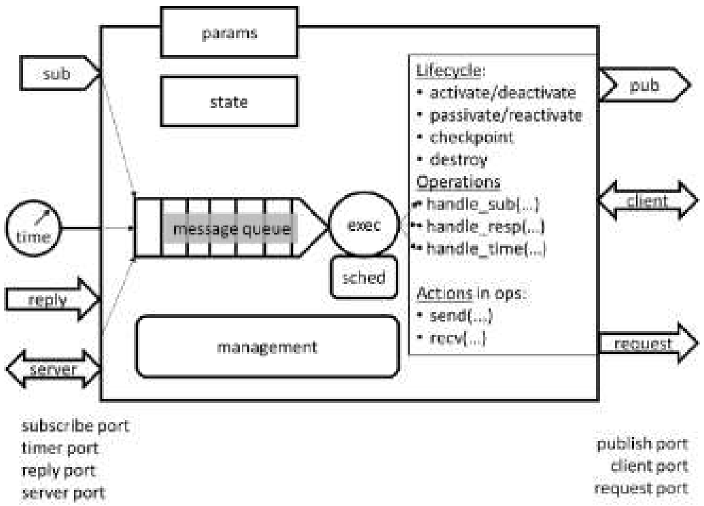
RIAPS Software Platform

- **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.*
    - Build apps from *components* and *actors*, use modular construction, focus on *interfaces* and *interactions*
    - Manage *resources*, *faults*, and *security* – rely on app's business logic
    - Provide common *services* for app deployment, integration, coordination, time synchronization, fault management, ….
  - How do we *manage* accidental complexities in the development process? – *Developers need tools to be productive.*
    - Separate app architecture from algorithms, make both explicit
    - Provide services for packaging, deployment, and operation
    - Supply tools for design automation (languages, code generators, etc.)

