NSI from CoUniverse Perspective

Petr Holub

Sitola – Laboratory of Advanced Networking Technologies
CESNET
Masaryk University

NSI-WG

Banff, AB, Canada, 2009–10–14

Talk Overview

CoUniverse

NSI Usage Scenarios

CoUniverse

- Self-organizing application orchestration for real-time media-based collaborative applications
- Requirements
  - self-organized system that can accommodate changes in underlying infrastructure
  - support for applications with bandwidth requirements comparable to link capacity
  - incorporation of external applications
  - support for multi-point data distribution
  - built-in monitoring and visualization
  - as user-empowered as possible
- Universe
  - each for single collaborating group

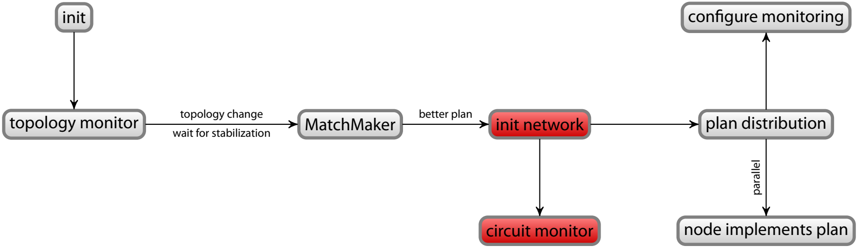
CoUniverse

- Architecture
  - control plane
    - distribution of control information
    - self-organizing P2P control plane
    - optimized for robustness
    - not optimized for throughput
  - data plane
    - uses native network
  - application encapsulation
    - start/stop/restart
    - on-the-fly control if supported
  - built-in monitoring
    - network (end-to-end)
    - nodes
    - applications
  - scheduler for media streams
    - including multi-point data distribution based on reflectors

CoUniverse

- Implementation
  - Java-based implementation
  - JXTA overlay network for control plane
  - Application Group Controller
    - scheduler based on constraints solver
    - creates plan for setting up nodes based on users' requirements
    - handles network setup if needed
    - distributes plan to required nodes to configure themselves
    - when change in the underlying infrastructure is detected, new plan is computed
  - currently supported applications
    - UltraGrid in various modes
    - MBone Tools
    - VideoLAN Client
    - several flavors of reflectors
    - Poycom H.323 devices

CoUniverse AGC Diagram



- MatchMaker
  - finds suitable source based on configuration of each receiver (if possible)
  - builds plan based on available network features (links, reflectors)
- network initialization
  - added for end-to-end circuit initialization
  - blocking stage to make sure we have the network prior to application startup

NSI Usage Scenarios

- What CoUniverse needs from NSI (and maybe others)?
    - information service
        - what networks are reachable from given interfrace (port) of a host
        - topology information if available (even partial)
    - allocation service
    - monitoring
- CoUniverse role from NSI perspective
    - Requesting Agent
    - credentials proxy

Information Service

- Information to what network is given port connected
- Information about what can be allocated
- Actual/estimated latency
    - for latency minimization optimization
    - can we get that for inactive links?
    - remeasured by the middleware/applications once the circuit is active

Multi-Point Networks

- "Flat" network among group of hosts
  - may be of interest for generic applications
  - e.g., MPI calculation with changing communication pattern
- Can be emulated
  - requesting full-mesh of circuits
  - creating various topologies with hints from users/middleware
  - more efficient would probably be to leave it for the layer that has detailed knowledge of topology and/or policies (with optional hints from users/middleware)

Multi-Point Distribution

- So far CoUniverse relies on application reflectors
- Application-level implementation gives also other options (transcoding, per-user processing, etc.)
- We would like to have also network doing multicast
- How can we request for that?
- May be implemented with the multipoint circuit with multicasting capability
- L1 vs. L3 multicasting

CESNET

UNIVERSITAS MASARYKIANA BRUNENSIS

AA(A) Issues

- How to handle things for which you need more than one identity?

$$\text{Host}_A \quad \underbrace{\longleftrightarrow}_{\text{auth:user}_1} \quad X \qquad \underbrace{\longleftrightarrow}_{\text{auth:user}_1,\text{user}_2,\text{user}_3} \qquad X \quad \underbrace{\longleftrightarrow}_{\text{auth:user}_3} \quad \text{Host}_B$$

  - there are ways but there are caveats
- All users in the virtual collaborative group sign all requests
  - doesn't require publishing information on who is allowed to request what
  - may require limited proxy certificates (in X.509 terms) to make things automatic
  - the circuit may be denied if even all the credentials are not sufficient
  - what should happen if a user leaves the group?

AA(A) Issues

- Problem of delegation – do we want it?
  - maintaining authorization database may be tedious
  - building delegation chain may help
  - circuit are rather limited resource, so we probably don't want to give them to everybody

Request Specifications

- Bandwidth – problem of bursts revisited
  - we are orchestrating *legacy* applications
  - sometimes there are good reasons for bursts (e2e latency with bursty source)
  - we may need to allocate burst size instead of average bandwidth
    - problem of wasted capacity
    - we can’t multiplex multiple streams like this in the way we did it in the past (3× 1.5 Gbps UltraGrid streams with 6 Gbps bursts)
  - in any case, we should give tools to the user to tell what they should ask for (normal users can’t tell how bursty their applications are)

Thank you for your attention!

Q?/A!

`<hopet@ics.muni.cz>`