December 13, 2012

Eric Boyd, Internet2

Bandwidth on Demand in the Wild Circuit Service Deployments Status and Examples

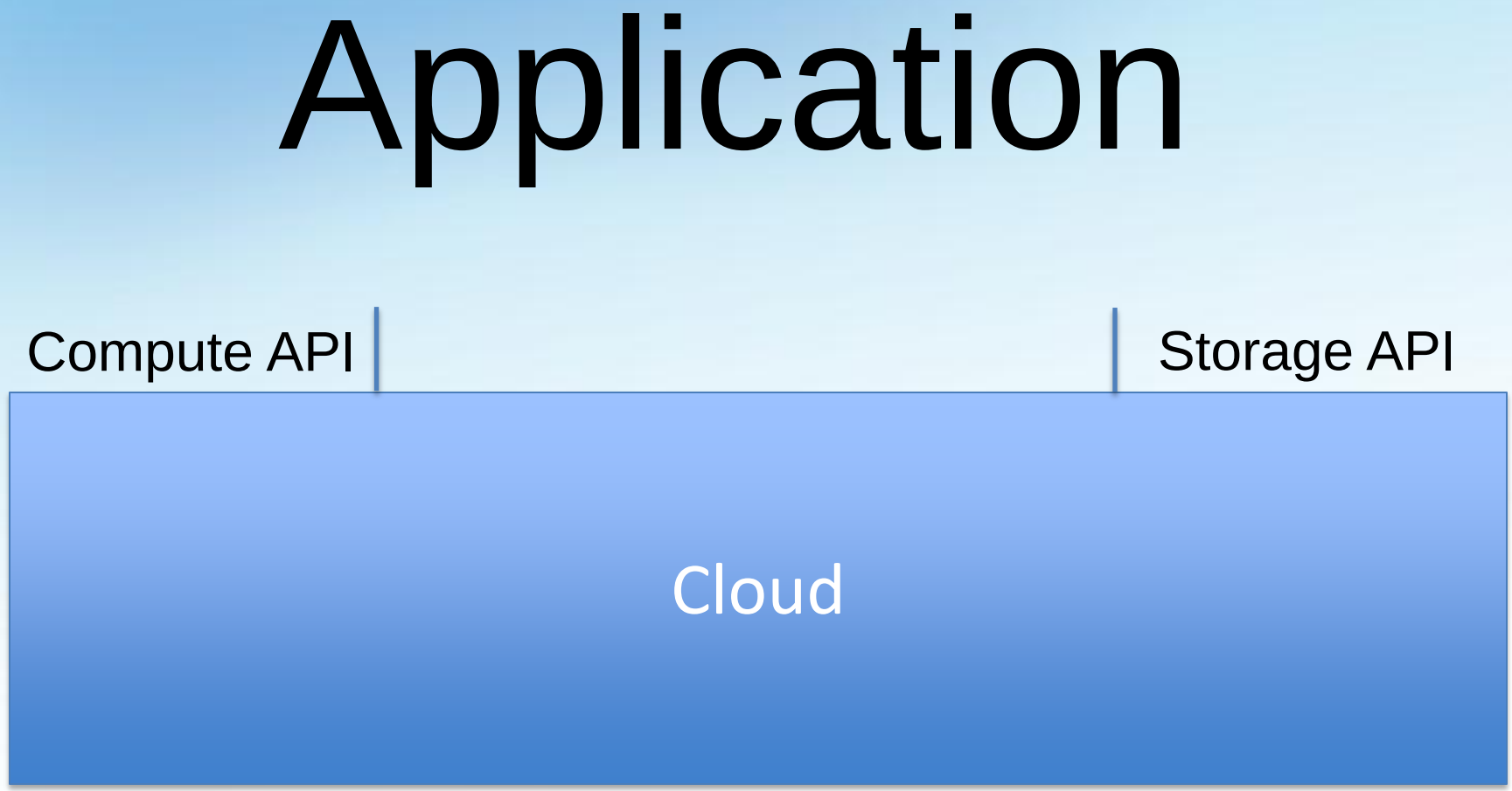
Disclaimer

- Thanks go to others for input
- My examples are not exhaustive
- Errors and oversights are my fault and to be corrected in posted version

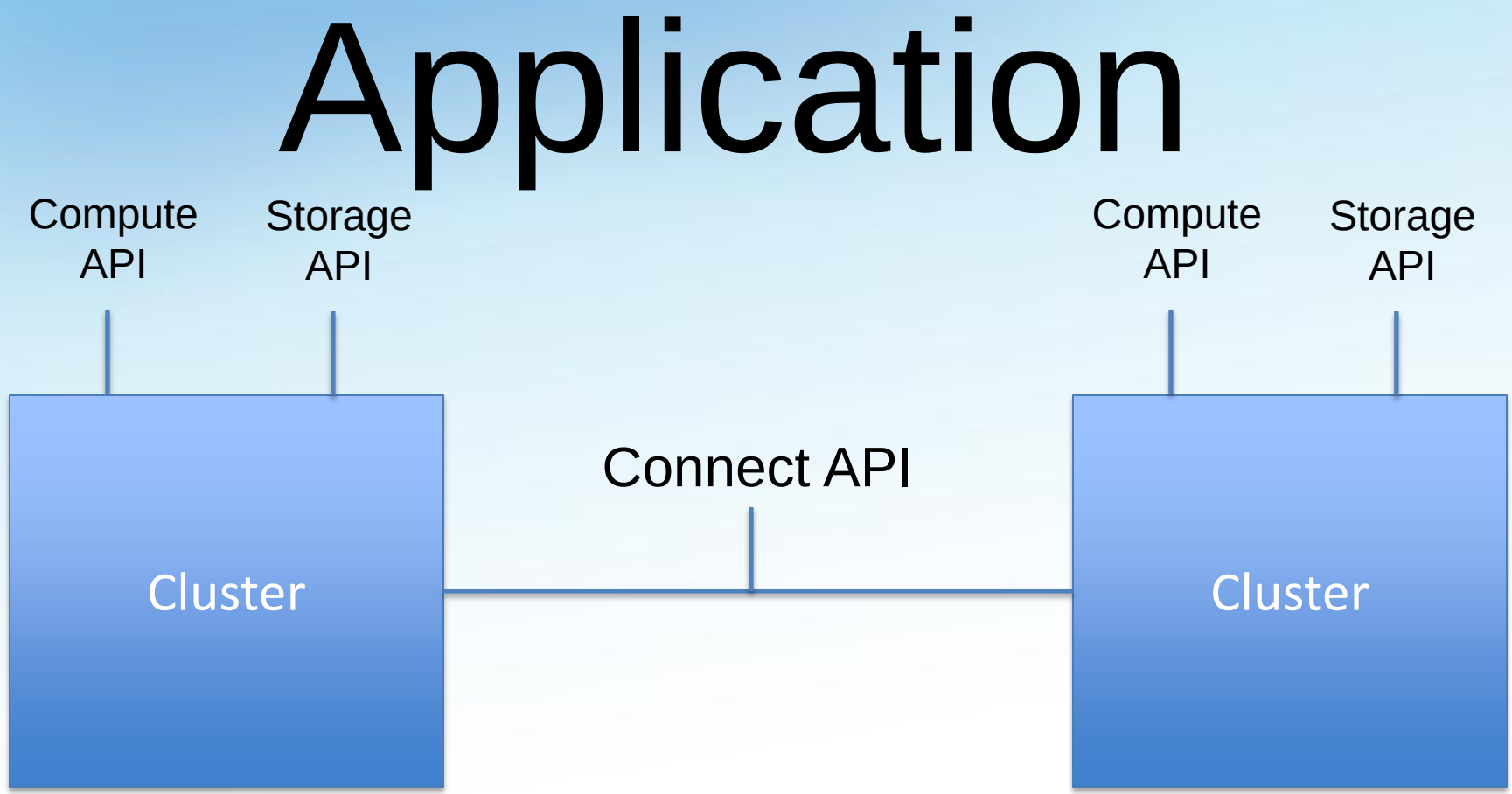
Overview

- View from the Application
- Connect API
- Reality is Messy: Opening up the Black Box
- The World Today
- When It All Goes Wrong