RIAPS Details
The Software Platform

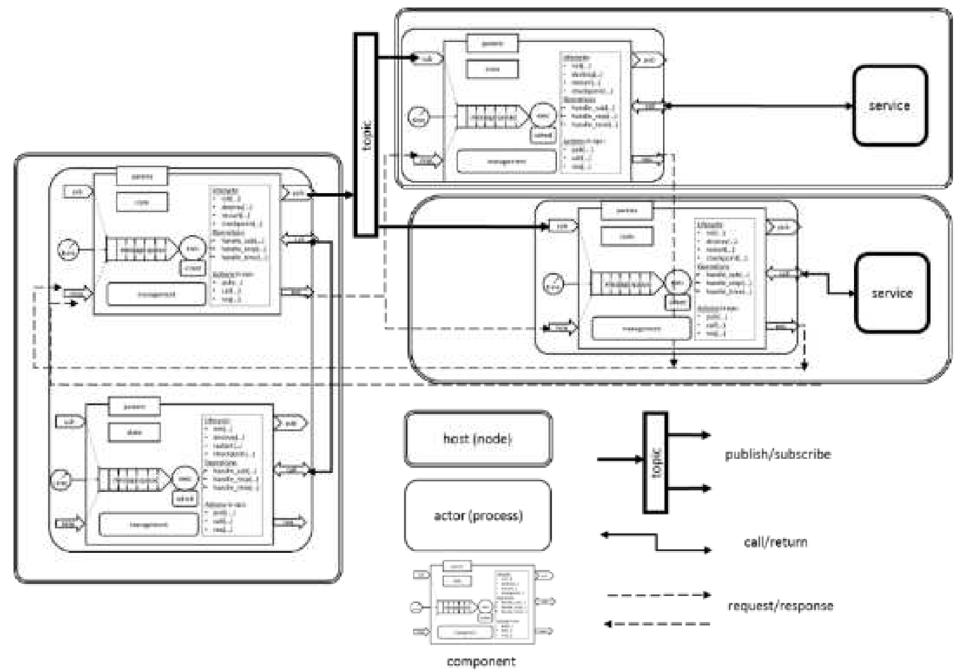


RIAPS Details
Apps = Components + Actors



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

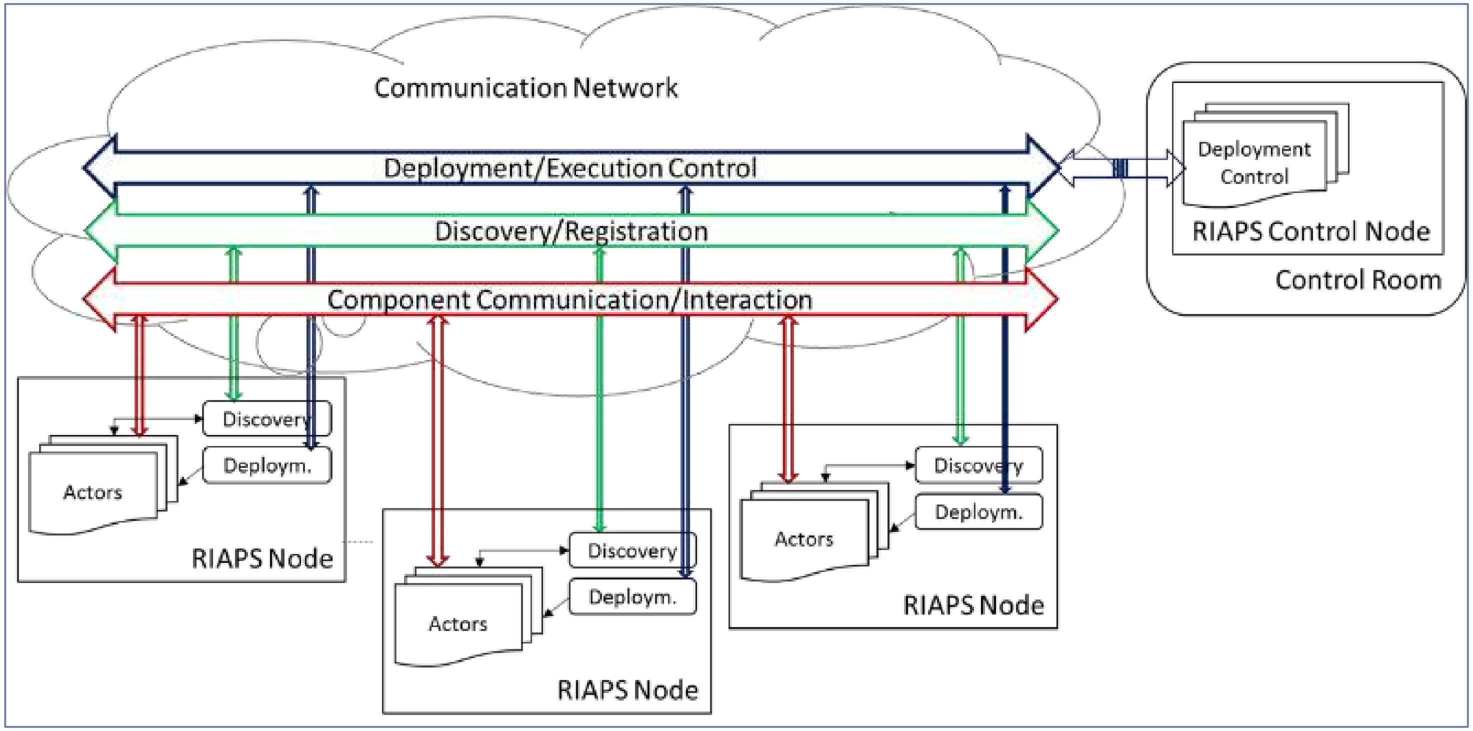
Actors are built from components that interact solely via messages and are deployed on computing nodes in a network.
All applications are built as a fabric of interacting components



Benefits: Reusable components + concurrency is handled in the framework (not in the 'business logic') + lends itself to timing analysis

