Move-with-the-Sun or Move-with-the-Moon?

Wide-Area CloudNet Migrations Under Latency and Resource Constraints

Johannes Grassler, Stefan Schmid, Anja Feldmann
Telekom Innovation Laboratories (T-Labs) & TU Berlin, D-10587 Berlin, Germany

Abstract—**This paper presents the architecture of our wide-area CloudNet migration demonstrator which supports two types of migrations: (1) A latency critical virtual video streaming service migrates according to the current user locations and access patterns ("move-with-the-sun"), e.g., to improve Quality-of-Service (QoS) parameters. (2) Latency uncritical CloudNets—virtual networks connecting cloud resources—are re-embedded at different locations in the substrate network, e.g., to make resources available for the latency critical applications ("move-with-the-moon").**

I. Introduction

The virtualization of resources and networks introduces a decoupling of services from the constraints of the underlying physical infrastructure, allowing for a more flexible and efficient allocation of resources: Resources can be allocated where it is most cost or performance effective, and may even be *migrated* over time and space. After node virtualization revolutionized the server business and heralded elastic computing services, we now observe an increasing interest in virtualized and software-defined *networking* (SDN), also in the *wide area* [4].

The vision of our *CloudNets* architecture is to provide a combined abstraction of both node and link virtualization by connecting cloud resources (e.g., storage and computational resources which are available at the PoPs of an ISP), with virtual networks. We have developed a federated prototype which supports different economical roles, from the Service Provider down to the Physical Infrastructure Provider, and allows to flexibly specify and allocate entire CloudNets. [2] In this paper, we focus on the *migration* support in the CloudNet prototype and describe the migration demonstrator: it adaptively migrates latency critical services closer to the users ("move with the sun"), e.g., to the ISP PoP with the lowest access latencies; on the other hand, to free up resources, it migrates latency uncritical CloudNets *away* from the users, e.g., to a location where resources or energy are cheap.

II. Service and CloudNet Migration

The migration demonstrator consists of two parts: (1) a *topology demonstrator* which demonstrates how independent CloudNets are embedded concurrently on a shared substrate network (e.g., an ISP network with global footprint), and (2) a *video demonstrator* which demonstrates how a migrating streaming server can improve access latency for the mobile terminals. The idea is that while the CloudNet can be placed anywhere in the substrate network (e.g., a computational CloudNet without many user interactions), the video server should always be close to the (dynamic or mobile) users. We set the resources (in our case: the RAM) in such a way that at any given time only one substrate node can accommodate the streaming server in addition to the different CloudNet nodes hosted on the shared substrate. Therefore, whenever the streaming server migrates (e.g., "move-with-the-sun"), parts of the other CloudNets need to be moved too ("with the moon").

The migrations of both demonstrators are *cost-aware*. The video demonstrator uses a "center-of-gravity" algorithm amortizing costs over time [1], and the topology demonstrator is based on optimal embeddings computed with Mixed Integer Programs (MIPs) [5].

Prototype and Wide-area Testbed. Our prototype and demo runs on a testbed that spans two geographically remote sites (the sites are connected by VPN tunnels), one at TU Berlin (the *Routerlab*) and one at NTT DoCoMo Eurolabs, Munich. Both testbeds have a Cisco 4500 series switch carrying both CloudNet *data plane* VLANs and *testbed management* VLANs and Sun X4150 machines hosting substrate resources and the virtual machines running physical infrastructure provider management software. Substrate nodes, management nodes and virtual nodes run Linux (`Ubuntu 8.04, Debian Wheezy`). Virtual nodes are *Xen* virtual machines running on the substrate nodes.

Both the topology and video demonstrator CloudNets are described in terms of our resource and network description language, [3] and sent to the physical infrastructure provider. It maps each CloudNet to its substrate using a MIP. Subsequently, the physical infrastructure provider creates virtual machines, establishes the links between them and hands control over to the customer (by providing console access to the virtual nodes).

