Document: FP7-PIRSES-GA-2010-269323-EVANS-D4.1
Date: 30-Oct-12 Security: Public
Status: Completed Version: 1.0

End-to-end Virtual Resource Management across Heterogeneous Networks and Services

(EVANS)

FP7- PIRSES-GA-2010-269323

Start date of the project: 1 May 2011

Duration: 36 months

DELIVERABLE D4.1

Intermediate Report on Horizontal Management of Virtualised Resources

Date of preparation: 28 September 2012

Deliverable Leading Partner: UniS

Contributing Author(s): Ning Wang (ed.) (UNIS), Juan-Luis Gorricho & Joan Serrat (UPC), Yan Zhang (SRL), Ke Xu & Mingwei Xu (THU), Ping Zhang & Zheng Hu (BUPT), Kun Yang (UEssex)

Consortium:

Partner name	Partner short name	Country
University of Essex	UEssex	UK
Universitat Politècnica de Catalunya	UPC	Spain
Simula Research Laboratory	SRL	Norway
University of Surrey	UniS	UK
Tsinghua University	THU	China
Beijing University of Post and Telecommunications	BUPT	China

Document location: http://www.fp7-evans.eu/evans_wiki

Project web site: http://www.fp7-evans.eu

Table of Contents

This document has been produced in the context of the EVANS Project. The EVANS Project is part of the European Community's Seventh Framework Program and is as such funded by the European Commission.

Executive Summary

The aim of this document is to provide an intermediate report about the research activities that have taken place in WP4 of the EVANS project. This WP is concerned about the horizontal management of the virtualized network resources, which is more a concern of service providers. This is about virtualized resource management across multiple individual network domains or about inter-domain resource management issues.

The first research issue to be addressed is virtual network embedding techniques that involve multiple interconnected ISP domains. Compared to the conventional virtual network embedding within a single administration domain, the technical challenge in its inter-domain counterpart is that, each participating domain does not have the global knowledge about the resource availability, and also they normally have distinct virtual resource management policies. To comprehensively tackle the problem, we first present the overall business model behind the inter-domain virtual network embedding problem, including the entities of physical infrastructure provider (PIP), virtual network provider (VNP) and service provider (SP). The fundamental target is to perform both virtual node and link mapping between VNPs and interconnected PIPs. Detailed problem formulation is presented in this document, and the proposed algorithms and their evaluation results will be presented at a later stage.

The second research topic is related to virtual resource negotiation and allocation among multiple providers based on the concept of intelligent agents. This work pursues the use of artificial intelligence techniques for the dynamic allocation of substrate resources to virtual networks. We propose that each virtual network can be represented by an intelligent agent, with an objective of satisfying specific goals, and that the overall global objective for all the agents is an efficient utilization of substrate network resources. Reinforced learning mechanisms are applied here to address the problem in dynamic network environments.

The third research topic is about energy-aware data centre (DC) management in content delivery networks (CDN) that cover multiple ISP domains. In order to optimize multiple DCs' energy consumption in CDNs, it is investigated that some servers in DCs can be put to sleep modes during off-peak hours to save energy. The key challenge is to avoid deteriorated end-to-end delay performance due to less live service capability, and dynamic user-to-DC request mapping decisions need to be made with respect to reduced number of active servers. It is shown through a detailed system-level framework that our proposed scheme can be integrated into existing CDN platforms.

1. Introduction

There are two types of important stakeholders in the Internet business market: infrastructure providers (InP) that own and manage the physical network infrastructure, and service providers (SP) that provide end-to-end services to end users without necessarily owning any physical infrastructure. Instead, SPs may "rent" network resources from the underlying InPs according to their specific business and service plans. In virtualised networks, an SP typically creates its own virtual networks by "concatenating" the rented (virtual) resources from multiple InPs in order to offer Internet-wide services. On the other hand, an InP needs to concern how to optimally slice its resources, for instance bandwidth, CPU time, memory etc, to various requesting SPs, such that the overall infrastructure resources can be efficiently allocated for maximising its own profits. Therefore, two orthogonal dimensions of management tasks in a virtualised network environment can be envisioned, as depicted in Figure 1.

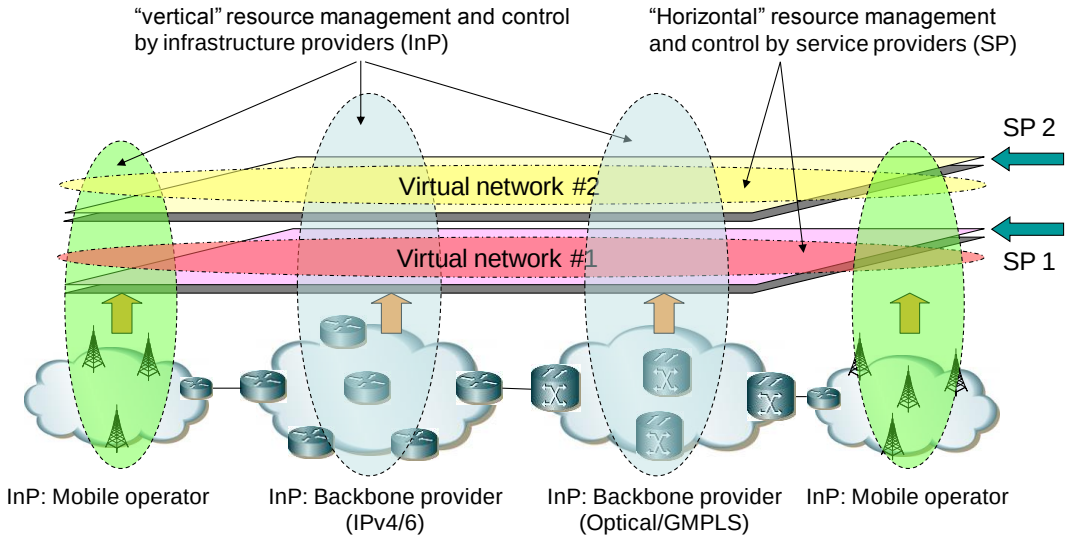


Figure 1. Two Dimensions of Management of Virtualised Networks

Firstly, an InP needs to manage its own physical resources, which involves tasks such as how to describe the physical resources, how to slice them, how to handle incoming resource requests from heterogeneous SPs and allocate virtual resources in a cost-efficient way, etc. This project names this type of management as *vertical* resource management for easy reference. Another dimension of network management is how an SP manages and controls its virtual network resources which are rented from multiple heterogeneous InPs in order to offer its specific services across the corresponding geographical area. This type of management is called *horizontal* resource management, in this project. This deliverable D4.1, which falls into WP4, deals with horizontal resource management across multiple InPs. Its counterpart in WP3, namely, D3.1, deals with vertical resource management.

Service providers shall have the ability to select and operate over different types of autonomous networks (thus *network heterogeneity*) in accordance to user requirements and network features (including cost) to maximize user's quality of experience and therefore

potential profits. Network heterogeneity includes not only the difference between physical carriers, but also diversity in network addressing/routing architectures such as IPv4 and IPv6. Therefore WP4, which is dedicated to SP's management system, considers: Horizontal Management of Virtualised Resource for Network Heterogeneity.

This document records the project's progress regarding investigations into management issues that need to be carried out at the service provider side in order to provide various services, for instance computation-intensive services such as grid/cloud computing applications and bandwidth-hungry services such as content delivery, as identified in Deliverable D2.1. This WP addresses inter-domain issues with each domain owned by one InP.

There are two categories of problems in network virtualization: virtual network mapping or embedding and resource allocation. The former deals with how virtual networks (in terms of virtual nodes and virtual links) are mapping onto the substrate physical networks. The latter is concerned with finer allocation of physical network resources (e.g., in terms of timeslots, radio frequency, power, etc.) to virtual networks. These two important aspects are addressed in Section 2 and Section 3 of this document, respectively. Within each section, both static (e.g., virtual resource selection and composition) and dynamic (e.g., run-time virtual resource control) mechanisms, as represented by Task 4.1 and Task 4.2 respectively in the DoW (Description of Work), are presented.

