GENI System Architecture: Overview and Narrow Waist

Facility Architecture Working Group / Narrow Waist Working Group

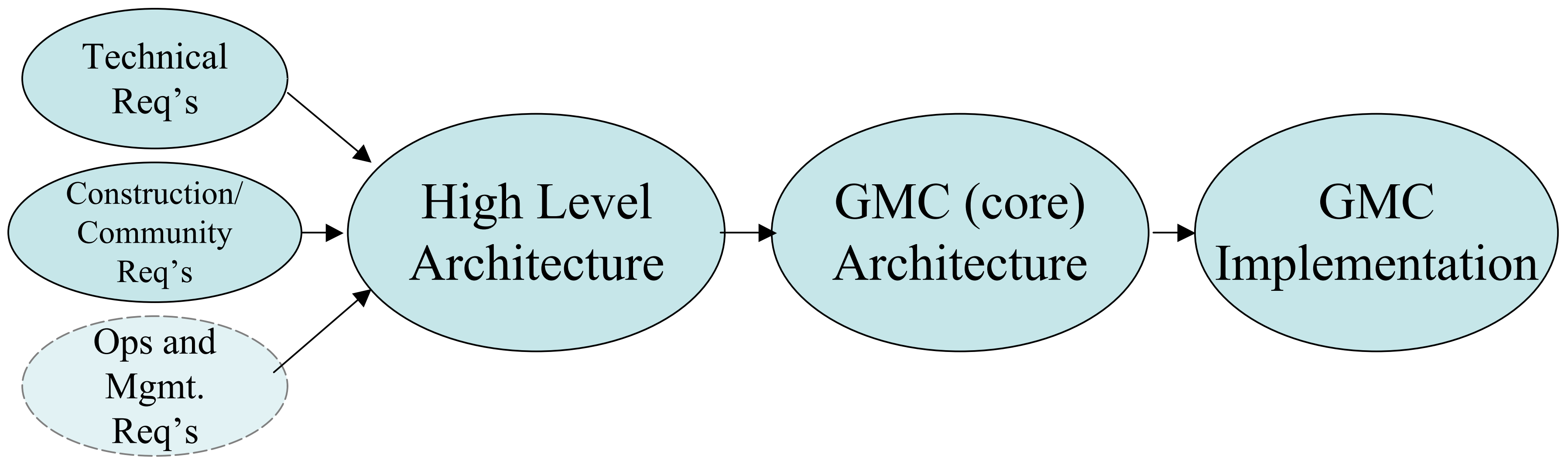
Larry Peterson and John Wroclawski

1st GENI Engineering Conference
Oct 10, 2007

Purpose of Talk

- Describe the (strawman) overall GENI system architecture
  - What
  - Why
- Describe the (strawman) core or “narrow waist” of this architecture
  - “Fixed points” around which the rest of GENI can be built and evolve
- Provide (initial) context for people interested in building other parts of GENI
  - Explanation of Narrow Waist abstractions and interfaces
  - Maturity levels
  - Open issues and future development directions

Design flow



Requirements $\searrow$

Overall Design: modularity and structure $\searrow$

Narrow Waist: abstractions and Interfaces $\searrow$

Narrow Waist: algorithms and implementation choices

Top-Level Requirements

1. Generality
   A. Minimal Constraints
      - allow new data formats, new functionality, new paradigms,…
      - allow freedom to experiment across the range of architectural issues (e.g., security, management,...)

   B. Breadth of Representative Technology
      - include a diverse and representative collection of networking technologies, since any future Internet must work across each of them, and the challenges/opportunities they bring
2. Sliceability
   - support many experiments in parallel
   - isolate experiments from each other, yet allow experiments to compose their experiments to build more complex systems

Top-Level Req (cont)

3. Fidelity
   - A. Device Level
     - ➤ expose useful level(s) of abstraction, giving the experimenter the freedom to reinvent above that level, while not forcing him or her to start from scratch (i.e., reinvent everything below that level)
     - ➤ these abstractions must faithfully emulate their real world equivalent (e.g., expose queues, not mask failures)
   - B. Network Level
     - ➤ arrange the nodes into representative topologies and/or distribute the nodes across physical space in a realistic way
     - ➤ scale to a representative size
     - ➤ expose the right network-wide abstractions (e.g., circuits, lightpaths)
   - C. GENI-Wide
     - ➤ end-to-end topology and relative performance
     - ➤ economic factors (e.g., relative costs, peering)

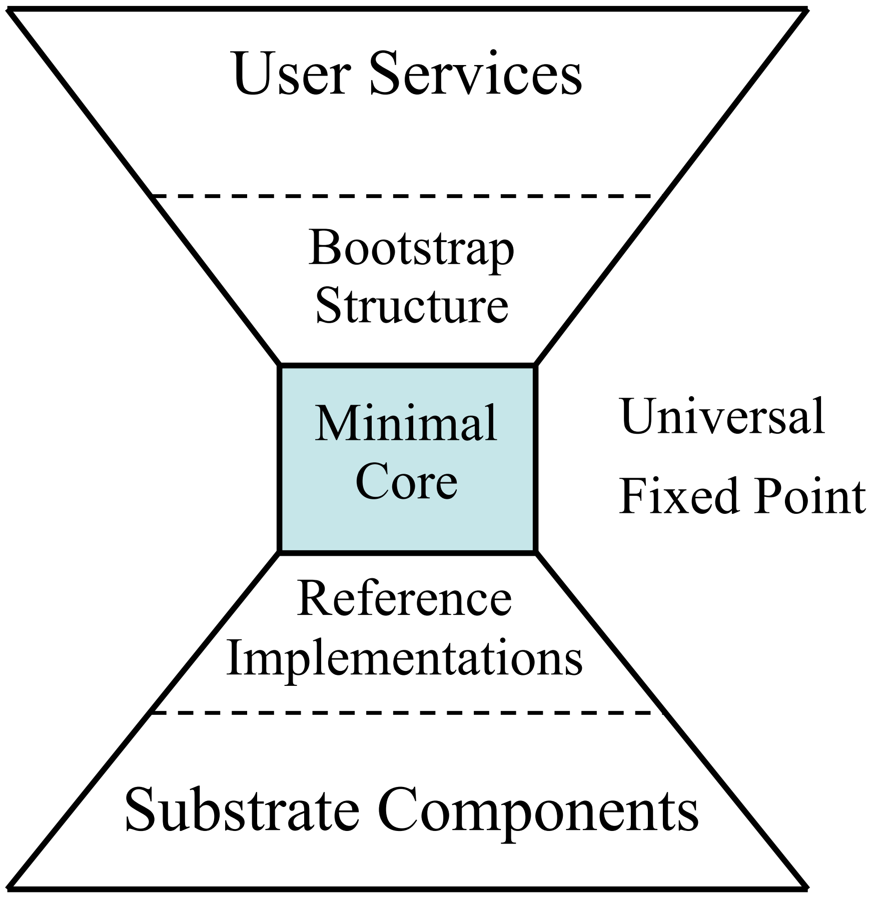
Top-Level Req (cont)

4. Real Users
   - allow real users to access real content using real applications
   - provide incentives and mechanisms to encourage this
   - Support *long-lived* experiments and services
5. Research Support
   - A. Ease-of-Use
     - provide tools and services that make the barrier-to-entry for using GENI as low as possible (e.g., a single PI and one grad student ought to be able to use GENI)
     - Key point: this community builds its own tools..
   - B. Observability
     - make it possible to observe and measure relevant activity

Top-Level Req (cont)

