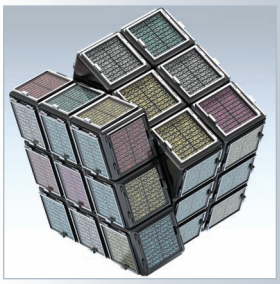

Virtual Slice Assignment in Large-Scale Cloud Interconnects

Software-defined networking is an emerging method for providing flexible and scalable network connectivity in both intra- and inter-datacenter interconnects with regard to various requirements, including traffic-awareness, quality of service, energy efficiency, and renewable-power intermittency. This article investigates issues and solutions for software-defined planning of virtual slices that involves multiple virtual machines with interdependent constraints spanning a network of distributed datacenters. A flexible and optimized virtual-slice assignment that considers server consolidation and multipath forwarding can address large-scale cloud computing services.

Kim-Khoa Nguyen and Mohamed Cheriet
École de Technologie Supérieure, University of Quebec

Yves Lemieux
Ericsson Research, Canada

Cloud computing is widely recognized as the key engine of next-generation Internet applications, which are characterized by mega-scale volumes of data. Today, large online service providers such as Google and Facebook routinely use datacenters for data warehousing, Internet search, and high-performance computing. Cloud hosting in datacenters is a rapidly growing industry that plays a crucial role in the future information and communications technology (ICT) sector. Modern datacenters deploy virtualization techniques to increase operational efficiency and enable dynamic resource provisioning in response to changing application needs.

In terms of network control, today's vertically integrated networking systems use a distributed control plane to manage each device and interface independently – that is, device by device.[1] Adding to this siloed architecture is a complex array of network protocols that creates unacceptable vendor lock-in. Bringing such legacy architecture to the cloud – which contains huge numbers of devices, subnetworks, multitenancy, and virtual machines (VMs) – presents scalability issues, exposing the architectural limits of networking systems and protocols. In addition, most current datacenter traffic consists of flows between adjacent servers, which is a very different traffic pattern from that in traditional networks.

To address these problems, researchers have proposed extensions to routing protocols and operational practices that prevent transient anomalies during changes. However, these solutions are

1089-7801/14/$31.00 © 2014 IEEE Published by the IEEE Computer Society

Related Work in Virtual Slice Assignment for Datacenters

Recent research in the field has made significant achievements in optimal virtual machine (VM) placement in datacenters with traffic-aware features,[1] and virtual network (both node and link mapping) embedding.[2] In particular, an implementation and deployment in the GENI network automatically coschedules and provisions heterogeneous networked resources.[3] Prior research and implementation models have presented VM consolidation algorithms in networks of datacenters (see http://greenstarnetwork.com). However, a problem combining virtual network mapping and VM consolidation has yet to be investigated. Moreover, no prior research has considered multipath forwarding in intra-datacenter interconnects, and scheduled forwarding in inter-datacenter interconnects.

References

1. X. Meng, V. Pappas, and L. Zhang, "Improving the Scalability of Data Center Networks with Traffic-Aware Virtual Machine Placement," *Proc. 29th Conf. Information Comm.*, 2010, pp. 1–9.
2. N.M.M.K. Chowdhury et al., "Virtual Network Embedding with Coordinated Node and Link Mapping," *Proc. IEEE INFOCOM 2009*, 2009, pp. 783–791.
3. I. Baldine et al., "Networked Cloud Orchestration: A GENI Perspective," *Proc. 2010 IEEE GLOBECOM Workshops*, 2010, pp. 573–578.

limited to specific protocols (such as the Border Gateway Protocol and Open Shortest Path First) and properties (for instance, loops and black holes), and they increase system complexity.[1]

To support a larger, global consumer base, cloud infrastructure providers (that is, infrastructure as a service, or IaaS, providers) have established datacenters in multiple geographical locations to equally distribute loads, provide redundancy, and ensure reliability in case of site failures. For example, Google and Amazon have several datacenters in the US, Europe, and Asia. Combining siloed, multitenant intra-datacenter interconnects with a highly distributed, ultra-speed inter-datacenter interconnect imposes new challenges for network planning, especially considering the massive increase in data traffic, flexible connectivity, and cost optimization.

Here, we examine these challenges and present a solution based on network virtualization that utilizes software-defined networking (SDN) — in particular, the OpenFlow framework.

Network Virtualization

Highly scalable, flexible, and reliable intra- and inter-datacenter interconnects designed with energy efficiency and cost optimization are an emerging requirement for next-generation cloud computing architecture. One key challenge is how to alter network topologies and routing schemes dynamically and flexibly according to traffic requirements and the capacity of the underlying physical infrastructure. In particular, datacenter networks might implement consolidation and "follow the sun" algorithms (for example, workload follows the time of day) to allocate or relocate VMs within a datacenter or across multiple nodes.[2] This results in various virtual datacenter network topologies, causing link capacity and energy consumption to vary as well.

