SDN Research Group
Internet-Draft
Intended status: Standards Track
Expires: August 29, 2015

LM. Contreras
Telefonica I+D
CJ. Bernardos
UC3M
D. Lopez
Telefonica I+D
M. Boucadair
France Telecom

February 25, 2015

Cooperating Layered Architecture for SDN
draft-contreras-sdnrg-layered-sdn-02

Abstract

Software Defined Networking proposes the separation of the control plane from the data plane in the network nodes and its logical centralization on a control entity. Most of the network intelligence is moved to this functional entity. Typically, such entity is seen as a compendium of interacting control functions in a vertical, tight integrated fashion. The relocation of the control functions from a number of distributed network nodes to a logical central entity conceptually places together a number of control capabilities with different purposes. As a consequence, the existing solutions do not provide a clear separation between transport control and services that relies upon transport capabilities.

This document describes a proposal named Cooperating Layered Architecture for SDN. The idea behind that is to differentiate the control functions associated to transport from those related to services, in such a way that they can be provided and maintained independently, and can follow their own evolution path.

This Internet-Draft will expire on August 29, 2015.

Copyright Notice

Table of Contents

1. Introduction

Software Defined Networking (SDN) proposes the separation of the control plane from the data plane in the network nodes and its logical centralization on a control entity. A programmatic interface is defined between such entity and the network nodes, which functionality is supposed to perform traffic forwarding (only). Through that interface, the control entity instructs the nodes involved in the forwarding plane and modifies their traffic forwarding behavior accordingly.

Most of the intelligence is moved to such functional entity. Typically, such entity is seen as a compendium of interacting control functions in a vertical, tight integrated fashion.

This approach presents a number of issues:

- o Unclear responsibilities between actors involved in a service provision and delivery.
- o Complex reuse of functions for the provision of services.
- o Closed, monolithic control architectures.
- o Difficult interoperability and interchangeability of functional components.
- o Blurred business boundaries among providers.
- o Complex service/network diagnosis and troubleshooting, particularly to determine which segment is responsible for a failure.

The relocation of the control functions from a number of distributed network nodes to another entity conceptually places together a number of control capabilities with different purposes. As a consequence, the existing solutions do not provide a clear separation between services and transport control.

This document describes a proposal named Cooperating Layered Architecture for SDN (CLAS). The idea behind that is to differentiate the control functions associated to transport from those related to services, in such a way that they can be provided

and maintained independently, and can follow their own evolution path.

Despite such differentiation it is required a close cooperation between service and transport layers and associated components to provide an efficient usage of the resources.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [RFC2119].

This document makes use of the following terms:

- o Transport: denotes the transfer capabilities offered by a networking infrastructure. The transfer capabilities can rely upon pure IP techniques, or other means such as MPLS or optics.

- o Service: denote a logical construct that make use of transport capabilities. This document does not make any assumption on the functional perimeter of a service that can be built above a transport infrastructure. As such, a service can be an offering that is offered to customers or be invoked for the delivery of another (added-value) service.

- o SDN intelligence: refers to the decision-making process that is hosted by a node or a set of nodes. The intelligence can be centralized or distributed. Both schemes are within the scope of this document. The SDN intelligence relies on inputs form various functional blocks such as: network topology discovery, service topology discovery, resource allocation, business guidelines, customer profiles, service profiles, etc. The exact decomposition of an SDN intelligence, apart from the layering discussed in this document, is out of scope.

Additionally, the following acronyms are used in this document.

CLAS: Cooperating Layered Architecture for SDN

FCAPS: Fault, Configuration, Accounting, Performance and Security

SDN: Software Defined Networking

SLA: Service Level Agreement

3. Architecture overview

Current operator networks support multiple services (e.g., VoIP, IPTV, mobile VoIP, critical mission applications, etc.) on a variety of transport technologies. The provision and delivery of a service independently of the underlying transport capabilities requires a separation of the service related functionalities and an abstraction of the transport network to hide the specificities of underlying transfer techniques while offering a common set of capabilities.

Such separation can provide configuration flexibility and adaptability from the point of view of either the services or the transport network. Multiple services can be provided on top of a common transport infrastructure, and similarly, different technologies can accommodate the connectivity requirements of a certain service. A close coordination among them is required for a consistent service delivery (inter-layer cooperation).

This document focuses particularly on:

- o Means to expose transport capabilities to external services.
- o Means to capture service requirements of services.
- o Means to notify service intelligence with underlying transport events, for example to adjust service decision-making process with underlying transport events.
- o Means to instruct the underlying transport capabilities to accommodate new requirements, etc.

An example is to guarantee some Quality of Service (QoS) levels. Different QoS-based offerings could be present at both service and transport layers. Vertical mechanisms for linking both service and transport QoS mechanisms should be in place to provide the quality guarantees to the end user.

CLAS architecture assumes that the logically centralized control functions are separated in two functional blocks or layers. One of the functional blocks comprises the service-related functions, whereas the other one contains the transport-related functions. The cooperation between the two layers is considered to be implemented through standard interfaces.

