isis^2

GridCloud:
A Platform for Controlling the Smart Grid from the Cloud

Cornell University (Birman, van Renesse) and WSU (Bakken, Bose, Hauser)

Cloud Computing

- The "next new thing"
  - Big data centers (probably hosted by power industry vendors, not Amazon.com or Google)
  - These permit "consolidation"
    - 10x or better reductions in cost of operation
    - Far better equipment utilization and management
    - New styles of elastic computing, potential to compute directly on *massive* data collections
    - Adds up to a new way of computing that forces us to undertake new kinds of thinking

Our vision?

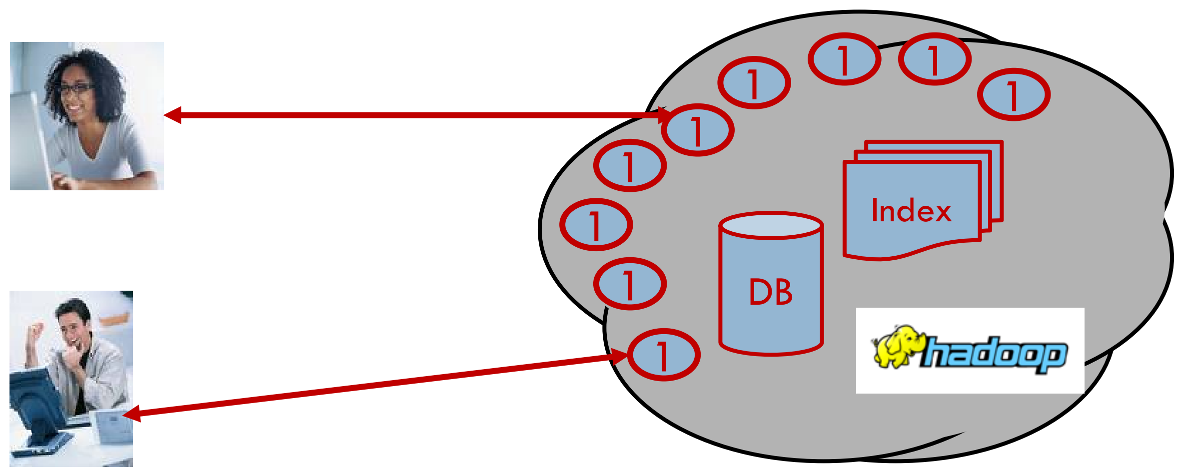
- Cloud technology in support of smart power grid
  - Use best-of-breed high-assurance distributed systems technologies to create a platform for hosting high-assurance cloud computing apps
  - Integrate with a technology base designed for monitoring and controlling smart-grid applications
  - Achieve a new kind of "operating system" for cloud-hosted embedded control of power grid
- Key: Building on powerful existing technologies

GridCloud architecture

- Our vision unites two existing technology bases
  - Cornell's new $Isis^2$ platform, for high assurance cloud computing.
  - WSU's GridStat technology, for supporting smart grid monitoring and control
- To demonstrate the capability we'll create demo programs using
  - WSU's GridSim tool, to emulate PMU data sources
  - NSF's GENI platform, to host these emulated PMU's
  - A kind of map-annotation application intended to play the role of SCADA and optimization technologies

Today’s commercial cloud architecture

- Structured in “tiers”, each with its own required programming style
- First-tier and associated services keep “soft state” and must be highly elastic
- Inner tiers host “services”
- Back-end can support large scale batch apps.



First tier: scalable, elastic services that interact with client and must respond instantly, often using cached data