Datacenter network virtualization is a promising solution to address these problems. By creating multiple virtual networks on top of a shared physical network infrastructure, virtualization separates logical networks from the underlying physical network, leading to aggregated traffic and reduced energy consumption. In terms of computing resources, VM consolidation is widely recognized as an approach to avoid over-provisioning physical servers. Although recent ultra-low-power techniques can decrease idle server power consumption,[3] consolidation is still required to maximize server utilization.

Here, we investigate the virtual flow assignment problem for VM networks, which results in flexible and dynamic data forwarding schemes in both intra- and inter-datacenter interconnects. Our approach maps user requirements — in terms of virtual compute and network resources — onto physical infrastructure. Although prior work has addressed this problem, we must consider new challenges in the context of cloud computing, in which server consolidation and multipath forwarding are enabled. Moreover, software-defined networking (SDN) provides a unique opportunity to effectively implement solutions. The OpenFlow framework,[4] in particular, lets us program network functions and protocols by decoupling the data plane and the control plane, allowing

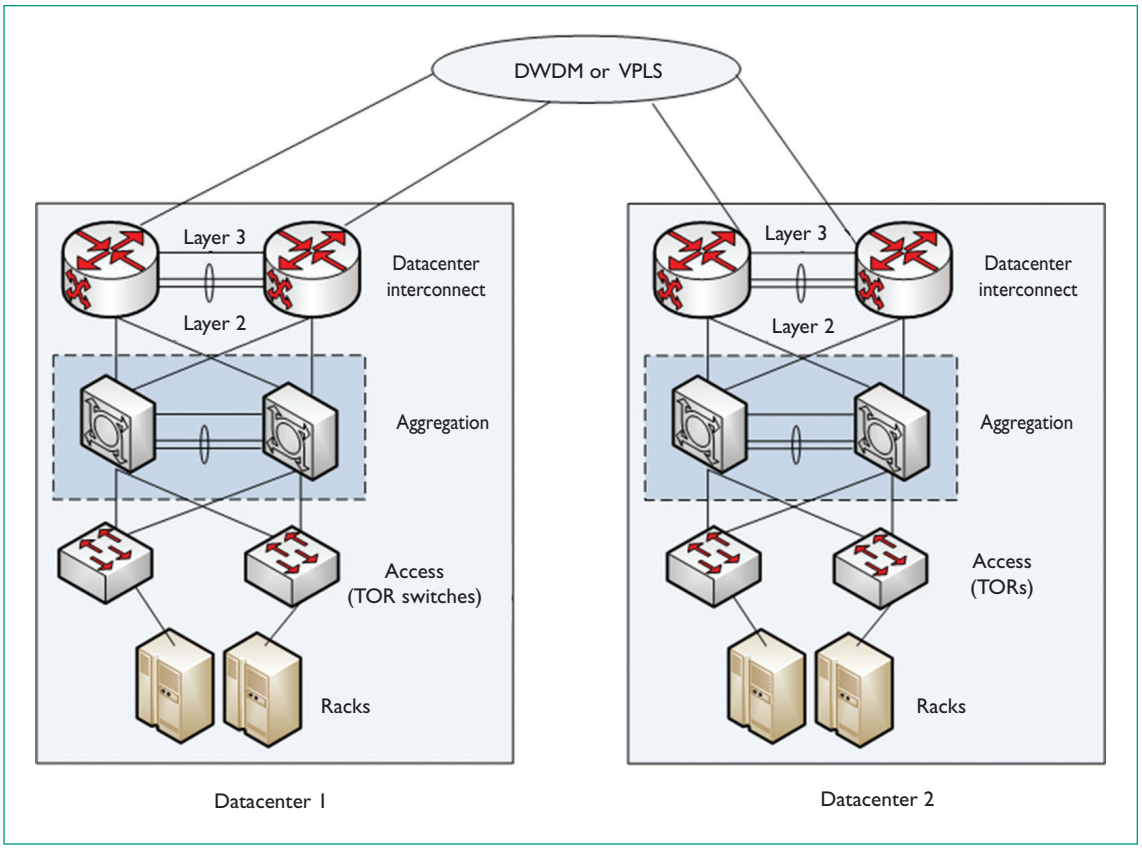

Figure 1. Intra- and inter-datacenter interconnect architectures. Dense wavelength division multiplexing (DWDM) and virtual private LAN services (VPLSs) are two technologies that can be used alternately to connect datacenters.

intelligent, user-controlled switching and routing service provisioning. This is the cornerstone for building a cloud management framework that implements our proposed solutions. We've implemented and deployed our software-defined solution in the GreenStar Network (GSN; http://greenstarnetwork.com) and Green Telco Cloud projects.