Figure 1 shows the CLAS architecture. It is based on functional separation in the NGN architecture defined by the ITU-T in [Y.2011]. Two strata of functionality are defined, namely the Service Stratum, comprising the service-related functions, and the Transport Stratum,

covering the transport ones. The functions on each of these layers are further grouped on control, management and user (or data) planes.

```
                                North Bound Interface

                                        /\
                                        ||
      +-------------------------------------||-------------+
      | Service Stratum                     ||             |
      |                                     \/             |
      |                       ............................  |
      |                       . SDN Controller           .  |
      |                       .                          .  |
      |  +--------------+     .        +--------------+ .  |
      |  | Resource Pl. |     .        |  Mngmt. Pl.  | .  |
      |  |              |<===>.  +--------------+     | .  |
      |  |              |     .  |  Control Pl. |     | .  |
      |  +--------------+     .  |              |-----+ .  |
      |                       .  |              |       .  |
      |                       .  +--------------+       .  |
      |                       ............................  |
      |                                     /\             |
      |                                     ||             |
      +-------------------------------------||-------------+
                                            ||
                                            ||
                                            ||
      +-------------------------------------||-------------+
      | Transport Stratum                   ||             |
      |                                     \/             |
      |                       ............................  |
      |                       . SDN Controller           .  |
      |                       .                          .  |
      |  +--------------+     .        +--------------+ .  |
      |  | Resource Pl. |     .        |  Mngmt. Pl.  | .  |
      |  |              |<===>.  +--------------+     | .  |
      |  |              |     .  |  Control Pl. |     | .  |
      |  +--------------+     .  |              |-----+ .  |
      |                       .  |              |       .  |
      |                       .  +--------------+       .  |
      |                       ............................  |
      |                                                     |
      |                                                     |
      +-----------------------------------------------------+
```

Figure 1: Cooperating Layered Architecture for SDN

In the CLAS architecture both the control and management functions are the ones logically centralized in one or a set of SDN controllers, in such a way that separated SDN controllers are present in the Service and Transport strata. Furthermore, the generic user or data plane functions included in the NGN architecture are referred here as resource plane functions. The resource plane in each stratum is controlled by the corresponding SDN controller through a standard interface.

The SDN controllers cooperate for the provision and delivery of services. There is a hierarchy in which the Service SDN controller requests transport capabilities to the Transport SDN controller. Furthermore, the Transport SDN controller interacts with the Service SDN controller to inform it about events in the transport network that can motivate actions in the service layer.

The Service SDN controller acts as a client of the Transport SDN controller.

Despite it is not shown in the figure, the Resource planes of each stratum could be connected. This will depend on the kind of service provided. Furthermore, the Service stratum could offer an interface towards external applications to expose network service capabilities to those applications or customers.

This document does assume that SDN techniques can be enabled jointly with other distributed means (e.g., IGP).

3.1. Functional strata

As described before, the functional split separates transport-related functions from service-related functions. Both strata cooperate for a consistent service delivery.

Consistecy is determined and characterized by the service layer.

Communication between these two components could be implemented using a variety of means (such as [I-D.boucadair-connectivity-provisioning-protocol], Intermediate-Controller Plane Interface (I-CPI) [ONFArch], etc).

3.1.1. Transport stratum

The Transport stratum comprises the functions focused on the transfer of data between the communication end points (e.g., between end-user devices, between two service gateways, etc.). The data forwarding nodes are controlled and managed by the Transport SDN component. The Control plane in the SDN controller is in charge of instructing the

forwarding devices to build the end to end data path for each communication or to make sure forwarding service is appropriately setup. Forwarding may not be rely on the sole pre-configured entries; dynamic means can be enabled so that involved nodes can build dynamically routing and forwarding paths. Finally, the Management plane performs management functions (i.e., FCAPS) on those devices, like fault or performance management, as part of the Transport stratum capabilities.

3.1.2. Service stratum

The Service stratum contains the functions related to the provision of services and the capabilities offered to external applications. The Resource plane consists of the resources involved in the service delivery, such as computing resources, registries, databases, etc. The Control plane is in charge of controlling and configuring those resources, as well as interacting with the Control plane of the Transport stratum in client mode for requesting transport capabilities for a given service. In the same way, the Management plane implements management actions on the service-related resources and interacts with the Management plane in the Transport stratum for a cooperating management between layers.

3.1.3. Recursiveness

Recursive layering can happen in some usage scenarios in which the Transport Stratum is itself structured in Service and Transport Stratum. This could be the case of the provision of a transport services complemented with advanced capabilities additional to the pure data transport (e.g., maintenance of a given SLA [RFC7297]).

3.2. Plane separation

The CLAS architecture leverages on the SDN proposition of plane separation. As mentioned before, three different planes are considered for each stratum. The communication among these three planes (and with the corresponding plane in other strata) is based on open, standard interfaces.

3.2.1. Control Plane

The Control plane logically centralizes the control functions of each stratum and directly controls the corresponding resources. [RFC7426] introduces the role of the control plane in a SDN architecture. This plane is part of an SDN controller, and can interact with other control planes in the same or different strata for accomplishing control functions.

3.2.2. Management Plane

The Management plane logically centralizes the management functions for each stratum, including the management of the Control and Resource planes. [RFC7426] describes the functions of the management plane in a SDN environment. This plane is also part of the SDN controller, and can interact with the corresponding management planes residing in SDN controllers of the same or different strata.