RIAPS Details
Services: Deployment

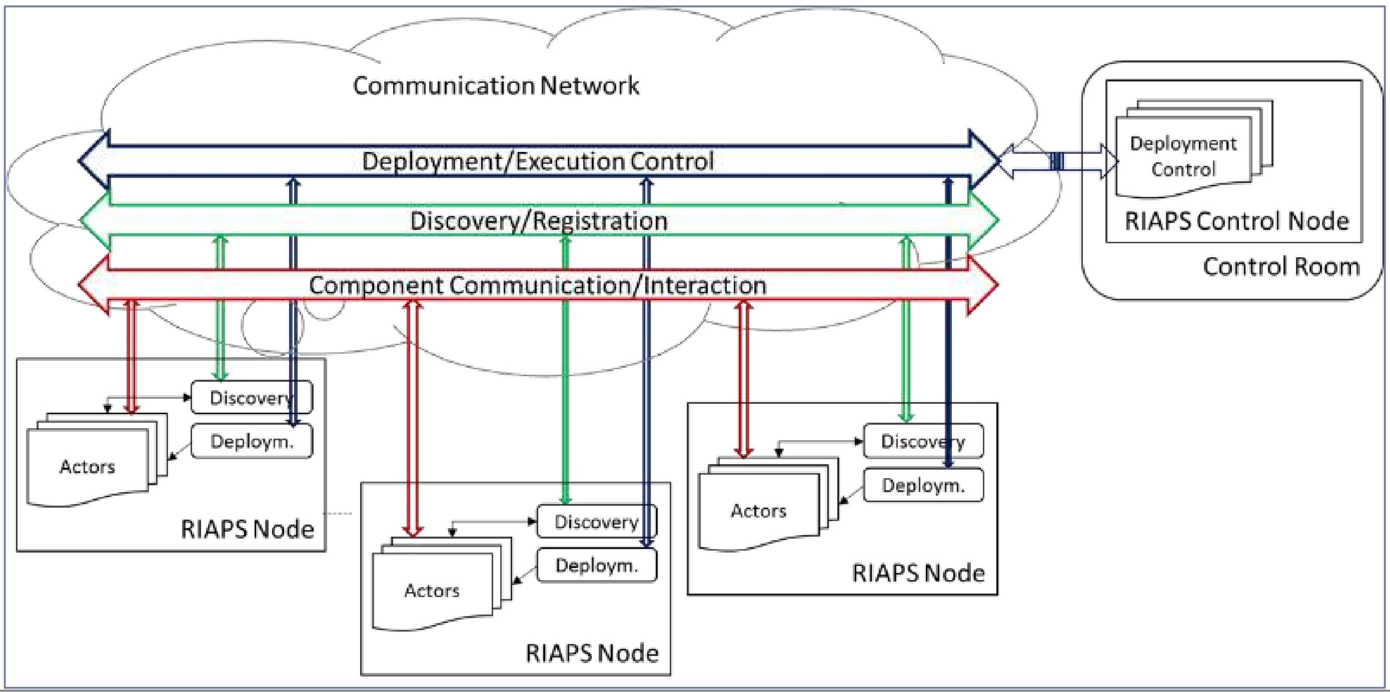
- RIAPS nodes and apps
  - are remotely managed by a system operator (control room)
  - can join and leave the network at any time



Benefit: Authoritative control over all software deployed on the RIAPS network.

RIAPS Details
Services: Discovery

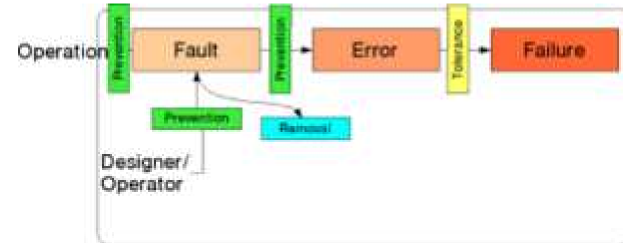
- RIAPS components form a peer-to-peer network, organized and configured via the Discovery Service
  - Service provider – service client match-up

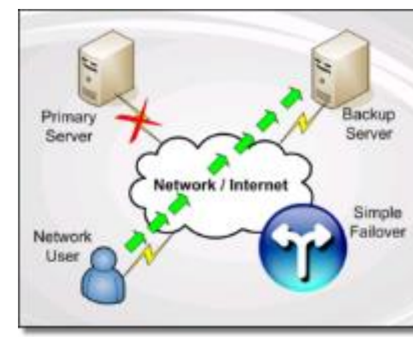


Benefit: Actors of a RIAPS app can join and leave at any time – yet able to connect to and operate within the group reliably.

RIAPS Details
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.
- **Principle**
  - The platform provides the *mechanics*, but app-specific behavior <u>must be </u>supplied by the app developer





Benefit: Complex mechanisms that allow the implementation of resilient apps.

RIAPS Details
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

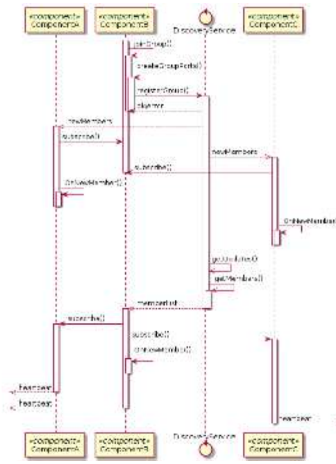
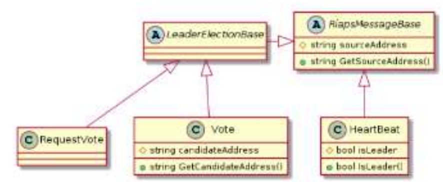
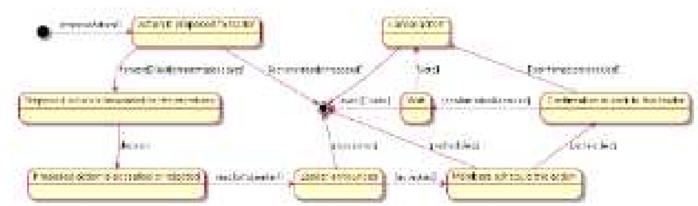