Background on Datacenter Interconnects

Today, most intra-datacenter interconnects are switch-centric, based on commodity switches for the datacenter fabric. The network is usually a canonical fat-tree two- or three-tier architecture (see Figure 1), consisting of dense, high-speed, and nonblocking switches placed in a hierarchal manner. The blade servers are accommodated in racks and connected through a top-of-the-rack (ToR) switch, typically using 1-Gbps links. These ToR switches are further interconnected through 10-Gbyte aggregate switches that are in turn connected to 10-Gbps or 100-Gbps core switches. Because switch-centric architecture limits the overall network bisection bandwidth, researchers have proposed server-centric architectures that combine switches and servers (such as the B-cube topology[5]) so that servers act as not only end hosts but also relay nodes for multihop communications. Furthermore, to face increased communication bandwidth demand and datacenter power consumption, new interconnection schemes use optical technology to provide high throughput, reduced latency, and low power consumption.[6]

To colocate datacenters with customers, make them tolerant to catastrophic failures, and reduce latency, energy, and personnel costs, online service providers have built networks of distributed datacenters to host cloud platforms in multiple locations across the globe. Although links are basically peer-to-peer, the biggest challenge faced when interconnecting these datacenters is synchronization, resulting in massive amounts of backup and replication data sent through the inter-datacenter network. Recent research reports

that Web-scale cloud providers can transfer petabytes of internal traffic daily between their datacenter networks,[7] thus doubling or tripling bandwidth requirements over the next few years.

Traditional solutions for dealing with this problem include periodically expanding network capacity, which is expensive and unsustainable, or using data manipulation techniques, such as compression and caching, which have technological limits and might always lag the growing rate of traffic requirements. Advanced services, such as Google's Effingo, have thus arisen to support large-scale copying and replication between datacenters.[8] The key feature is to scale and sustain very high traffic throughput using distributed scheduling decisions on constrained resources with multipoint topological efficiency, high availability, and scalability. This is enabled by a store-and-forward mechanism in which data transit through one or more locations before being forwarded to the destination when restrictions no longer exist on network usage.[8] To schedule data transfers at the best time, such services might also consider network-related information such as local peaks.

The Case for a Virtual Slice

Virtual networks are increasingly used in new datacenters, enabling providers to partition available resources and share them among different users. Moreover, providers are increasingly deploying cloud services across multiple infrastructure providers, creating a new class of cloud federation techniques. One major task to federate the cloud is stitching different pieces of virtualized resources from geographically distributed network substrates into a single connected configuration. For example, the Global Environment for Network Innovations (GENI) has implemented three proposed approaches to connect datacenters[9]: hop-by-hop stitching, in which each datacenter negotiates locally with its neighbors; centralized stitching, in which a central entity negotiates with all datacenters to set up a network; and coordinated stitching, which combines the first two approaches.

Based on network virtualization paradigms, SDN offers a new way to divide, or *slice*, network resources so that researchers and network administrators can use them in parallel. Network slicing implies that actions in one slice don't negatively affect other slices, even if they share the same underlying physical hardware. Virtual LANs, a traditional slicing technique, let network administrators partition the network via switch ports and map all traffic to the VLAN by input port or explicit tag. However, VLAN techniques depend heavily on routing and forwarding protocols, and VLANs aren't easily configured. Network elements connecting to a VLAN aren't easily transported to a new location. This imposes limitations on cloud services, which are scalable and mobile. More importantly, VLAN is a network-centric technique that doesn't consider computing resources, such as servers and VMs.

In the cloud computing context, a virtual slice is composed of several VMs linked via dedicated flows. This definition addresses both computing and network resources involved in a slice, providing users with the means to flexibly program, manage, and control their cloud services.

The GSN Testbed

The GSN is a typical testbed for cloud mobility on a global scale; it's based on a "follow the wind, follow the sun" algorithm.[2] All GSN servers are virtualized by hypervisors, and users rent computing power through VMs. The network slice service lets users actively create and manage their VM networks. Providers such as Amazon offer a similar concept called a virtual private cloud (VPC; http://aws.amazon.com/vpc). However, links in a VPC are fixed when it's created, whereas a slice in the GSN is scalable and flexible thanks to an SDN architecture that uses OpenFlow technology.

In most clouds in the market, the hypervisor links VMs directly to the server's physical network interface controller (NIC), which then connects to a physical switch in the datacenter. In contrast, each server in the GSN has a built-in virtual smart switch (called a vSwitch); VMs connect to their vSwitch before the physical switch. A vSwitch isolates or groups VMs running on a server according to users' demand. An OpenFlow controller running on a dedicated VM handles the entire network's control plane. It controls the vSwitch's flow tables in such a way that all VMs belonging to a user slice are put in a VLAN, which might span multiple vSwitches. Users configure their slices through a Web-based graphical interface (http://greenstarnetwork.com) that translates and then relays user requests to the controller through the GSN cloud middleware. When a VM moves around servers, the controller dynamically reconfigures vSwitches so that the

VM network slice remains unchanged. Virtual routers can configure dynamic tunnels when VMs move between datacenters. Flow classification algorithms let the controller provide different quality-of-service (QoS) levels to different user categories.

OpenFlow-based virtual slice networking has several advantages for cloud computing services deployed in a network of distributed datacenters. First, it simplifies path definition in a network. Rather than defining a new header containing a label, OpenFlow lets network operators define a "flow" with an arbitrary combination of header fields. Second, it enables a variety of actions. In addition to regular packet output actions, packet copy, packet discard, header modifications, and other actions can be indicated. Finally, it exhibits flexible behavior. OpenFlow switches can be reactive so that new flow table entries are dynamically created when a switch encounters an undefined flow. Moreover, OpenFlow switches can be proactive so that flow table entries are set in advance of packet arrivals.