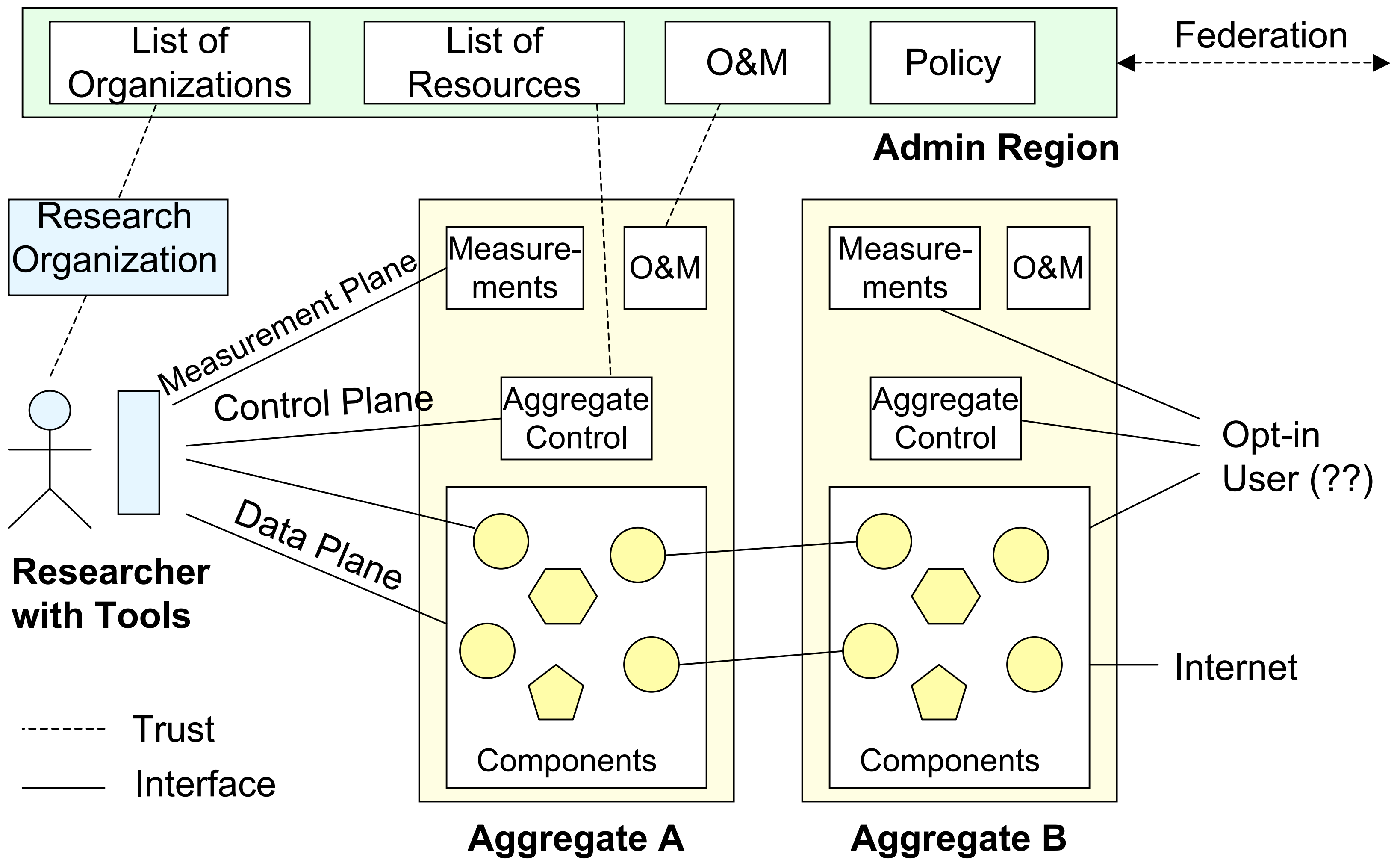
6. Sustainability
   A. Extensible and evolvable
      - ➢ accommodate network technologies contributed by various partners
      - ➢ accommodate new technologies that are likely to emerge in next several years
      - ➢ support technology roll-over without disruption
   B. Operational Costs
      - ➢ the community should be able to continue to use and support the facility long after construction is complete
      - ➢ Trade off increased capital cost for decreased operational cost when appropriate

Facility Architecture (View 1)



- name spaces, registries, etc
  -for key system elements
  (users, slices, & components
- set of interfaces
  -("plug in" new substrate
  components)
- support for federation
  -("plug in" new partners)

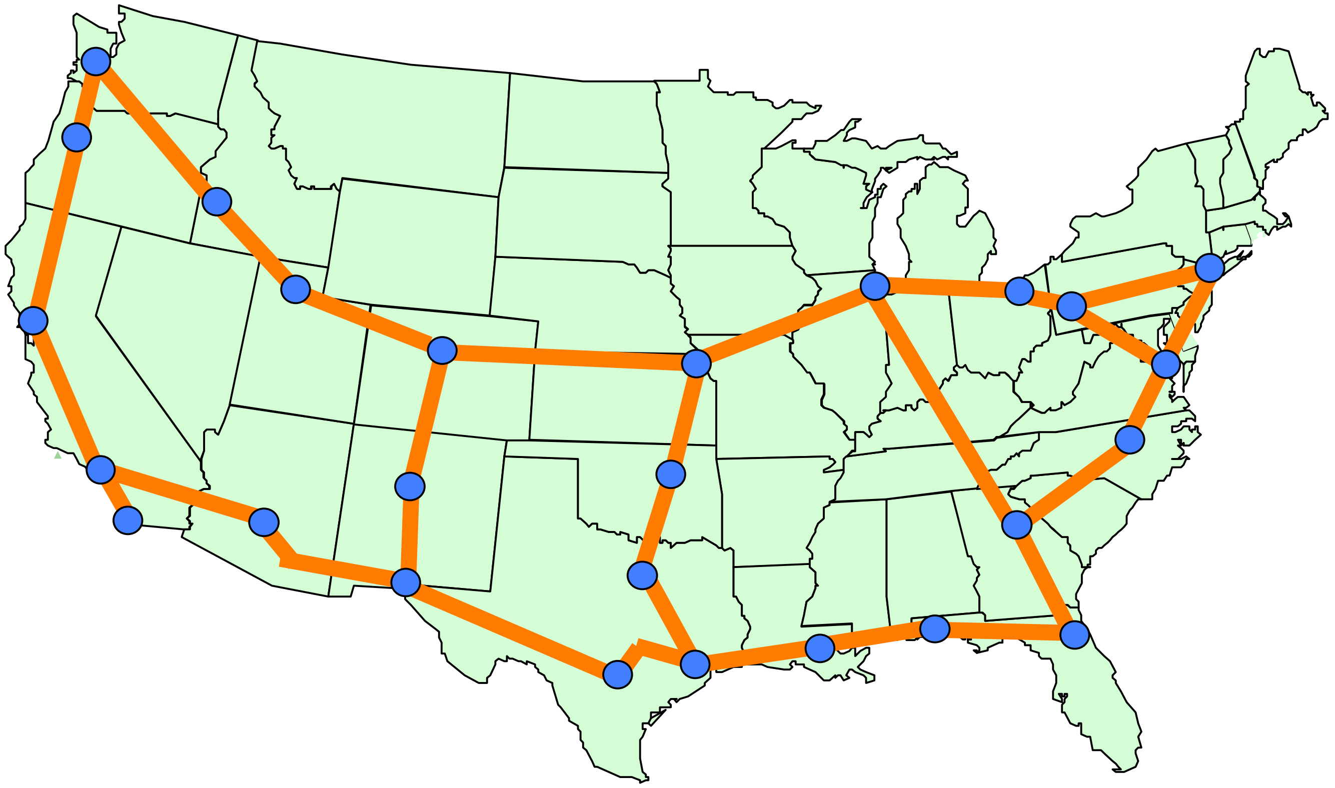
Facility Architecture (View 2)



