Dynamic Network Adjustments for Cloud Service Scaling

draft-dunbar-neotec-net-adjust-cloud-scaling-01

ldunbar@futurewei.com

xiechf@chinatelecom.cn

kausik.majumdar@oracle.com

sunqiong@chinatelecom.com

IETF 121 Dublin

Problem Statement

- **Key Challenges**:
  - Lack of coordination between dynamic cloud service scaling and network configuration.
  - Proprietary solutions limit interoperability across multi-vendor environments.
  - Manual adjustments lead to delays and potential service disruptions.
  - No standardized framework for automating network responses to cloud scaling.
- **Solution**:
  - A framework that automates network adjustments triggered by cloud service changes using standardized YANG models.
  - Extending RFC8969

Framework Overview

- **Unified Resource Model (URM)**: Abstracts network and cloud resources, providing a unified interface for multi-vendor environments.
- **Cloud Orchestration Systems**: Detect changes in cloud services (e.g., traffic increases) and notify the network controllers.
- **Network Controllers**: Software-Defined Networking (SDN) controllers or network orchestrators adjust network resources in real-time based on cloud service needs.

Existing Resource Abstraction Layer

Abstraction Layer	Compute	Storage	Networking	Multi-Cloud Support
Kubernetes	Containers	Persistent Volumes	Services, Ingress	AWS, GCP, Azure, On-prem
Cloud Foundry	Diego Cells	Service Broker	Gorouter	AWS, GCP, Azure
Terraform	HCL Config Files	Block & Object Storage	VPC, Security Groups	AWS, GCP, Azure
OpenStack	Nova	Cinder, Swift	Neutron	On-premise, Cloud
AWS Outposts/Azure Arc	EC2, VMs	S3, Blob Storage	VPC, Virtual Networks	AWS, Azure
ONAP	VNFs, CNFs	Cloud Storage Integrations	SDN Controllers	Multi-cloud

Dynamic-Bandwidth YANG Model:

E.g. when a cloud orchestration system detects increased traffic, it can dynamically request an increase in bandwidth to 1000 Mbps (1 Gbps) on network link link-123:

```
module dynamic-bandwidth {
namespace "urn:ietf:params:xml:ns:yang:dynamic-bandwidth";
prefix dbw;

import ietf-network-topology {
  prefix nt;
}

organization "IETF";
contact "IETF Routing Area";
description
      "YANG model for dynamically updating bandwidth on network links.";

revision "2024-10-18" {
  description "Initial version.";
}

augment "/nt:networks/nt:network/nt:link" {
  description
    "Augment the network topology YANG model to update
       the bandwidth dynamically.";

  leaf requested-bandwidth {
    type uint64;
    description "Requested bandwidth in Mbps.";
  }
 }
```

```
{
  "nt:networks": {
    "nt:network": [
      {
        "network-id": "cloud-network-1",
        "nt:link": [
          {
            "link-id": "link-123",
            "source-device": "device-1",
            "destination-device": "device-2",
            "requested-bandwidth": 1000  // 1000 Mbps bandwidth
          }
        ]
      }
    ]
  }
}
```

Dynamic-Load-Balancer YANG Model:

- E.g., when the cloud service expansion triggers the addition of a new backend server (server-3 with IP 192.168.1.12) to the load balancer lb-1, the cloud orchestration system can automatically trigger the change using the following JSON code:

```
module: dynamic-load-balancer
  +--rw load-balancer
    +--rw balancer* [balancer-id]
      +--rw balancer-id      string
      +--rw algorithm        enumeration
      |     +--:(round-robin)
      |     +--:(least-connections)
      |     +--:(ip-hash)
      +--rw backend-servers* [server-id]
        +--rw server-id      string
        +--rw ip-address     inet:ipv4-address
        +--rw port           uint16
```

```
{
  "load-balancer": {
    "balancer": [
      {
        "balancer-id": "lb-1",
        "algorithm": "least-connections",
        "backend-servers": [
          {
            "server-id": "server-1",
            "ip-address": "192.168.1.10",
            "port": 8080
          },
          {
            "server-id": "server-2",
            "ip-address": "192.168.1.11",
            "port": 8080
          },
          {
            "server-id": "server-3",   // New server added dynamically
            "ip-address": "192.168.1.12",
            "port": 8080
          }
        ]
      }
    ]
  }
}
```

Dynamic-ACL YANG Model:

•E.g., JSON code to add a new rule (rule-3) to the ACL acl-123, allowing SSH traffic (port 22) from source IP 192.168.1.101 to destination IP 10.0.0.10. The existing rules, rule-1 and rule-2, control HTTPS (port 443) and block HTTP traffic (port 80), respectively

```
{
  "acl": [
    {
      "acl-id": "acl-123",
      "rules": [
        {
          "rule-id": "rule-1",
          "action": "permit",
          "src-ip": "192.168.1.100",
          "dst-ip": "10.0.0.10",
          "protocol": "tcp",
          "port": 443
        },
        {
          "rule-id": "rule-2",
          "action": "deny",
          "src-ip": "0.0.0.0",
          "dst-ip": "10.0.0.10",
          "protocol": "tcp",
          "port": 80
        },
        {
          "rule-id": "rule-3",   // New rule added dynamically
          "action": "permit",
          "src-ip": "192.168.1.101",
          "dst-ip": "10.0.0.10",
          "protocol": "tcp",
          "port": 22
        }
      ]
    }
  ]
}
```

```
module: dynamic-acl
 +--rw acl* [acl-id]
   +--rw acl-id          string
   +--rw rules* [rule-id]
     +--rw rule-id       string
     +--rw action        enumeration
     |    +-- permit
     |    +-- deny
     +--rw src-ip        inet:ipv4-address
     +--rw dst-ip        inet:ipv4-address
     +--rw protocol      enumeration
     |    +-- tcp
     |    +-- udp
     |    +-- icmp
     +--rw port          uint16
```

Security Considerations

- **Authentication and Authorization:**
  - Use mutual authentication methods such as TLS certificates to verify the identities of both the cloud orchestrator and the network controller before any configuration commands are accepted.
  - OAuth or API Key-Based Access: For REST API-based communications, secure token-based authentication (e.g., OAuth 2.0) or unique API keys can be employed to validate requests from legitimate sources.
- **Data Integrity:**
  - Use TLS to encrypt communication channels, protecting the integrity of the transmitted data.
  - Employ checksums or hash functions on critical configuration messages to detect any tampering or unintended modifications during transit.
- **Monitoring and Auditing:**
  - Maintain detailed logs of all configuration changes initiated by cloud scaling events, including timestamps, source entities, and specific parameters modified.
  - Conduct periodic audits of the authorization policies, access logs, and configuration adjustments to ensure compliance with security policies and to detect any anomalies.