3.2.3. Resource Plane

The Resource plane comprises the resources for either the transport or the service functions. In some cases the service resources can be connected to the transport ones (e.g., being the terminating points of a transport function) whereas in other cases it can be decoupled from the transport resources (e.g., one database keeping some register for the end user). Both forwarding and operational planes proposed in [RFC7426] would be part of the Resource plane in this architecture.

4. Deployment scenarios

Different situations can be found depending on the characteristics of the networks involved in a given deployment.

4.1. Full SDN environments

This case considers the fact that the networks involved in the provision and delivery of a given service have SDN capabilities.

4.1.1. Multiple Service strata associated to a single Transport stratum

A single Transport stratum can provide transfer functions to more than one Service strata. The Transport stratum offers a standard interface to each of the Service strata. The Service strata are the clients of the Transport stratum. Some of the capabilities offered by the Transport stratum can be isolation of the transport resources (slicing), independent routing, etc.

4.1.2. Single service stratum associated to multiple Transport strata

A single Service stratum can make use of different Transport strata for the provision of a certain service. The Service stratum interfaces each of the Transport strata with standard protocols, and orchestrates the provided transfer capabilities for building the end to end transport needs.

4.2. Hybrid environments

This case considers scenarios where one of the strata is legacy totally or in part.

4.2.1. SDN Service stratum associated to a legacy Transport stratum

An SDN service stratum can interact with a legacy Transport stratum through some interworking function able to adapt SDN-based control and management service-related commands to legacy transport-related protocols, as expected by the legacy Transport stratum. The SDN controller in the Service stratum is not aware of the legacy nature of the underlying Transport stratum.

4.2.2. Legacy Service stratum associated to an SDN Transport stratum

A legacy Service stratum can work with an SDN-enabled Transport stratum through the mediation of and interworking function capable to interpret commands from the legacy service functions and translate them into SDN protocols for operating with the SDN-enabled Transport stratum.

4.3. Multi-domain scenarios in Transport Stratum

The Transport Stratum can be composed by transport resources being part of different administrative, topological or technological domains. The Service Stratum can yet interact with a single entity in the Transport Stratum in case some abstraction capabilities are provided in the transport part to emulate a single stratum.

Those abstraction capabilities constitute a service itself offered by the Transport Stratum to the services making use of it. This service is focused on the provision of transport capabilities, then different of the final communication service using such capabilities.

In this particular case this recursion allows multi-domain scenarios at transport level.

5. Use cases

This section presents a number of use cases as examples of applicability of this proposal

5.1. Network Function Virtualization

To be completed.

5.2. Abstraction and Control of Transport Networks

To be completed.

6. IANA Considerations

TBD.

7. Security Considerations

TBD. Security in the communication between strata to be addressed.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[Y.2011] "General principles and general reference model for Next Generation Networks", ITU-T Recommendation Y.2011 , October 2004.

8.2. Informative References

[I-D.boucadair-connectivity-provisioning-protocol] Boucadair, M., Jacquenet, C., Zhang, D., and P. Georgatsos, "Connectivity Provisioning Negotiation Protocol (CPNP)", draft-boucadair-connectivity-provisioning-protocol-08 (work in progress), September 2014.

[ONFArch] Open Networking Foundation, "SDN Architecture, Issue 1", June 2014, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf>.

[RFC7297] Boucadair, M., Jacquenet, C., and N. Wang, "IP Connectivity Provisioning Profile (CPP)", RFC 7297, July 2014.

[RFC7426] Haleplidis, E., Pentikousis, K., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, January 2015.

Authors' Addresses

Luis M. Contreras
Telefonica I+D
Ronda de la Comunicacion, s/n
Sur-3 building, 3rd floor
Madrid 28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com
URI: http://people.tid.es/LuisM.Contreras/

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Phone: +34 91624 6236
Email: cjbc@it.uc3m.es
URI: http://www.it.uc3m.es/cjbc/

Diego R. Lopez
Telefonica I+D
Ronda de la Comunicacion, s/n
Sur-3 building, 3rd floor
Madrid 28050
Spain

Email: diego.r.lopez@telefonica.com

Mohamed Boucadair
France Telecom
Rennes 35000
France

Email: mohamed.boucadair@orange.com

SDNRG
Internet-Draft
Intended status: Informational
Expires: September 9, 2015

R. Gu, Ed.
C. Li
China Mobile
March 8, 2015

SDN Controller Requirement
draft-gu-sdnrg-sdn-controller-requirement-00

Abstract

The requirements of SDN controllers including fundamental technical requirements, requirements of the SDN controller architecture and the requirements of the SDN controller functionality are provided. All these requirements raised are focused on the scalability, reliability, programmability, intercommunity, security and the network management of the SDN controller.

Status of This Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 9, 2015.

Table of Contents

1. Introduction

Software-defined networking (SDN) is an intelligent network, especially used in Data Centers, with configuration and operation through a centralized software controller. SDN controller is a core entity of the SDN architecture indicating how the network behaves and where the traffic is sent. Network intelligence is logically centralized in software-based SDN controllers that maintain an abstract view of the network, which appears to applications and policy engines as a single, logical switch.