Virtual Slice Assignment Problem

Allocating a slice of virtual resources with its specific performance requirements onto physical datacenter infrastructure is challenging, especially when new datacenter interconnects and software-defined capabilities are taken into account. In this section, we will review new requirements and propose a solution for the problem.

Motivation

Large-scale applications, such as telecommunications, often have stringent QoS, availability, and reliability requirements. Virtualizing these applications and deploying them in a cloud environment presents resource allocation problems — in particular, how to allocate network resources when traffic increases. Figures 2a and 2b show an experiment carried out on a telco cloud (in this case, a cloud environment hosting telecommunications applications developed by Ericsson), where a virtual IP multimedia subsystem (IMS) is deployed on two identical VMs and stressed with 600 calls (SIP registration messages). The VMs are hosted on two blade servers in our Green Telco Cloud testbed and linked via a 10-Gbps backplane. One VM handles the virtual IMS's *call session control functions* (CSCF) component, and the other runs the *home subscriber server* (HSS) component.

As shown, CPU consumption increases rapidly in the HSS when the system is stressed because of database lookup requests. Traffic between the two VMs also increases quickly, whereas the memory requirement remains stable. This suggests that an appropriate strategy to deal with this sudden increase in service requests would be to dynamically scale up the processing capacity of the VM hosting the HSS, and the flow between the two VMs. The VM hosting the CSCFs can remain intact.

The experiment shows a correlation between the two VMs in terms of CPU, memory, and network requirements during a service request. So, mapping a slice of these two VMs onto a cloud's physical infrastructure requires computing all needed processing and network resources, as well as interdependence between the VMs. We must also consider the capacity of intercloud links, especially when store-and-forward mechanisms are deployed. For example, the synchronization of the distributed HSSs can be scheduled according to the intercloud scheduling algorithm. Also, the intracloud interconnect architecture must be involved in the mapping decisions. Figure 2c illustrates assigning a virtual slice into a B-cube (server-centric) datacenter architecture, in which all physical links are 1 Gbps, and virtual flows among VMs are 2 Gbps each. The figure shows a possible mapping: VM1 to Host 1, VM2 to Host 6, VM3 to Host 6, and the virtual flow VM1–VM3 to two physical links H1-S1.0–H5-S0.1-H6 and H1-S0.0–H2-S1.1-H6. We assume in Figure 2c that each host has four CPUs, and each VM requires two CPUs' processing power. The proposed mapping consolidates VM2 and VM3 in Host 6; thus communications between these two VMs don't consume network bandwidth.

Assumptions

Because switches often operate at a constant level regardless of conveyed traffic, we reduce energy consumption from links, assuming that an adaptive link rate mechanism is available. Our goal is to minimize the energy that a user slice of VMs consumes. Thus, idle servers aren't turned off, but their resources are used for other applications running competitively on the cloud. In addition, isolation is also an issue when multiple slices are hosted on the same infrastructure. Our solution addresses traffic interference by deploying OpenFlow-enabled virtual switches directly on physical servers, where switch ports are assigned to different user slices based on OpenFlow tables.