Figure 6: States of the leader election

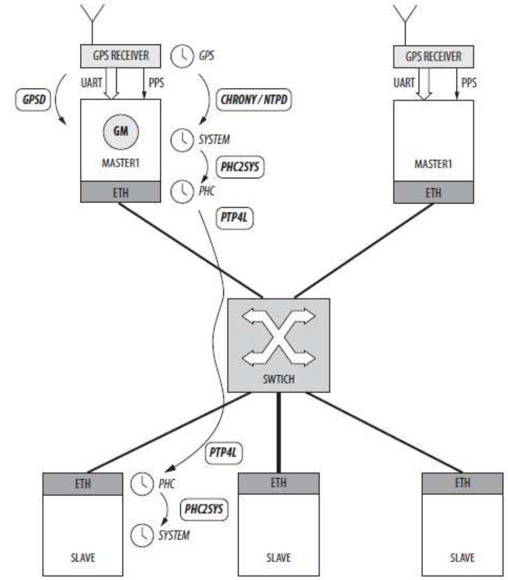


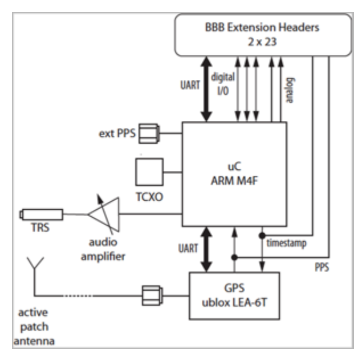


Benefit: Reusable implementation of complex algorithms – available as a service.

RIAPS Details
Services: Time Synchronization

- **High-precision Time Synchronization**
  - Maintains a cluster-wide synchronized notion of time
  - Applications can: (1) query the global time, (2) sleep until a specified point in time, (3) query the status of the service
  - Architecture:
    - Use PTP (IEEE-1588)
    - Some nodes may have a GPS
    - GPS clock is distributed
    - Fallback: NTP
    - Accuracy: ~10 usec
  - Board support: GPS receiver

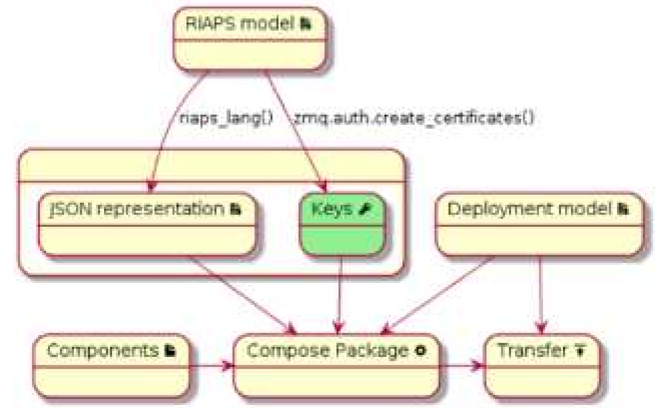




Benefit: Precisely synchronized time base available to all apps on the RIAPS network.

RIAPS Security
Application deployment

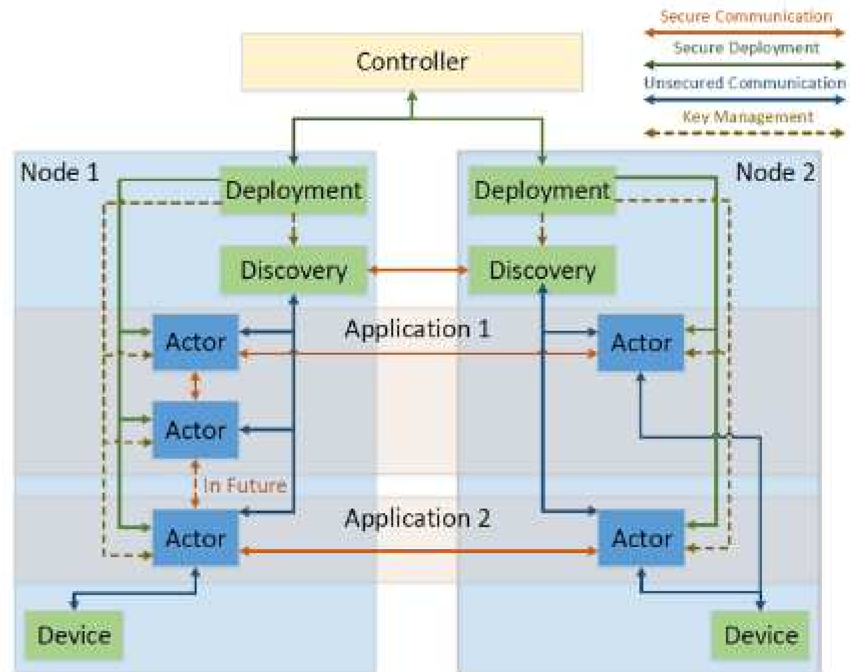
- **Secure applications**
  - Application packages are compressed, encrypted and cryptographically signed before deployment. The recipient nodes verify cryptographic signatures, decrypt, and install the app.
  - All app-level communications are protected by the CurveCP (elliptic curve encryption) on the messaging layer. All communications are protected via public/private key-pairs that are generated dynamically when the app is deployed. Keys are installed whenever an app-level network connection is established, and they are part of the deployment package, stored in a certificate store on the target nodes.