2. Virtual Network Embedding across Multiple Physical Network Domains

2.1. Background of virtual network embedding problem

Recently network virtualization has emerged as a powerful solution to mitigate the great pressure of traditional Internet Service Providers (ISPs), by decoupling infrastructure providers (who are responsible for managing the network connectivity) from service providers (who are dedicated to deploy network protocols and provide value-added services to end users). In the network virtualization environment, multiple virtual networks (VNs) share a common substrate network, each of which can be customized to offer a specific service (e.g., VoIP).

Each VN is a collection of virtual nodes (e.g., virtual routers) interconnected via a set of virtual links. Therefore, both virtual nodes and links in the VN should be assigned to a specific set of physical nodes and physical paths in the substrate network, with certain constraints on virtual and physical components, which is known as the VN embedding (or mapping) problem. Since multiple VNs share physical resources in the substrate network, effective and efficient embedding of VN requests is of great importance to increase the utilization of the substrate network resources and the capability of satisfying further requests as well as the revenue of infrastructure providers.

VN embedding problem has received significant attention in the research community. Many interesting and efficient embedding algorithms and mechanisms have been proposed. However, these literatures only focus on embedding VNs within the same substrate domain (referred as intra-domain embedding problem), leaving the embedding problem across multiple substrate domains (referred as inter-domain embedding problem) unaddressed. Since a single infrastructure provider rarely controls an entire path, it is inevitable to embed a virtual network across multiple substrate domains so as to offer end-to-end services.

We argue that the inter-domain embedding problems are of great difference from the intra-domain ones. Within the same substrate domain, both the detailed physical network topology and required VN topologies are visible to the infrastructure provider who makes embedding decisions according to its operational goals (e.g., maximize the overall embedding revenue). However, in the inter-domain case, one infrastructure provider has few knowledge about the topology information of another provider that is viewed as commercial secret. Therefore, the algorithms proposed for intra-domain problem cannot be directly used in inter-domain cases. We need new algorithms and mechanisms for inter-domain problem.

A major challenge to achieve this goal is how to define a reasonable information sharing scheme among different parties. Many exact or heuristic solutions have been developed to efficiently embed VN requests within a single domain, which all rely on the whole picture of (un)used network resources for the decision maker. The same assumption will not make sense in the inter-domain case, because PIPs are traditionally reluctant to have their private network topologies exposed, especially to their competitors. However, embedding decisions made by incomplete information will result in undesirable long embedding delays (i.e., the decentralized manner in [1] without extra knowledge of network resources in other domains)

or high embedding failures (i.e., the centralized manner in [2] with limited information of resources in other domains). Therefore, it is crucial to define a reasonable information sharing scheme to facilitate an efficient embedding process meanwhile protect commercial secrets for each individual operator.

2.2. Business model

Considering the functionalities of all participated parties in network virtualization environment, here we present a new business role model with redefined interconnection.

- Physical Infrastructure Provider (PIP), which owns the physical infrastructures, manages the connectivity of the substrate network and provides virtual resources (e.g., routers slices) to support network virtualization.

- Virtual Network Provider (VNP), which will lease virtual resources from one or multiple PIPs to operate a customized end-to-end virtual network according to the requirements of a virtual network user.

- Service Provider (SP), whichmakes specific virtual network requirements to VNP and then applies the virtual network provided by VNP to offer services to end users.

To address this tussle among VNP and PIPs, we define a reasonable information sharing scheme, from which each party involved can benefit while maintaining the core commercial secrets of PIPs. Two kinds of resource information in each substrate network domain will be provided by its PIP, including:

- Node: The location (e.g., geographic coordinates) and available capacity (e.g., CPU and memory) of each substrate node that can support virtual nodes, as well as its price (e.g., money paid for each unit of capacity). Each node exposed to public will be allocated with a global unique index, which can be easily realized by prefixing the local node index with its AS Number.

- Link: the available capacity and the original node index of each substrate link that connects with another substrate domain, as well as its price.

Using the above information sharing scheme, the complete substrate network topology in each domain is invisible to VNP and its competitors. Therefore, each party involved in the Inter-domain VN embedding problem will have a different and partial picture of the whole network.

For VN users (i.e., SPs) who only care about planning an end-to-end virtual network, it is not necessary for them to know the details of underlay networks. For VNP, beside many available substrate nodes located in isolation, it can also know how these substrate networks are interconnected by inter-domain links. For PIPs, they can only control and manage their own substrate network.

2.3 Problem formulation

1) Substrate Network. We model the substrate network operated by all PIPs as an undirected graph and denote it by GS = (NS, LS), where NS and LS represent the set of substrate nodes and links, respectively. Different from the single domain case, here we consider domain, location and CPU capacity as node attributes, and link type and capacity as link attributes. Each node $n^s \in N^S$ is associated with the geographic location $g(n^s)$ and available CPU capacity $c(n^s)$, as well as $dom(n^s)$ to denote the domain it belongs to. Each substrate link $l(i,j)$ for node pair (i,j) is associated with the bandwidth capacity $c(l^s)$. We introduce l^s_{ex} and l^s_{in} to indicate the inter-domain substrate link connecting two different domains and intra-domain substrate link, respectively.

2) Virtual network request. Similar to the substrate network, we also model the virtual network request as an undirected graph and denote it by GV = (NV, LV), where NV and LV represent the set of virtual nodes and links, respectively. For each virtual node $n^v \in N^V$ with location requirement $g(n^v)$, we associate it with a non-negative value d^v as an embedding constraint to indicate how far this virtual node can be placed from its specified location $g(n^v)$. In general, d^v can be measured by the geographical distance or hop counts. The capacity requirements for each virtual node $n^v \in N^V$ and virtual link $l^v \in L^V$ are denoted by $c(n^v)$ and $c(l^s)$, respectively.

When a VN request arrives, VNP will first get the whole VN request decomposed into several sub-VN requests for minimizing the overall provisioning cost.The virtual network embedding for a VN request is defined as a mapping M_{VNP} from G^V to a subset of G^S such that the constrains in G^V are satisfied. The VN mapping can be naturally divided into two components, namely the node mapping phase denoted by M^N_{VNP} and the link mapping phase denoted by M^L_{VNP}.

Node mapping: each virtual node from the same VN request is assigned to a different substrate node.

$$M^N_{VNP}(n^v) \in N^S$$

$$M^N_{VNP}(m^v) = M^N_{VNP}(n^v), \text{iff} m^v = n^v$$

subject to

$$c(n^v) \leq c(M^N_{VNP}(n^v))$$

$$dis\left(g(n^v), g\left(M^N_{VNP}(n^v)\right)\right) \leq d^v$$

where $dis()$ is the distance between two locations.

Link mapping: if two vertices of a virtual link are mapped into the same substrate network domain, then it will be mapped with that domain by the correspondingPIP. Therefore, when mapping virtual links by VNP, only those links cross multiple domains will be considered,

which can be defined by a mapping $M_{VNP}^{L}: L^{V} \rightarrow P^{S}$ from virtual links to inter-domain substrate paths such that for all $l^{v} = (m^{v}, n^{v})$.

$$M_{VNP}^{L}(m^{v}, n^{v}) \subseteq P^{S}(M_{VNP}^{N}(m^{v}), M_{VNP}^{N}(n^{v}))$$

$$dom(M_{VNP}^{N}(m^{v})) \neq dom(M_{VNP}^{N}(n^{v}))$$

Subject to

$$c(l^{v}) \leq c(P^{S}(M_{VNP}^{N}(m^{v}), M_{VNP}^{N}(n^{v})))$$

3. Virtual Resource Allocation

Network Virtualisation is gaining considerable attention as a solution to ossification of the Internet. However, the success of network virtualisation will depend in part on how efficiently the virtual networks utilise substrate network resources. In this section, we propose a machine learning-based approach for the virtual network resource management. We propose to model the substrate network as a decentralised system and introduce a learning algorithm in each substrate node and substrate link, providing self-organization capabilities. We propose a multi-agent learning algorithm that carries out the substrate network resource management in a coordinated and decentralised way. The task of these agents is to use evaluative feedback to learn an optimal policy so as to dynamically allocate network resources to virtual nodes and links. The agents ensure that, while the virtual networks have the resources they need at any given time, only the required resources are reserved for this purpose. Simulations will show that our dynamic approach significantly improves the virtual network acceptance ratio and the maximum number of accepted virtual network requests at any time, while ensuring that virtual network quality of service requirements such as packet drop rate and virtual link delay are not affected. The complete content of this section, section number 3 of this document, devoted to the work developed to solve the problem of the Virtual Resource Allocation, has been taken from the accepted for publication paper [22], as one of the contributions made by the authors within the EVANS project.