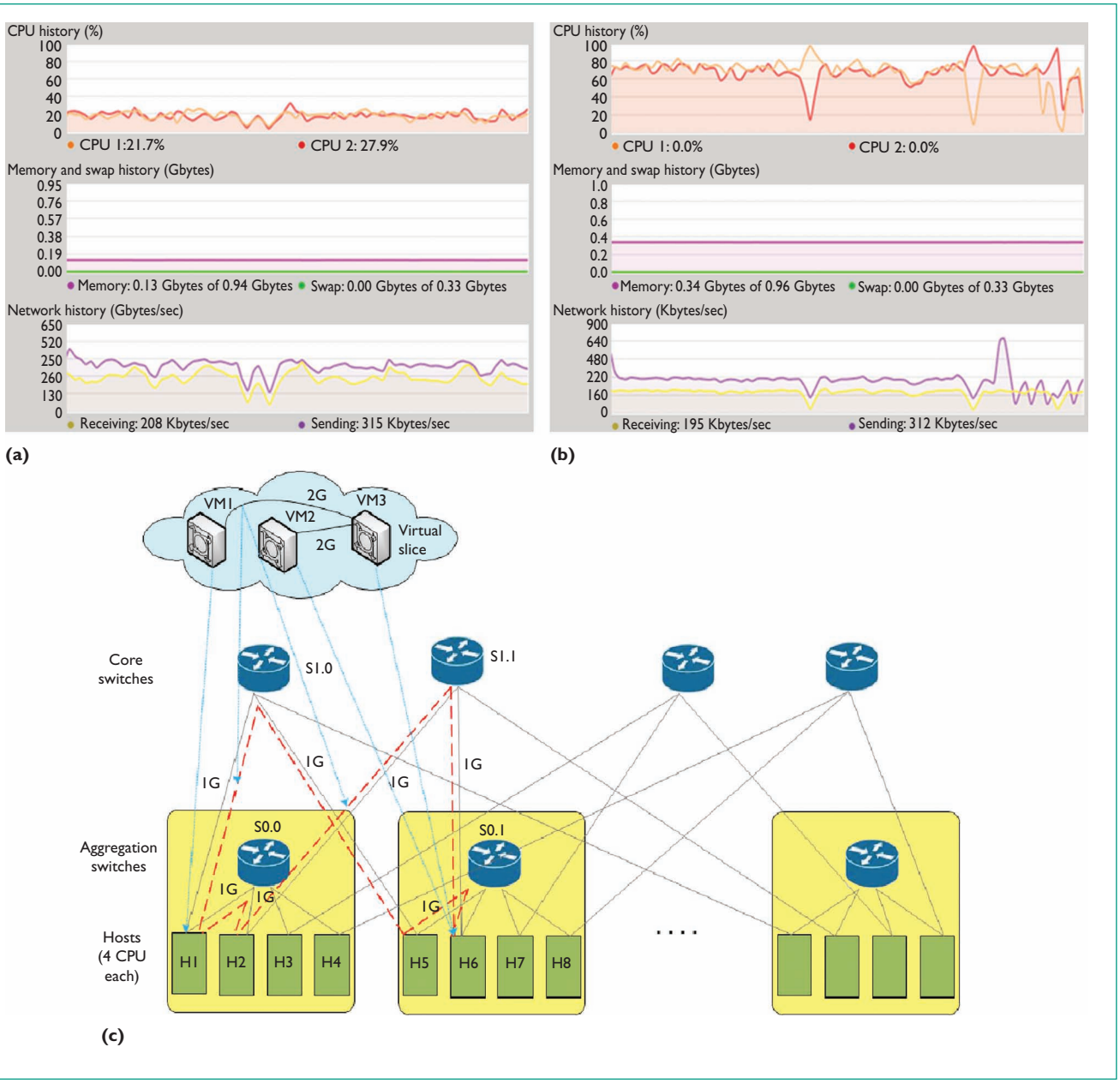

Figure 2. Telco cloud experiment: CPU, memory consumption, and traffic between (a) the call session control function (CSCF) components of a virtual open IP multimedia subsystem (IMS), and (b) the home subscriber server (HSS) components of a virtual open IMS; and (c) a virtual slice assignment into a B-cube intracloud interconnect.

Formal Problem Definition

The virtual slice assignment problem recurs to a set of flow and node assignment problems in an undirected graph. Unlike the traditional substrate network mapping problem, virtual slice assignment puts a new computing power constraint on a set of nodes in the graph (for instance, the set of hosts). Given an objective function, such as minimizing the power consumption of links, switches, and hosts, the problem will result in an optimal virtual network of VMs allocated on physical hosts distributed across a datacenter network. This will satisfy the resource requirements of VMs and the flows between them. Note that we don't address VM relocation (migration) here.

Considering an objective function that minimizes energy consumption in the cloud, we formulate the problem of virtual slice assignment as follows. Given an undirected graph $G(V, E)$, V is the set of vertices and $E \in V \times V$ is the set of

edges. $V = H \cup S$, in which H is the set of hosts, and S is the set of switches (including the datacenter gateway switches). Each host i has available capacity (CPU and memory) $C_i = \{C_{ik}\}$ and consumes an amount of energy p_i for a resource allocation (in terms of memory and CPU). Each edge $(i, j) \in E$ represents a communication link between two nodes i and j, which are hosts or switches. The available bandwidth capacity of an edge is denoted by $c(i, j)$ ($c(i, i) = \infty$). The bandwidth capacity between two datacenter gateways (i, j) isn't a fixed value, but is rather a function representing the intercloud scheduling algorithm. We calculate it from the normalized diurnal periodicity patterns ($T(t \ mod \ 24)$), the over-provisioning level (O) of the link type (peripheral or core), and the nominal $c(i, j)$ of the link.[8]

Let $p(i, j)$ refer to energy consumed on the physical link when sending a bit from node i to node j and vice versa. Consider that a virtual slice contains N VMs. Each VM j has requirements in terms of CPU and memory $C_j = \{C_{jk}\}$. We can calculate a VM's power consumption from its requirements and the power consumption of the host.[2] M virtual flows link these VMs. We denote a flow between each pair of VMs in N hosts by F_m (s_m, t_m, d_m), in which s_m and t_m are, respectively, the source and destination hosts of the mth flow, $s_m, t_m \in N$; and d_m is the demanded bandwidth of the flow. D is the number of datacenters in the network; each datacenter $d \in D$ is accessible through a gateway g_d, which is considered a special node.

Given that any external request must go through the gateway to a VM, we model external traffic to a slice by N flows from the gateway to N VMs of the slice. If no external traffic goes to the kth VM, the kth flow's demanded bandwidth is 0. The slice thus contains $M' = M + D \times N$ flows, in which M is the number of flows among VMs, and $D \times N$ is the number of flows from the gateways to the VMs in the slice.