Benefit: State-of-the-art, industrial-grade security for app deployment and communications.

RIAPS Security
Secure deployment / communications

- **Secure messaging between services**
  - Unsecured – communication is among processes on the same host
    - Deployment service ←→ actor
    - Deployment service ←→ discovery service
    - Actor ←→ discovery service
  - Discovery service
    - DHT already encrypts all service registrations
    - Discovery service instances use a single shared key across the network
    - Private key on node is protected via file access control



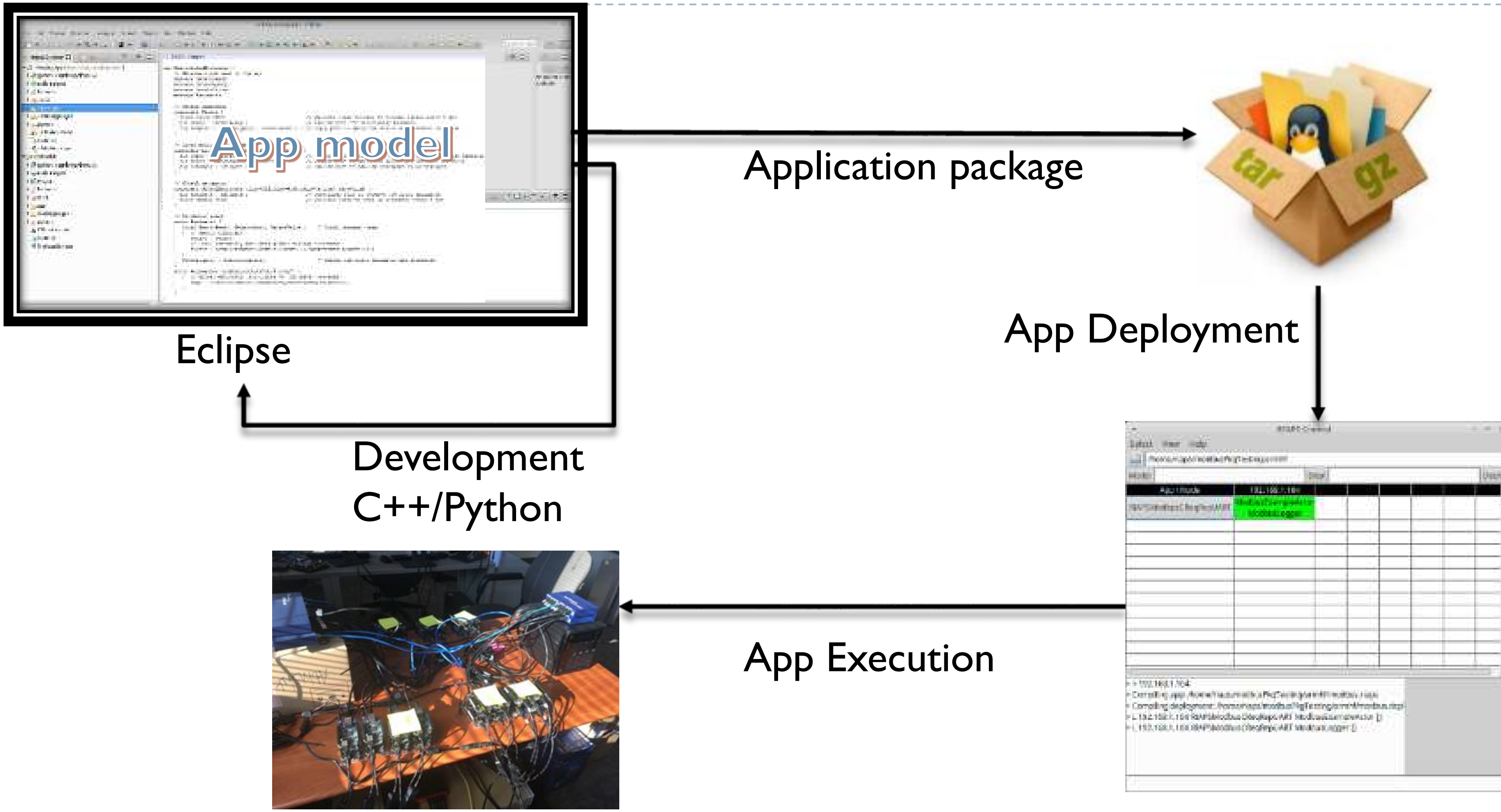
Benefit: State-of-the-art, industrial-grade security for app deployment and communications.