Due to the importance of SDN controllers to the SDN architecture, the requirements of SDN controllers should be come up with. The requirements are divided into three parts: fundamental technical requirements, requirements of the SDN controller architecture and the requirements of the SDN controller functionality.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Fundamental technical requirements of SDN controllers

The fundamental technical requirements include scalability, reliability, programmability, intercommunity, security, and the network-based management.

Scalability:

SDN controller should meet the requirement of scalability in order to adapt the changes and adjustments of the network. The computing and controlling ability can be extended as the performance of hardware increases.

Reliability:

SDN controller should meet the carrier-level requirement with rapid fail-over mechanism.

Programmability:

SDN controller should offer APIs in order to provide rapid deployment of new service through executing scripts such as Python and Java or loading third-party module dynamically.

Intercommunity:

One SDN controller should support standard protocols in interacting with other SDN controllers or with traditional network.

Security:

SDN controller should qualify the security requirements including the communication security between the controllers and the switches, the access control security of controllers and switches, TLS and IPsec mechanism of the communication channels, DoS attacks prevention, digital certificate of third-party support.

Network-based management:

SDN controller should provide tools for basic network management and trouble diagnosis, such as secure access, status report, statistics, forwarding operations and so on.

4. Requirements of the SDN controller architecture

SDN controller should support both traditional distributed forwarding and centralized forwarding based on openflow. SDN controller interacts with switch through southbound interface.

SDN controller is logically divided into several models, including subsystem of protocol, forwarding abstraction layer (FAL), topology management, route management, host management, flow table management, interface management, database management, OAM interface management and inter-application subsystems.

```
 ----------------------------------------------------------------------
|    |------------------|             |------------------|           |
|    |   Orchestrator   |             |        EPC       |           |
|    |------------------|             |------------------|           |
|   External application layer                                        |
```

```
  -----+---------------------------+-----------------------------------
       |                           |
       |                           |
       |  -------------------------+-------------------------
       |  |         |------------------|  |--------------|  |
       |  |         | L2/L3 forwarding |  |  ARP reply   |  |
       |  |         |------------------|  |--------------|  |
       |  |                                                 |
       |  |         |-----|   |-----|  |-----| |-----|      |
       |  |         | BGP |   | IGP |  |  TE | | ... |      |
       |  |         |-----|   |-----|  |-----| |-----|      |
       |  | Internal application layer                     |
       |  -------------------------------------------------
       |
-------+-------------------------------------------------------------------
| -------- ------------------------------------------------------ -------- |
| |      | |                                                    | |      | |
| |      | |                      Route management               | |      | |
| |      | |                                                    | |      | |
| |      | -----------------------------------------------------| |      | |
| |      |                                                      | |      | |
| |      | |-----------||-----------||-----------||-----------| | |      | |
| |      | | Topology  ||    Host   || Flow table|| Interface | | |      | |
| |      | | Management|| Management|| Management|| Management| | |      | |
| |      | |-----------||-----------||-----------||-----------| | |      | |
| |      |                                                      | |      | |
| |  DB  | ----------------------------------------------------- | OAM  | |
| |subsys| |          Forwarding abstraction layer              | |manage| |
| |      | ----------------------------------------------------- | ment | |
| |      | ----------------------------------------------------- | |      | |
| |      | |     Protocol subsystem                            | |      | |
| |      | |                                                   | |      | |
| |      | | ------------ ------------  ----------  --------- | |      | |
| |      | | | Openflow | | OF-Config|  | BGP-LS |  |  XMPP | | |      | |
| |      | | ------------ ------------  ----------  --------- | |      | |
| |      | | ------------ ------------  ----------            | |      | |
| |      | | |   OVSDB  | | Netconf  |  |   ...  |            | |      | |
| |      | | ------------ ------------  ----------            | |      | |
| |      | ----------------------------------------------------- | |      | |
| -------- ------------------------------------------------------ -------- |
-------------------------------------------------------------------------
```

Figure 1: Sample Calibration Permutation

Protocol subsystem:

The protocol subsystem of the SDN controller focuses on southbound interface with protocols such as openflow, OF-Config, BGP-LS, OVSDB, Netconf, XMPP and so on.

Forwarding abstraction layer (FAL):

FAL translates the different forwarding plane into the unified interface upside in order to realize the abstraction of SDN controller node.

Topology management:

Topology is calculated through the status of port reported by the switch with the protocol such as LLDP, BGP-LS and so on. Logical networks are supported by SDN controller. Physical network can be divided into several logical networks with physical port and host corresponding to the virtual networks.

Route management:

Centralized computing of every virtual network is supported by controller. Forwarding path is calculated according to the ability of switch and the constraint conditions such as link cost, and bandwidth and network information.

Host management:

Host management takes the function of MAC and ARP learning. Host position and ARP information is recorded and aging at a certain time.

Flow table management:

Basic functions such as forwarding table storage, routing coalescence and re-forwarding are realized by the flow table management. It’s suggested that both distributed and centralized forwarding models are supported.

Interface management:

Interface configurations are maintained in the interface management, including dynamic and static interface configuration information. Virtual ARP table is also generated in the interface management model.

Database management:

Forwarding table and openflow table are managed in the database management with data synchronization.

OAM interface management:

Configuration command of command-line terminal and visualized network management server is written into database. Management interface is provided.

Inter-application subsystem:

Inter-application subsystem supports the interface to openstack and cloud platform by restful. Layer 2 and Layer 3 forwarding, traffic engineering, and ARP reply features are equipped. IGP/BGP protocols are supported.