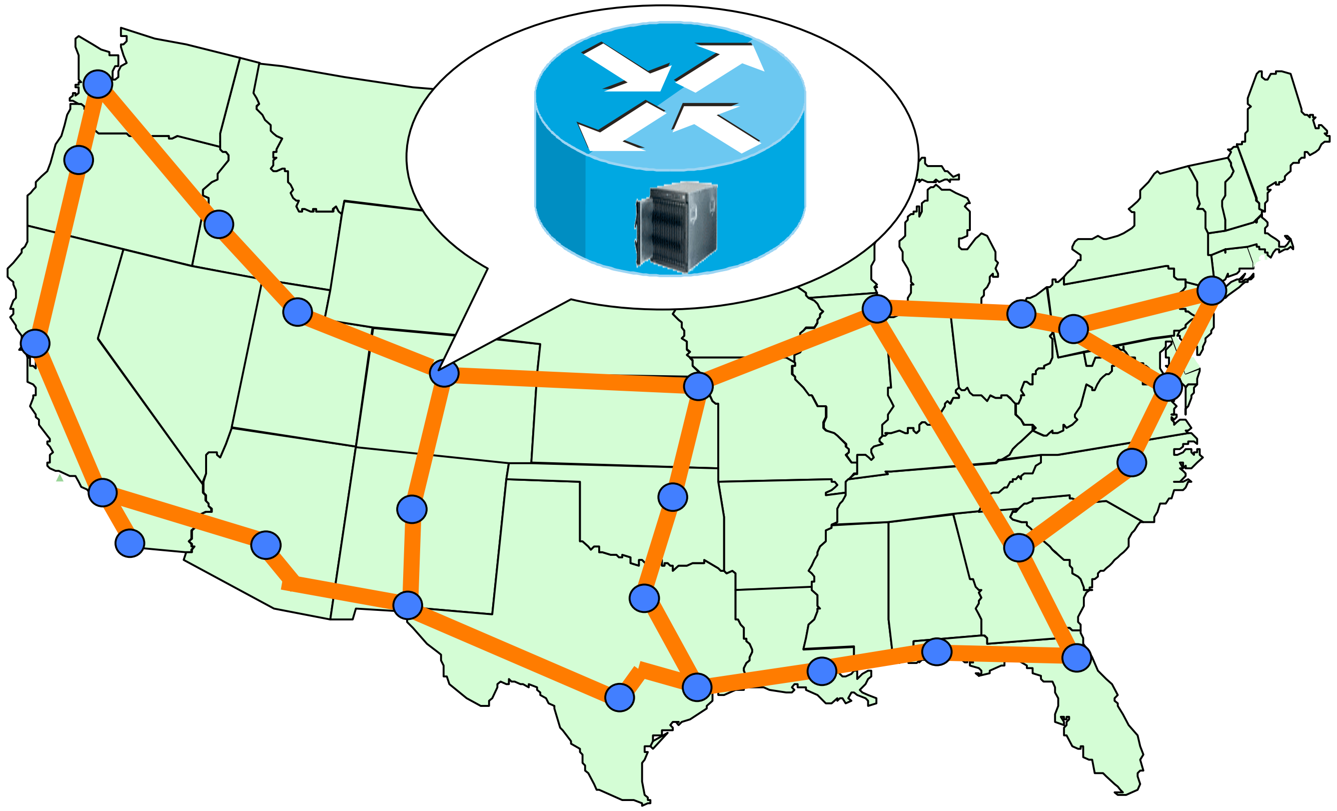
Diversion: the physical substrate

- The key function of the GENI system is to support flexible, useful embedding of *experiments* within a shared *physical substrate*.
- To understand the system architecture, it is helpful to first understand the sorts of physical resources the architecture is designed to control.
- I will walk through this quite quickly.
  I will tell lies.