Adapting this for GridCloud

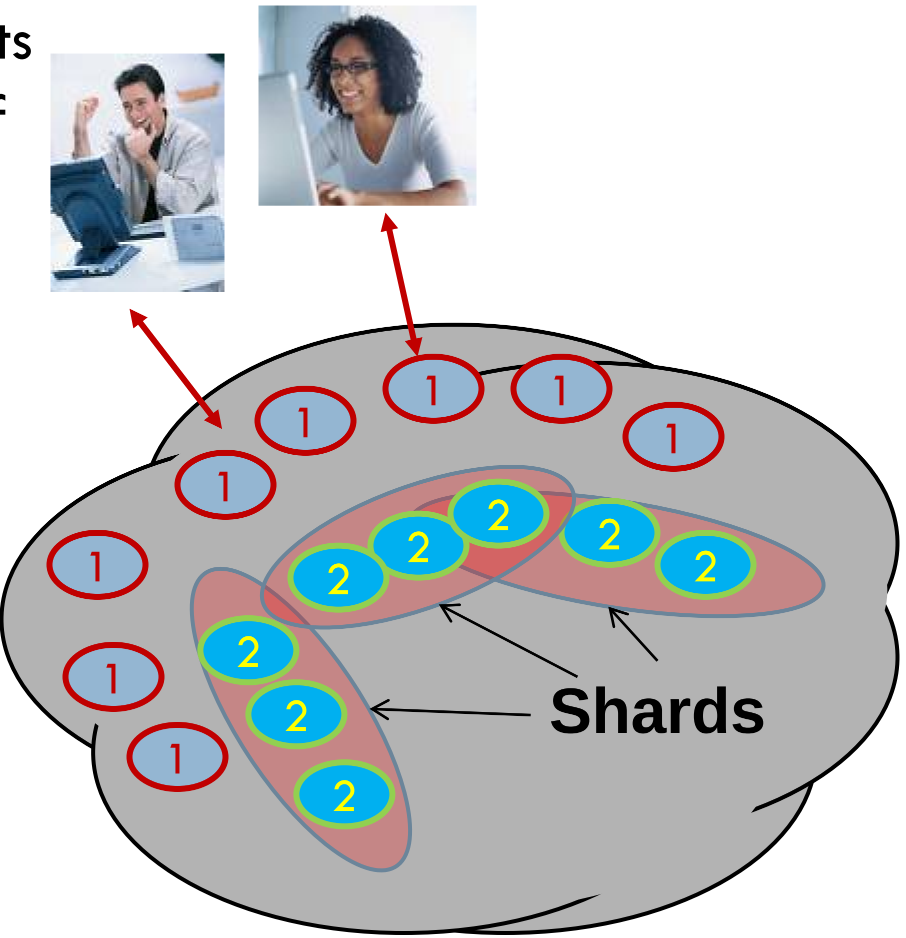
- A first challenge is that in today's first-tier, systems are "inconsistent" by design
  - We need to guarantee fast response and yet can't abandon the key elasticity properties that permit scalability.
- A second challenge is that the amounts of data being gathered dwarf what one machine can manage
  - Forces data to be "sharded" and demands a new style of computing
  - Applications will have all the data (not just models), but on the other hand, the data isn't at one place

Sharding: A form of key-value replication (key maps to a shard)

- Cloud can collect huge amounts of data from huge numbers of sources

- But scalability centers on a scheme for fragmenting the data into shard

- Our new Isis^2 library makes it easy to create secure, high reliability shard based cloud software

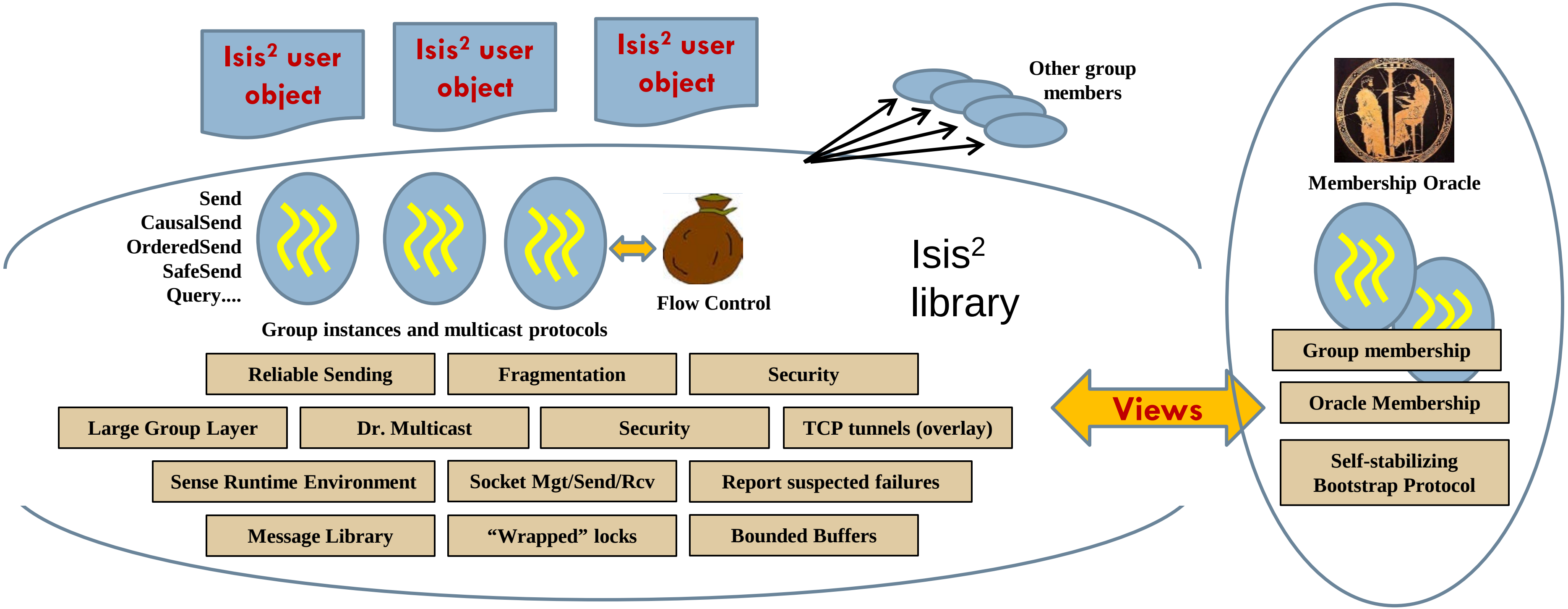


Our new Isis^2 library

- Named for an old Cornell story
  - In 1990 our first Isis Toolkit became the core of the NYSE, French Air Traffic Control System and US Navy AEGIS
- Isis^2 : A completely new system but same idea
  - Makes it easy to create high-assurance cloud apps
  - Offers consistency, fault-tolerance, security
  - About 8500 "semicolons" in C# (half of which are debugging self-checks)
- FreeBSD code release just completed!