3.1. Background and Motivation

One key aspect in network virtualisation is the allocation of physical resources to Virtual Networks (VNs). This involves embedding VNs onto Substrate Networks (SNs), and the management of the allocated resources throughout the lifecycle of the virtual network. The virtual network embedding (VNE) problem involves embedding virtual nodes and links to substrate nodes and links respectively. The efficiency, optimality and flexibility of this resource allocation are fundamental factors for network virtualisation to be successful.

VNE is a well-studied problem [22]. However, most current solutions perform static embeddings where they do not consider the possibility of remapping or adjusting resource allocation to one of more virtual networks. Even approaches that propose dynamic virtual network embedding solutions still allocate a fixed amount of resources to the virtual nodes and links for their entire lifetime. As Internet traffic is not static, this could lead to an inefficient usage of overall network resources, especially during those periods when the virtual nodes and/or links are lightly loaded.

In our work, instead of allocating a fixed amount of resources to a given VN throughout its lifetime, we dynamically and opportunistically allocate resources to virtual nodes and links depending on the perceived needs. The opportunistic use of resources involves carefully taking advantage of unused virtual node and link resources to ensure that VN requests are not rejected when resources reserved to already embedded requests are idle. To this end, we use a *demand-driven* dynamic approach that allocates resources to virtual nodes and links using reinforcement learning (RL) [22].

3.2. Problem description

The virtual network resource allocation problem is made up of two stages; VNE and dynamic resource management. As shown in Fig. 3.1, VNE involves embedding of VNs onto a SN and is initiated by a virtual network provider specifying resource requirements for both nodes and links to the substrate network provider. The specification of virtual network resource requirements can be represented by a weighted undirected graph. Each virtual link connecting the virtual nodes has a maximum delay and bandwidth (data rate), while each virtual node has a queue size (the queue size is a measure of the maximum number of packets (or Bytes) a given node can have in its buffer before dropping packets) and a location as well as a constraint on its location which specifies the maximum allowed deviation from its coordinates. In the same way, a substrate network can be modelled as an undirected graph denoted by the sets of substrate nodes and links, respectively. Each substrate link connecting the substrate nodes has a delay and a bandwidth, while each substrate node has a queue size and a location.

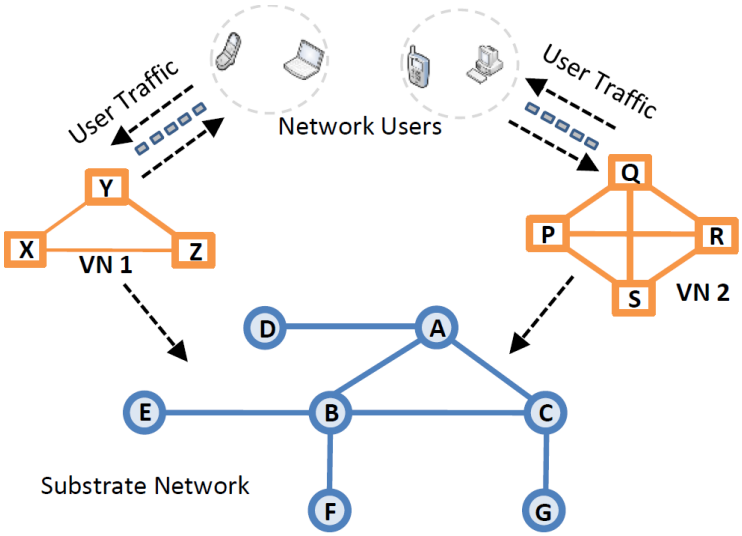


Figure 3.1. Virtual Network Resource Allocation.

The VNE problem involves the mapping of each virtual node to one of the possible substrate nodes within the set defined as a set of all substrate nodes that have enough available queue size and are located within the maximum allowed deviation of the virtual node location. For a successful mapping, each virtual node must be mapped onto a substrate node and any given substrate node can map at most one virtual node from the same request. Similarly, all the virtual links have to be mapped to one or more substrate links connecting the nodes to which the virtual nodes at their ends have been mapped. Each of the substrate links must have sufficient data rate to support the virtual link. In addition, the total delay of all the substrate links used to map a given virtual link must not exceed the maximum delay specified by the virtual link. Solving the VNE problem is out of the scope of this work. Any of the static approaches [22] - [22] can be used for this stage (in this work, a mathematical programming

formulation that performs both node and link embedding in one step, and solved using ILOG CPLEX 12.5 [22] is used to represent a static solution for the evaluations).

The second stage - which is the focus of the work in this report - follows a successful embedding of each VN, in this case the resources allocated/reserved for the embedded VN should be managed to ensure optimal utilisation of the overall SN resources. For this work, we simulate the use of VN resources by transmitting user traffic in the form of packets over the virtual network. By monitoring the actual use, the resources allocated to the VN are then dynamically managed. This is however performed carefully to ensure that quality of service parameters such as packet drop rate and delay for the VNs are not affected.

3.3. Reinforcement Learning (RL) Model for Dynamic Resource Allocation

Virtual network embedding allocates resources to virtual nodes and links based on the specification in the VN requests. Stopping at the embedding stage would result into a static allocation in which a fixed amount of substrate network resources is reserved for each virtual link and node irrespective of actual utilisation. This would lead to underutilisation in situations of light loading for the VNs. The approach proposed in this report is to dynamically adjust the resource allocation using RL. To this end, we start by modelling the overall system showing the interaction of the different elements as shown in Fig. 3.2. The modelling mainly involves the learning environment, the learning algorithm, and a reward function to evaluate the effectiveness of the agents' learning.

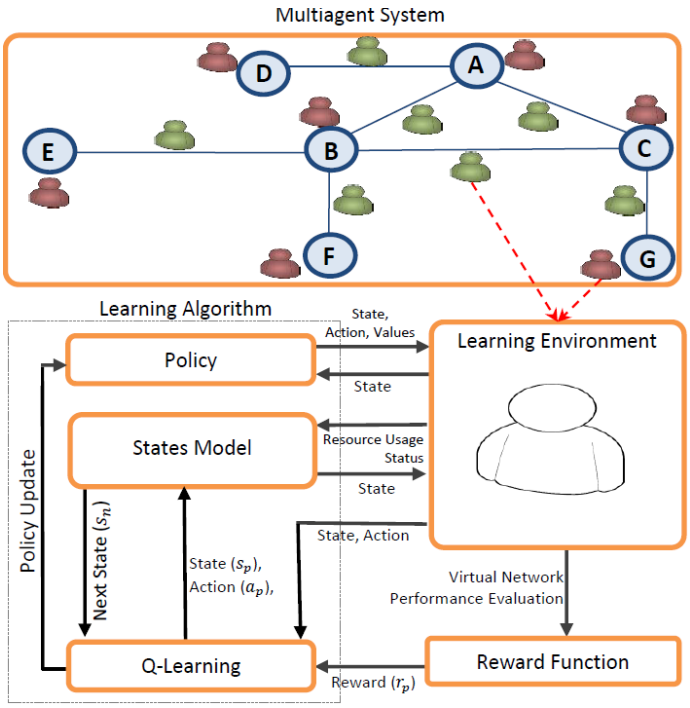


Figure 3.2. Reinforcement Learning Model

The learning environment consists of all the agents that represent the substrate network (the multi-agent system). Specifically, each substrate node and link is represented by a node agent and a link agent. The node agents manage node queue sizes while the link agents manage link bandwidths. The agents dynamically adjust the resources allocated to virtual nodes and links, ensuring that resources are not left underutilised, and that enough resources are available to serve user requests. We consider that each node agent has information about the substrate node resource availability as well as the resource allocation and utilisation of all virtual nodes mapped onto the substrate node. In the same way, we expect that each link agent has information about substrate link bandwidth as well as the allocation and utilisation of these resources by all virtual links mapped to it. In case a given virtual link is mapped onto more than one substrate link, then each of the link agents coordinate to ensure that their allocations do not conflict.

