Novel Network Services for Supporting Big Data Science Research

JOAQUIN CHUNG, SEAN DONOVAN, JERONIMO BEZERRA, HEIDI MORGAN, JULIO IBARRA, RUSS CLARK, HENRY OWEN
Motivation
Motivation

- These circuits are typically provisioned by network operators.
- The interface for requesting advance reservations is designed by network operators for network operators.
Motivation

- Not intuitive for domain-expert scientists
- If reservation fails, user have to start again (cycle of trial and error) [1]
- Manual provisioning might take weeks [2]
Outline

- Background
- AtlanticWave/SDX Architecture
- Future Generation Science Network Services
- Related Work
- Conclusions
Outline

- Background
  - Software-Defined Exchange (SDX)
  - Software-Defined Networking (SDN)
- AtlanticWave/SDX Architecture
- Future Generation Science Network Services
- Related Work
- Conclusions
Software-Defined Exchange (SDX)

An SDX is a novel cyberinfrastructure that allows multiple independent administrative domains to share computing, storage, and networking resources in a programmatic way.
What is SDN?

Software Defined Networking (SDN) separates the control plane from the data plane
Outline

- Background
- AtlanticWave/SDX Architecture
- Future Generation Science Network Services
- Related Work
- Conclusions
We need a more intuitive interface for domain scientist to request science network services.

The Stack

Front-end → Python Flask
  ◦ Web interface
  ◦ REST API

SDX and Local Controllers
  ◦ Ryu SDN framework → Written in Python

SDN switch configuration
  ◦ OpenFlow v1.3
  ◦ Corsa switches
SDX User Interface

Request a Pipe
Users can request for a pipe based on their requirements and role

Request a Data Transfer
Users can request for a data transfer based on their requirements and role
SDX User Interface Demo

Network Operator: https://youtu.be/EczfnoeHbgQ
Scientist: https://youtu.be/tjoKZNM41Qk
REST API

GET /api/v1/policies/

List all visible policies. Administrators are able to view all policies, while regular users are only able to see their own policies.

GET /api/v1/policies/number/<policynumber>

Get details of a given policy specified by policynumber. Each policy type will return different style of information, so we've sequestered the details into a sub-piece
REST API

POST /api/v1/policies/type/scitunnel/

- Create a new L2 Tunnel Policy from a scientist request.

Request JSON Object

- size (int) - Dataset size in bytes
- deadline (string) - Deadline for the data transfer. String should be in RFC3339 format: "2017-04-12T23:20:50"
- srcdn (string) - Name of source data transfer node.
- dstdn (string) - Name of destination data transfer node.

Response JSON Object

- policy (dict) - Link to the newly created policy.
REST API

POST /api/v1/policies/type/l2tunnel

Create a new L2 Tunnel Policy.

Request JSON Object

- `starttime (string)` - Start time of the L2 Tunnel. String should be in RFC3339 format: "2017-04-12T23:20:50"
- `endtime (string)` - End time of the L2 Tunnel. String should be in RFC3339 format: "2017-04-12T23:20:50"
- `srcswitch (string)` - Name of source switch. See the /api/v1/localcontrollers/<clname>/switches/ endpoint for switch names.
- `dstswitch (string)` - Name of destination switch. See the /api/v1/localcontrollers/<clname>/switches/ endpoint for switch names.
- `srcport (int)` - Port number on source switch. See the /api/v1/localcontrollers/<clname>/switches/<switchname>/ports endpoint for switch port information
- `dstport (int)` - Port number on source switch. See the /api/v1/localcontrollers/<clname>/switches/<switchname>/ports endpoint for switch port information
- `srcvlan (int)` - VLAN at source port.
- `dstvlan (int)` - VLAN at destination port.
- `bandwidth (int)` - Bandwidth in kbit/sec.

Response JSON Object

- `policy (dict)` - Link to the newly created policy
REST API

Example Response

HTTP/1.1 200 OK
Content-Type: application/json

{
    "policy": {
        "href": "http://awavesdx/api/v1/policy/number/3",
        "policynumber": 3,
        "user": "sdonovan",
        "type": "12tunnel",
        "json": {
            "12tunnel": {
                "starttime": "1985-04-12T23:20:50",
                "endtime": "1985-04-12T23:20:50+0400",
                "srcswitch": "atl-switch",
                "dstswitch": "mia-switch",
                "srcport": 5,
                "dstport": 7,
                "srcvlan": 1492,
                "dstvlan": 1789,
                "bandwidth": 1
            }
        }
    }`
}
Outline

- Background
- AtlanticWave/SDX Architecture
- Future Generation Science Network Services
- Related Work
- Conclusions
Future Generation Science Network Services

Simplify current science network services

Bandwidth calendaring

- Augmented by external sources (e.g. weather data)
- Predictive

Fetch the nearest dataset

- Physical proximity
- Network congestion
- Green paths
Outline

- Background
- AtlanticWave/SDX Architecture
- Future Generation Science Network Services
- Related Work
- Conclusions
Related Work

Software-Defined Networking (SDN) bandwidth reservation
- Developing applications with networking capabilities via end-to-end SDN (DANCES) [5] → BW management (SLASH2 and GridFTP)

Intent-based Networking
- Intelligent Network Deployment Intent Renderer Application (iINDIRA) [6]
Outline

- Background
- AtlanticWave/SDX Architecture
- Future Generation Science Network Services
- Related Work
- Conclusions
Conclusion

We presented **AtlanticWave/SDX**, an architecture for novel network services, that leverages SDX

We proposed **interfaces** that allow **domain-expert scientists** and data workflow management systems to reserve resources of the scientific network.

We proposed **future generation science network service** such as augmented, predictive bandwidth calendaring, and fetch the closest dataset.
References


Questions/Comments

HTTP://WWW.ATLANTICWAVE-SDX.NET/
HTTPS://GITHUB.COM/ATLANTICWAVE-SDX/ATLANTICWAVE-PROTO
Backup Slides
Software-defined Networking

Decoupling of control and data planes

- The control plane is physically distributed, yet logically centralized (SDN controller)
- The data plane is distributed on the network devices (SDN switches)
- Agile programmability, rapid innovation, and independent evolution

Interfaces:

- Applications to controller (e.g., IDS, load balancer, and traffic eng.) → Northbound
- Controller to SDN switches (e.g., OpenFlow) → Southbound
- Between controllers → West/East
Centralized SDX Architecture Interconnecting Independent SDN Domains

SDX controller interfaces:
- Applications to SDX controller (e.g., science workflow manager or resource scheduler) → **Northbound**
- Controller to SDN participant domains (match SDN northbound interface) → **Southbound**
- Between SDX controllers → **West/East**

SDX controller functions:
- Resource management
- Path computation
- Resource provisioning