It takes a community!

- Isis² was "incredibly hard" to build... it took years of effort by domain experts
- This is easily understood when we consider how many issues needed to be addressed



Coding style: C#, C++, Python

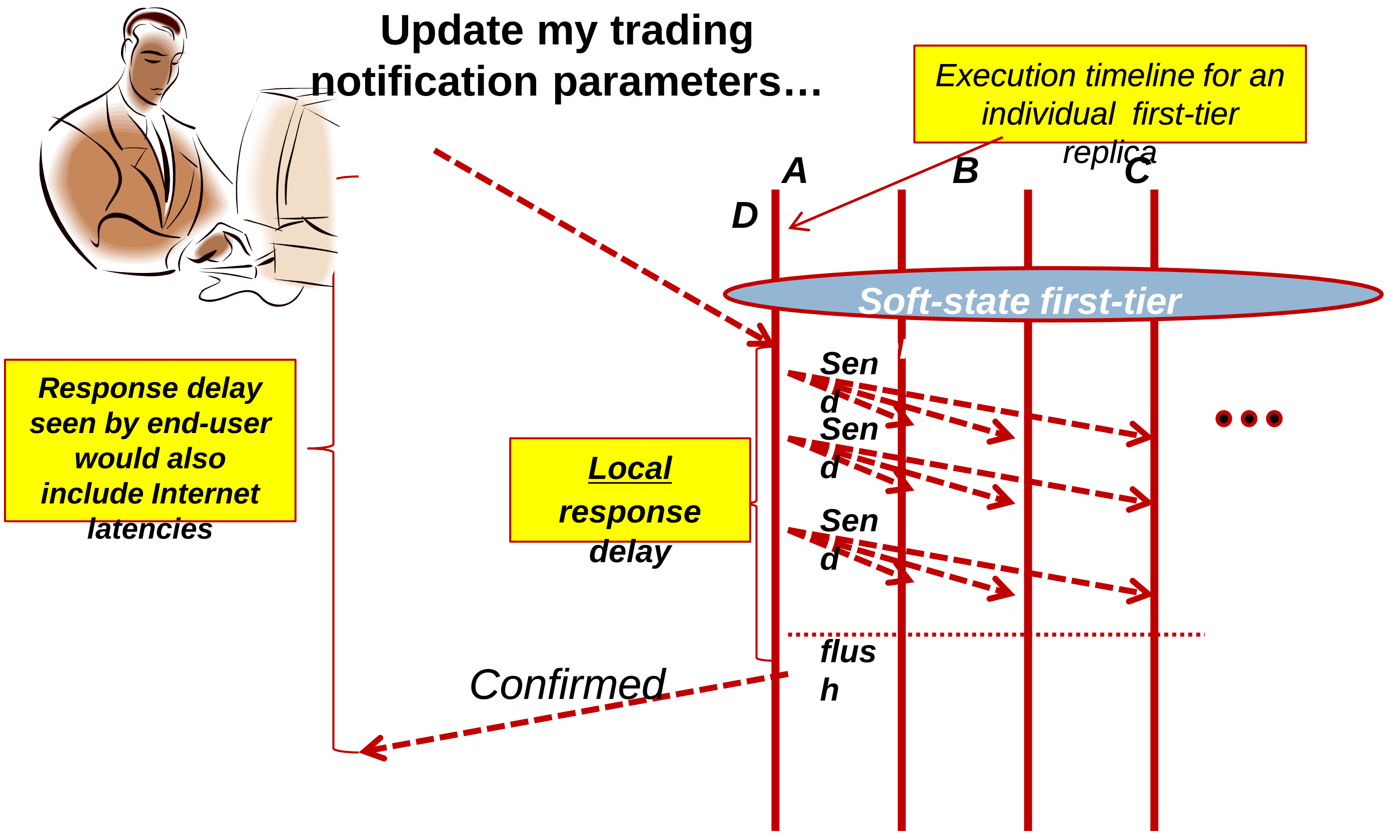
```
Group g = new Group("myGroup");
g.ViewHandlers += delegate(View v) {
    Console.Title = "myGroup members: "+v.members;
};
g.Handlers[UPDATE] += delegate(string s, double v) {
    Values[s] = v;
};
g.Handlers[LOOKUP] += delegate(string s) {
    Reply(Values[s]);
};
g.Join();

g.Send(UPDATE, "Harry", 20.75);

List<double> resultlist = new List<double>;
nr = g.Query(LOOKUP, ALL, "Harry", EOL, resultlist);
```