RIAPS Security
Application level protection

- **Network threats**
  - Each app actor is allowed to accept network packets only from hosts participating in the same app: App-level firewall on the incoming messages
- **Insider threats (malicious / flawed app)**
  - Network protection
    - App's view of the network is explicitly modeled and used in configuring firewalls on the hosts
    - Firewall allows only communication within the RIAPS app's network (both directions)
    - Exceptions are configurable by system integrator ('owner')
  - Information flow protection
    - AppArmor (a Linux Mandatory Access Control [MAC]) system is used to constrain the app's access
    - Security profile is enforced by the trusted installer (Deployment Manager)
    - Default access: own files, core system packages, TCP/UDP protocols – very constrained – maybe necessary to allow app-specific overrides

Benefit: Strict isolation of apps from each other, access control on shared resources.

RIAPS Development Tools



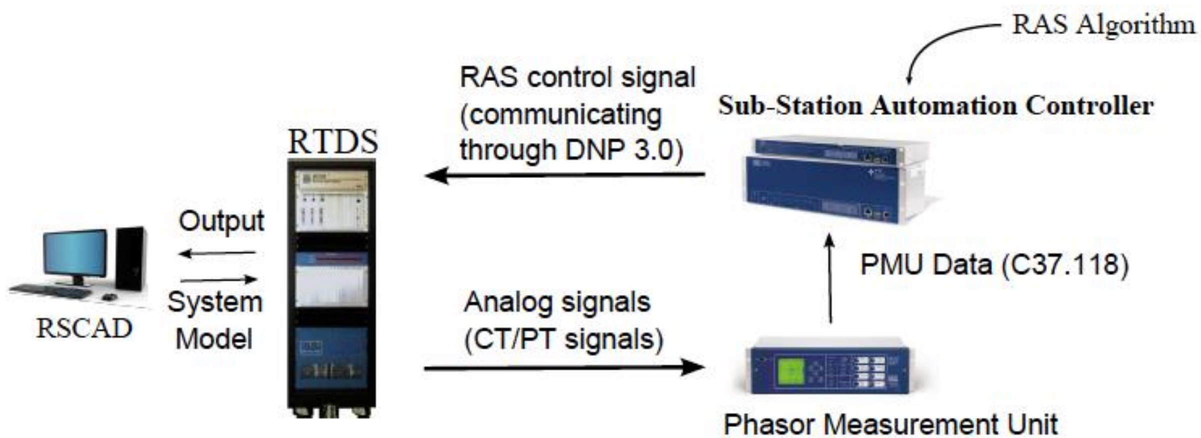
Benefits: Developer can focus on the core logic of the application (the 'algorithms') – the composition and configuration is done on a higher-level of abstraction.

19

RIAPS Apps
RAS, Microgrid, Transactive Energy

Application1:
Response Based Remedial Action Scheme (WSU)

- RAS is a key mechanism to protect electric power grid, generally used as the last line of automatic defense
- Existing RAS are pre-determined, inflexible and do not factor in changing system conditions and might take control actions good for small system but not optimal for the overall power grid
- RIAPS enables dynamic coordinated response based RAS (DCRB-RAS), which uses measurements, changing network conditions, control settings to dynamically decide control decisions



Application1:
Response Based Remedial Action Scheme (WSU)