When an agent takes an action, the networks are monitored, recording the link delays, packet drops and virtual and substrate network resource utilisation so as to determine a reward. Specifically, the reward resulting from a learning episode of any agent is a vector in which each term corresponds to the reward of an allocation to the virtual resource (we use the term virtual resource to mean either a virtual node queue or virtual link bandwidth), and is dependent on the percentage resource allocation, the percentage resource utilisation, the link delay and the number of dropped packets. The objective of the reward function is to encourage high virtual resource utilisation while punishing node agents for dropping packets and link agents for having a high delay. We also assign a large negative punitive reward to resource allocations below 25% to ensure that this is the minimum allocation to a virtual resource.

Learning algorithm

The policy is implemented by means of a lookup table which, for each state, maintains an up to date evaluation of all the possible actions. Since we have 9 possible actions and 512 possible states, the size of our policy is 9 x 512 = 4608 state-action values.

The state of any agent is a vector with each term representing the state of one of the virtual links/nodes mapped onto it. The states in this work are discrete. We consider that the total resource demand of each virtual node or link can be divided into at least 8 resource chunks, each representing 12.5% of its total resource demand. For example, a virtual node could be allocated 12.5%, 25%, 37.5%, 50%, 62.5%, 75%, 87.5% and 100% of its total demand. It is important to remark that these re-allocations are performed after a successful embedding. Therefore, all embeddings are performed based on the total demand of any given virtual node or link.

The output of each agent is a vector indicating an action for each of the virtual node/link mapped onto it. An agent can choose to increase or reduce the resources (queue size or bandwidth) allocated to any virtual node or link respectively. Specifically, at any point, each agent can choose 1 out of the 9 possible actions, each of which leads to a discrete change in resource allocation (in all cases, the percentage change is with respect to the total demand of the virtual node or link).

The states' model mimics the behaviour of the environment. When provided with a given status of the substrate and virtual networks, a states' model returns a state. In the same way, when provided with a given state and action, the states' model provides the next state. It is in general a model of the substrate and virtual network resources and how the different possible actions affect the allocation of substrate resources to virtual networks.

In this work, we propose a decentralised Q-learning based algorithm to iteratively approximate the state-action value, and then use these values to *select actions* for the allocation of substrate resources to the virtual nodes and links. The learning algorithm is made up of three major steps: policy initialisation, policy update and action selection.

3.4. Performance Evaluation

To evaluate the performance of the proposed approach, we added a network virtualisation module to NS3 [22]. Table 3.1 shows the NS3 parameters used in our simulations. The implementation is such that every time a virtual network request is accepted by the substrate network, the virtual network topology is created in NS3, and a traffic application starts transferring packets over the virtual network. The traffic used in this paper is based on real traffic traces from CAIDA anonymized Internet traces [22]. This dataset contains anonymized passive traffic traces from CAIDA's equinix-chicago and equinix-sanjose monitors on high-speed Internet backbone links, and is mainly used for research on the characteristics of Internet traffic, including flow volume and duration. The trace source used in this work was collected on 20th December 2012 and contains over 3.5Million packets. We divide these packets among 1000 virtual networks, so that each virtual network receives about 3500 packets. These traces are used to obtain packet sizes and time between packet arrivals for each VN. As the source and destination of the packets are anonymized, for each packet in a given VN, we generate a source and destination IP address in NS-3 using a uniform distribution.

The substrate and virtual network topologies are generated using Brite [22] with settings shown in Table 3.2. Simulations were run on an Ubuntu 12.04 LTS Virtual Machine with 4.00GB RAM and 3.00GHz CPU specifications. Both substrate and virtual networks were generated on a 25x25 grid. The queue size and bandwidth capacities of substrate nodes and links as well as the demands of virtual networks are all uniformly distributed between values shown in Table 3.3. Link delays are as determined by Brite. Each virtual node is allowed to be located within a uniformly distributed distance $7.5 <= x <= 15$ of its requested location, measured in grid units. We assumed that virtual network requests arrive following a Poisson distribution with an average rate of 1 per minute. The average service time of each virtual network is 60 minutes and is assumed to follow a negative exponential distribution.

Parameter	Value
Queue Type	Drop Tail
Queue drop Mode	Bytes
Maximum Queue Size	6,553,500 Bytes
Maximum Packets Per VN	3500 Packets
Number of VNs	1024
Network Mask	255.255.224.0
IP Adress Range	10.0.0.0 – 10.255.224.0
Network Protocol	IPv4
Transport Protocol	TCP
Packet MTU	1518 Bytes
Packet Error Rate	0.000001 per Byte
Error distribution	Uniform (0, 1)
Port	8080

Table 3.1 NS3 Parameters

Parameter	Substrate Network	Virtual Network
Name (Model)	Router Waxman	Router Waxman
Number of nodes (N)	25	[5 – 10]
Size of main plane (HS)	250	250
Size of inner plane (LS)	250	250
Node Placement	Random	Random
GrowthType	Incremental	Incremental
Neighbouring Nodes	3	2
alpha (Waxman Parameter)	0.15	0.15
beta (Waxman Parameter)	0.2	0.2
BWDist	Uniform	Uniform
Minimum BW (BWMin)	2×10^6 bps	1×10^6 bps
Maximum Dev. (BWMax)	8×10^6 bps	1×10^6 bps

Table 3.2 Brite Network Topology Generation Parameters

Parameter	Substrate Network	Virtual Network
Minimum Number of Nodes	25	5
Maximum Number of Nodes	25	10
Minimum Node Queue Size	(100 × 1518) Bytes	(10 × 1518) Bytes
Maximum Node Queue Size	(200 × 1518) Bytes	(20 × 1518) Bytes
Minimum Link Bandwidth	2.0Mbps	1.0Mbps
Maximum Link Bandwidth	10.0Mbps	2.0Mbps

Table 3.3 Substrate and Virtual Network Properties

Performance Metrics

We evaluate the performance of our proposal on two fronts; the quality of the embeddings, as well as the quality of service to the virtual networks. The idea is that the opportunistic use of virtual network resources should not be at the expense of the service quality expectations of the network users.

- Embedding Quality

This is evaluated using the acceptance ratio and total instantaneous accepted virtual networks. The acceptance ratio is a measure of the long term number of virtual network requests that are accepted by the substrate network. The total instantaneous accepted virtual networks is a measure of the embedding cost incurred by a given substrate network, as a substrate network that incurs a lower embedding cost normally has more extra resources at any point and hence is able to have many embedded virtual networks at any point.

- Quality of Service

We use the packet delay and drop rate as indications of the quality of service. We define the packet delay as the total time a packet takes to travel from its source to its final destination. The drop rate is defined as the ratio of the number of packets dropped by the network to the total number of packets sent. As shown in Table 3.1, we model the networks to drop packets due to both node buffer overflow as well as packet errors. In addition, as it is more important in some applications, we define the variations of these two parameters. The jitter (delay variation) is defined as the difference between delays during different time periods, while the drop rate variation is defined as the variation between packet drops in different time periods. The time interval to update the measurements corresponds to the transmission of 50 packets.

Discussion of Results

The simulation results are shown in Fig. 3.3 – 3.8. As can be seen from Fig. 3.3, the dynamic approach performs better than the static one in terms of virtual network acceptance ratio. This can be attributed to the fact that in the dynamic approach the substrate network always has more available resources than in the static case, as only the resources needed for actual transfer of packets is allocated and/or reserved for virtual networks. This is further confirmed by Fig. 3.6 which shows that at any given point a substrate network that dynamically manages its resources is able to embed more VNs than a static one.

Fig. 3.4 shows that the packet drop rate of the static approach is in general constant (due to packet errors as well as buffer overflows) while that of the dynamic approach is initially high, but gradually reduces. The poor performance of the dynamic approach at the start of the simulations can be attributed to the fact that at the beginning of the simulation when the agents are still learning, the virtual node queue sizes are allocated varying node buffers that lead to more packet drops. In fact, this initial number of packet drops affects the rate at which the overall drop rate reduces towards the one for the static approach. This can be confirmed by observing the actual periodic drops in packets as shown in Fig. 3.7 which show that the

total number of packets dropped by both approaches is comparable towards the end of the simulation.