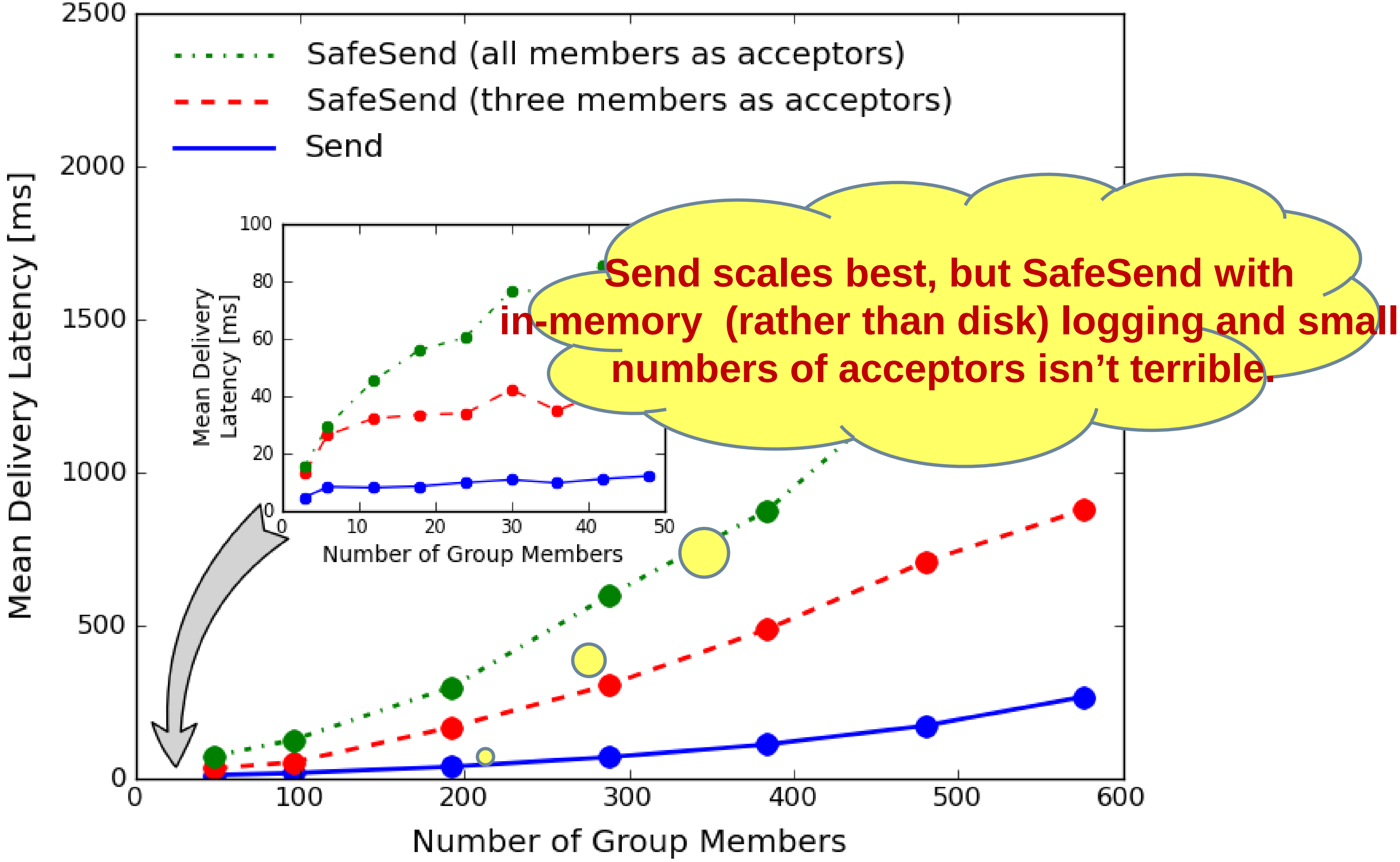
- First sets up group
- Join makes this entity a member. State transfer isn't shown
- Then can multicast, query. Runtime callbacks to the "delegates" as events arrive
- Easy to request security (g.SetSecure), persistence
- "Consistency" model dictates the ordering seen for event upcalls and the assumptions user can make