5. Requirements of the SDN controller functionality

Due to the fundamental techinical requirements of SDN controllers, the follow functionality aspects need to be considered.

1. Requirement of multi-tenants and self-service

Multi-tenants with their self-service are typical scenarios of SDN. Multi tenants are existed in data centers with several virtual networks per tenant. IP address pool is allocated in every virtual network. Virtual network is logically isolated with each other. Same IP addresses can be assigned to different tenants. Virtual routers are used in different virtual network communications.

2. Requirement of network function

Basic network functions SDN controller needs to support list as follows.

(a) The number of tenants should be over 4000 by tunneling technique.

(b)Virtual machines in one subnet can communicate with each other by unicast of layer 2.

(c) Virtual machines in different subnets can’t communicate with each other.

(d)Virtual machines in different subnets can communicate with other by configuring a virtual router.

(e)Virtual machine can access to the network by assigning a public IP address.

(f)Tenants can translate private IP address into public IP address by NAT.

(g)Different tenants can use the same IP address and VLAN ID.

(h)Network can be recovered rapidly when fails.

(i)ARP Broadcast storm should be suppressed.

(j) Equal-Cost Load Sharing is supported in both underlay and overlay networks.

(k)Traditional protocols such as IGP , BGP and others are supported.

3. Requirement of administrator features

Administrators are responsible for tenants creation and deletion, network creation and deletion, unbinding the relation between tenants and network, query for tenants' information, query for physical and virtual information, virtual machine immigration and so on.

4. Requirement of network management

The information of switches, hosts and network topologies can be queried by management. Monitoring on network traffic is supported by network management. Network management is also responsible for network policies release and flow table configuration.

5. Requirement of reliability and scalability

Reliability of SDN controller relies on active-standby mode by controller node, secure connection between controller and switch nodes, multi-controllers based on openflow and so on. Scalability of SDN controller relies on node upgrading without service interruption and unique node upgrade in the distribute systems without any influence on the whole system.

6. Requirement of performance

Performance of SDN controller is reflected in the number of forwarding nodes supported per controller node, the capacity of flow table per controller node, speed of forwarding table processing per node and standby time of controller node.

7. Requirement of northbound and southbound interface

The northbound interface of the SDN controller is to achieve the requirement of the administrators and network management. While the southbound interface of the SDN controller is including the interface of status/configuration information such as OVSDB, OF-Config, XMPP

and the interface of routing/forwarding information such as Openflow, XMPP, IGP, BGP and so on.

8. Requirement of processing flow

The process of packet-forwarding network networks added or modified, physical network topology discovered and network failure advertised should be required.

6. Conclusion

All the requirements provided above are recommended to be taken into consideration for the SDN controllers.

7. Security Considerations

None.

8. IANA Considerations

None.

9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

Authors' Addresses

Rong Gu (editor)
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing 100053
China

Email: gurong_cmcc@outlook.com

Chen Li
China Mobile
32 Xuanwumen West Ave, Xicheng District
Beijing 100053
China

Email: lichenyj@chinamobile.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 10, 2015

Y. Tsuzaki
Kyoto University
R. Atarashi
IIJ Innovation Institute Inc.
S. Suzuki
Keio University
K. Mitsuya
K. Okada
Lepidum Co. Ltd.
March 09, 2015

Network configuration Web API for Bandwidth Reservation
draft-tsuzaki-netconfig-webapi-00.txt

Abstract

This draft introduces a framework for a dynamic bandwidth reservation via Web API for Web applications. In this document, we propose Web APIs for Web clients to request bandwidth allocation to network controllers. The network controller could be both of SDN compliant or Non-SDN compliant one. In this document, a network specification definition language is also proposed.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 10, 2015.

Table of Contents

1. Introduction

This draft proposes a framework for a dynamic bandwidth reservation via Web API for Web applications. We assume that there are network controllers to control the network devices and gather information about their control domain. Those controllers equip Web APIs so that Web clients can request bandwidth allocated virtual private paths between contents Web servers and the clients. Network administrators describe the service specifications with "service description language", and the bandwidth are allocated to the clients according to the service specifications. This draft explains the overview of this architecture and how resource reservations are made.

2. Requirement

- o From the Viewpoint of Network Administrators
  Based on the service specifications configured by the administrators, network management controllers automatically respond to the client requests via Web APIs.

- o From the Viewpoint of Clients
  By accessing the Web API for the network resource reservations, clients can reserve QoS guaranteed communication bandwidth for Web contents downloads.

- o Use Case
  The network administrators prepare Web APIs for configuring network paths and bandwidth reservations. When a client need to download large contents from a Web server, the client send the requiring resource information to the network management server via Web APIs. The network management server construct a QoS guaranteed communication path for the client based on the information received from the client.

3. Terminology

- o Management Server: Servers which control the network devices in a domain. These servers also provide the application interfaces for Media Clients to signal resource requests. Administrators describe the network configurations and policies of networks by SDL/NDL and put them to Management Servers. Management servers are also referred as Network Management Servers.

- o Media server: Kind of a web server, which delivers media contents to Media Clients.

- o Media client: Client application run on a Web Browser, which receives and present media contents to an end user.