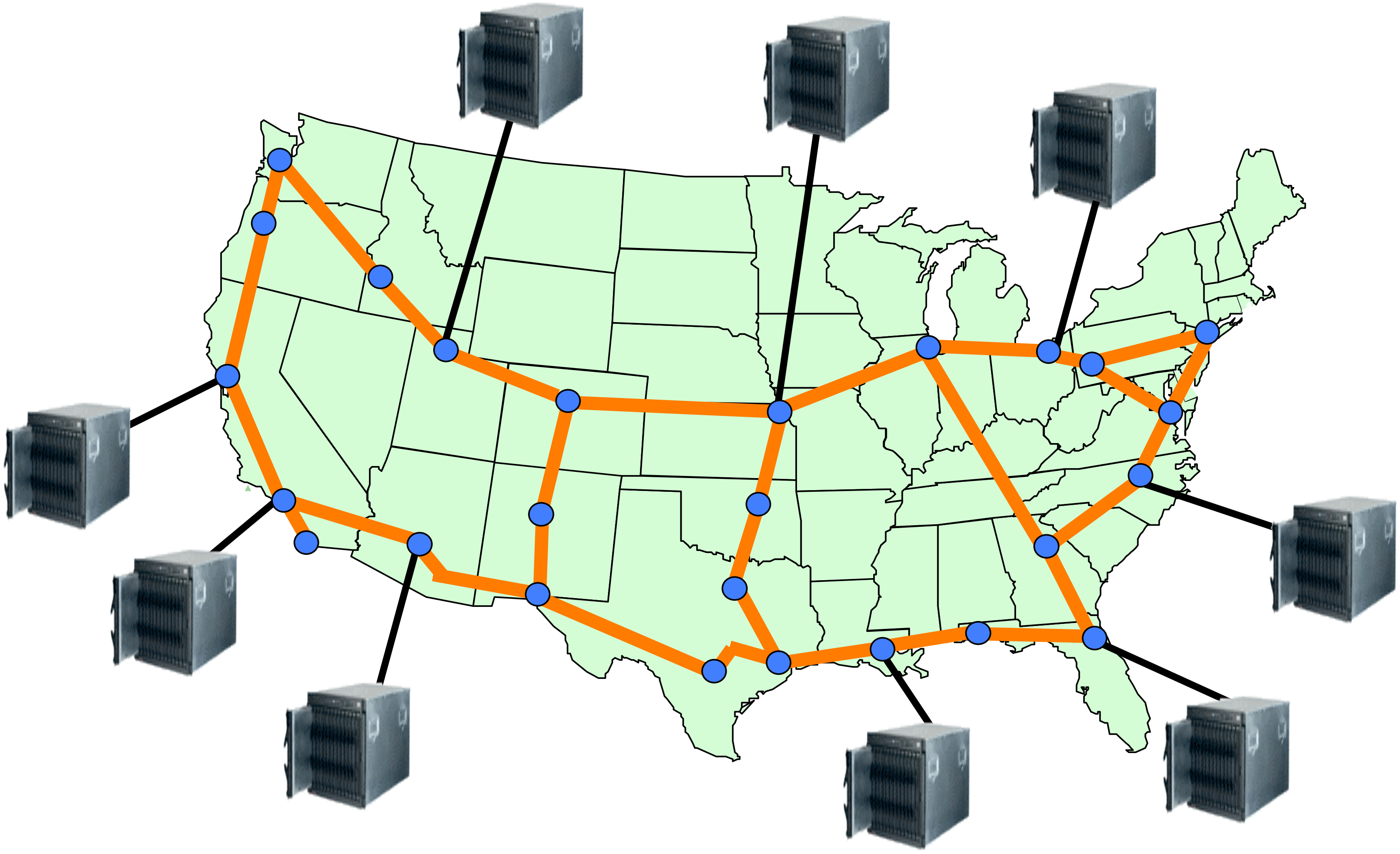
National Fiber Facility



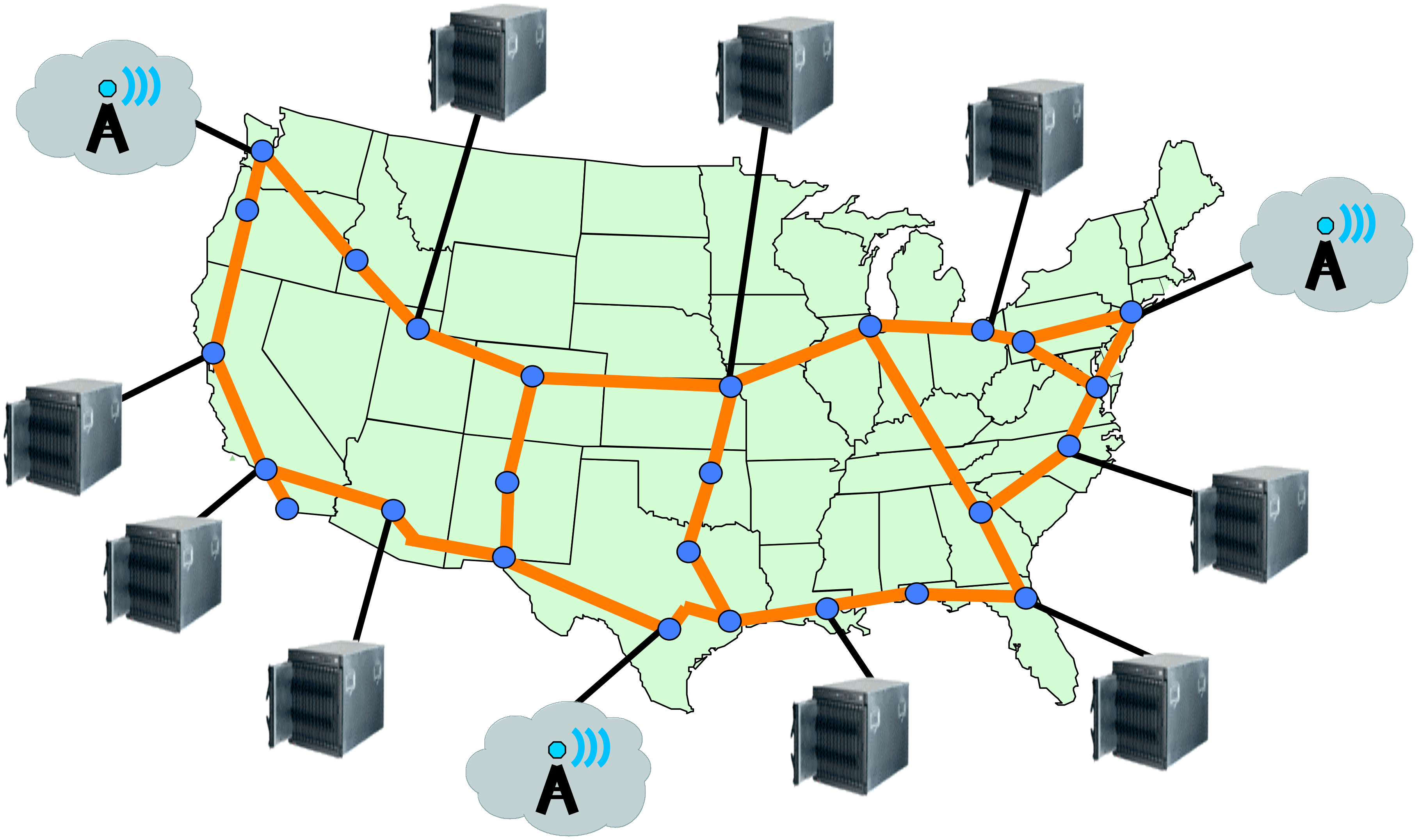
+ Programmable Switch/Routers



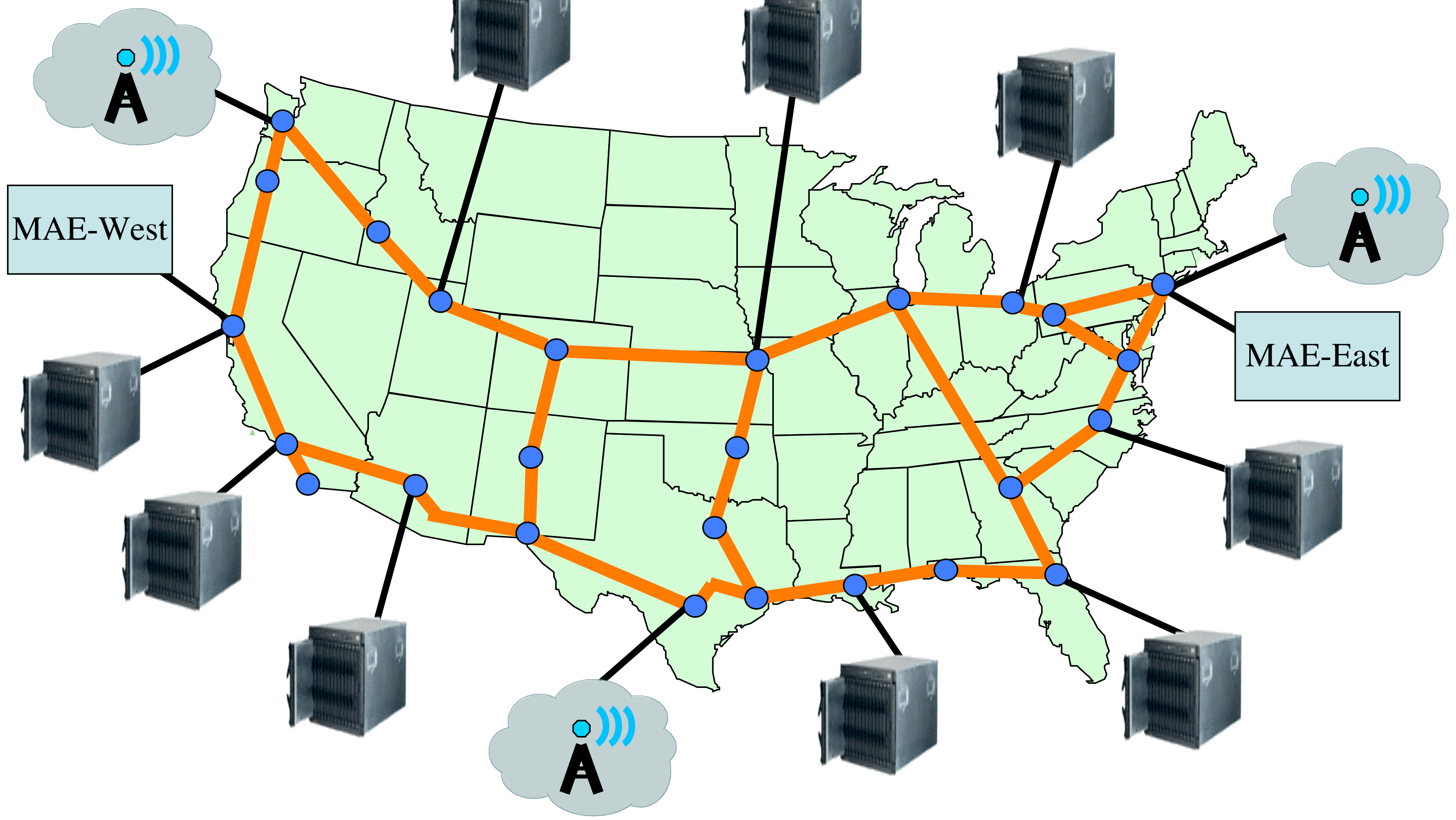
+ Clusters at Edge Sites

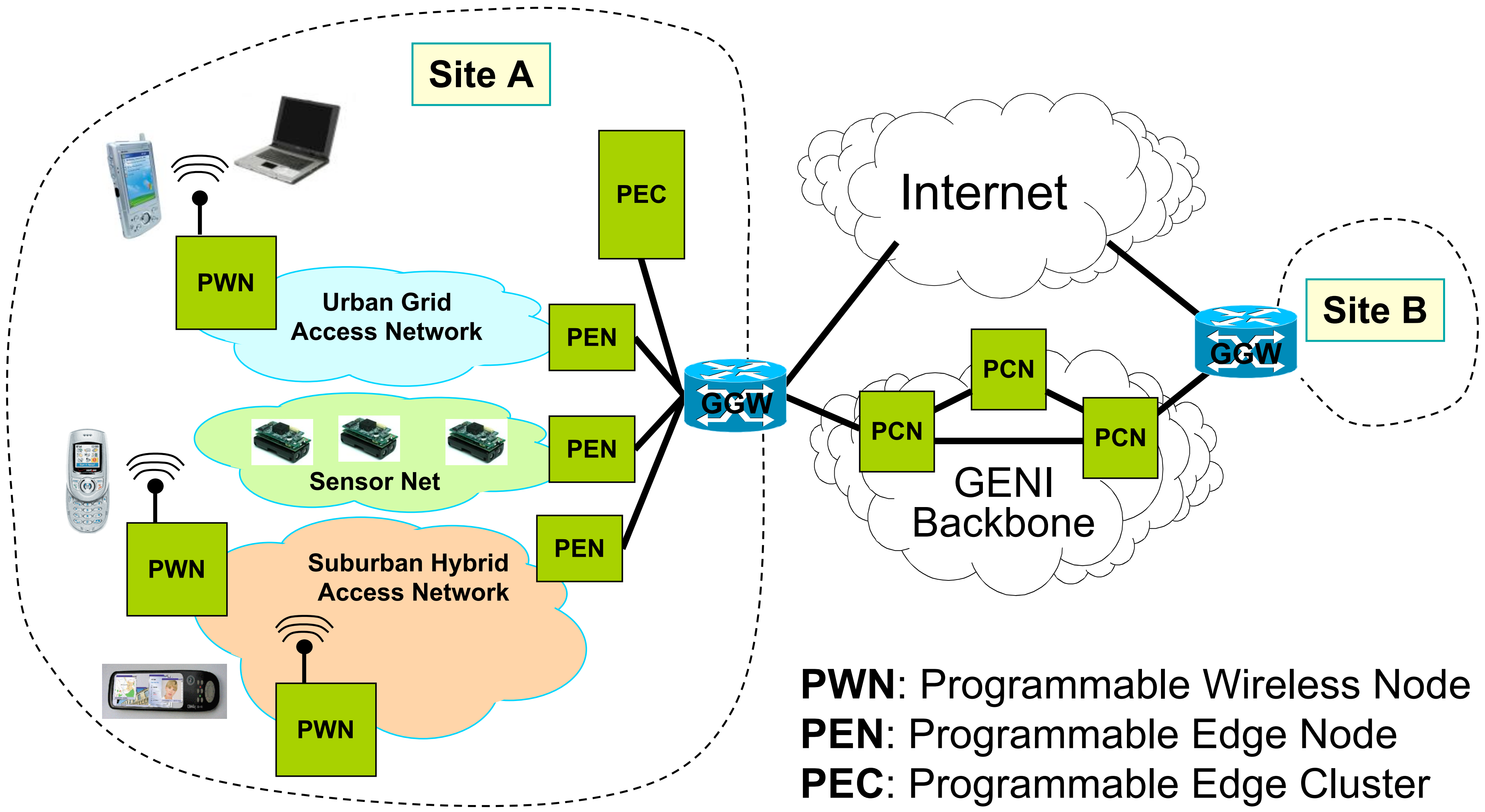


+ Wireless Subnets



+ ISP Peers





PWN: Programmable Wireless Node
PEN: Programmable Edge Node
PEC: Programmable Edge Cluster
PCN: Programmable Core Node
GGW: GENI Gateway

