GEANT Testbeds Service

Jerry Sobieski (NORDUnet)
GN3plus-SA2 Activity Leader

Presented to the Internet2 Global Summit
Denver, US
Apr 8, 2014
Network Innovation requires testing to prove out...

- “Production” environments have the required scale but are highly risk averse.
- Testing in live networks can have unintended effects on non-combatants.
- Other users and network providers don’t like being crash test dummies.
- Production services have tangential concerns as well – e.g. privacy, legal requirements, etc.

How do we accelerate innovation from concept to testing to production, rapidly, yet with minimal risk to existing stable and reliable infrastructure, services, and applications??
Networking R&D Laboratories

- The network research community needs “Laboratories” to test novel concepts...
  - Environments that allow high risk experiments to be carried out...
  - Yet secure to prevent unexpected or errant behaviour from interfering with existing services or research activities
  - Agile: Ability to rapidly prototype new ideas or integrate new results/findings
  - Constructed from stable underlying infrastructure
  - Extensible: Enable a broad range of innovation and technologies
  - Scalable: Ability to construct large scale test environments
  - Geographical reach – a realistic global environment
Key Testbed Service capabilities:

- Flexible virtual network resources
  - VMs, VCs, VXs, ...
- Flexible distributed network topologies
  - Object oriented, domain specific description language
- Under control of the researcher
- Insulated environments to prevent collateral damage
- Extensible support for novel user hardware
- Pan-European footprint

The GEANT Testbed Service is a Production Service

- The “testbed” capabilities are expected to be reliable and consistently available 24x7x365
- The Testbed Service functional processes must be stable and secure
- Integrated “multi-species” virtualization represents new technology that is still evolving in the community ...
Dynamic Testbed Provisioning - How it works

Network testbed concept to test brilliant idea

Testbed template doc “alpha” network

Resource Manager allocates resources

Testbed is activated and user controls it via the TCA

Researcher logs in, builds a testbed DSL via a web GUI

Resource A: port p0, p1;
Resource B: port out1, out2;
Adj B/out1==A/p0;

Researcher has a brilliant idea

Switch “B”

Virtual Circuit “L1”

Virtual Machine “A”

VM “C”

Switch “B”

Virtual Circuit “L2”

Virtual Machine “A”

VM “C”

Switch “B”

Virtual Circuit “L3”

Virtual Machine “A”

VM “C”

Switch “B”

Virtual Circuit “L3”

Virtual Machine “A”

VM “C”

Switch “B”

Virtual Circuit “L3”

Virtual Machine “A”

VM “C”

Switch “B”

Virtual Circuit “L3”

Virtual Machine “A”

VM “C”
The TaaS Architecture treats all [testbed] networks as *graphs*.

**Testbed “Alpha” Description**

Internally, all testbed components are treated as generalized **virtual Resources**. Resources all have a set of explicitly defined **data flow Ports**. User specified Port adjacency relations define the testbed topology.

**“Derived Resource Graph” data plane**
GEANT “Testbeds as a Service” Virtualization, Management, and Control Layers

SA2 Core Resource Manager and Resource Control Methods

TaaS Virtualization Layer Services

OpenNSA/BoD

NFS

OpenStack

JunOS/HP

...
The Domain Specific Language – the “DSL”

- Class NodeLink using a host and a link

```python
Class NodeLink {
  def myvm1 {
    host {
      location = Ams
    }
  }
  def l1 {
    link {
      bandwidth = 100
    }
  }
  adjacency myvm1.p2, l1.p1
  port { id = ”nlp1” }
  port { id = ”nlp2” }
  adjacency nlp1, myvm1.p1
  adjacency nlp2, l1.p2
}
```
The Domain Specific Language – the “DSL”

- Describing a resource called “testbed”

testbed {
    name = "Hosts connected in a triangle topology."
    def hosts = []
    def links = []
    3.times { idx ->
        def h1 = host {
            id = "host$idx"
            cpuCores = 3 - idx
            port { id = "p1" }
            port { id = "p2" }
        }
        hosts << h1
        def l1 = link {
            id = "link$idx"
            bandwidth = 1000
            rtt = 50
            port { id = "l1"; mode = "bidirectional" }
            port { id = "l2"; mode = "bidirectional" }
        }
        links << l1
        adjacency h1.p1, l1.l1
    }
    3.times { idx -> adjacency
        hosts[(idx + 1) % 3].p2,
        links[idx].l2 }
}
Testbed Resources

- Resources are organized by their shared functional characteristics:

  - Computational
    - Virtual Machine
    - Virtual Switch
    - LinuxVM
    - OpenFlowFabric
  - Transport
    - Virtual Circuit
    - EftsVC
    - Broadcast Domain
    - UserAccessGW
  - Composite
The “Gang of Five” Common Resource Control Primitives:

- **Reserve()** – A request from TCA to RM to find resources and to reserve them for this user/project.
- **Activate()** – Given a reserved resource, the primitive instructs the RM to place the reserved resource into service.
- **Query()** – Obtain the resource specific state information for a particular resource instance
- **Deactivate()** – Take a resource instance out of service, but retain the reservation.
- **Release()** – deactivate a resource and release the entire reservation for that instance.
Each Resource Class defines methods (control primitives) that translate high level TaaS control semantics to resource specific command sequences.