- o Service specification: the description of network service components described by SDL/NDL

- o Service Description Language (SDL): A language by which administrators describes network device information. Administrators describe SDL and put the descriptions to Management Servers.

- o Network Description Language (NDL): A language by which administrators describes network service information. Administrators describe NDL and put the descriptions to Management Servers.

o Resource request: action by which Media Clients obtain resource reserved communication path to Media Servers.

4. System architecture

```
                   +----------------+
                   |   Management   |
         Resource  |     server     |
         Request  +-+---+-------+--+-+
         +------> | WEB API |   |  |
         |        +---------+   |  |  Path
         |                      |  |  Setup
         |                      |  |
         |                 , - \+  +/- ,
         |              , '      \  /     ' ,
         |            ,           \/          ,
    +----+---+   +-------------------------------+   +--------+
    |  Media |               Reserved                |  Media |
    | client |                 Path                  | server |
    +--------+   +-------------------------------+   +--------+
                    ,        SDN/Non-SDN      ,
                     ,        BackBone       ,
                       ,                  , ,
                         ' + , _ _ _ ,   '
```

Figure 1: System Architecture

Figure 1 depicts the system overview of application triggered resource reservation architecture. All the network components except for the end clients in the domain (routers, servers) are under the control of the management server. The network management server gathers network management information such as status of network devices or links in the network, and also command those devices to set up QoS guaranteed communication paths between Media Servers and Media Clients.

The scope of this architecture is to define Web interfaces to signal resource allocation from Web browser applications to management servers.

4.1. Management server

Management servers are servers that control network components on the networks. Network administrators describe network device groups and network service description language with language called "service definition language". Service definition language is detailed in Section 5.1.

Management servers serves Web APIs for clients to make resource reservations. To trigger network resource reservations, Media Clients access these Web APIs on the management servers. Upon receiving requests from Media Clients, a management server calculate appropriate communication paths between Media Servers and Media Clients. The intermediate nodes (routers or switches) can be both of SDN compliant and Non-SDN compliant devices, but each of those devices have to be configurable by the management server via some remote configuration methods such as Netconf[RFC6241] or SSH.

4.2. Media client

Media client is a client application program run on a Web browser, which receives and present media contents to an end user. Media client receives Media Program, which is a list of contents can be presented, from a Program Server. When the user selects a content from the presented list of contents, Media client start playing the content.

4.3. Program Server

Program Server store and provides a Media Program, which is a list of available contents. We use HTTP to provide a Media Program to a Media Client.

The content specified in the Media Program consists of the title of the content and URL of the content. We expect content URL point to a location of a MEPGDash[MPEGDASH] file. The Media Program can be generated either by statically or dynamically.

At this moment, we do not define how Media client finds a Program Server. We assume this information is already available in Media client.

4.4. Media Server

Media server is a server program which store and provide metadata of a program as a MPEGDash format, and the contents of each media referenced from the MPEGDash formatted metadata.

Contents can be split into multiple segments by duration, or prepared in multiple bit rates.

Since the links between Media Program, MPEGDash file and segmented contents are described as a URL, all types of contents can store on one Media Server or among multiple Media Servers.

4.5. System components

```
+----------------+        +--------------+
|Media Client    |        |Program Server|
+----------------+        +--------------+
| +------------+ |        | +---------+  |
| |  Program   +----------> | Program |  |
| |Information | |        | |   List  |  |
| |  Manager   +--------+ | +---------+  |
| +--+---------+ |      | +--------------+
|    |           |      |
|    v           |      | +--------------+
| +------------+ |      | |Media Server  |
| |  Resource  | |      | +--------------+
| |  Manager   +----+   | | +----------+ |
| +--+---------+ |  |   +-> |  Media   | |
+----|------^----+  +-----> | Contents | |
     |      |             | +----------+ |
     |      +-------+     +--------------+
     |              |
+----|--------------|--------------------+
|    |              |  Management Server |
+----v--------------|--------------------+
|  +------+       +-+-------+            |
|  | Web  |       | Client  |            |
|  | APIs |       | Manager |            |
|  +------+       +---------+            |
|        +---------+     ^               |
|                  v     |               |
|  +----------+   +------+-----+         |
|  | Topology +-->| Topology   |         |
|  | Database |   | Calculator |         |
|  +----------+   +------------+         |
+----------------------------------------+
```

Figure 2: System component

Figure 2 shows a simple component diagram of this architecture. When a Media Client starts to obtain media contents from Media Servers, the program information manager of the client first get the program list from program servers. The program lists are described in the media presentation description(MPD) format of MPEGDash. The resource manager then access to the resource usage request Web API on the management server. When received a request from a client, the management server calculate the allocatable bandwidth between the Media Client and the Media Server via the topology calculator. Then the client manager of the management server respond a resource usage report to the Media Client. Based on the information in the resource

usage report, the Media Client trigger resource allocation by accessing the resource request Web API on the management server. Then the management server allocate bandwidth to the client via the topology calculator and send success message back to the client.