Experimental Scenario

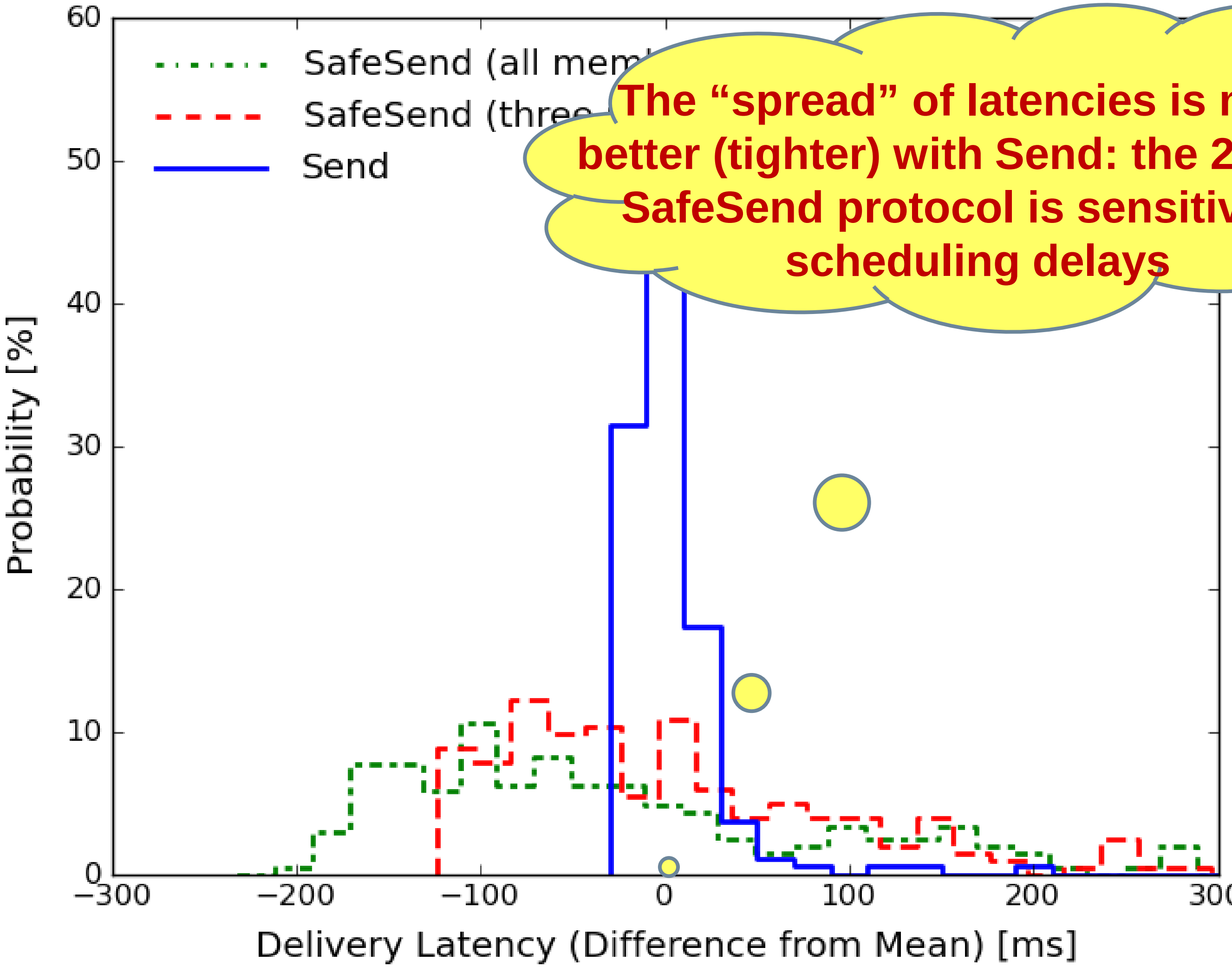


An online monitoring system must focus on real-time response delays

User trades guarantees for speed

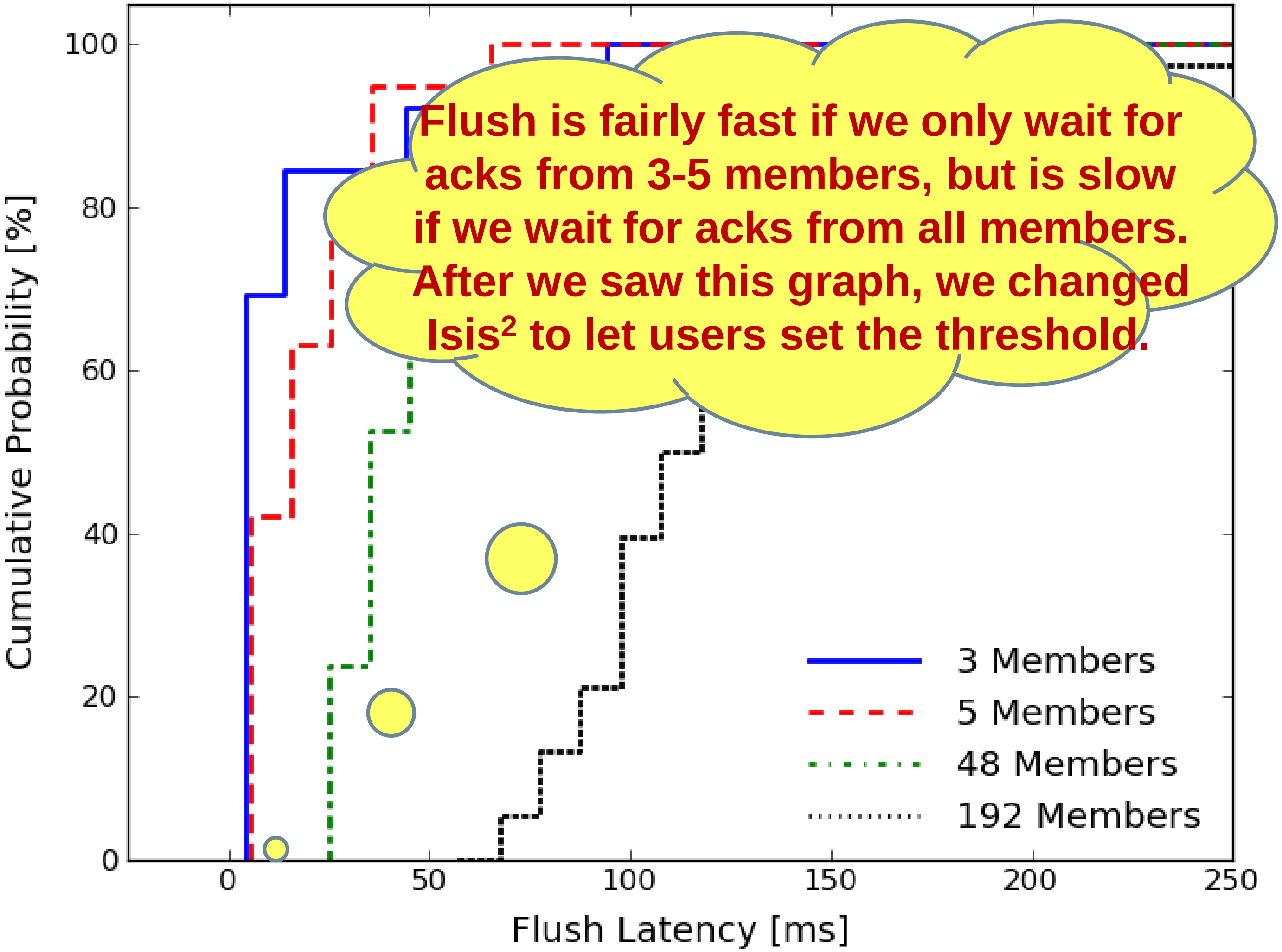


Tightness of timing guarantees



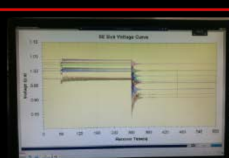
The "spread" of latencies is much better (tighter) with Send: the 2-phase SafeSend protocol is sensitive to scheduling delays