Similarly, Fig. 3.5 shows that the packets in the dynamic approach initially have higher delays than those in the static approach. Once more, the reason for this is the initial learning period of the agents. This is again confirmed by observing that the delay variations in Fig. 3.8 easily converge to those of the static approach. It is however worth noting that unlike the packet drop rate (Fig. 3.4), the actual delay (Fig. 3.5) of the dynamic approach finally converges to that of the static approach. Again, this could confirm that the slow convergence of the drop rate is due to the initial packet drops, since initial packets delays would not affect the delays of other packets, yet initial packet drops remain factors in the final drop rate.

We are however mindful that it could require a much more number of learning episodes for the overall drop rate in Fig. 3.4 to finally converge to that of the static approach. This is because we used a learning policy with 4608 state-action values. With this high number of state-action values, the agents require a lot of time to learn an optimal policy. Moreover, it could improve the accuracy and precision of the agents' actions even more if the state-action values were increased. It would therefore be better to use function approximation or a more compact parameterized function representation to model the agents' policy other than a look-up table. We will investigate this approach more in the future.

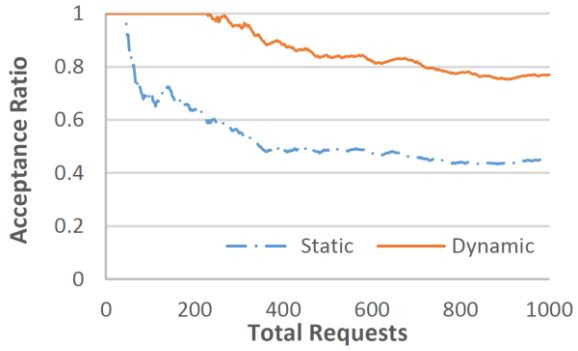


Figure 3.3. VN Acceptance Ratio

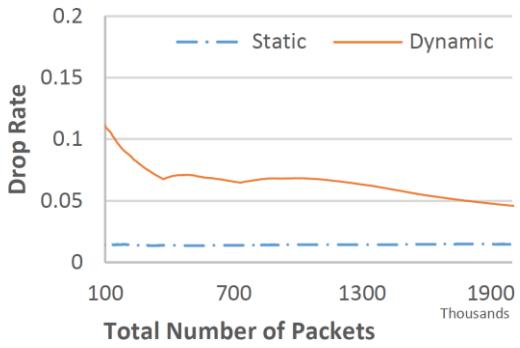


Figure 3.4. Node Packet Drop Rate

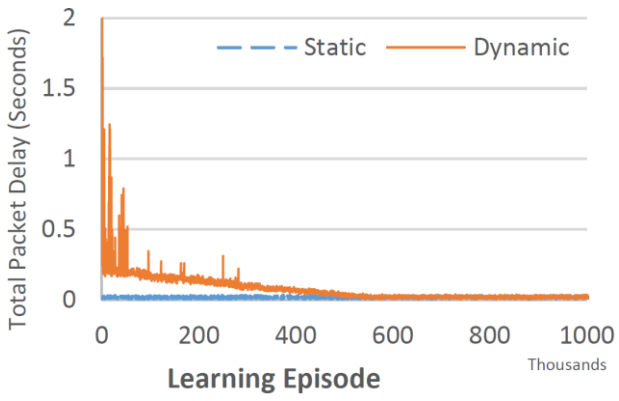


Figure 3.5. Link Packet Delay

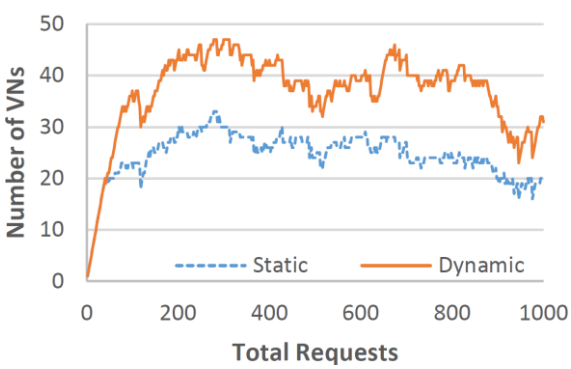
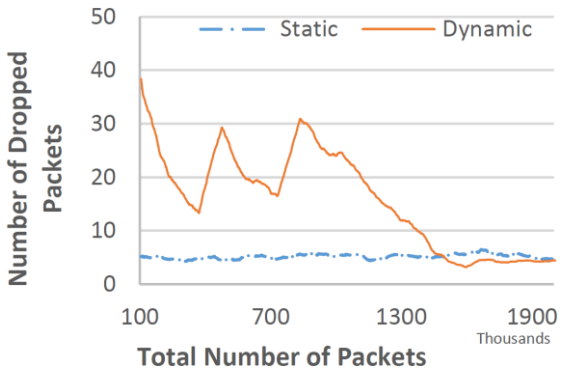


Figure 3.6. Number of Accepted Virtual Networks



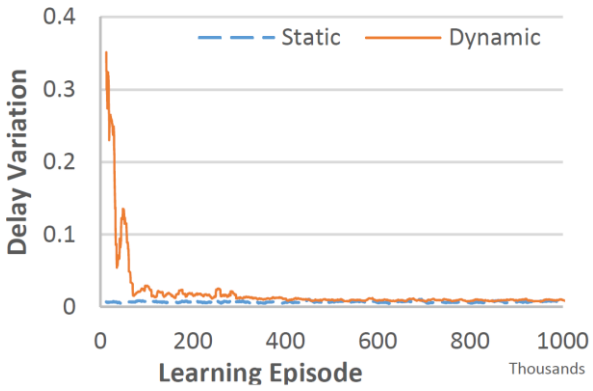


Figure 3.7. Node Packet Drop Rate Variation Figure 3.8. Link Packet Delay Variation

4. Energy-aware management of Cross-domain CDNs

4.1. Background and Motivation

In recent years, energy efficiency in the information communication technologies (ICT) sector has attracted more and more interests from both the research community and the industry, which is especially the case for DCs and CDNs. In the literature, most relevant research works have focused on *standalone* DCs, which involves power management, performance management or both [17]. Meanwhile, there have been research works on reducing CDN energy consumption in recent years, which aim to optimize CDN energy consumption from either a theoretical perspective [18] or a practical perspective [19][20]. However, the scenario of energy saving in large-scale CDN infrastructures that typically cover *multiple inter-connected* ISP domains has not been well addressed.

In this article, we propose an energy management scheme that is specifically developed for cross-domain CDN infrastructures. The fundamental idea is to strategically reconfigure DC servers to the sleep mode without deteriorating the service capabilities on both the DC side and the network side. It is observed that user activities in a CDN typically follow a dynamic but relatively regular daily pattern [21], which implies that during off-peak hours, only a subset of all servers is needed to serve the requests. Therefore, in our scheme, content servers are dynamically provisioned with respect to the present content request volume instead of being kept active all the time. However, the realization of such an operation needs careful consideration on the tradeoff between the DC energy consumption and the CDN QoS performance. On one hand, the proposed scheme tries to put as many servers as possible to the sleep mode, as long as the remaining servers are capable of handling the present content requests. On the other hand, the server status reconfigurations also lead to changes in bandwidth utilization of the underlying ISP networks, as some end-users need to fetch content objects from more remote DCs. Intuitively, such request redirections will incur increased bandwidth consumption due to the growth of inter-PoP content traffic. In order to cope with such energy-performance tradeoff, specific constraints, which keep both DC servers and virtual links under their load capacities, need to be taken into account to avoid QoS deterioration in content consumption sessions with reduced content serving capability.

4.2. Scheme Overview

We first illustrate our main idea for energy saving in CDNs in Figure 2. The conventional request mapping operation is shown in Figure 2(a), in which five DCs are established to serve content requests from nine PoP nodes within two ISP domains. We can see that since the DCs are attached to five of the PoP nodes, requests from these nodes can be resolved locally without incurring inter-PoP bandwidth consumptions. For the other four PoP nodes, their requests are resolved to the DCs that are the closest to them, which is the common practice in real CDN environments to minimize user-experienced latency [22].