- Each resource class must implement the gang of five..
- Each resource class may define additional control primitives/semantics that may be specific to that class of resource only

For instance:

- A “LinuxVM” Class may implement a “ColdStart()” primitive that fully re-initializes the VM via OpenStack and boots it.
- That same class may also implement a WarmStart() primitive that simply reboots the OS for a VM.
- These primitives do not make sense for Virtual Circuit resource instance..

New resources classes [user defined?] may be introduced into the TaaS service by developing these control primitives.
The Testbeds ‘GUI’

The web GUI allows users to access the Testbed Service from any common web browser. From the GUI, testbed admins and users are able to manage their resources through the entire lifecycle.
The SA2 TaaS service domain engineering (Pilot Phase)

Red = Control & Mgmt Net
Blue = Data plane infra.
Greenish = external service

Internet

X.Y.Z.0/29

CSF

GN3+ Core (waves)

AMS

PRA

BRA

LJU

802.1 EoMPLS/10G

Ethernet framed Wave

Private VRF

pod

pod
Deployment sequence is a function of space in the pops, and budget. Post LJU sites/seq are still TBD.

GOFF=FRA, AMS, VIE, LON, ZAG
Timeline

- LOTS (!) of SW development happening
- HW production line: order, staging, deployment...
- User Guides, Resource guides, operations/monitoring
- 2014-Q2 TaaS v1.0 production service Phase 1

- 2014-H2 scaling up
  - Training/Seminars/Workshops/online video tutorials
  - Richer selection of resources/attributes
  - Consolidation of GEANT “testing” services/environments, and expanded geo footprint
  - Scaling target: 10^3 VMs, 10^3 VCs
  - Reach: Inter-domain multi-domain interoperability
SA2 Features Roadmap

- **TaaS 1.0(beta) Pilot** – April/May 2014 thru Q2
  - Shake down with small set of [willing] real users
  - Locations: CM1/CM2; + CPH, BRA, AMS, LJU
  - Resource Classes:
    - “Big” VMs (1 core/4GB/500G),
    - VCs (up to 1Gbps in pilot)
    - VXs virtual switches – OpenFlow capable hw (!) HP5900 now, others in future (TBD).
    - Composite Resources
    - Simple Internet Access Resource (non-trivial feature!)

- Hands on training for early adopters
- Full 24x7 NOC support and operations procedures
- GOFF continues (sw OF)
SA2 Features Roadmap

- **TaaS v1.0 Full –2014-Q2**
  - Open user base
  - Refine resource base
  - Begin migrating GOFF users to TaaS
  - NAS storage, 10Gbps VCs,...

- **TaaS 1.1 2014-Q4**
  - Initial multi-domain resource brokering
  - Improved DSL
  - More dynamic resource creation from infrastructure
  - Interface with BoD and NRENs for VCs
  - Bare Metal servers (blade servers (?))
  - Dynamic in-situ testbed modification.
CY15Q1  TaaS version 2.0

- “Intelligent” Composite resources  e.g.:
  - Resources that can sense the presence of other resources
  - “Resources” need not be just HW analogs...they can be services/agent analogs (IP networks, shared storage, etc.)

- New [atomic] resource species (e.g. photonic/spectrum based virtual resources, ...)

- Broader global applications/use cases  e.g.:
  - Science workflows,
  - High performance global video transport facilities/services,
  - CDNs/distributed repositories

- Global multi-domain VNE provisioning/realtime [user] control
Testbeds (virtual network environments / slices) must scale globally - yet preserve security, insulation, control, privacy, etc

GEANT SA2-T4 is exploring a strategy
  - EU deployment within NRENs and Campus service implementations
  - Inter-domain interoperation with similar projects in other international regions
    - US/NA (Internet2 AL2S, GENI )
    - SA (RNP, et al)
    - APAC
  - Common service semantics, common inter-domain architecture, consensus interoperability protocol(s), ...
SA2 Task Leaders

- T1: Hardware and Systems Engineering: 
  Michal Hazlinsky (CESnet)

- T2: Software Development: 
  Blazej Pietrzak (PSNC)

- T3: Service Management: 
  Peter Szegedi (TERENA)

- T4: Multi-Domain Interoperability: 
  Fabio Farina (GARR)
SA2 Conspirators:

- GARR
- PSNC
- TERENA
- DANTE
- CESnet
- AMRES
- GRnet
- RedIRIS
- DFN
- RENETER
- HEAnet
- NIIFII
- NORDUnet
Finally... What’s in a Name?

- “TaaS” is not ideal... sounds like a Russian news agency, ... Or a cartoon character

- How about the World Testbed Facility... “wtf” ...?
  
  
  (beta version...)
Questions?