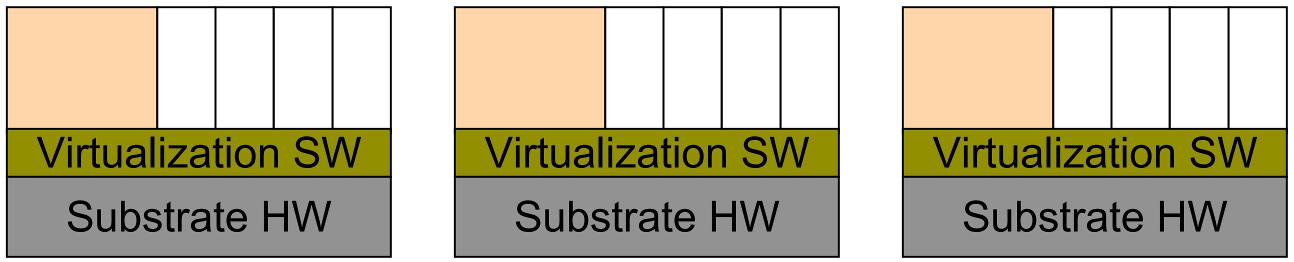
Overall system architecture

- Previous slides described a strawman physical substrate
- Next slides describe GENI’s system architecture - the software abstractions, objects, and functions that make this physical substrate available to GENI’s researchers as experiments.

Substrate Hardware

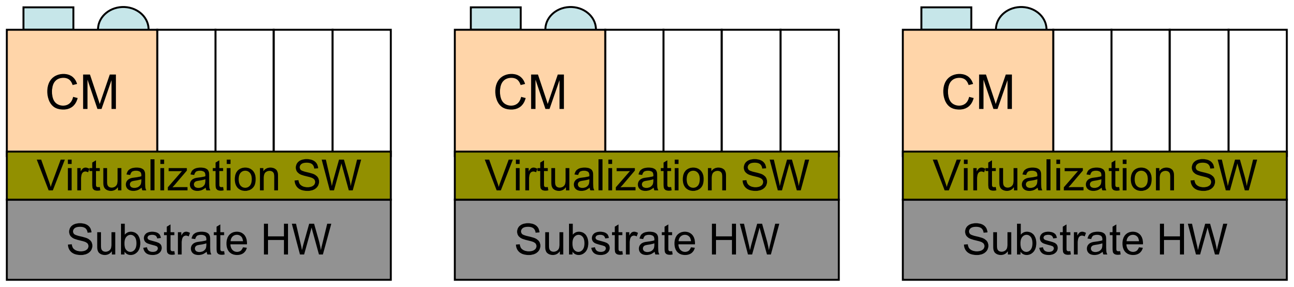
Substrate HW
Substrate HW
Substrate HW

Slicing Model and Software (often, “virtualization”)