5. API

5.1. Service definition language

Network administrators define the service specifications utilizing the service specification language, Network Description Language(NDL) and Service Description Language(SDL). NDL is to define the group of network components such as router groups. With SDL, administrators can define service specifications on the network. Service specifications are the descriptions which define the relationship between network devices or network groups that compose network services. Examples of service definitions are network configurations such as segment IP address blocks or VLAN id for the segment. An example of NDL/SDL is shown in Figure 3 and Figure 4

```
node {
  ovs1;
  ovs2;
  media1;
  media2;
  pc11;
  pc12;
  pc13;
  pc14;
  pc21;
  pc22;
  pc23;
  pc24;
}
location {
  loc1 {
    media1;
    ovs1;
    pc11;
    pc12;
    pc13;
    pc14;
  }
  loc2 {
    media2;
    ovs2;
    pc21;
    pc22;
```

```
      pc23;
      pc24;
    }
  }
  group {
    group101 {
      media1;
      media2;
      pc11;
      pc12;
      pc13;
      pc14;
      pc21;
      pc22;
      pc23;
      pc24;
      ovs1;
      ovs2;
    }
    group1623 {
      ovs1;
      ovs2;
    }
    group1624 {
      ovs1;
      ovs2;
    }
    group1625 {
      ovs1;
      ovs2;
    }
  }
  link {
    type = layer1;
    edge1 = pc11;
    edge2 = pc12;
  }
```

Figure 3: Example of NDL

```
networks {
  network group101 {
    address = "192.168.1.0/24";
    vlan = 101;

    device ovs1 {
      type = L2Switch;
      address = "192.168.1.1";
    }

    device ovs2 {
      type = L2Switch;
      address = "192.168.1.2";
    }

    device media1 {
      type = Server;
      address = "192.168.1.30";
    }

    device media2 {
      type = Server;
      address = "102.168.1.31";
    }
  }
}
```

Figure 4: Example of SDL

SDL also enables registrations of events on the network and event bound actions. For example, if the traffics from certain source IP address exceeds the defined per-flow bandwidth limitation on the certain physical link, the traffic can be automatically shaped according to the definitions of SDL. Administrators define resource usage limitation using this functionality of SDL. For example, administrators can limit the usage of bandwidth per the domain to which user equipments attached. The bandwidth allocation for each user is determined based on these service specifications.

5.2. Web API

5.2.1. Resource usage report

Media servers advertise resource usage of links to Media Servers. The resource usage reports have two types. One is periodic resource usage reports broadcasted from management servers. Periodic usage reports include the uplink bandwidth usage of each servers(Figure 5). Another resource usage type is solicited usage report which is

delivered to clients through WebAPI on the management servers. In a solicited usage report request(Figure 6), a Media Client specify the server from which it want to download media contents. The Media Server which received the solicited usage reports calculates the physical link set which connect the client and the server, and report available bandwidth the management server afford to allocate to the client(Figure 7). If multiple paths between the client and the server exist, the max available bandwidth will be returned to the client. At a solicited resource usage report request, a Media Client opens a web socket to the management server.

```
{
  [
    {
      "server": <String>
      "resource": {
        "bandwidth": <Num> // Option
        "latency": <Num> // Option
      }
    },
    ...
  ]
}
```

Figure 5: Unsolicited resource usage report json format

o server: server IP address or FQDN in string

o resource: available resource of the server

```
{

  "from": <String>
  "to": <String>
}
```

Figure 6: Solicited resource usage report request json format

o from: from IP address or FQDN in string

o to: to IP address or FQDN in string

```
{

  "resource": {
    "bandwidth": <Num> // Option
    "latency": <Num> // Option
  }
}
```

Figure 7: Solicited resource usage report response json format

o resource: available resource of the server

5.2.2. Resource request

Media clients acquire reserved communication paths by accessing resource requests API on the management server. The resource requests have three types, initial resource request, resource modification request from clients and management server trigger resource modification request. We explain these types of resource requests in this section. According to the session_id information in the request, management server associate the web socket object of the request source client and the session-id.

The clients post json format requests on the reservation. Figure 8 is the format of the resource request json.

```
{
  "session_id": <String>
  "class": <Num>
  "type": <Num>
  "server": <String>
  "resource": {
    "bandwidth": <Num> // Option
    "latency": <Num> // Option
  }
}
```

Figure 8: Resource request json format

o session_id: random created UUID to identify the session

o class: user priority class in digit number

o type: 0: Initial 1: Modification

o server: the server to which the client willing to connect

o resource: resource object contains bandwidth and latency

5.2.2.1. Initial resource request

```
   +------+      +------------+      +------+
   |Media |      | Management |      |Media |
   |Client|      |   Server   |      |Server|
   +--+---+      +------------+      +------+
      |                                  |
      |                                  |

      |                 |                |
      |      RRRQ       |                |
      +---------------> |                |
      |                 +                |
      |                                  |
      |               RRRQR              |
      +--------------------------------> |
      |                                  |
      |                 +      RRRQ      |
      |                 | <--------------+
      |                 |                |
      |                 |      RRRS      |
      |                 +--------------> |
      |                 |                |
      |       RRRS      |                |
      | <---------------+                |
      |                 +                |
      |               RRRQS              |
      | <--------------------------------+
      |                                  |
      +                                  +
```

Figure 9: Initial resource request sequence

- o RURR: Resource Usage Report Request
- o RURA: Resource Usage Report Advertisement
- o RRRQ: Resource Reservation ReQuest
- o RRRS: Resource Reservation ReSponse
- o RRRQR: Resource Reservation ReQuest Request
- o RRRQS: Resource Reservation ReQuest Response