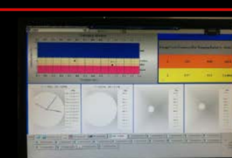
Response delay/level of replication



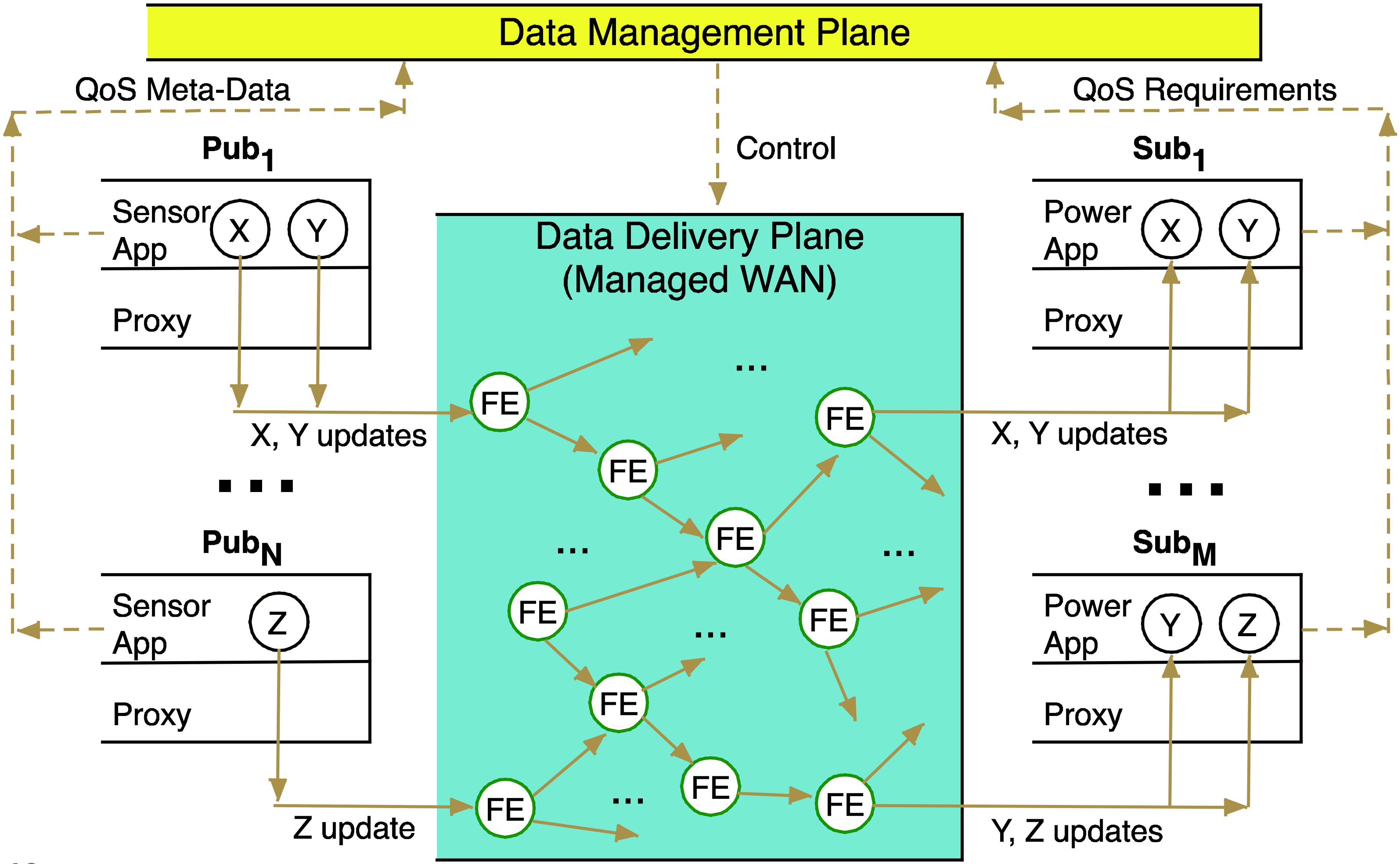
GridSim: A Virtual Smart Grid

- GridStat (data delivery)
- Hierarchical Linear State Estimator
- Other components
  - Power Tech's TSAT transient stability simulator equipped with virtual PMUs
  - Oscillation monitor (OM) , an openPDC application
  - Glue components: static data, measurement and C37.118 generators
- Supported by DOE (thru 8/12)





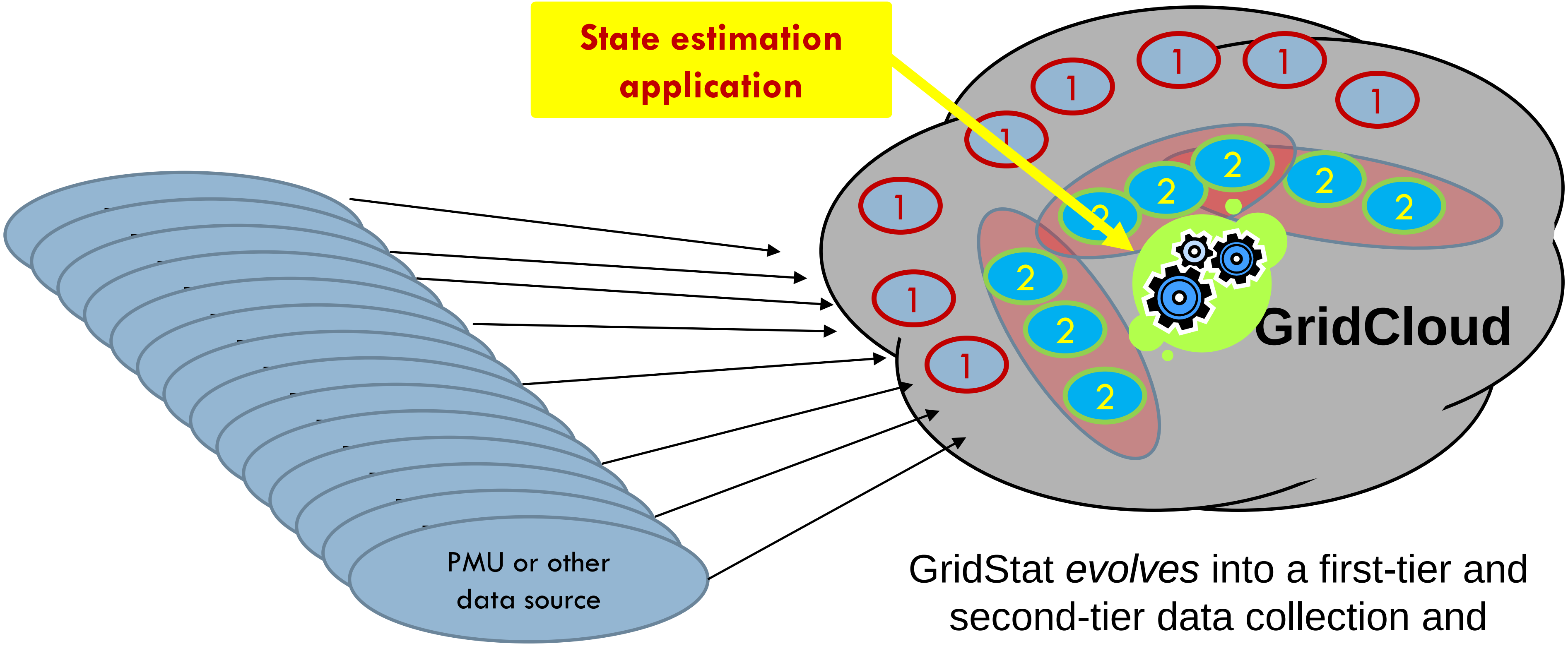
GridStat Architecture



How will GridCloud control the grid?