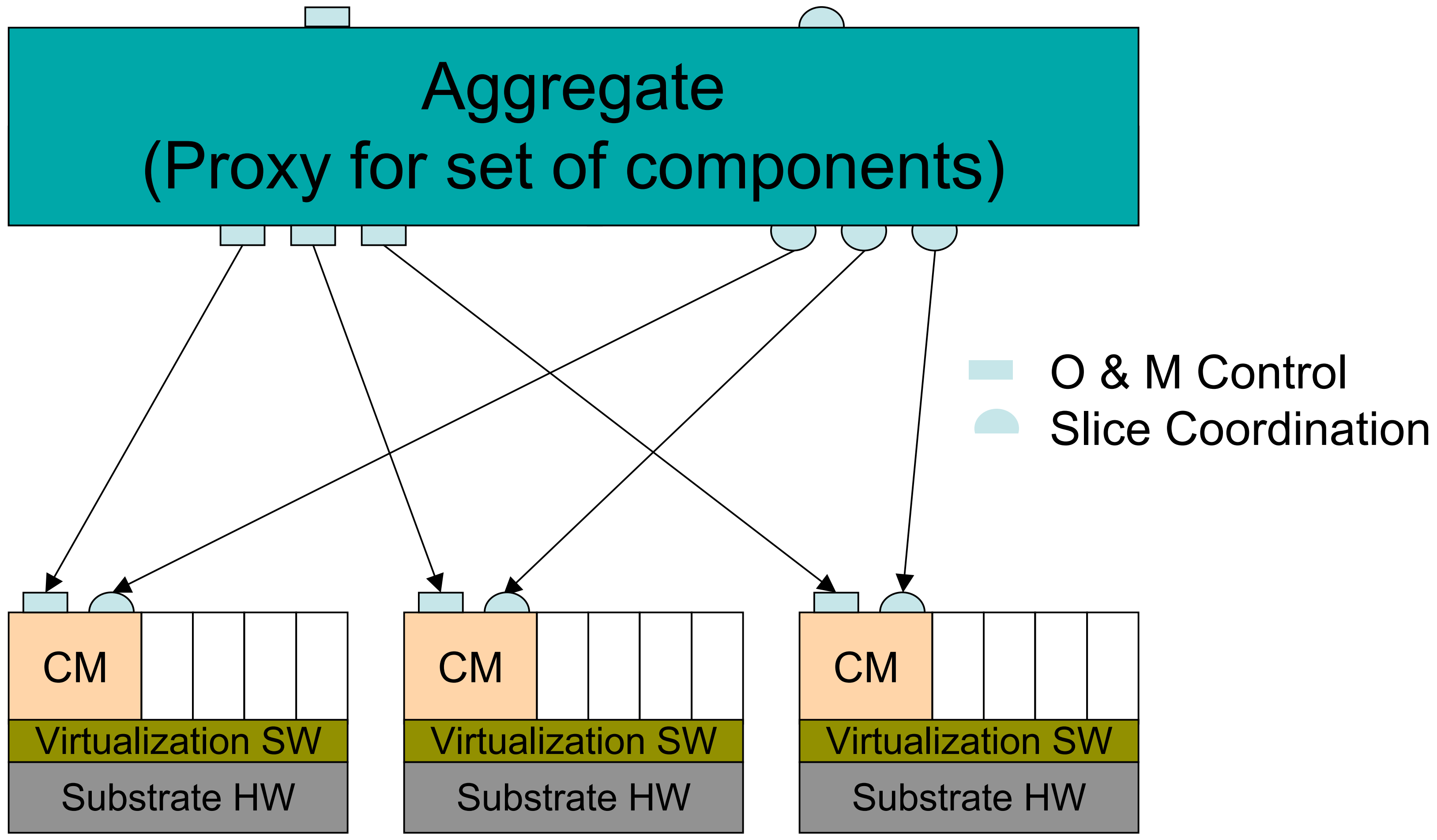


Components

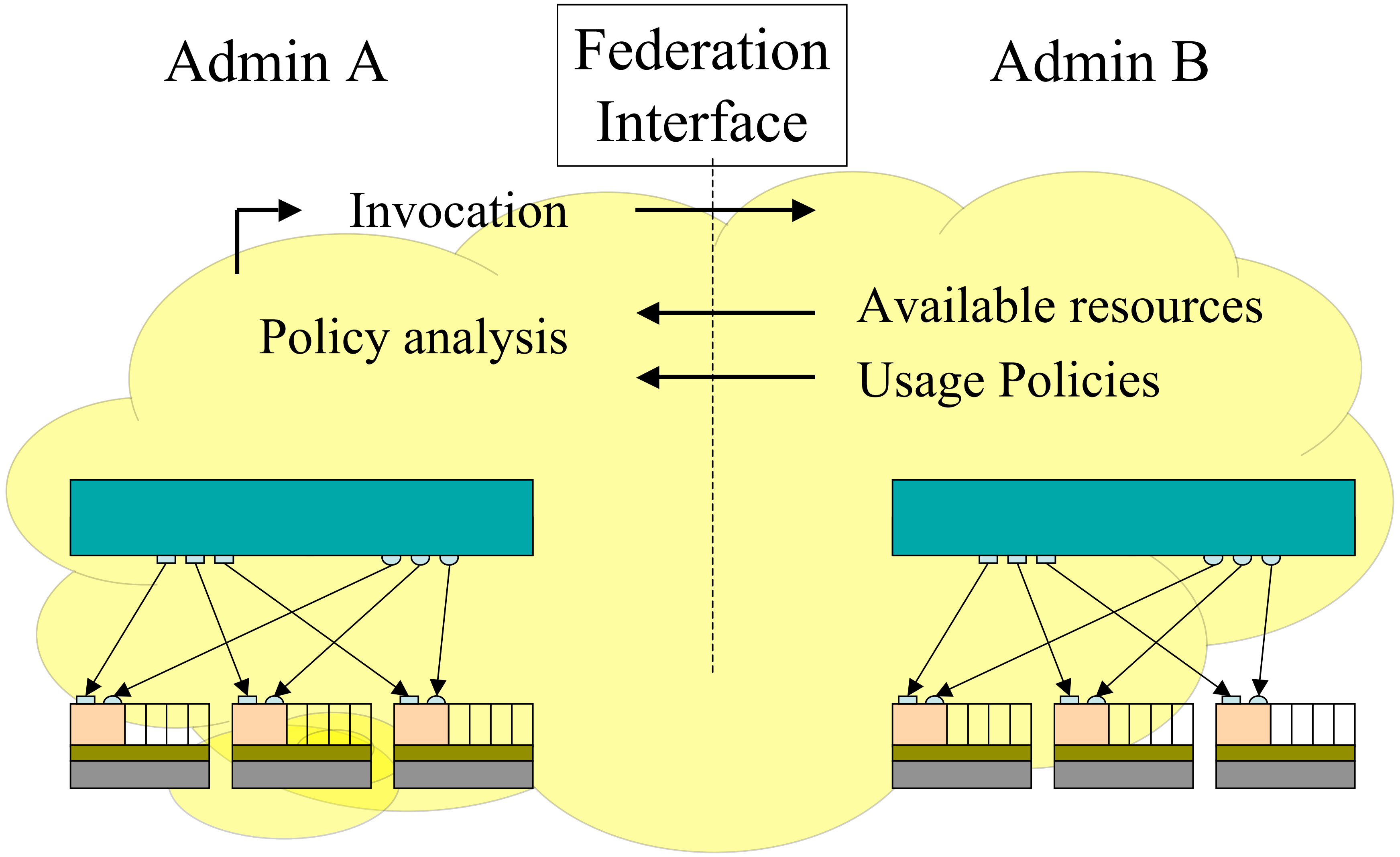
- Export a standard *component manager* interface
  - Resource allocation (to slices) and control
  - Minimal management