The problem is therefore to determine a set of edges $E' \in E$ linking VMs in the slice, thus satisfying the traffic demand, CPU, and memory requirements, and minimizing the slice's total power consumption (the total power consumption of links and VMs). We use two matrixes X, Y to represent the results:

$$x_m(i,j) = \begin{cases} \text{bandwidth on the link } (i,j) \\ \quad \text{reserved for the flow } F_m \in M' \\ 0 \text{ if the flow } m \text{ is not routed} \\ \quad \text{through the link } (i,j) \end{cases} \quad (1)$$

$$y(i,j) = \begin{cases} 1 \ \text{VM } j \text{ is hosted by the physical server } i \\ 0 \ \text{otherwise} \end{cases} \quad (2)$$

Minimize

$$\sum_{i,j \in N, m \in M'} x_m(i,j) \times p(i,j) \times d_m + \sum_{i \in H, j \in N} y(i,j) \times p_i \times C_j \quad (3)$$

such

$$\sum_{m \in M'} x_m(i,j) \le c(i,j), \quad \forall i, j \in V \quad (4)$$

$$\sum_{k \in N} x_m(s_m, k) = \sum_{k \in N} x_m(k, t_m) = d_m, \quad \forall m \in M' \quad (5)$$

$$\sum_{k \in N} x_m(i,k) = \sum_{k \in N} x_m(k,i), \quad \forall m \in M', \quad \forall i \in V \setminus \{s_m, t_m\} \quad (6)$$

$$\sum_{i \in H} y(i, s_m) \times C_{s_m} \le C_i, \forall m \in M' \quad (7)$$

$$\sum_{i \in H} y(i, d_m) \times C_{d_m} \le C_i, \forall m \in M' \quad (8)$$

Equation 3 defines the objective function that is the virtual slice's energy consumption. Equations 4, 5, 6, 7, and 8 state the problem's constraints. Equation 4 states that the total bandwidth allocated for M' flows on any link (i, j) doesn't exceed the available bandwidth capacity of (i, j). Equation 5 states the demand satisfaction constraint — that is, for any flow, the outgoing traffic at the source or the incoming traffic at the destination is equal to the flow's demand. Equation 6 states that for any intermediate node of a flow, the incoming traffic is equal to the outgoing traffic. Equations 7 and 8 state the constraints of computing resources for two ends of a flow (that is, the total requirements of allocated VMs can't exceed a server's available CPU and memory capacity).

Solution

This problem is NP-hard, and has been solved in polynomial time using mixed linear programming formulation.[10] However, the calculation time is often high because a cloud has many servers. So, before applying the algorithm,[10] we propose a three-step heuristic based on knowledge about intra- and intercloud interconnects

1. Run a Best-Fit Decreasing algorithm[11] to map and consolidate VMs onto servers in datacenters, starting from servers with the lowest energy consumption.
2. Remove flows between VMs hosted by the same server.
3. **For each** flow m in the list of flows **do**
4. Determine the location of s_m (source node) and t_m (destination node) of m.
5. If s_m and t_m are located in the same data center, go to line 6; otherwise go to line 7.
6. Run a Depth First Search (DFS) algorithm to select intermediate switches. The algorithm is executed starting from s_m. At each intermediate switch, try to allocate the link with the largest available bandwidth possible. If the sum of all links' bandwidth does not meet constraint 5, backtrack to the previous switch. This step loops until either t_m or s_m is reached. If the algorithm returns back to s_m, the problem is unsolvable.
7. Sort inter-datacenter links in descending order of available bandwidth, and then execute line 6 with t_m as the gateway g_{s_m} of the source node s_m.
8. Determine the gateway g_{s_m} of the destination node t_m. Execute line 6 with g_{s_m} and t_m.

Figure 3. Algorithm for allocating a virtual slice.

to reduce the computing space. Figure 3 presents the proposed algorithm's pseudocode.

We apply the heuristic as follows:

- Step 1. *Server consolidation* involves lines 1 and 2 of the algorithm. The VMs are first consolidated on datacenters with the lowest amount of energy consumed per resource allocation.[11] The number of virtual flows between VMs is reduced after this step because of VM consolidation.
- Step 2. *Multipath maximization* includes lines 3, 4, 5, 6, 7, and 8 of the algorithm. Node assignment starts with those leaf nodes that have the largest number of links and bandwidth capacity. This strategy helps increase the successful assignment rate because it reduces the number of critical flows (those with a low chance of being accommodated).
- Step 3. *Local privilege* is embedded in line 1 of the algorithm. We maximize the number of VMs assigned to a datacenter to reduce the flows going through the intercloud network.

Given that the complexity of the DFS algorithm is $O(E)$, the proposed algorithm's complexity is $O(M \times E)$.

Experimental Evaluation

We implemented our proposed solution in the cloud middleware developed for the GSN and the Green Telco Cloud (http://greenstarnetwork.com). In experimental simulations, we evaluated network power consumption with regard to the requirements of an experimental application called GeoChronos.[12] This infrastructure enables the Earth observation community to share data and scientific applications, and collaborate effectively. The application runs on a multiprocessor clustered system with 48 total cores.