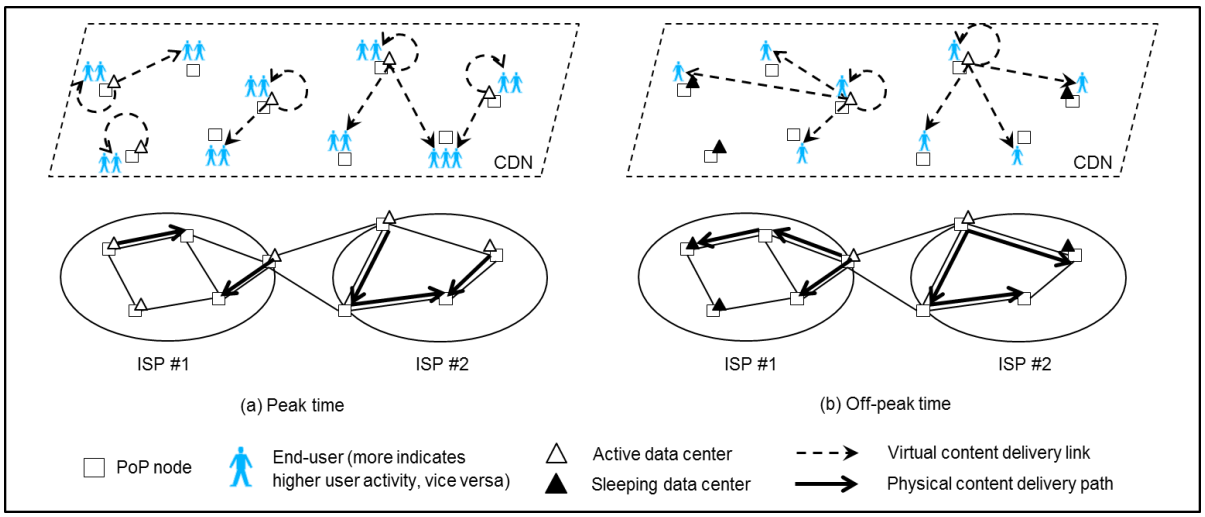


Figure 2. Energy management scheme illustration: a) Peak time, when all DCs are active and serving user requests; and b) Off-peak time, when some DCs are active and serving user requests; and b) Off-peak time, when some DCs are in sleep mode and requests are resolved to fewer active DCs

At present, the CDN operators normally keep all DCs and servers up and running, so that end-users could experience optimized latency when requesting web contents [23]. Although this is necessary at peak hours as illustrated in Figure 2(a), such a practice may lead to significant waste of CDN energy during off-peak hours. The main reasons are discussed as follows:

- From a single PoP node's perspective, the local end-users' request volume varies over different time periods within a single day, which is usually very low at midnight and early morning. Such a pattern has been observed in [21], which reflects the relatively-steady characteristic of CDN request volume at individual PoP nodes.
- From a CDN's perspective, content requests originate from geographically-distributed PoP nodes that are attached with end-users. Considering the difference in the global time zones among the PoP nodes or ISP domains, as well as the daily fluctuating pattern in end-user activities described above, it is unlikely that all domains experience high request volume simultaneously if we consider cross-continental ISPs (which is the case in typical large-scale CDNs).

Based on the two reasons above, it can be inferred that during off-peak hours, only a subset of servers is sufficient to serve all requests due to the low content request volume from local end-users. In this case, the remaining servers can be safely reconfigured to the sleep mode without introducing negative impact on the CDN performance, which will lead to considerable gains in energy saving of CDN data centers. Such a scenario is illustrated in Figure 2(b), which shows that the servers in three DCs are put to the sleep mode during off-peak time in order to save energy. Meanwhile, all requests are served by the remaining two active DCs.

While considering energy savings, we also aim to assure the CDN performance in terms of end-to-end QoS. Such an objective is achieved by considering the following two aspects.

Firstly, basic performance assurance is provided through preventing DC servers and virtual content delivery links from reaching their full load capacities. As servers are responsible for handling user requests and delivering content objects, their response time degrades rapidly as they approach load capacities, which in turn affects the CDN QoS performance. Regarding CDN virtual links, as illustrated in Figure 2, they are used to establish mappings between DCs and PoP nodes that are attached with end-users. Furthermore, network traffic incurred by content delivery will be mapped to underlying ISP networks for the actual delivery. Therefore, limiting traffic over virtual links reduces the risk of congestion in ISP networks, which has implications to end-to-end QoS performance.

Secondly, the CDN performance is further assured through avoiding the generation of unnecessary inter-domain content traffic. This is based on the following considerations. As DC servers are being put to the sleep mode, some requests will have to be redirected to an alternative DC, which could be located in another ISP domain. This should be avoided because a) inter-domain content delivery incurs significantly longer network distance, which increases end-to-end delay; and b) since inter-domain links are more critical compared to intra-domain links, they are more vulnerable to traffic congestions. Therefore, in order to reduce risk of congestion and assure user-experienced latency, inter-domain content delivery is avoided in our scheme when saving CDN energy. Such a scenario is illustrated in Figure 2(b), in which content delivery sessions are restricted within each domain as three of the five DCs are put to sleep.

4.3. Energy Management Framework and Policies

4.3.1. Centralized Energy-Aware CDN Management

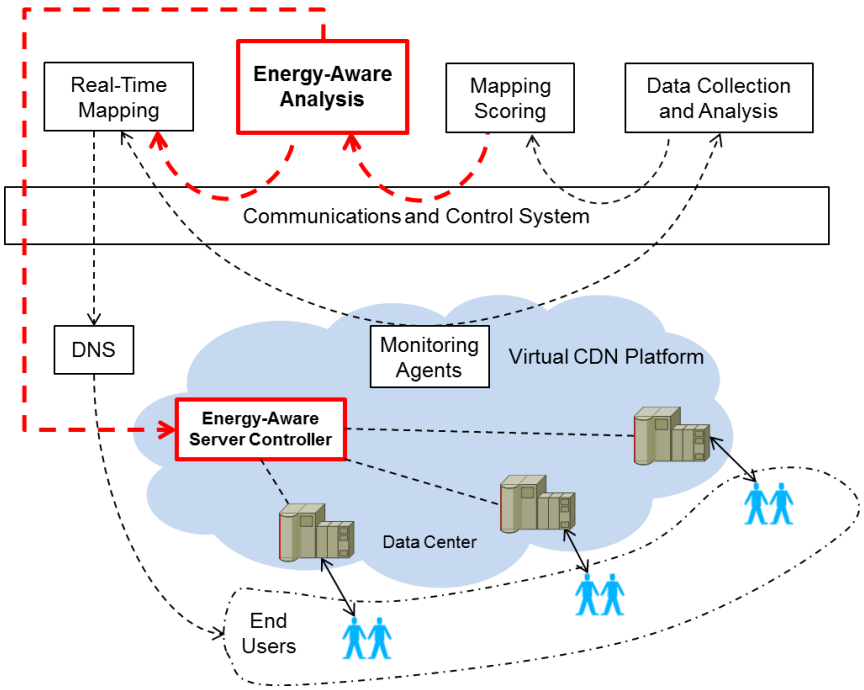


Figure 3. Functional block diagram of proposed energy-aware CDN management system, based on [16]

Since a CDN is a virtual overlay network built on top of physical ISP networks, it is essentially an application layer infrastructure over the Internet. In Figure 3, we illustrate the functional blocks of a typical CDN platform with energy-awareness, which involves the following key components:

- **Monitoring Agents (MA).** In a CDN, it is necessary to continuously monitor the conditions of DCs, servers and virtual links, since it is important that such information is up-to-date so that other components are able to make appropriate decisions on CDN management. The MA units keep reporting necessary context information above to data collection and analysis units and real-time mapping units, so that responses can be immediately made to cope with the events occurred in the CDN.
- **Data Collection and Analysis (DCA).** The DCA unit is responsible for collecting data for general routine purposes such as logging, analyzing and billing.
- **Mapping Scoring (MS).** The MS unit is part of the request mapping system in a CDN, and its responsibility includes creating an up-to-date Internet topology map including network connectivity, latency and loss information. The map is continuously refreshed by the MS unit.
- **Real-Time Mapping (RTM).** The RTM unit is the other part of the CDN request mapping system. It creates a map with only DCs, servers and end-users, so that content requests are resolved to the best DC and server. It is responsible for request

resolutions at both PoP-to-DC and DC-to-server levels, which are based on real-time information received from the MS and MA units. The RTM unit then instructs the DNS (domain name system) to resolve individual requests to their designated DCs and servers.