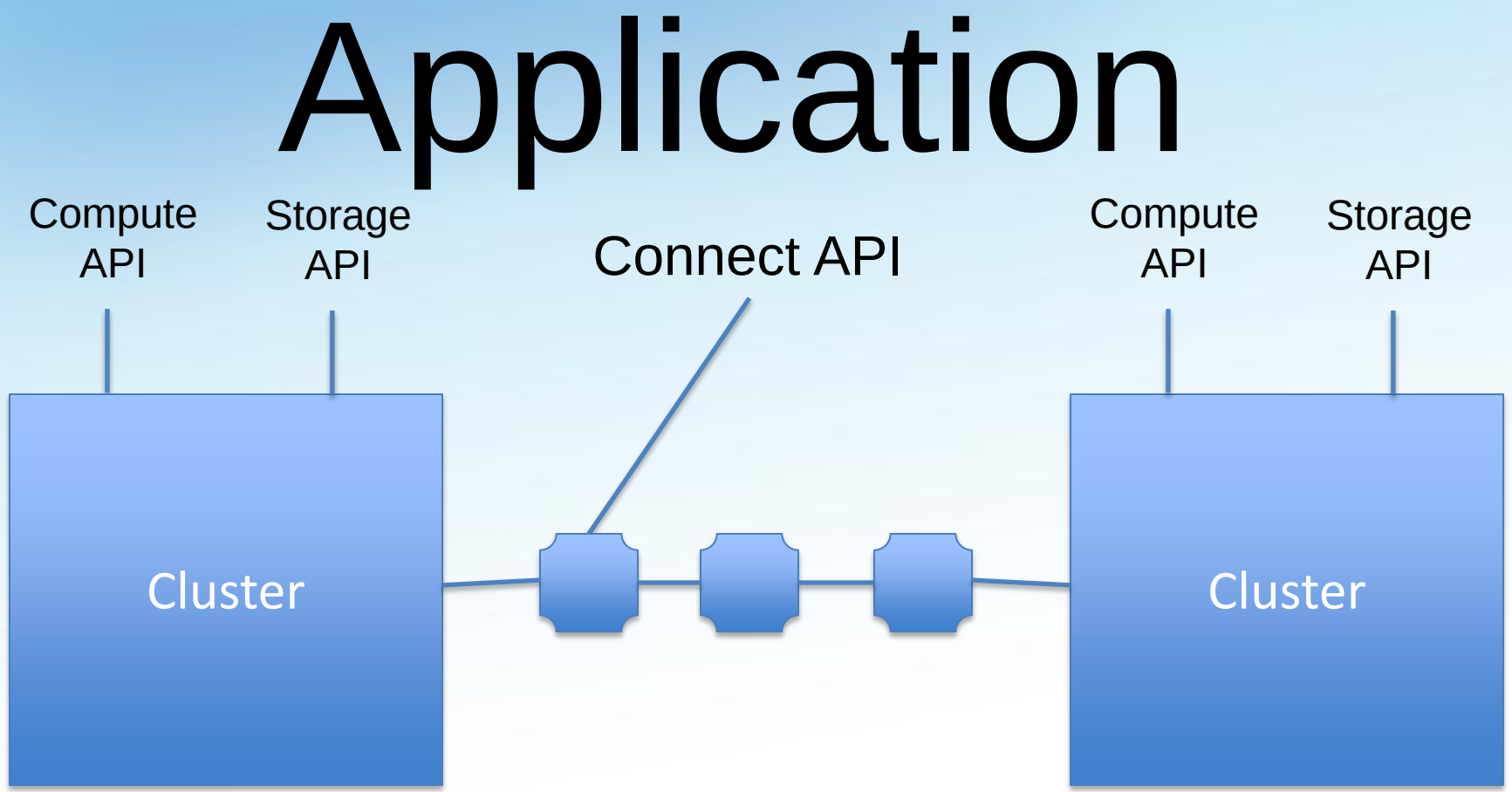
Black box



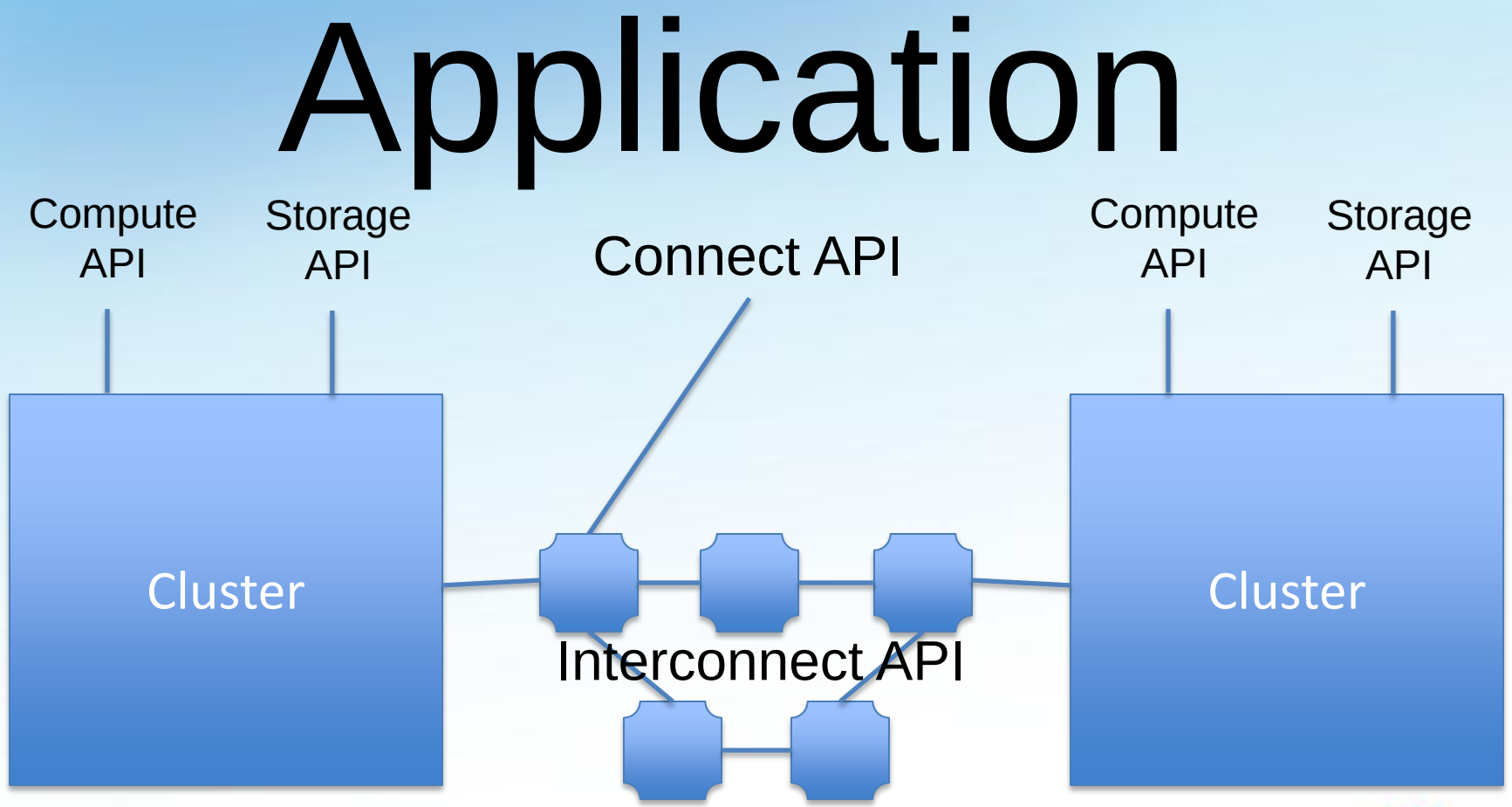
Distributed Black Boxes



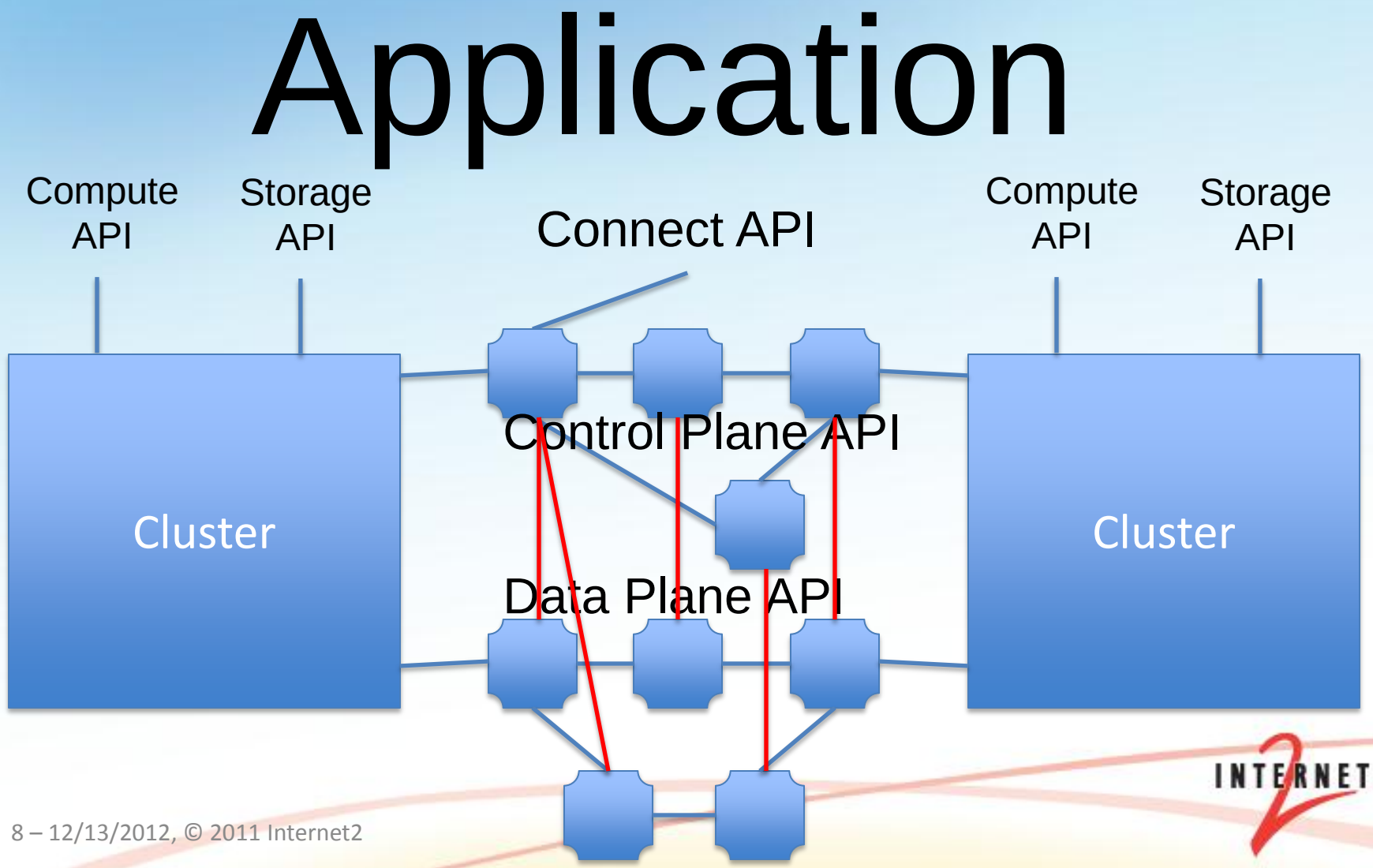
Connected by Network Elements



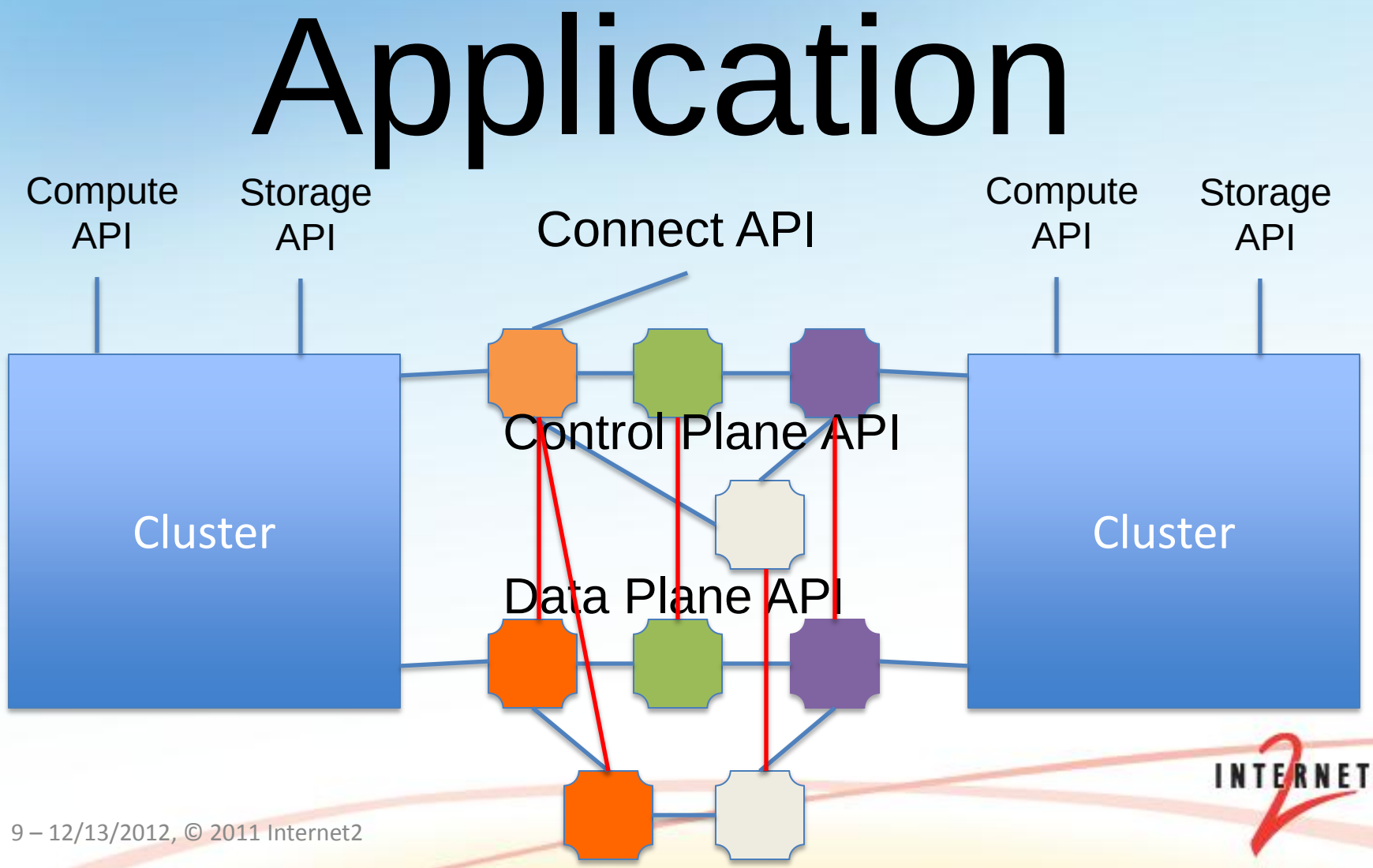
Connected by Network Elements



Control Plane directs Data Plane



Control Plane directs Data Plane



Observation

- The network community spends a lot of time thinking about interconnect APIs
- We conflate interconnect APIs and connect APIs on a regular basis
- Most of my mistakes later in this presentation are probably due to that lack of clarity
- Application programmers don’t care about the interconnect APIs (unless they have to debug them)

Application Flowchart (1)

- Figure out in the application being used where the requirement for a circuit occurs.
  - In a file transfer application this might be when the transfer takes place for instance