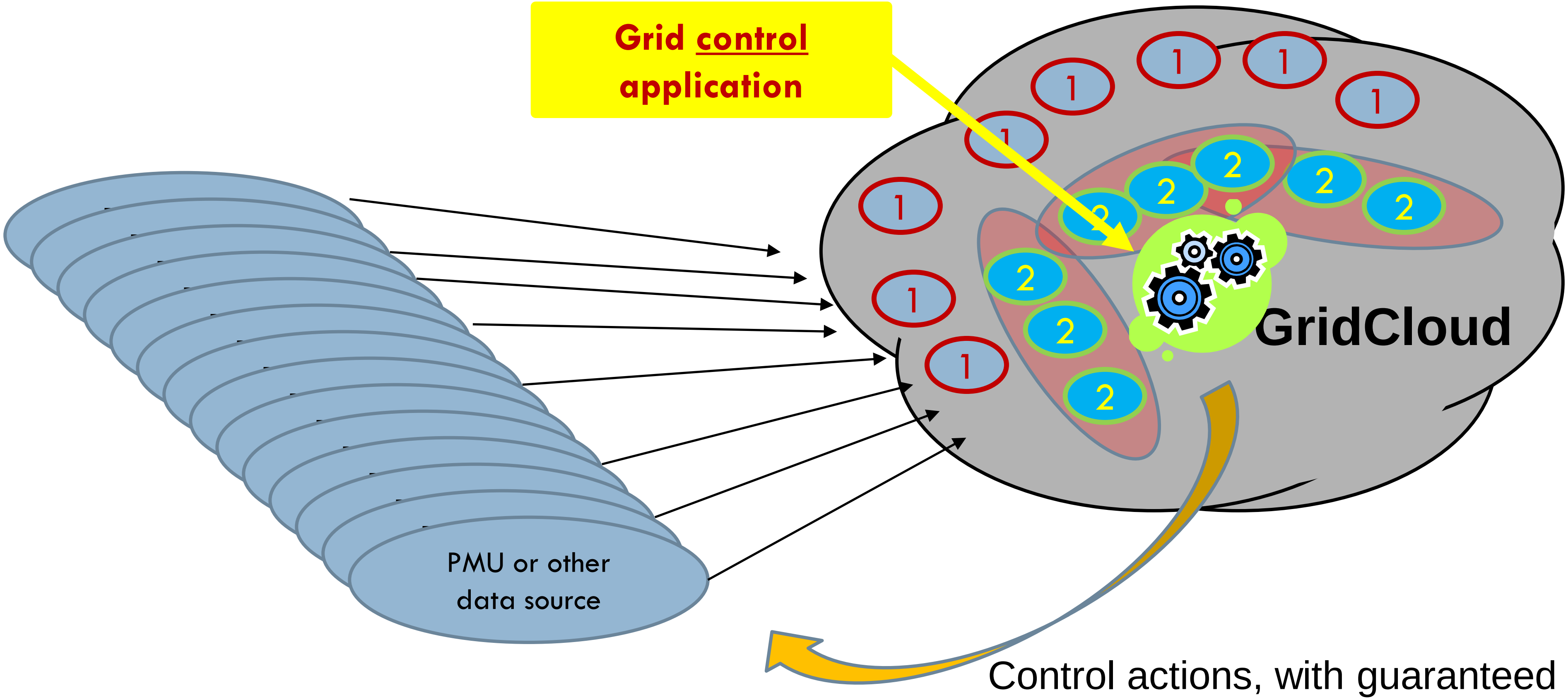
- You get to tell us!
  - GridCloud is like a new kind of operating system
  - A platform on which you can implement and run customized control solutions
  - “Plug in” your own state estimator and control code and we’ll do the rest

Bringing it together



GridStat *evolves* into a first-tier and second-tier data collection and management infrastructure that can host control applications

Eventually, close the loop



Control actions, with guaranteed real-time response

Summary?

- The word on the street is that cloud computing will rule but that the cloud can't support high assurance control apps

- The Cornell/WSU team just doesn't accept that limitation
  - We'll build GridCloud to prove our point
  - But we'll need to replace our "demo" applications with real ones to actually deploy GridControl in real smart-grid scenarios, and for that, we need you!