- **Communications and Control System (CCS).** CCS is the channel for disseminating management information such as control messages and status updates. As shown in Figure 3, all management instructions between the functional units need to go through CCS.

So far we have described the basic functional components in a typical CDN platform. The question is - how can our proposed scheme be embedded in such an infrastructure? In Figure 3, we also illustrate two additional management units that enable our scheme to be deployed in a CDN.

- **Energy-Aware Analysis (EAA).** The EAA unit is the key component that is needed in our scheme, which is used to determine which specific servers need to be turned on or put to sleep. In order to make such a decision, it takes real-time CDN information as an input from the MS unit. With this information, The EAA unit is able to determine *how many* servers need to be turned on or off. Next, the CDN operator specifies a request-mapping policy to EAA, which is then used to determine mappings from user PoPs to DCs and servers to achieve optimized energy consumption and assured CDN QoS performance. Two examples of such request-mapping policies will be described in the next subsection. Afterwards, the EAA unit sends the mapping instructions to the RTM unit, which in turn instructs DNS to resolve the user requests.
- **Energy-Aware Server Controller (EASC).** As a controller unit, EASC is built on top of the CDN platform to establish control over DCs and servers. As EASC receives instructions from the EAA unit on state reconfiguration of servers, it sends instruction signals to the specified servers for on/off status change. Such a technique is readily available in modern CDN platforms [16].

With the illustration of Figure 3, we have shown that our energy-saving scheme is able to be embedded in modern CDN infrastructure in the form of functional blocks. By coordinating with other management components, our scheme is capable of utilizing real-time CDN information to make decisions to optimize the CDN energy consumption with assured QoS performance.

4.3.2. Request Mapping Policies

In the EAA unit, the CDN operator is able to specify different request-mapping policies that are suitable for CDNs with different characteristics and requirements. In this subsection, we show two examples of such policies, *i.e.*, Min-Energy-Dist and Min-Energy-DC.

- **Min-Energy-Dist**: This policy simultaneously optimizes DC energy consumption and network distances traversed by content delivery sessions. In other words, as the number of servers that are actively serving requests is minimal, requests are resolved to local or nearby DCs to optimize network distance and end-to-end delay. Under this

policy, user requests are resolved in a distributed manner, and it is likely that every DC in the CDN will have some servers running to serve requests.

- **Min-Energy-DC**: While minimizing DC energy consumption, this policy aims to consolidate active servers into as few DCs as possible. In other words, content requests are resolved to the fewest possible number of active DCs, and the remaining DCs will have no active server running within them. This policy corresponds to the scenario illustrated in Figure 2(b).

These two policies both have their own advantages and disadvantages. Regarding Min-Energy-Dist, since every DC has a subset of servers running, servers in each DC are provisioned in a more flexible way. For example, if the request volume suddenly increases in a specific area, the DCs in that area will have the flexibility to activate their own sleeping servers to handle the spike in local demand. However, since every DC has some servers running, the overall DC operational costs (*e.g.*, cooling system energy consumption) will be kept at relatively high levels. Regarding Min-Energy-DC, consolidating active servers to fewer DCs leads to easier DC and server management from a centralized point of view. Furthermore, if all servers in a DC are not needed for a certain period of time, then the entire DC could be scheduled for maintenance or sleeping for a given period of time, which introduces possibility of further saving in DC operational costs. However, the CDN becomes more prone to sharply-increased request volumes as entire DCs are being scheduled for sleeping mode reconfigurations. Therefore, Min-Energy-Dist is more suitable for CDNs with more dynamic user demands, where optimizing CDN QoS is prioritized over saving energy consumption. On the other hand, Min-Energy-DC is better for CDNs with regular user activity patterns, where DCs can be safely reconfigured to the sleep mode during off-peak hours without the need to worry about sudden increase in request volumes.

4.3.3. Working as a Whole System

We now briefly describe how they work together as an energy-aware CDN management system. As MA and DCA continuously report up-to-date condition information on CDN network, DCs and servers to MS, the management system is always aware of the current CDN condition. Hence, MS forwards updated information to EAA, which is the key component of our scheme for decision-making purpose. EAA has two main responsibilities. Firstly, as EAA takes input from MS (as illustrated in Figure 3), it makes decisions on which specific DC that each content request is resolved to, and send corresponding request mapping instructions to RTM. RTM then instructs DNS to resolve individual content requests to their designated DCs. Secondly, after EAA makes decisions on request mapping, it is able to determine which specific servers are safe to be turned on or off. It then instructs EASC to remotely configure the specified servers to on or off status. Such a management process repeats as MA and DCA keep refreshing CDN condition information in the system. As a result, the CDN energy consumption is optimized by EAA's instructions to RTM and EASC on request mapping and server on/off reconfigurations respectively. Meanwhile, the CDN QoS performance is assured during EAA's decision-making process by considering server and network load conditions.

4.4. Performance Evaluation

In order to evaluate the performance of the proposed scheme, we use the real topologies from GEANT [24] and Internet2 networks [25], which are two interconnected autonomous domains in Europe and US respectively. Altogether, there are 34 PoP nodes that are distributed over Europe (25 nodes) and US (9 nodes), as well as 55 (bi-directional) network links interconnecting them (including 5 inter-domain links). We set 9 DCs to be deployed in the network (5 in GEANT and 4 in Internet2) and assume a 1:1 over-subscription ratio, which means the DC servers have *just* enough capabilities to serve all users in the network *simultaneously*. With respect to the conventional DC deployment strategy in a CDN [22], these 9 DCs are deployed to the 9 PoP nodes whose associated cities have the highest local populations.

We use the content request and traffic traces from the ClarkNet WWW server (an ISP covering Washington DC metro area) which is publicly available at the Internet Traffic Archive [26], and apply them to all PoP nodes according to the following approach. Firstly, we assume that end-users at all PoP nodes initiate content requests by following the same *pattern* as shown in the trace. Thereafter, the exact number of requests at each PoP node is approximated by applying a specific scaling factor to the request volume in the trace, which is proportional to the PoP's local population. This is necessary as the population associated to each PoP can vary substantially in reality.

As previously described, the proposed scheme takes into account time zone differences among PoP nodes in the CDN. Therefore, in the experiments, we identify three representative snapshots that reflect the changes in the CDN request volume within a single day while considering the geographical factor. For each snapshot, we calculate the overall request volume over one-minute period. These snapshots are listed in Table 1, where they are referred to as scenarios #1, #2 and #3.

Table 1 – Representative User Activity Scenarios

#	**USA (GMT-8 to GMT-5)**	**EU (GMT to GMT+2)**
	User Activity	*User Activity*
#1	Medium	Off-Peak
#2	Off-Peak	Peak
#3	Peak	Medium

With the inputs specified above, we compare the performances of the following schemes.

Firstly, with the objective of minimizing DC energy consumption, we compare the request mapping policies of Min-Energy-Dist and Min-Energy-DC, which have been described in the previous section. Meanwhile, we also consider a reference scheme as a benchmark comparison:

Min-Dist: The scheme, as described in [27], optimizes content delivery efficiency through mapping requests locally or nearby. However, no energy awareness was considered and all servers are kept active regardless of the current request volume, which matches the common practice in DCs [23].