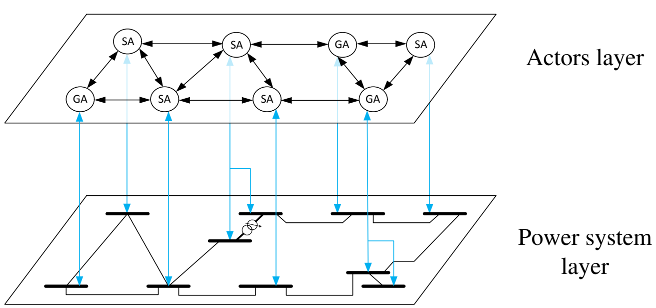
Two applications

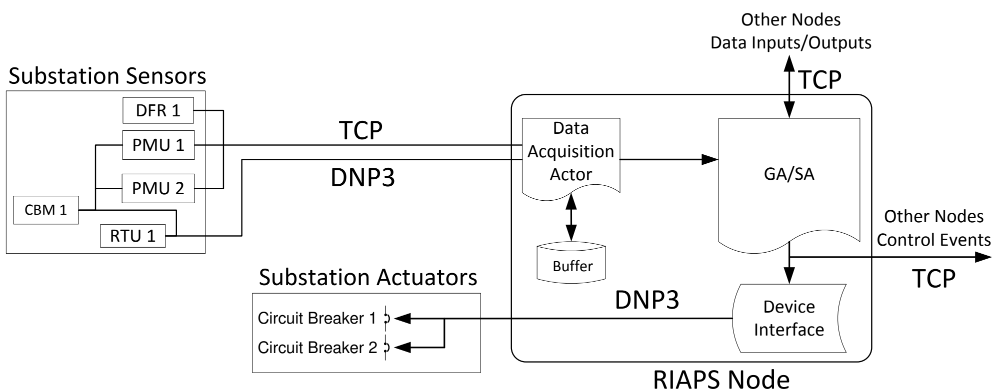
RAS I for managing wind generation: curtailment

- Use a distributed state estimation to determine the current state of the network
- If generation exceeds demand, calculate an optimal curtailment of wind turbine generation

RAS II for under-frequency control: load shedding

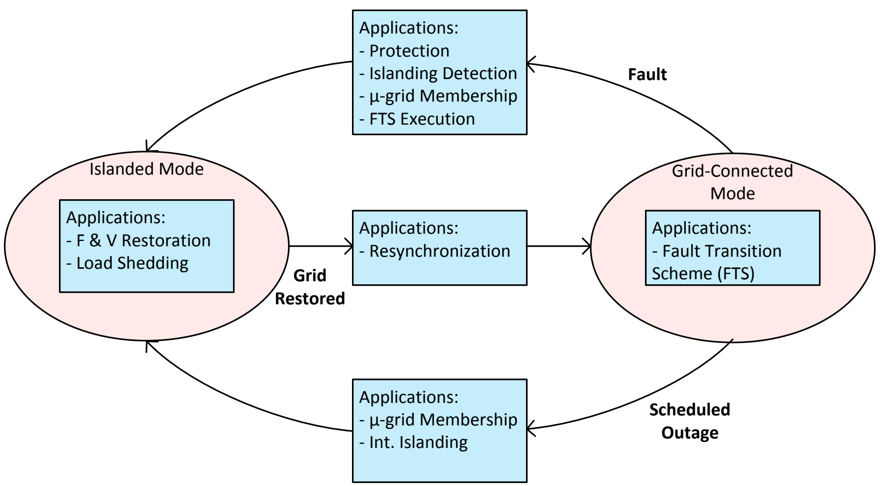
- Detect if system frequency drops below acceptable limit due to high load
- Calculate which loads to shed using a distributed algorithm

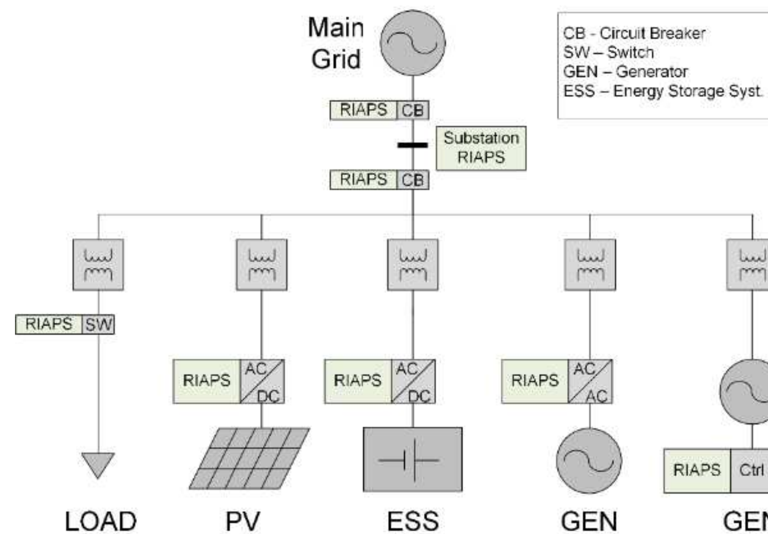




Application 2:
Microgrid Control (NCSU)