Video Demonstrator. In the Video Demonstrator, the following challenges have to be addressed: (1) *Client connectivity:* Streaming clients outside the testbed (and hence outside the scope of CloudNet management) need to receive a video stream from a virtual node, the streaming server, inside the video demonstrator's CloudNet. (Currently, our prototype lacks a terminal attachment mechanism.) (2) *Latency augmentation:* The latency of local (connecting through the access point co-located with the `Server` virtual node) versus remote (connecting through a different access point) clients is too small to have a detectable (to the human eye) impact on a video stream. Hence the egress latency of links between substrate nodes is increased using a traffic engineering / buffering mechanism. (3) *Virtual node control:* Clients are enabled/disabled by controlling the stream on the `Server` virtual node. To this end we needed to develop a mechanism to non-interactively send commands to virtual nodes (the only out-of-VNet access to the prototype's virtual nodes is through a serial console). (4) *VNet negotiation/provisioning*: In order to be able to issue CloudNet requests through a web interface we developed a negotiation mechanism that allows for sending request graphs to the prototype's mapping engine without further manual intervention.

Concretely, the Video Demonstrator incorporates a range of backend scripts on the infrastructure provider, some code running on the streaming service virtual node, and a multi-layered VPN tunnel architecture to connect the streaming clients to the testbed. See Figure 1.

In the demonstrator, there are three *client machines*. Each client machine runs three VLC instances in UDP listening mode that represent the clients connected through the access point connecting this client machine. A client is associated with a unique client port, and the server runs a VLC instance for every client, each of which sends the same video stream to its assigned client. Clients are enabled/disabled on the streaming server by inserting and removing `iptables` rules that pass or drop UDP traffic to the client's port. This does not cause any disruption (such as clients timing out eventually) since VLC's streaming protocol is stateless.

Clients connect to the testbed via a OpenVPN `tap` style tunnel. This VPN tunnel provides layer 2 connectivity (i.e., it tunnels IEEE 802.3 frames over IP). It terminates in a tap device on each client machine and its counterpart on the `tbr1` (tunnel bridge) substrate node.

`tbr1` bridges the clients' `tap` devices onto its CloudNet data plane interface, `eth2`. Each client machine/access point pair is assigned a dedicated VLAN tag. This VLAN tag identifies the client machine's traffic

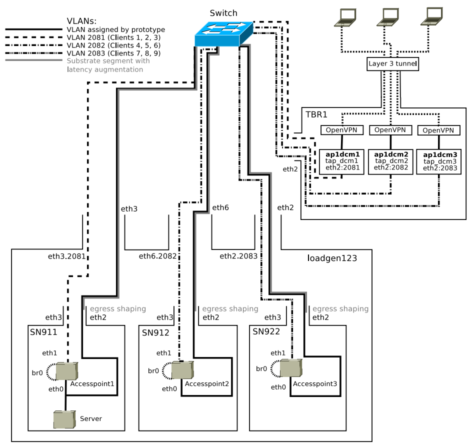

Fig. 1. Video Demonstrator overview schematic. An automatically assigned VLAN connects all three access points and the streaming server. Each triple of clients shares a dedicated VLAN. The client VLANs are bridged into the CloudNet by the access points. Note the egress shaped segments between the substrate nodes and the switch: video streams going to clients 4 through 9 will pass these links and is slowed down. Streams to clients 1 through 3 will be unaffected due to passing through the unshaped link between `accesspoint1` and `server`.

as it passes through the VNet data plane network from the access point to `tbr1`. Finally, a bridge device on each access point, interconnecting its `eth0` (CloudNet side) and `eth1` (`tbr1` side) bridges traffic from the virtual network to the streaming client VLAN.

When a client is added to an access point, the center of gravity of the demand is recomputed. If the newly computed center of gravity differs from the current one, a CloudNet request placing the `server` virtual node with the access point at the new center of gravity will be issued.

It will cause the mapping engine to replace the currently embedded video demonstrator CloudNet topology by one that co-locates the server with the new center of gravity (displacing topology demonstrator resources as required).

Acknowledgments: The authors would like to thank Gregor Schaffrath.

REFERENCES

[1] D. Arora et al. Online strategies for intra and inter provider service migration in virtual networks. In *Proc. IPTComm*, 2011.
[2] G. Schaffrath et al. Network virtualization architecture: Proposal and initial prototype. In *Proc. ACM SIGCOMM VISA*, 2009.
[3] G. Schaffrath et al. A resource description language with vagueness support for multi-provider cloud networks. In *Proc. ICCCN*, 2012.
[4] OpenFlowHub. SDN Use Case: Multipath TCP at Caltech and CERN. In *http://www.openflowhub.org/ (Dec 3)*, 2012.
[5] G. Schaffrath, S. Schmid, and A. Feldmann. Optimizing long-lived cloudnets with migrations. In *Proc. 5th IEEE/ACM UCC*, 2012.