  - Understand how long it typically takes to set up an circuit so that the application can start that process when it needs to
- Put a hook in the application to signal the control plane server via the connect API
- Send appropriate info to the control plane
  - From where, to where
  - Starting when, lasting how long
  - Needed capacity
  - Optional: Via what path

Application Flowchart (2)

- Now the end nodes in the file transfer need to be made aware of each other.
  - Authorize application to configure end nodes
  - Configure end nodes
    - e.g. assign IP addresses, any other required Layer (3) setup
- Await setup of circuit and begin sending data

Overview

- View from the Application
- Connect API
- Reality is Messy: Opening up the Black Box
- The World Today
- When It All Goes Wrong

Connect API: Goals

- Primary Goal
  - Application support for Bandwidth on Demand
- Secondary Goals
  - Minimize manual intervention in this process
  - Maximize automation of circuit setup, usage and teardown.

Connect API:
Typical Set of Primitives

- Make
- Modify
- Cancel
- Query

Connect API: Circuit Creation

- Make circuit
  - Create circuit between ingress and egress points
  - Capacity (bits-per-second)
  - Maximum payload size
  - Start / duration
  - Path (optional)
  - Authorization
  - The application must deliver frames of the type supported by the service.
    - e.g. 802.1q frames
    - Other types are possible also

INTERNET 2

Connect API: Other Primitives

- Modify circuit
  - Can be done as a cancel and make
  - Could also be done on an existing reservation
- Cancel circuit
  - Terminate circuit of finite (prematurely) or indefinite duration
- Query circuit
  - Inquire about status of circuit

INTERNET 2

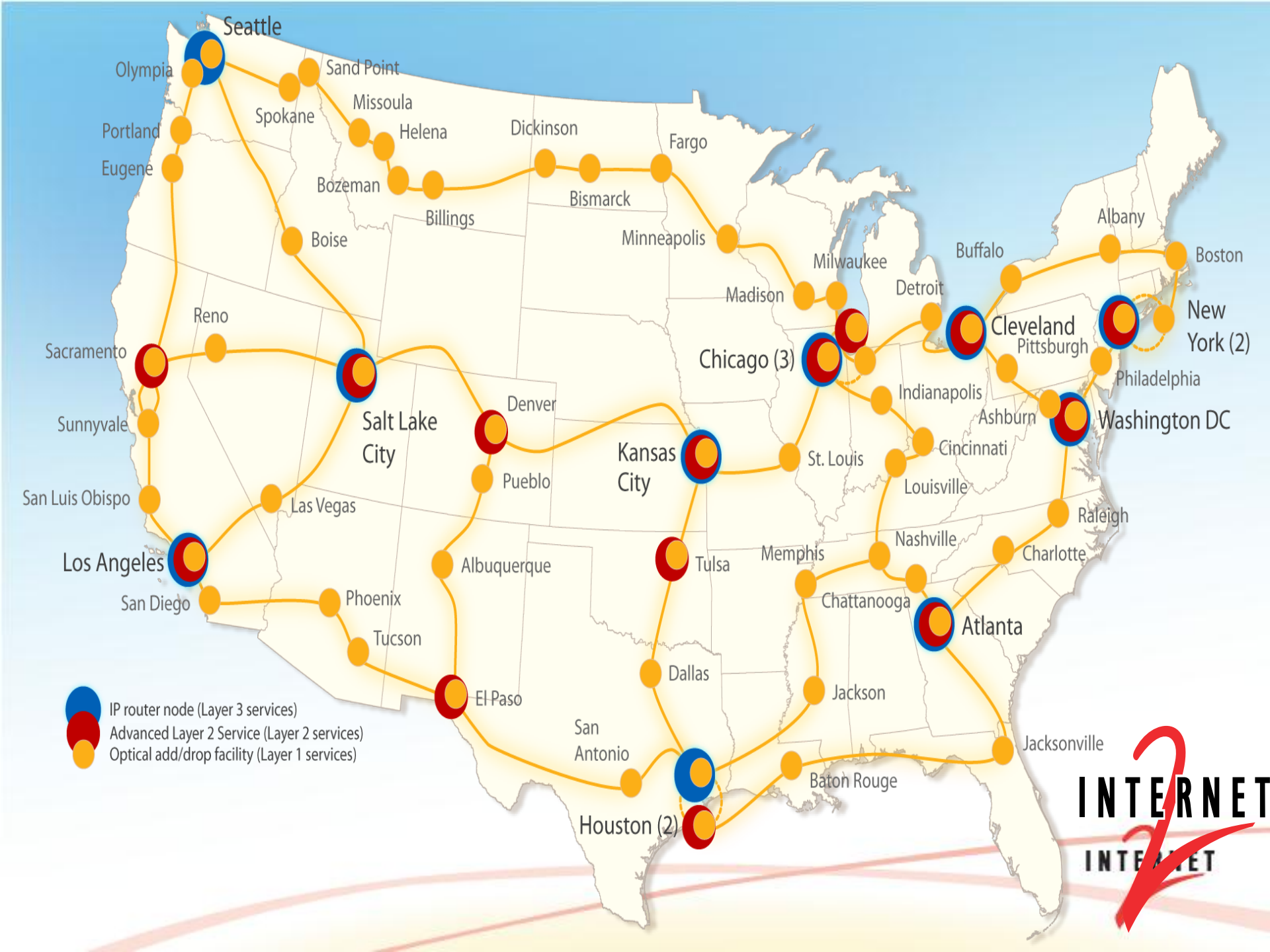
Iterative Dialog through the Connect API