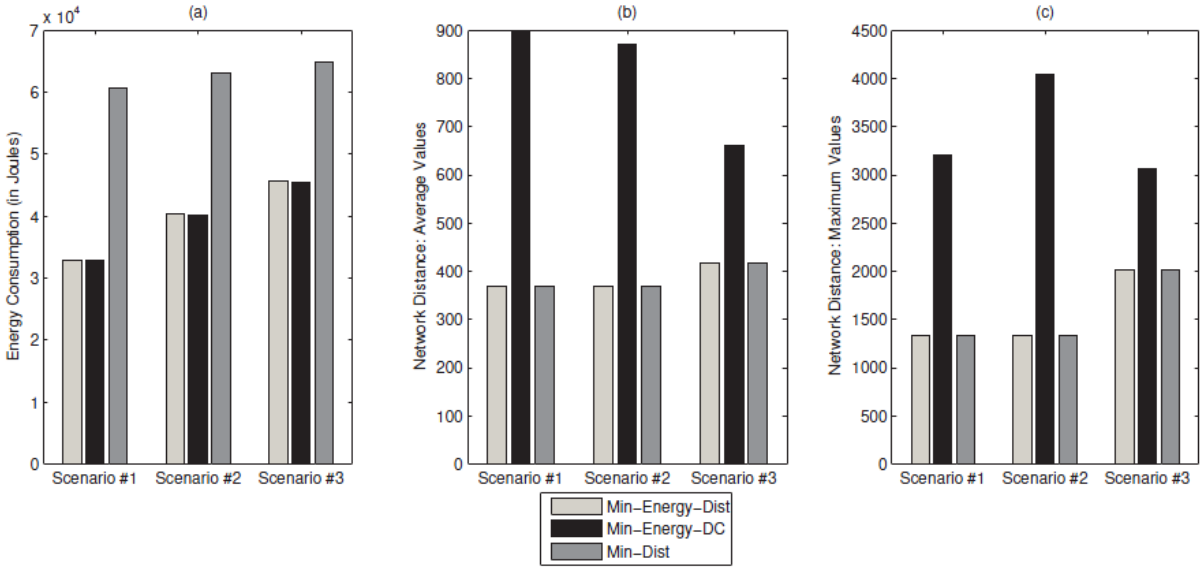


Figure 4. Performance comparison among schemes with respect to different user activity scenarios: a) energy consumption; b) average network distance; and c) maximum network distance

The experimental results are shown in Figure 4, in which the following two metrics are compared in each of the three user activity scenarios:

- Energy consumption (in Joules): the amount of energy consumed by each scheme over the one-minute period, which is used to evaluate *energy-saving performance*.
- Network distance: the network distances traversed by content objects in each scheme while they are being delivered from DCs to end-users. This is calculated through aggregating the weights of links along the content delivery paths, which are proportional to real end-to-end delay according to the settings of GEANT and Internet2 networks [24][25]. Both *mean* and *maximum* values are investigated to reflect both average and worst cases in end-to-end delay, which are used to evaluate the CDN QoS performance.

It can be immediately observed from Figure 4(a) that the proposed scheme can substantially reduce energy consumption of the servers in CDN data centers. In the three user activity scenarios, energy consumption has been reduced by 45.9%, 36.4% and 30.0% respectively. Generally speaking, the higher the content request volumes are, the less energy consumption can be saved as more active servers are needed to serve the requests.

From Figure 4(b) and (c), it can be seen that regarding the average and maximum network distances traversed by content requests, Min-Energy-Dist and Min-Dist have achieved the

same results. This shows that the Min-Energy-Dist policy is capable of significantly reducing DC energy consumption without compromising the CDN QoS performance. Compared with these two policies, Min-Energy-DC has increased network distances while achieving the same amount of energy saving as Min-Energy-Dist. This is because requests are mapped to fewer DCs instead of being resolved locally, and some requests need to travel among PoP nodes in order to be served.

5. Conclusions

The issues related to horizontal management of virtualized resources can be categorized for wireless and wired virtualized networks separately due to the different nature of these networks. The key research challenge is to deal with end-to-end resource management by taking into account network heterogeneity. We first address the issue of virtual network embedding in multi-domain environments with an initial problem formulation to be presented in section 2. In addition, section 3 introduces a systematic agent-based virtual resource allocation that aims to achieve coordinated and dynamic resource re-optimizations by means of intelligent and learning agents. Detailed solutions will be available in our future work.

References

[1] M. Chowdhury*et al.*, "PolyViNE: policy-basedvirtual network embedding across multiple domains," *Proc. ACM VISA*,2010, pp. 49-56.

[2] I. Houidi*et al.*, "Virtual networkprovisioning across multiple substrate networks," *Comput. Netw.*, vol. 55,no. 4, pp. 1011-1023, March 2011.

[3] R. Mijumbi, J.L. Gorricho, J. Serrat, M. Claeys, F. De Turck and S. Latre, "Design and Evaluation of Learning Algorithms for Dynamic Resource Management in Virtual Networks", IEEE/IFIP Network Operations and Management Symposium (NOMS), Krakow, Poland, 2014. Accepted November 2013.

[4] A. Fischer, J. Botero, M. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey", IEEE Communications Surveys Tutorials, PP 1-19, 2013.

[5] S. Sutton G. Barto, "Reinforcement Learning: An Introduction", Cambridge MA USA MIT Press, 1998.

[6] J. Lu, and J. Turner, "Efficient mapping of virtual networks onto a shared substrate", DCSE department, Washington University in St Louis, Technical Report, Vol.35, pp.1-11, 2006.

[7] M. Chowdhury , M. Rahman and R. Boutaba, "ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping", IEEE/ACM Transactions on Networking, VOL. 20, NO. 1, 2012.

[8] S. J. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach", Prentice Hall, Englewood Cliffs, New Jersey, 2010.

[9] I. Houidi, W. Louati and D. Zeghlache, "A Distributed Virtual Network Mapping Algorithm", IEEE International Conference on Communications, pp 5634 - 5640, 2008.

[10] J. Infuhr, and G. R. Raidl, "Introducing the virtual network mapping problem with delay, routing and location constraints", 5th international conference on Network optimization, Springer-Verlag, pp. 105 - 117, 2011.

[11] A. Jarray and A. Karmouch, "VCG auction-based approach for efficient Virtual Network embedding", IFIP/IEEE International Symposium on Integrated Network Management, pp. 609-615, 2013.

[12] IBM, "IBM ILOG CPLEX Optimizer", http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/about/

[13] NS-3 Consortium, "Network Simulator 3", http://www.nsnam.org/

[14] The CAIDA Anonymized Internet Traces 2012 - 20 December 2012, equinix sanjose.dirB.20121220-140100.UTC.anon.pcap.gz, http://www.caida.org/data/passive/passive_2012_dataset.xml

[15] A. Medina, A. Lakhina, I. Matta, J. Byers, "BRITE: An Approach to Universal Topology Generation", 9th IEEE International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Page 346, 2001.

[16] E. Nygren, R. K. Sitaraman, and J. Sun, "The Akamai network: a platform for high-performance internet applications," *SIGOPS Oper. Syst. Rev.*, vol. 44, pp. 2–19, August 2010.

[17] C. Ge, Z. Sun, and N. Wang, "A survey of power-saving techniques on data centers and content delivery networks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1334 –1354, 2013.

[18] L. Chiaraviglio and I. Matta, "Greencoop: cooperative green routing with energy-efficient servers," in Proc. *ACM e-Energy'10*. New York, NY: ACM, 2010, pp. 191–194.

[19] V. Mathew, R. K. Sitaraman, and P. Shenoy, "Energy-aware load balancing in content delivery networks," in Proc. *IEEE INFOCOM'12*. Florida, OL: IEEE, March 2012, pp. 954–962.

[20] P. X. Gao, A. R. Curtis, B. Wong, and S. Keshav, "It's not easy being green," in Proc. *ACM SIGCOMM'12*. New York, NY, USA: ACM, 2012, pp. 211–222.

[21] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: a view from the edge," in Proc. *ACM IMC'07*. New York, NY, USA: ACM, 2007, pp. 15–28.

[22] R. Krishnan, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao, "Moving beyond end-to-end path information to optimize cdn performance," in Proc. *ACM IMC'09*. New York, NY, USA: ACM, 2009, pp. 190–201.

[23] C. Ge, N. Wang, and Z. Sun, "Optimizing server power consumption in cross-domain content distribution infrastructures," in Proc. *IEEE ICC'12*, June 2012, pp. 2628 –2633.

[24] Geant project home. [Online]. Available: www.geant.net

[25] The internet2 network. [Online]. Available: www.internet2.edu/network/

[26] Traces in the internet traffic archive. [Online]. Available: http://ita.ee.lbl.gov/html/traces.html

[27] J. M. Almeida, D. L. Eager, M. K. Vernon, and S. J. Wright, "Minimizing delivery cost in scalable streaming content distribution systems," *IEEE Trans. on Multimedia*, vol. 6, no. 2, pp. 356–365, 2004.