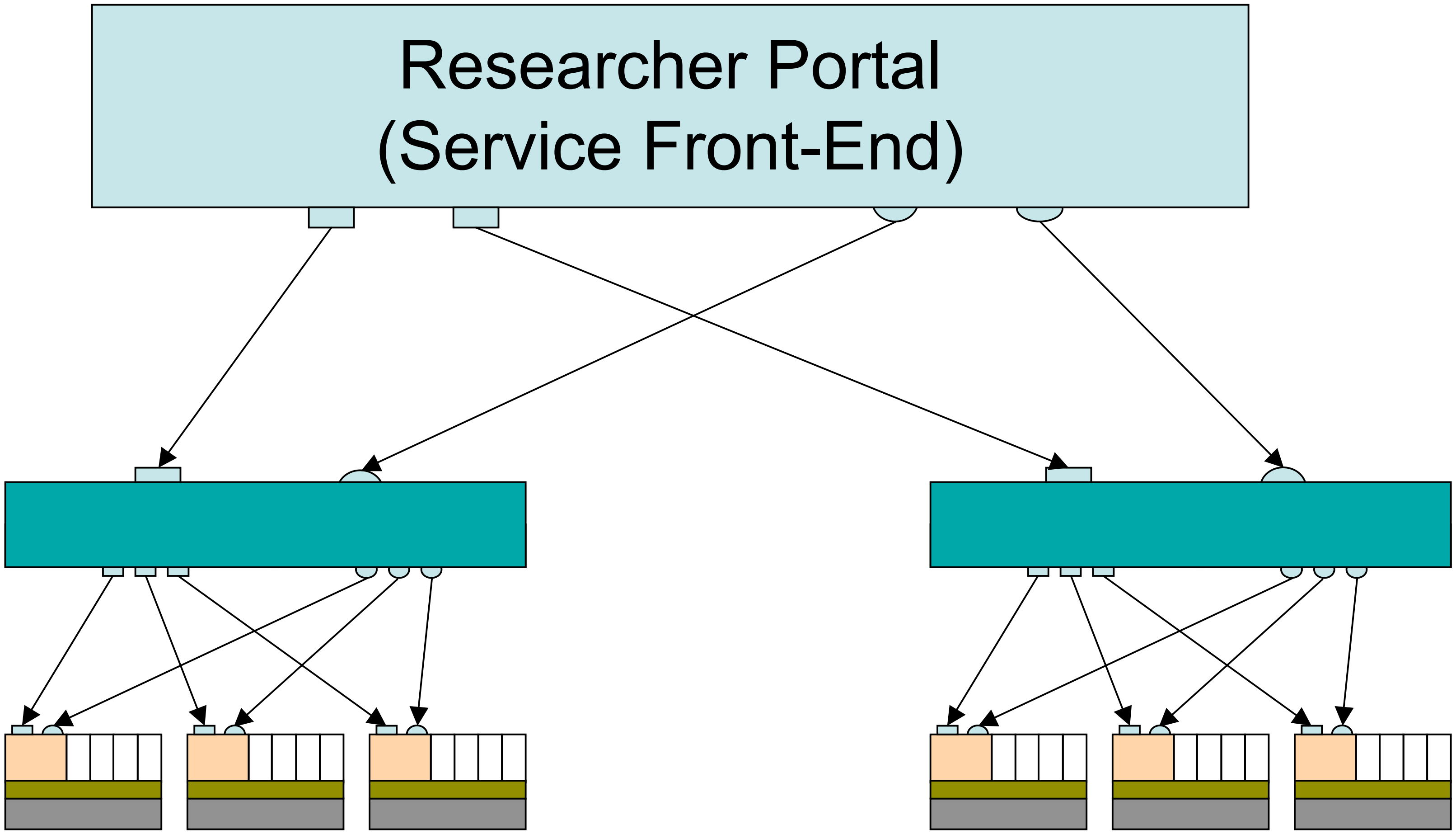
Aggregates

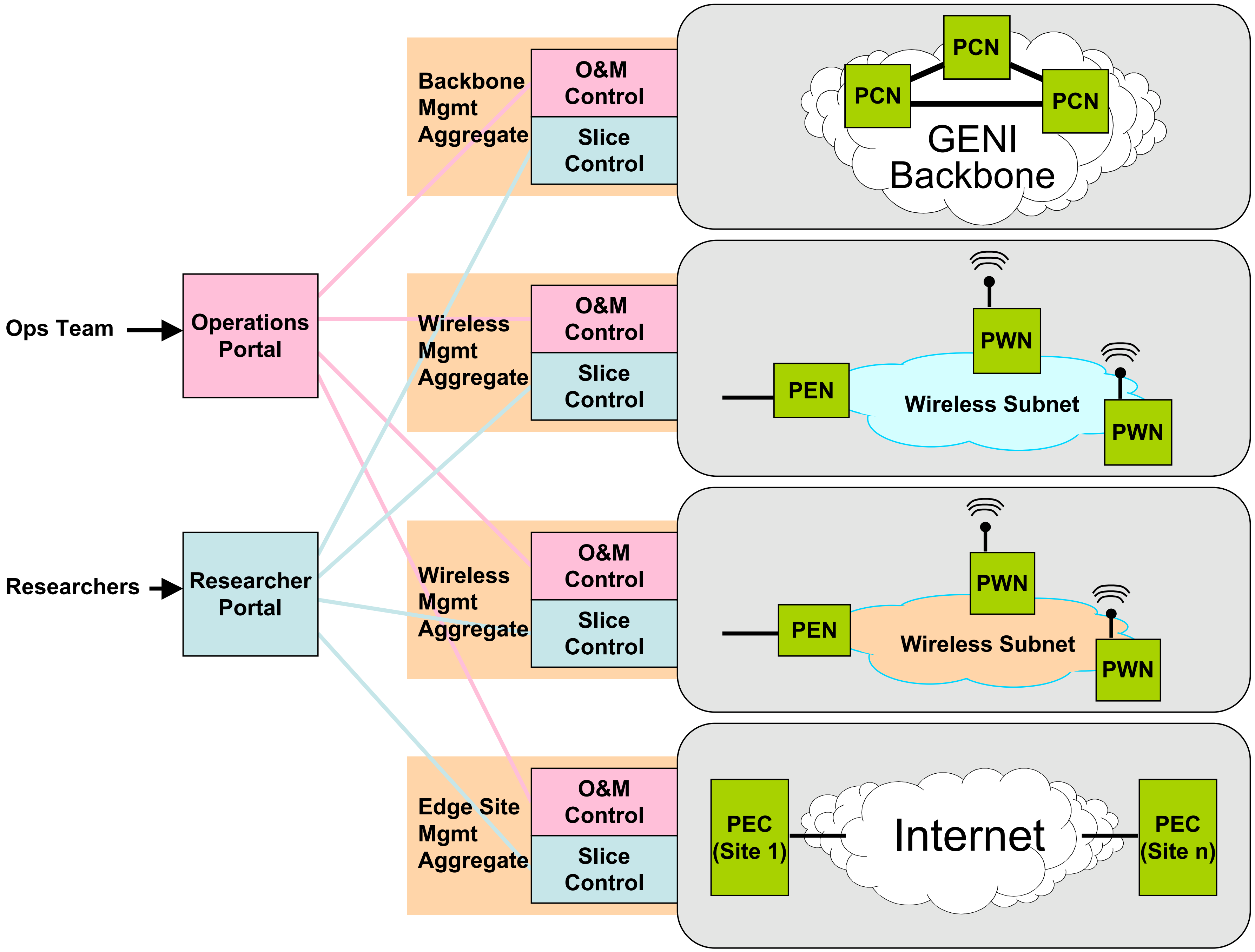

Aggregate
(Proxy for set of components)
O & M Control
Slice Coordination
CM
Virtualization SW
Substrate HW
CM
Virtualization SW
Substrate HW
CM
Virtualization SW
Substrate HW

Federation


Admin A
Federation Interface
Admin B
Invocation
Policy analysis
Available resources
Usage Policies

User Portals

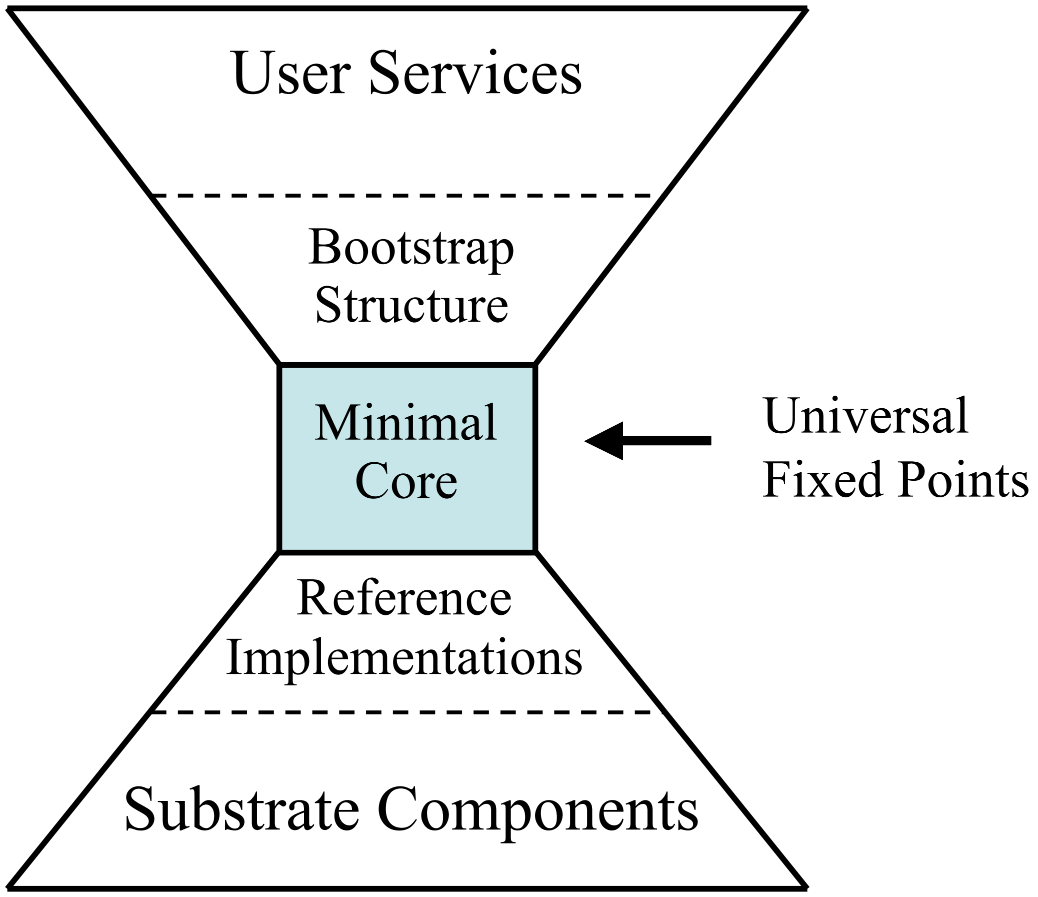



Ops Team
Operations Portal
Researchers
Researcher Portal
Backbone Mgmt Aggregate
O&M Control
Slice Control
PCN
PCN
PCN
GENI Backbone
Wireless Mgmt Aggregate
O&M Control
Slice Control
PWN
PEN
Wireless Subnet
PWN
Wireless Mgmt Aggregate
O&M Control
Slice Control
PWN
PEN
Wireless Subnet
PWN
Edge Site Mgmt Aggregate
O&M Control
Slice Control
PEC (Site 1)
Internet
PEC (Site n)

The narrow waist

- Previous slides have described the physical substrate, and some aspects of the overall software architecture that supports it
- Next slides describe the *narrow waist* - the core abstractions that serve as “fixed points” for the GENI architecture
  - GENI’s fixed “narrow waist” allows other system elements to evolve flexibly and independently
    - ➢ New components, services, partners, …
    - ➢ Particularly relevant during this early development / prototyping phase.

Hour-Glass Revisited



Minimal Core

- Principals
  - Slice Authorities (SA)
  - Component Management Authorities (MA)
  - User (researcher/experimenter, not “end user”)
- Objects
  - Slices - containers for experiments
    - Registered, Embedded, Accessed
  - Components - providers of resources
- Data Types
  - Global Identifiers (GID)
  - RSpec: resource specification
  - Tickets (credentials issued by component MA)
  - Slice Credentials (express live-ness, issued by SA)

Core (cont)

- Default Name Registries
  - Slice Registry (e.g., geni.us.princeton.codeen)
  - Component Registry (e.g., geni.us.backbone.nyc)
- Component Interface
  - Get/Split/Redeem Tickets
  - Create and Control Slices (“Slivers”)
  - Query Status

Core (cont)

- Management Interfaces (Minimal common elements)
  - Return to known state
  - Start/stop
  - Become more intelligent (boot load)
- Federation Interfaces
  - Cross-domain accountability
  - Policy expression and management

Maturity levels

- General statement
  - The strawman design builds on significant community experience.
    - PlanetLab, Emulab, DETER, others.
  - The strawman design is *work in progress*.
    - It is not implemented.
    - Some parts are incomplete.
    - Some are in great flux.
  - … but the initial version is *incremental and "simple"*
  - Component and service prototypers will work *in parallel* with the narrow waist / core development.