- Can the application assess the needed bandwidth?
  - If not, how is that information entered?
- Should the application be able to re-assess time vs bandwidth if a setup rejection occurs?
  - Is it ok to take longer at less bandwidth?
  - Does this happen automatically or does the user intervene?
- Does the API provide sufficient information if a circuit setup fails to allow the application to try with different parameters?
  - Parameters might be speed, duration, or path selection.

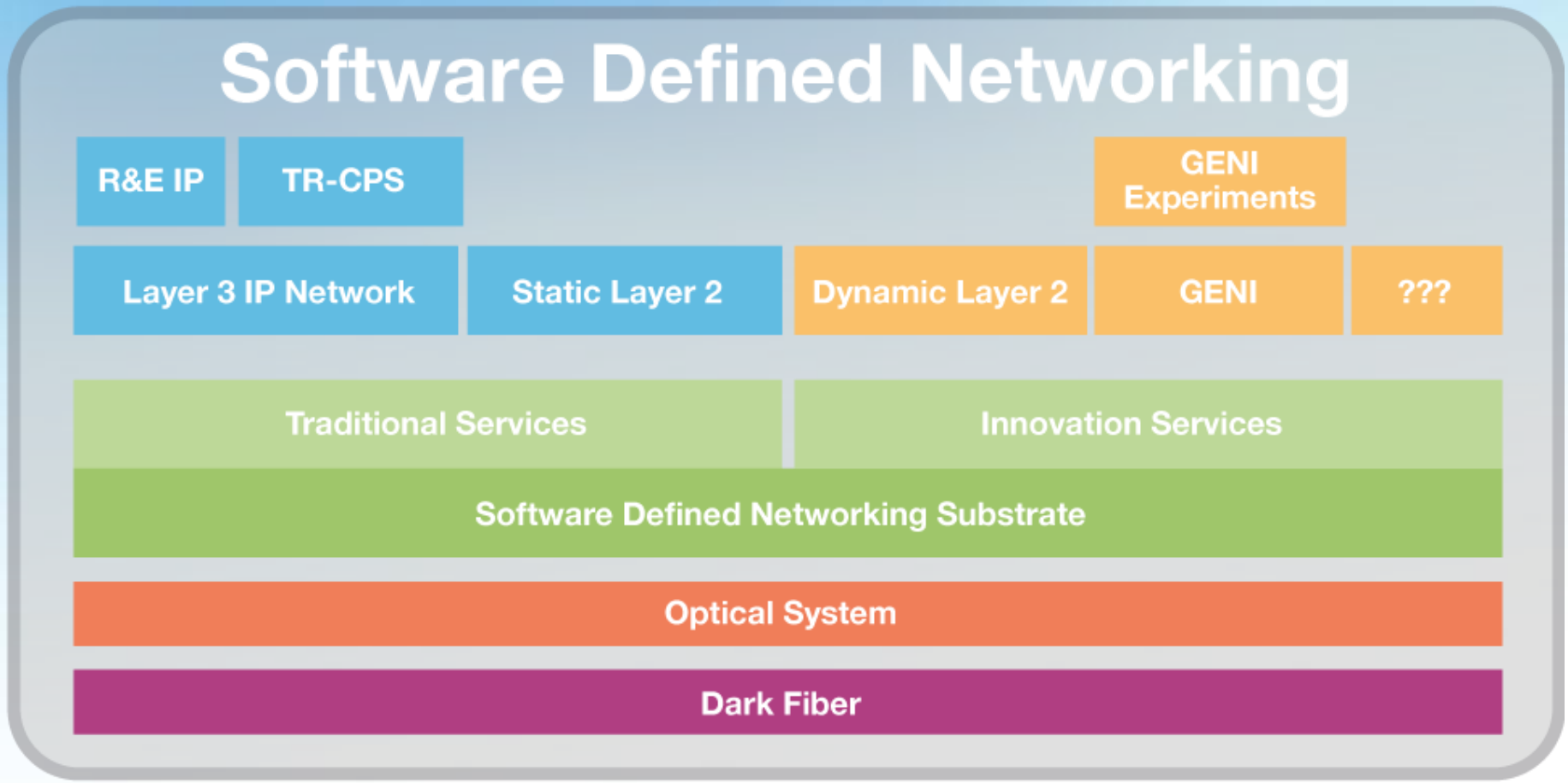
INTERNET 2

Overview

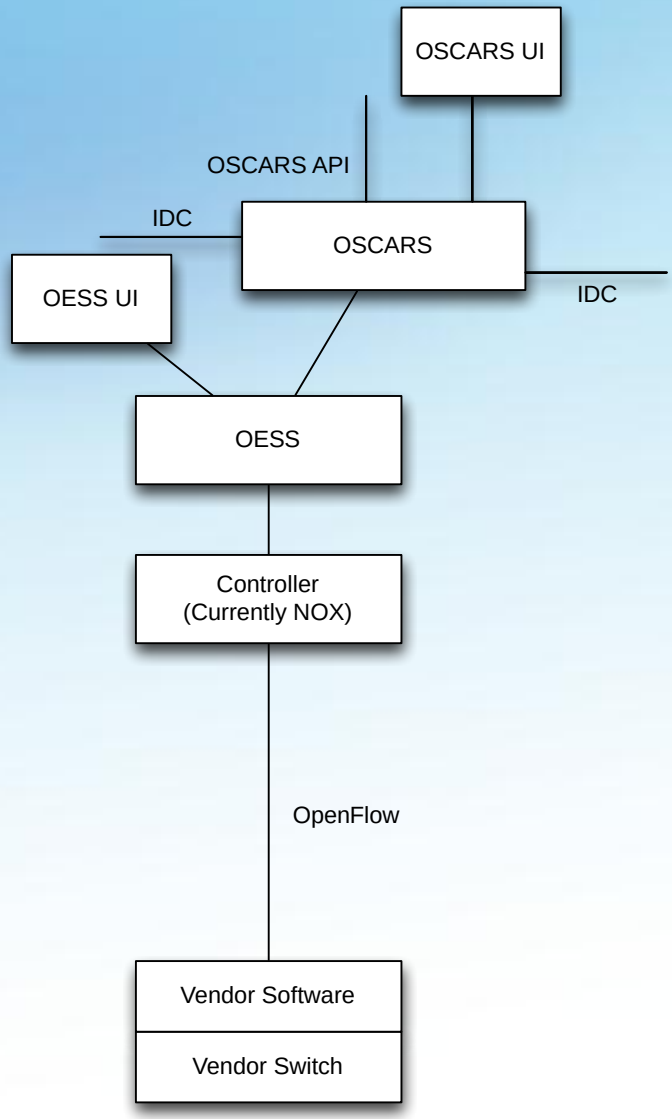
- View from the Application
- Connect API
- Reality is Messy: Opening up the Black Box
- The World Today
- When It All Goes Wrong