A Media Client initially obtains a contents list on the Media Server. This contents list is described in the media presentation description (MPD) format of MpegDash. The acquisition of contents list is done

by ordinal HTTP GET method. Then the client request resource usage reports to the management server as mentioned in Section 5.2.1. Based on information in the resource usage report and contents list, the client determine the contents bitrate and send a resource request to the management server based on the determined contents bitrate. The resource request contains a session-id randomly generated on clients(ex) UUID). The client simultaneously send a resource reservation request request to Media Server to trigger Media Server to send a request to the management server. The RRRQR also contains same session-id as resource request. The management server verify the request from the Media Server and the Media Client, and send response to both side if the information from the client and the server correspond. The management server stores the session-id, web socket information and allocated resources. These information are used for resource modifications and keep-alive. After received RRRS indicating the resource reservation was done successfully, the Media Server send RRRQS to the Media Client. Then the client get to be able to download the media contents with guaranteed quality.

5.2.2.2. Client trigger resource modification request

A Media Client MAY offer resource modification requests when resource usage reports say the uplink capacity of the Media Server from which the client downloads the media contents.

```
   +------+      +------------+      +------+
   |Media |      | Management |      |Media |
   |Client|      |   Server   |      |Server|
   +--+---+      +------------+      +------+
      |                                  |
      |      RRRQ       +                |
      +---------------> |                |
      |                 +                |
      |                                  |
      |               RRRQR              |
      +--------------------------------> |
      |                                  |
      |                 +      RRRQ      |
      |                 | <--------------+
      |                 |                |
      |                 |      RRRS      |
      |                 +--------------> |
      |                 |                |
      |       RRRS      |                |
      | <---------------+                |
      |                 +                |
      |               RRRQS              |
      | <--------------------------------+
      |                                  |
      +                                  +
```

Figure 10: Resource modification sequence (client trigger)

5.2.2.3. Management server trigger resource modification request

```
   +------+      +------------+      +------+
   |Media |      | Management |      |Media |
   |Client|      |   Server   |      |Server|
   +--+---+      +------------+      +------+
      |                                  |
      |      RRMRQ      +                |
      | <---------------+                |
      |                 |                |
      |      RRRQ       |                |
      +---------------> |                |
      |                 +                |
      |                                  |
      |               RRRQR              |
      +----------------------------------> |
      |                                  |
      |                 +      RRRQ      |
      |                 | <--------------+
      |                 |                |
      |                 |      RRRS      |
      |                 +--------------> |
      |                 |                |
      |       RRRS      |                |
      | <---------------+                |
      |                 +                |
      |               RRRQS              |
      | <--------------------------------+
      |                                  |
      +                                  +
```

Figure 11: Resource modification sequence (management server trigger)

RRMRQ: Resource reservation modification request.

Management servers MAY trigger resource downgrade/upgrade requests to Media Clients, when the used bandwidth of a certain link saturate or become to have room. This push messaging can be done by Web sockets or WebRTC. As resource reservation modification request contains available resource for the received client, client determine the contents bitrate based on the information contained in RRMRQ and pre-downloaded contents list information at the initial resource reservation. The following process is similar to the initial resource reservation described in Section 5.2.2.1.

5.2.2.4. Resource cancellation

When a client do not need the allocated resources, the client can explicitly stop using the resource by posting a json described in Figure 12. Upon receiving cancellation message, the management

server disassociate session-id from the client websocket, and release the resource bound to the session-id.

```
{
  "session_id": <String>
}
```

Figure 12: Resource cancel json format

5.2.3. Keep-alive

Management servers MAY keep-alive the clients to keep monitoring the usage of the reserved resources. While the clients can send resource free messages explicitly at the end of media streaming, client computers tend to disconnect from the networks suddenly or the browser applications can be reloaded by user operations. To avoid such kind of wasted resources, management servers send keep-alive messages include the session-ids sent from the clients at the initial resource reservations. When received a keep-alive message, the client verify the session-id contained in the keep-alive message. If the keep-alive message is not the one the client stores, the client ignore the keep-alive messages. If the server do not receive the keep-alive responses from the client for certain configured times, the server free the resource bound to the session-id. By default, the keep-alive interval is 300 seconds and the default keep-alive timeout count is 3.

6. Security Considerations

TBD

7. IANA Considerations

TBD

8. Acknowledgement

The author would like to thank Yasuo Okabe, Osamu Nakamura and Kaoru Maeda for their good contributions to this document.

9. References

9.1. Normative References

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

9.2. Informative References

[MPEGDASH] "ISO/IEC 23009-1:2014: Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats", May 2014, <http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html>.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

Authors' Addresses

Yoshiharu Tsuzaki
Kyoto University

Email: tsuzakiyo@net.ist.i.kyoto-u.ac.jp

Ray Atarashi
IIJ Innovation Institute Inc.

Email: ray@iijlab.net

Shigeya Suzuki
Keio University

Email: shigeya@wide.ad.jp

Koshiro Mitsuya
Lepidum Co. Ltd.

Email: mitsuya@lepidum.co.jp

Kouji Okada
Lepidum Co. Ltd.

Email: okd@lepidum.co.jp