Figure 4a shows GeoChronos's architecture, which is composed of six components, including gateways, and application, compute, and database servers. When being virtualized, each component can be hosted by a VM. Six internal flows link these components. The GSN's backbone network, used for intercloud communications, includes five nodes located in Canada's provinces; the energy consumed by a bit of data flowing between the two datacenter gateways is available elsewhere.[2] The datacenters are connected respectively using tree and fat-tree architectures. Table 1 provides the simulation configuration. The energy consumption of servers, VMs, and network equipment is also available in our prior work.[2]

All datacenters are tree-based in the first experiment, and fat-tree-based in the second. Figures 4b and 4c compare the proposed solution and the coordinated node and link mapping method (that is, R-ViNE[10]) when we assign the Geochronos slice in the cloud network in two respective experiments. In both the tree and fat-tree architectures, the proposed solution achieves better performance than the coordinated mapping method in terms of energy consumption for slice allocation because it incorporates all network- and computing-related information to make an optimal decision, especially with

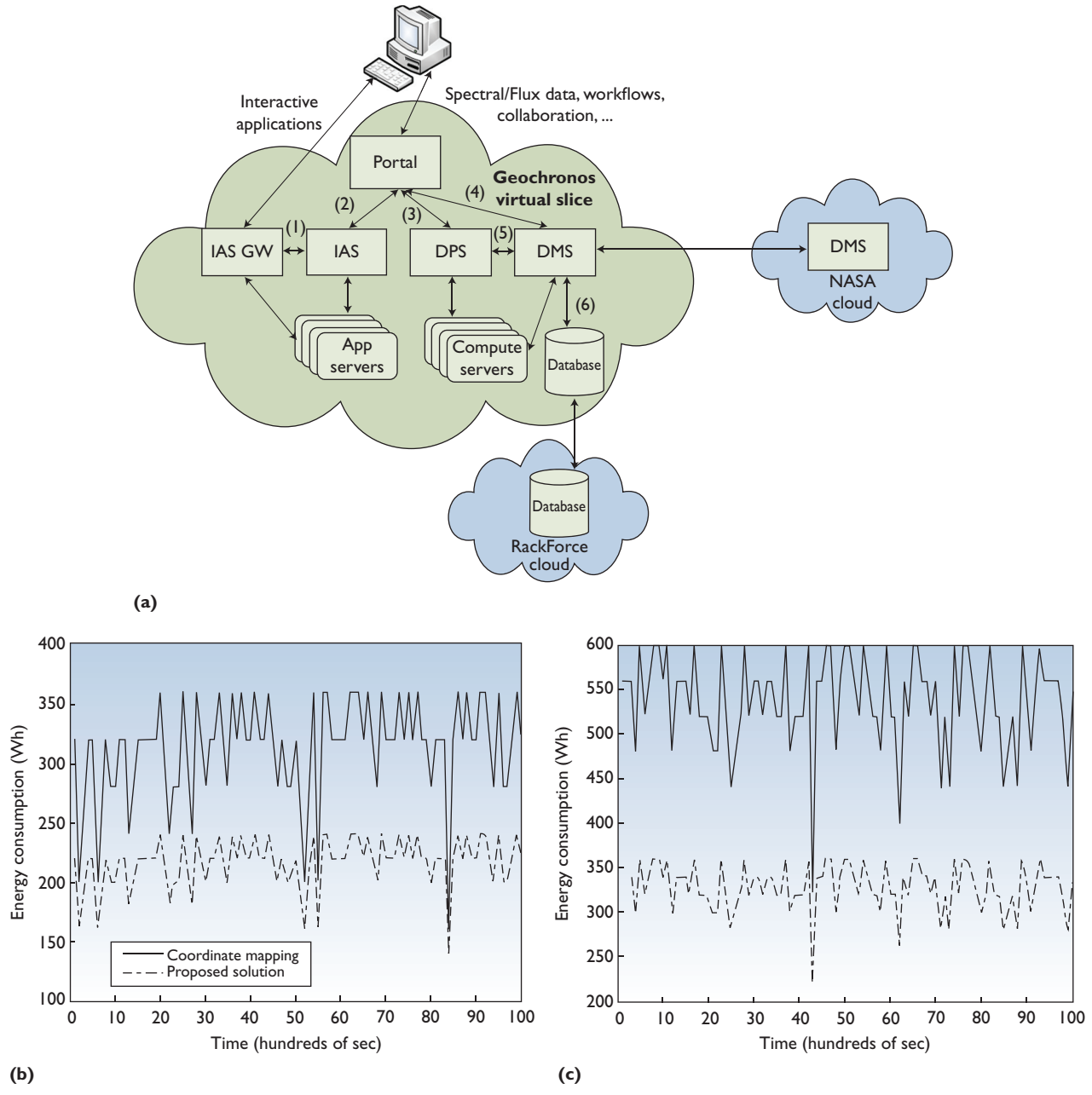

Figure 4. Simulations. (a) The GeoChronos architecture is composed of six components including application, compute, and database servers, and its virtual slice. We compare our proposed solution and the coordinated node and link mapping method (R-ViNE[10]) when we assign the Geochronos slice in the cloud network via two experiments with (b) tree and (c) fat-tree architectures.

respect to VM consolidation and flow multipath supports.

