Computer Networks 68 (2014) 166–186

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Application delivery in multi-cloud environments using software defined networking

Subharthi Paul[a,*,1], Raj Jain[a,1], Mohammed Samaka[b,2], Jianli Pan[a,1]

[a] Washington University in St. Louis, School of Engineering & Applied Science, Department of Computer Science & Engineering, Bryan Hall, CB 1045, 1 Brookings Drive, Saint Louis, MO 63130, USA

[b] University of Qatar, College of Engineering, Computer Science and Engineering Department, P.O. Box 2713, Doha, Qatar

ARTICLE INFO

Article history:
Received 5 June 2013
Received in revised form 4 November 2013
Accepted 4 December 2013
Available online 22 February 2014

Keywords:
Application delivery
Layer 7 traffic steering
Cloud computing
Software defined networks
Service centric networking
Middleboxes

ABSTRACT

Today, most large Application Service Providers (ASPs) such as Google, Microsoft, Yahoo, Amazon and Facebook operate multiple geographically distributed datacenters, serving a global user population that are often mobile. However, the *service-centric* deployment and delivery semantics of these modern Internet-scale applications do not fit naturally into the Internet's host-centric design. In this service-centric model, users connect to a *service*, and not a particular *host*. A *service* virtualizes the application endpoint, and could be replicated, partitioned, distributed and composed over many different hosts in many different locations. To address this gap between design and use, ASPs deploy a *service-centric* network infrastructure within their enterprise datacenter environments while maintaining a (virtual) host-centric service access interface with the rest-of-the-Internet. This is done using data-plane mechanisms including *data-plane proxying* (virtualizing the service endpoint) and *Layer 7 (L7) traffic steering* (dynamically mapping service requests to different application servers and orchestrating service composition and chaining). However, deploying and managing a wide-area distributed infrastructure providing these service-centric mechanisms to support multi-data center environments is prohibitively expensive and difficult even for the largest of ASPs. Therefore, although recent advances in cloud computing make distributed computing resources easily available to smaller ASPs on a very flexible and dynamic pay-as-you-go resource-leasing model, it is difficult for these ASPs to leverage the opportunities provided by such multi-cloud environments without general architectural support for a *service-centric Internet.* In this paper, we present a new service-centric networking architecture for the current Internet called OpenADN. OpenADN will allow ASPs to be able to fully leverage multi-cloud environments for deploying and delivering their applications over a shared, service-centric, wide-area network infrastructure provided by third-party providers including Internet Service Providers (ISPs), Cloud Service Providers (CSPs) and Content Delivery Networks (CDNs). The OpenADN design leverages the recently proposed framework of Software Defined Networking (SDN) to implement and manage the deployment of OpenADN-aware devices. This paper focuses mostly on the data-plane design of OpenADN.

* Corresponding author. Tel.: +1 (773) 679 7723.
E-mail address: spaul@wustl.edu (S. Paul).
[1] Tel: +(314) 935 6160, Fax: +(314) 935 7302.
[2] Tel: +974 4403 4240, Fax: +974 4403 4241.

http://dx.doi.org/10.1016/j.comnet.2013.12.005
1389-1286/© 2014 Elsevier B.V. All rights reserved.

1. Introduction

In this paper, we address the problem of application delivery in a multi-cloud environment. All enterprises, including banks, retailers, social networking sites like Facebook, face this problem. The enterprises are what we call Application Service Providers (ASPs) that want to deploy, manage, and delivery their applications over third party computing infrastructure leased from multiple Cloud Service Providers (CSPs) located all over the world. The Internet Service Providers (ISPs) provide the network connecting the users, ASPs, and CSPs. We describe the problem in two steps. First, we explain how the problem is handled in a private data center and then how it becomes more complicated with the emergence of cloud computing.

1.1. Service partitioning in a datacenter

The Internet's host-to-host application delivery model does not naturally map to the *service-centric* deployment and delivery semantics of modern Internet-scale applications. In this service-centric model, users connect to a *service,* and not a particular *host*. A *service* virtualizes the application endpoint, and could be replicated, partitioned, distributed and composed over many different hosts. However, in-spite of this, the Internet has successfully sustained the proliferation of services, without any fundamental changes to its architecture. This has been achieved by introducing *service-centric* deployment and delivery techniques within the enterprise infrastructure boundaries while maintaining a standard *host-centric* application access interface with the rest of the Internet. Initially, load balancers were introduced to support ***service replication*** over multiple hosts. A load balancer is a data plane entity that intercepts service requests in the data plane and dynamically *maps* it to the least loaded server that can serve the request. Although interposed in the application's data plane, the load balancer actually makes its mapping decision only on the end-to-end *connection setup* control messages (e.g., TCP SYN, flow setup over UDP, etc.) that are exchanged between the two end-hosts (in the host-centric design) to setup their transport/application layer connection state. Therefore, since the underlying communication protocols in the data plane are still host-centric, the load balancer needs to masquerade itself in the middle, using network address translation (NAT). A number of other DNS (domain name system) related techniques can be used to allow ***service replication***. However, what is more difficult is ***service partitioning***. Service partitioning is required to optimize the performance of a service by partitioning a monolithic service into smaller service components. Services may be partitioned based on:

Application content: For example, hosting video, images, user profile data and accounting data for the same service on separate server groups.
Application context: For example, different service API (Application Programming Interface) calls requiring different processing time and resources sent to different server groups.
User context: e.g., service access from different user devices (smart phones, laptops, and desktops), user access network capability (wired vs. wireless) being served differently.

Service partitioning improves performance, resilience, and also enables more efficient use of computing resources by allowing each service partition to scale independently through replication. For example, certain partitions may need to be replicated more than others owing to their higher resource requirements, criticality, popularity, etc. Data-plane proxying and L7 (Layer 7) traffic steering techniques are used to support service partitioning within enterprise environments. Data-plane proxying, unlike a network-layer load balancer, terminates L4 (TCP) connections and acts as a virtual application endpoint, thus allowing the service to maintain a standard host-centric interface for service access. The data-plane proxy then applies L7 traffic steering policies on each application message and independently routes them to a different application servers hosting the different service partition.

Also, in modern enterprise environments, different L3–L7 middleboxes provide several application delivery functions including message/packet filtering and monitoring (firewalls, intrusion detection, access control), message/packet transformation (transcoders, compression, SSL offload, protocol gateways) and application delivery optimization (WAN optimizers, content caches). Some of these services are message-level (L4–L7) and hence need to be implemented as data-plane proxies, whereas others are packet-level and may be deployed as network services. Different messages for the **same** service may need to be steered differently through a different set of middleboxes depending on the message context. Therefore, approaches that pre-establish an end-middle-end path through explicit control plane signaling such as in [1] are not sufficient. Instead a more dynamic approach of L7 traffic steering is required that enables *context-aware* service chaining where each message may be directed through a different sequence of middlebox services.

Therefore, data-plane proxying and L7 traffic steering, which we generically refer to as **Application Policy Routing** (APR), are the key techniques underlying intelligent enterprise fabrics that allow enterprises to support service-centric application deployments over a host-centric Internet design. Although not directly related to L7 traffic steering, but the application-level context needed to make APR decisions also informs the QoS requirements of delivering the message over the network. If the underlying network provides differentiated data transport services, such services may be invoked more efficiently (at per-message granularity rather than to all messages of the flow) and accurately.

1.2. Service partitioning in multi-cloud environments

Modern Internet-scale services are increasingly under pressure to move out of their centralized single datacenter environments to distributed multi-datacenter environ-