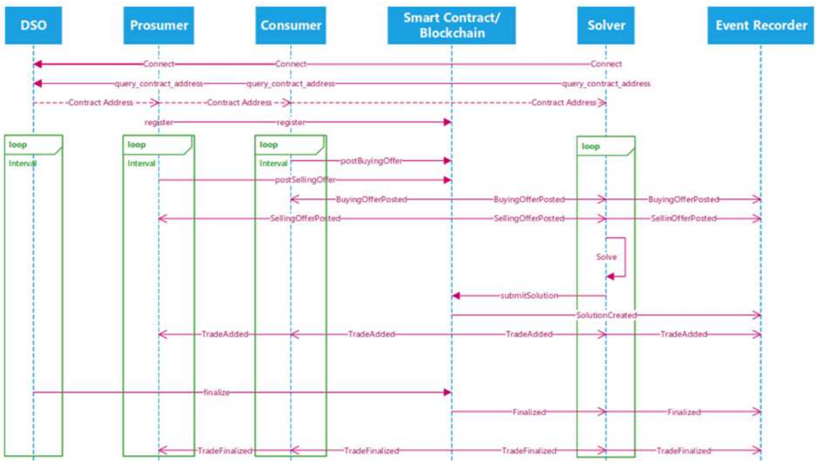
- Formation and interactions of microgrids (with local generation and energy storage) on a distribution feeder
- Focus: power management
- Main application scenario:
  - Unplanned transition from grid-connected to islanded mode and re-synchronization.
  - Distributed control and protection framework will be used to implement a fast transition scheme

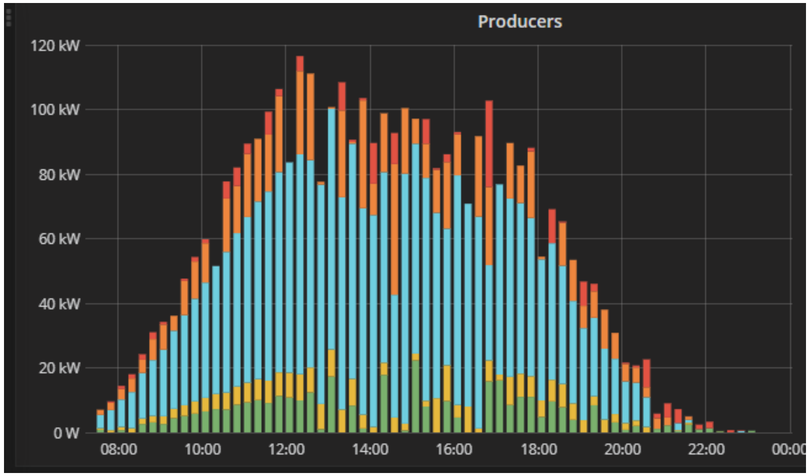




Application 3:
Blockchain based Energy Trading (VU)

- Transactive energy is a system of economic and control mechanisms that allows the dynamic balance of supply and demand across the entire electrical infrastructure using value as a key operational parameter.
- Challenges
  - Safety and efficiency
  - Integrity and auditability
  - Information privacy
- Solution is distributed system using blockchain and smart contracts
- Offers posted, broadcast, matched and traders notified.





Project Summary

- **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
- Websites:
  - https://riaps.isis.vanderbilt.edu/ - Project
  - https://github.com/RIAPS - Code base
  - https://riaps.github.io/ - Documents
  - https://www.youtube.com/channel/UCwfT8KeF-8M7GKhHS0muawg - Youtube channel

RIAPS: An LF Energy Project

- Speed technological innovation and enable the energy transition, globally.
- Facilitates standardizing, normalizing, and removing competition for shared “plumbing” to expedite the delivery of new products and services

YOUR VALUE

Spend your valuable time and resources developing the 20-30% that is **your secret sauce.**

LF ENERGY SOFTWARE STACK

Multi-vendor open source: collaboratively develop and support 70-80% of the starting point for a production-ready project – collaborating across the industry in order to **achieve scale and value FASTER!**

https://www.lfenergy.org/

RIAPS Team Members

- Vanderbilt University
  - Gabor Karsai
  - Abhishek Dubey
  - Istvan Madari
  - Mary Metelko
  - Peter Volgyesi
  - Tim Krentz
  - Purboday Ghosh
  - Scott Eisele [Supported by Siemens]
  - Joey Holliday [Undergrad Researcher]
- North Carolina State University
  - Srdjan Lukic
  - David Lubkeman
  - Yuhua Du
  - Hao Tu
  - Hui Yu
- Washington State University
  - Anurag Srivastava
  - Chen-Ching Liu
    - Moved to VT with Joint Appointment at WSU
  - Dave Bakken
  - Jing Xie
  - Vignesh Krishnan
  - Alex Askerman
  - Shyam Gopal
  - Zhijie Nie