We also observed the following:

- During the night time, when datacenters are synchronized, the slice's power consumption increases as a result of heavy traffic in the core network.
- A slice's power consumption is lowest when it's hosted entirely at a datacenter. In other words, inter-datacenter network equipment consumes a large part of the slice's total energy consumption.
- Computation time is on the order of minutes for a network of 13 nodes and a few hundred servers.

Table 1. Simulation configuration.

Simulation parameter	Description
Server CPU capacity	48 cores/server
Server memory	32 Gbytes/server
Intra-datacenter link capacity (Ethernet)	1 Gbps/link
Inter-datacenter link capacity	10 Gbps/link
Intra-datacenter link energy consumption (Ethernet)	10 W/link
Inter-datacenter link energy consumption	See reference 2
Virtual machine (VM) CPU requirement	2 cores/VM
VM memory requirement	4 Gbytes/VM
VM flow bandwidth requirement	100 Mbps/flow

Solving the virtual slice assignment problem we present is essential to building an optimization cloud-network-planning framework that considers computing, network, and storage resources. Solving the problem helps apply software-defined paradigms to cloud environments effectively — in particular, for resource allocation and service deployment. Our future work will address the environmental impacts of virtual slice allocation.

References

1. C.J. Sher Decusatis et al., "Communication within Clouds: Open Standards and Proprietary Protocols for Data Center Networking," *IEEE Communications*, vol. 50, no. 9, 2012, pp. 26–33.
2. K.K. Nguyen, M. Cheriet, and M. Lemay, "Powering a Data Center Network via Renewable Energy: A Green Testbed," *IEEE Internet Computing*, vol. 17, no. 1, 2013, pp. 40–49.
3. D. Meisner et al., "PowerNap: Eliminating Server Idle Power," *ACM SIGPLAN Notices*, vol. 44, no. 3, 2009, pp. 205–216.
4. N. McKeown, T. Anderson, and H. Balakrishnan, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Rev.*, vol. 38, no. 2, 2008, pp. 69–74.
5. C. Guo et al., "Bcube: A High Performance, Server-Centric Network Architecture for Modular Data Centers," *Proc. ACM SIGCOMM 2009 Conf. Data Communication*, 2009, pp. 63–74.
6. N. Farrington, G. Porter, and S. Radhakrishnan, "Helios: A Hybrid Electrical/Optical Switch Architecture for Modular Data Centers," *ACM SIGCOMM Computer Communication Rev.*, vol. 40, no. 4, 2010, pp. 339–350.
7. Z. Zhang et al., "Optimizing Cost and Performance in Online Service Provider Networks," *Proc. 7th Usenix Conf. Networked Systems Design and Implementation*, 2010, pp. 3–3.
8. M. Sirivianos, N. Laoutaris, and X. Yang, "Inter-Datacenter Bulk Transfers with Netstitcher," *Proc. ACM SIGCOMM 2011 Conf.*, 2011, pp. 74–85.
9. I. Baldine et al., "Networked Cloud Orchestration: A GENI Perspective," *Proc. 2010 IEEE GLOBECOM Workshops*, 2010, pp. 573–578.
10. N.M.M.K. Chowdhury et al., "Virtual Network Embedding with Coordinated Node and Link Mapping," *Proc. IEEE INFOCOM 2009*, 2009, pp. 783–791.
11. K.K. Nguyen et al., "Environmental-Aware Virtual Data Center Network," *J. Computer Networks*, vol. 56, no. 10, 2012, pp. 2538–2550.
12. R. Curry et al., "An On-line Collaborative Data Management System," *Proc. IEEE Gateway Computing Environments Workshop*, 2010, pp. 1–10.

Kim-Khoa Nguyen is a research fellow at École de Technologie Supérieure, University of Quebec, where he is key architect of the GreenStar Network project and responsible for R&D on the Green Telco Cloud project. His research includes green ICT, cloud computing, the smart grid, router architecture, and wireless networks. Nguyen received a PhD in electrical and computer engineering from Concordia University. Contact him at knguyen@synchromedia.ca.

Mohamed Cheriet is a full professor in the Automation Engineering Department at École de Technologie Supérieure, University of Quebec. His expertise includes document image analysis, optical character recognition, mathematical models for image processing, pattern classification models, and learning algorithms, as well as perception in computer vision. Cheriet received a PhD in computer science from the University of Pierre et Marie Curie. He cofounded the Laboratory for Imagery, Vision, and Artificial Intelligence (LIVIA) and founded and directs the Synchromedia Consortium (Multimedia Communication in Telepresence) at the University of Quebec. Contact him at mohamed.cheriet@etsmtl.ca.

Yves Lemieux is a research engineer at Ericsson. His main interests are in 3GPP-based end-to-end quality of service and virtualization for cloud computing. Lemieux received an MS in computer engineering from École Polytechnique de Montréal. Yves has several patents and publications in the fields of cellular system synchronization selection, network resiliency, and Long-Term Evolution core network congestion control, among others. Contact him at yves.lemieux@ericsson.com.