Seattle
Olympia
Sand Point
Spokane
Missoula
Portland
Helena
Dickinson
Fargo
Eugene
Bozeman
Bismarck
Billings
Albany
Boise
Minneapolis
Buffalo
Boston
Milwaukee
Madison
Detroit
New York (2)
Reno
Cleveland
Sacramento
Chicago (3)
Pittsburgh
Philadelphia
Indianapolis
Denver
Ashburn
Washington DC
Sunnyvale
Salt Lake City
Kansas City
St. Louis
Cincinnati
Pueblo
Louisville
San Luis Obispo
Las Vegas
Raleigh
Nashville
Memphis
Charlotte
Los Angeles
Albuquerque
Tulsa
Chattanooga
San Diego
Phoenix
Atlanta
Tucson
Dallas
Jackson
El Paso
San Antonio
Jacksonville
Baton Rouge
Houston (2)
IP router node (Layer 3 services)
Advanced Layer 2 Service (Layer 2 services)
Optical add/drop facility (Layer 1 services)
INTERNET
INTERNET

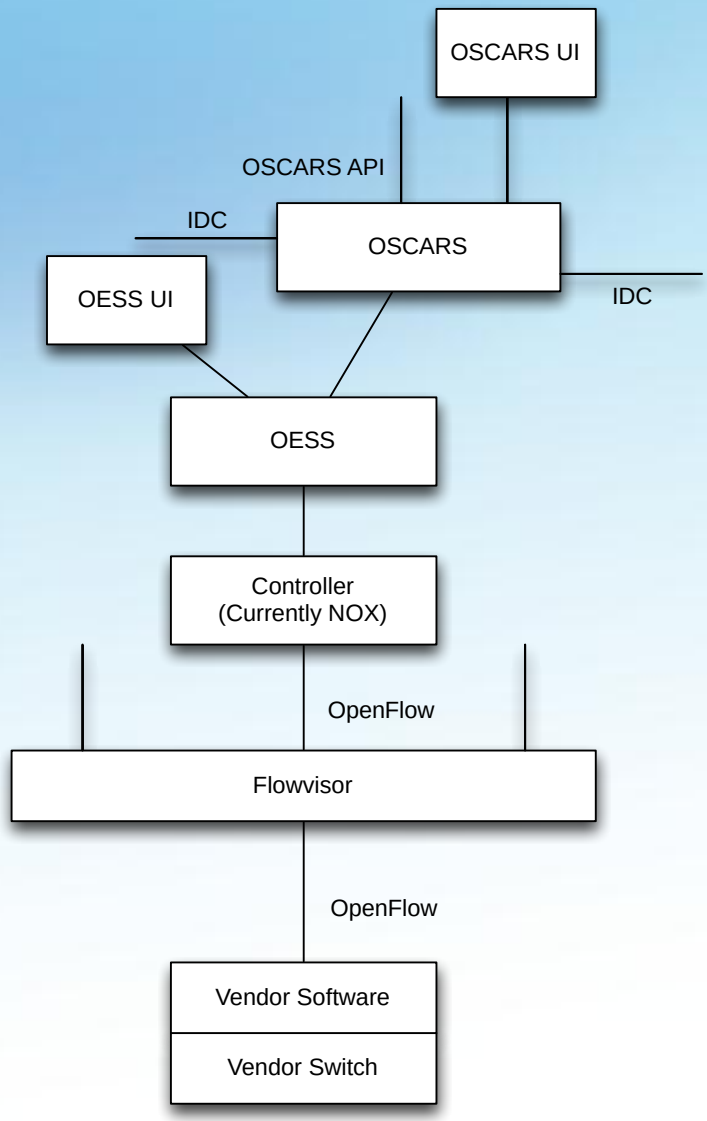
Long-term Architectural Aspiration



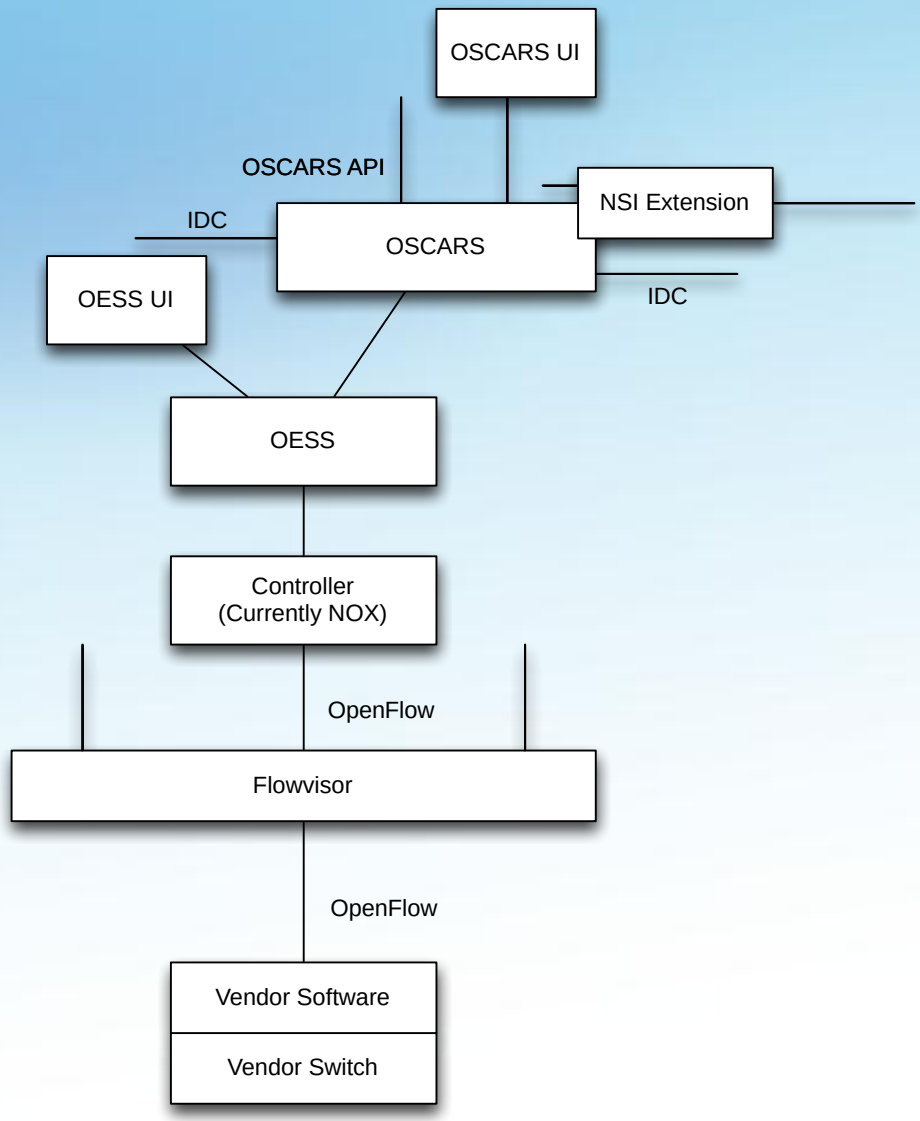
Software Stack — Today



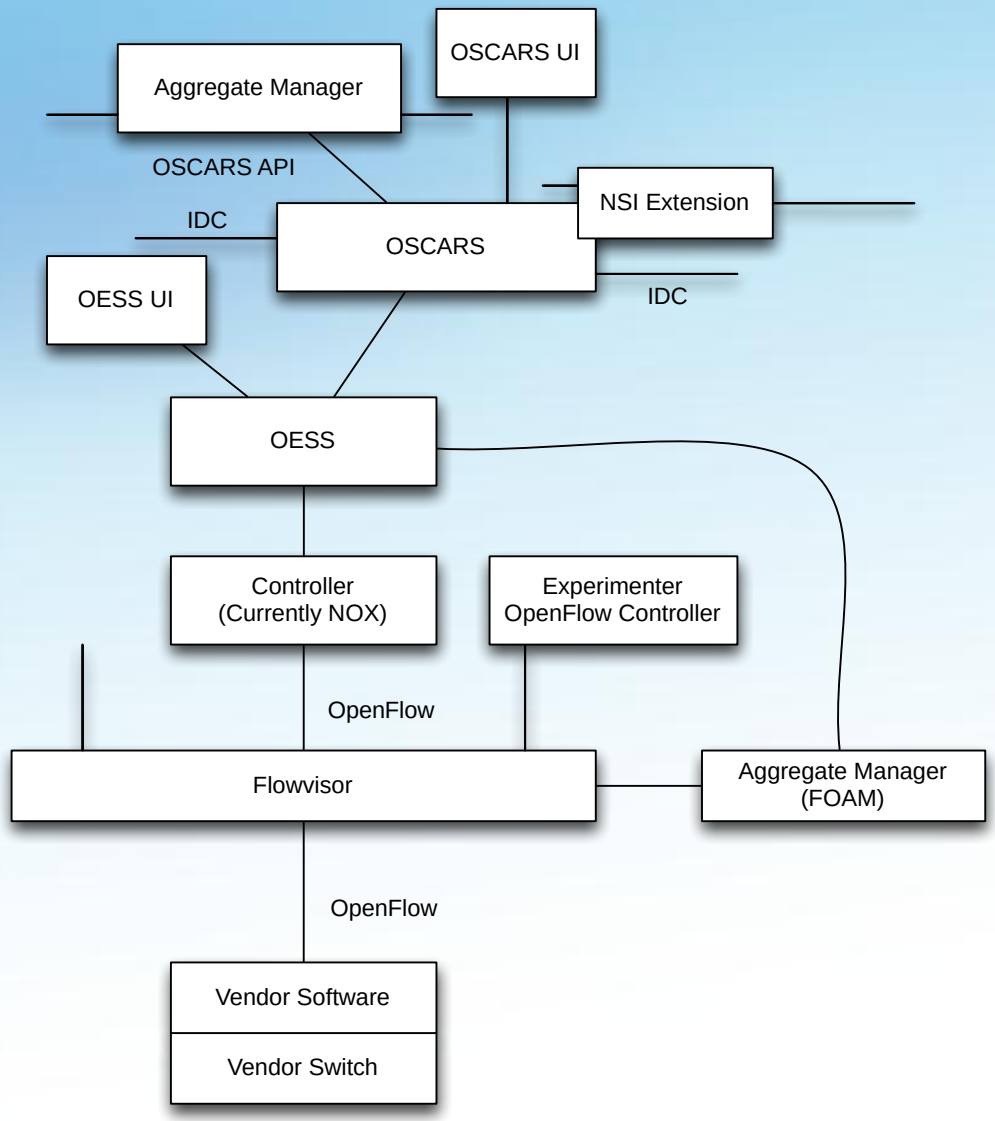
Software Stack — Q1 2013



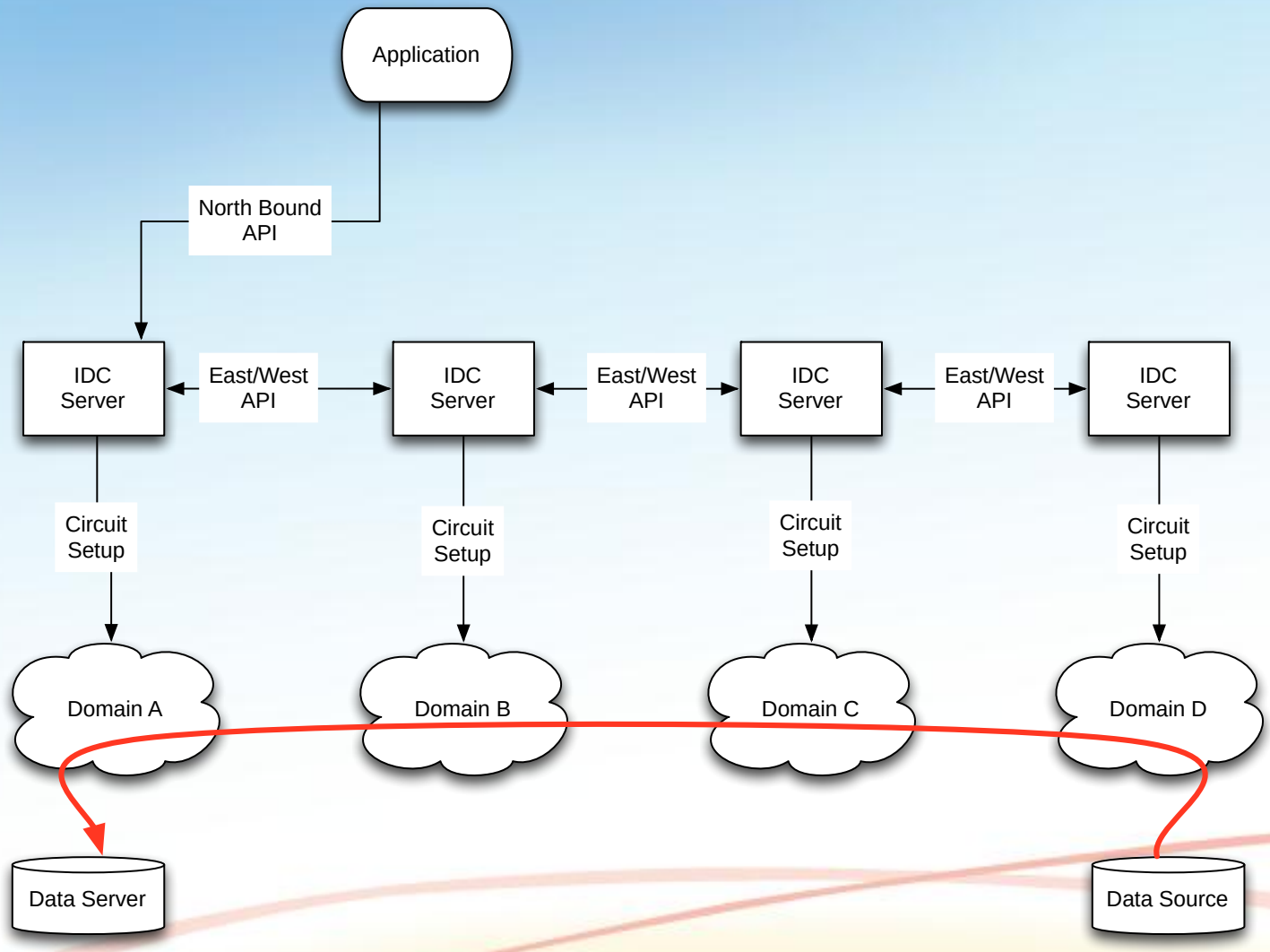
Software Stack — Future



Software Stack — Not fully baked



Internet2 Multi-Domain Environment



Existing R&E software that implements Connect and Interconnect APIs (1)