Relatively Mature

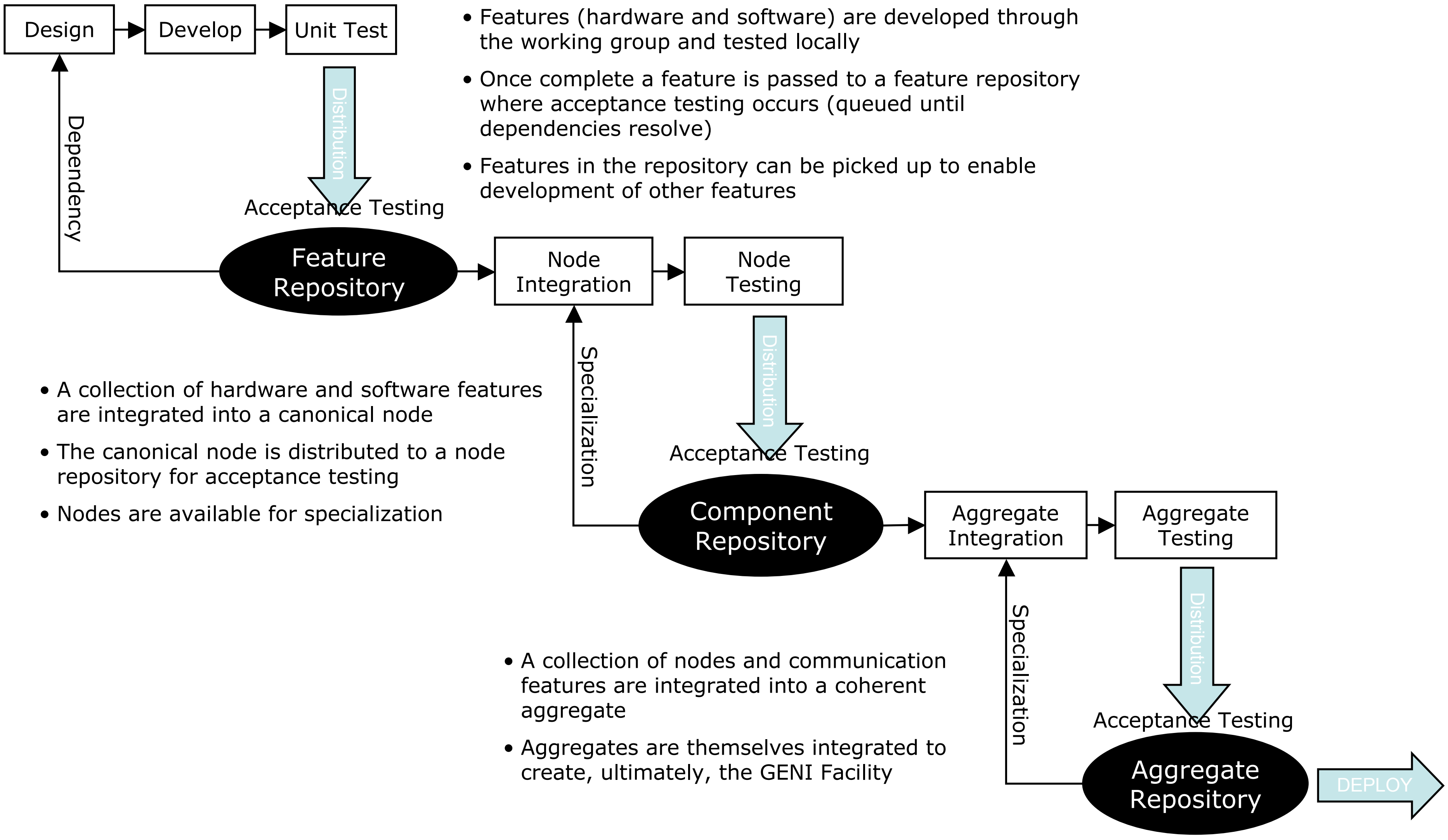
- Core system objects - users, components, ..
- Concept & function of components
- Concept & function of universal identifiers
- Concept of resource specifications
- Simple model for slice and component registries
- …

Less Mature

- Configuration management
  - How are components at a site related?
- Federation model and interfaces
  - How do we build experiments across different administrative regions?
- Operations and management interfaces
  - Ensure sufficient reliability, accessibility
  - Track usage to plan for future neede
- …

Construction issues

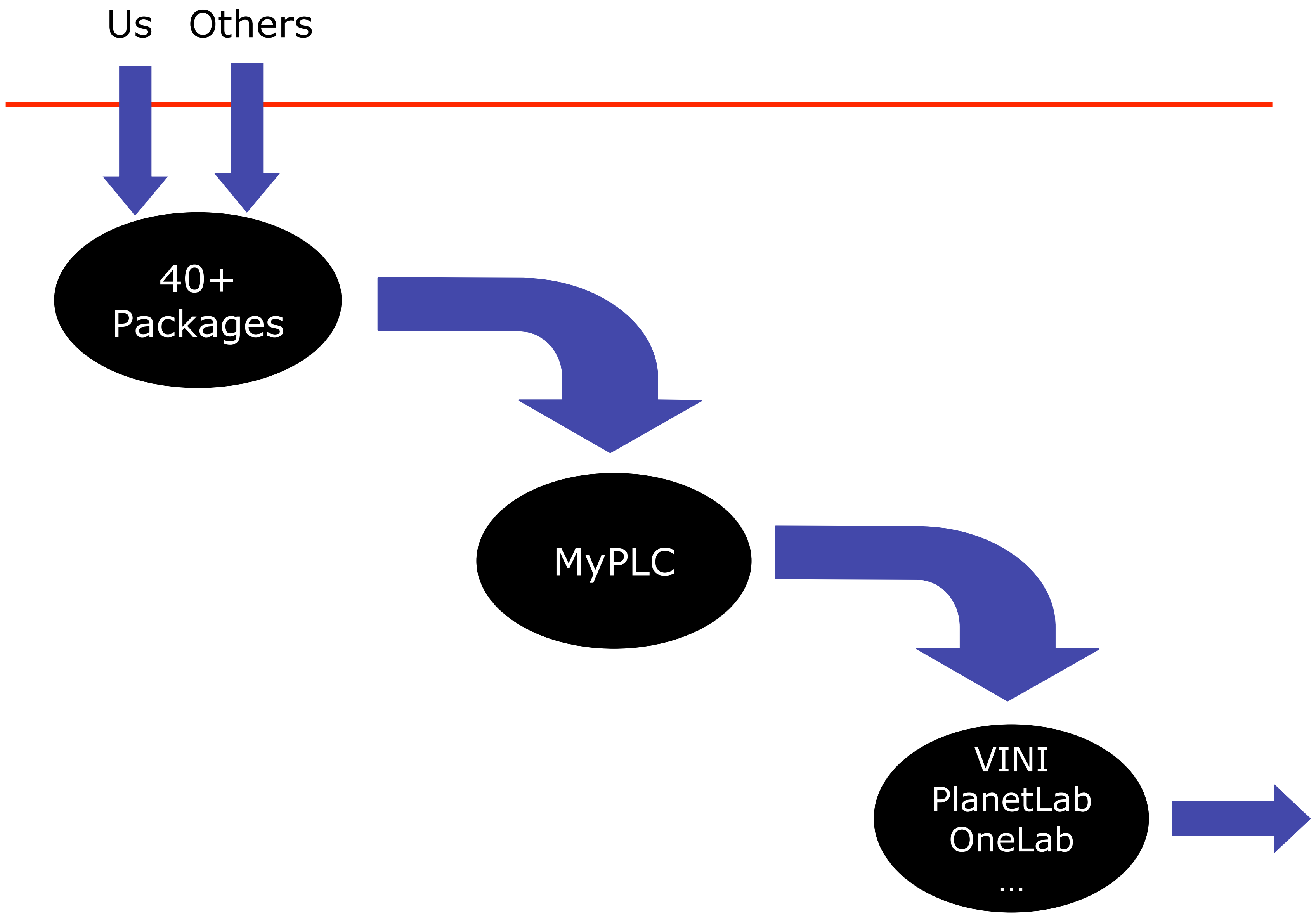
- Objectives
  - Allow broader community to contribute (not just subs)
  - Scale (federate) the integration effort
- Strategy
  - Feature Development
    - Roughly equivalent to open source development process
  - Component/Aggregate Integration
    - Roughly equivalent to preparing a Linux distribution



- Features (hardware and software) are developed through the working group and tested locally
- Once complete a feature is passed to a feature repository where acceptance testing occurs (queued until dependencies resolve)
- Features in the repository can be picked up to enable development of other features

- A collection of hardware and software features are integrated into a canonical node
- The canonical node is distributed to a node repository for acceptance testing
- Nodes are available for specialization

- A collection of nodes and communication features are integrated into a coherent aggregate
- Aggregates are themselves integrated to create, ultimately, the GENI Facility


Us
Others
40+ Packages
MyPLC
VINI
PlanetLab
OneLab
...

[redacted]