- Connect and Interconnect Protocols
  - NSI
    - Most recent, standards-based protocol development effort
    - Comes in 2 flavors: 1.0 and 2.0 interconnect API
  - IDCP
    - Predecessor to NSI, not a standard, but widely deployed in R&E production environments in some countries
    - Both a connect and interconnect API
  - Aggregate Manager
    - Network research community in the US
    - Both a connect and interconnect API

INTERNET 2

Existing R&E software that implements Connect and Interconnect APIs (2)

- Protocol Implementations
  - OSCARS v0.5 and v0.6
    - Supports IDCP connect / interconnect API
    - Being extended to support NSI 2.0 interconnect API
  - AutoBahn
    - Supports IDCP connect / interconnect API, NSI 1.0 interconnect API
    - Being extended to support NSI 2.0 interconnect API
  - OESS
    - Supports OESS and OSCARS (i.e. IDCP) connect API
    - Uses OSCARS (i.e. IDCP and eventually NSI 2.0) for interconnect API
  - Open NSA
    - Implements NSI 1.0 interconnect API
    - Being extended to support NSI 2.0 interconnect API and a connect API
  - Open DRAC
    - Implements NSI 1.0 interconnect API
    - Being extended to support NSI2.0 interconnect API and a connect API

INTERNET 2

Overview

- View from the Application
- Connect API
- Reality is Messy: Opening up the Black Box
- The World Today
- When It All Goes Wrong

Exchange Points and Networks

- The connect infrastructure is made up of exchange points and networks
- From the point of view of a exchange point, the “connect infrastructure” is a set of nodes interconnected by wires
- From the point of view of a network, the “connect infrastructure” is a set of networks interconnected directly or at exchange points

INTERNET 2

Exchange Point Examples

- AMPATH
  - Implements IDCP connect API via OSCARS / OESS (GENI) and OSCARS / DRAGON (DYNES)
- MAN LAN and WIX
  - Implementing IDCP connect API via OSCARS now (December)
  - Will support NSI connect API as soon as OSCARS implements it
- NetherLight
  - Implements NSI 1.0 interconnect API via OpenDRAC
- PacWave
  - Implements ??? connect API via ???
- Starlight
  - Implements NSI 1.0 interconnect connect API via ???
  - Plans to implement NSI connect API

Network Examples (1)

- Internet2 Network
  - Implements IDCP connect API
    - Via OSCARS v0.6 / OESS / OpenFlow (Advanced Layer 2 Service)
    - Via OSCARS v0.5 over MPLS (ION)
  - Plans to implement NSI 2.0 connect API
- Internet2 Regionals and Campuses in the DYNES project
  - Implement IDCP connect API
    - Via OSCARS v0.6 / OESS
    - Via OSCARS v0.5 and v0.6 / DRAGON
  - Possibility of eventually implementing NSI 2.0 connect API
    - Caltech / UMich / Vanderbilt CC-NIE award
- ESnet Science Data Network (SDN)
  - Implements IDCP connect API
    - Via OSCARS v0.6 over MPLS (Science Data Network)
  - Plans to implement NSI 2.0 connect API

Network Examples (2)

- GEANT Network
  - Implements IDCP connect API
    - Via AutoBAHN / cNIS
  - Implements NSI 1.0 connect API
    - Via AutoBAHN / cNIS
  - Plans to implement NSI 2.0 connect API
- Some European NRENs
  - Implements IDCP connect API
    - Via AutoBAHN / cNIS
  - Implements NSI 1.0 connect API
    - Via AutoBAHN / cNIS
  - Plans to implement NSI 2.0 connect API

Network Examples (3)

- Nordunet
  - Implements NSI 1.0 interconnect API and internal WS connect API
    - Via OpenNSA
  - Plans to implement NSI 2.0 connect API

Overview

- View from the Application
- Connect API
- Reality is Messy: Opening up the Black Box
- The World Today
- When It All Goes Wrong

Why look inside the black box?

- Understanding usage patterns / capacity planning
- Debugging
- Application learning / adaptive feedback

Types of Insights

- Why did circuit creation fail?
  - Rejected due to policy?
  - Rejected due to capacity?
  - Failure?
- What does success look like?
  - What do things look like peering down the wire?
    - Active tests like one-way latency measurements or throughput measurements
  - What do things look like when spying on an network element?
    - Passive observation of capacity, usage, failures
- What should change in the future?
  - Capacity planning
  - Diversity of routes
  - Improved applications
  - Better NOC support

State of the Art

- Monitoring of Layer 2 circuits is less developed than connect API
- A subset of R&E networks have agreed to a very basic Layer 3 monitoring framework
  - One-way latency
  - Throughput
- An architecture exists that has been “agreed to” by a subset of R&E networks to monitor Layer 2 circuits
  - Implemented by Internet2
  - Not aware of it being implemented anywhere else …

L2 Monitoring (1)

- Number of reservations requested.
- Number of reservations that succeed.
  - Number of reservations that fail.
- Duration of reservations - how many were for 15 min, 30 min etc. up to >1 month for example
- Amount of bandwidth reserved

INTERNET 2

L2 Monitoring (2)

- Other necessary metrics, which will be more of a challenge to collect are:
  - Utilization of the reserved circuit during the entire reservation period
  - Reliability of the circuits
- Overall for the infrastructure there are two sorts of measures that would be valuable. For every given segment in the network we should know both
  - The percentage of available reservation space that is reserved.
  - The percent of use of that segment that is due to circuits

L2 Monitoring (3)

- Useful metrics for assessing and debugging
  - Ingress/egress points for circuit both failed and succeeded.
  - Interactions between control plane nodes, Filtering on some marker to determine what project is behind the initiating request
- Root cause of request failure?
  - Control plane?
  - Data plane?
  - Oversubscription?
  - Something else

Summary

- Ideally, compute, storage, and connect are a single black box
- Reality is messy
  - Moving towards a reasonable architecture to code to
  - BoD software is not fully standardized yet
  - Deployment is not fully there yet
  - Monitoring of Layer 2 circuits is lagging
- But enough exists to get started

December 13, 2012

Eric Boyd, Internet2

eboyd@internet2.edu

Bandwidth on Demand in the Wild