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 topic is the description of Veganet, a virtualised experimentation platform built on top of actual production networks. VegaNet uses a lightweight probing mechanism to provide a consistent connectivity view as in the underlying physical production network, so that experimentation can be performed under the realistic network conditions. It uses virtualization to host multiple experiments on a single physical machine, while reflecting the current connectivity status to each hosted experiment in an accurate and timely manner. We prototype VegaNet and empirically evaluate its effectiveness atop a real-life production network that is currently deployed in a national country.

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.

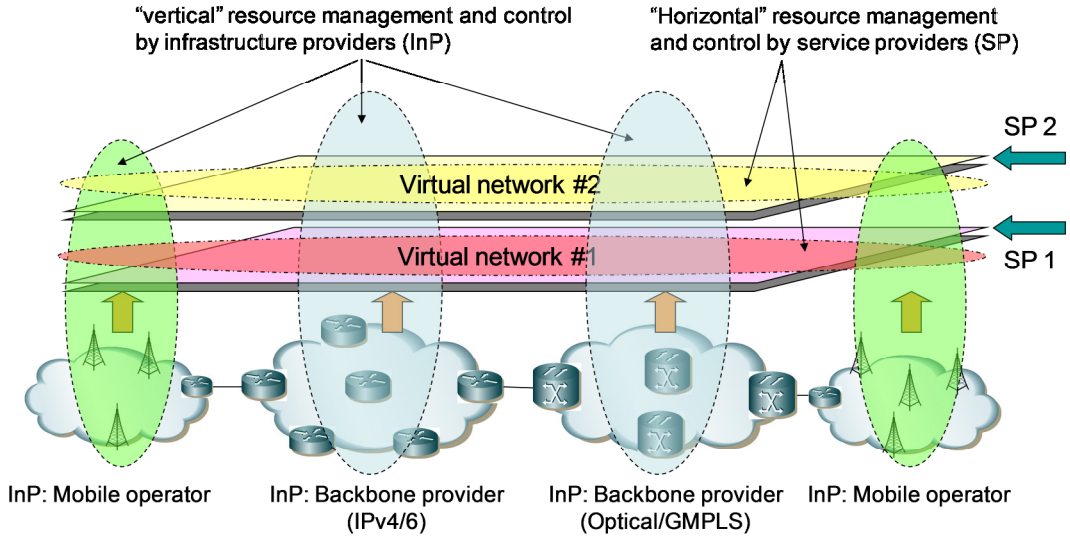

Figure 1. Two Dimensions of Management of Virtualised Networks

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 three important aspects are addressed in Sections 2, 3 and 4of 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 little 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), which makes 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.

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

$$M_{VNP}^{N}(m^{v}) = M_{VNP}^{N}(n^{v}), \text{iff} m^{v} = n^{v}$$

subject to

$$c(n^{v}) \leq c(M_{VNP}^{N}(n^{v}))$$

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

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 corresponding PIP. 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 and Negotiation

In recent times, there has been a growing interest from both industry and academia in the concept of Autonomous systems [3]. This concept has been supported by the growing complexity of systems, and the overall idea is to allow a given system take care of its maintenance with minimal human intervention. As defined by [3], the term "autonomic" comes from an analogy to the autonomic central nervous system in the human body, which adjusts itself to many situations automatically without any external help. According to [3], an autonomous system should posses the different self-CHOP characteristics, which are: self-**C**onfiguring – ability to adapt to changes in the system, self-**H**ealing – ability to recover from detected errors, self-**O**ptimizing – ability to improve use of resources and self-**P**rotecting – ability to anticipate and cure intrusions. We show these characteristics diagrammatically in Figure 2.

Figure 2: Attributes of an Autonomous System

For a system to be autonomous, the actors involved should be able to manage themselves, while at the same time communicating in order to influence each other. An important aspect of such systems is that while each of them can take its own decisions, they should cooperate and coordinate at a higher level so as to achieve an overall system goal.

3.1. Multi-Agent Systems

When more than one agent interacts with each other, the resulting system is called a multi-agent system [4]. A multi-agent system (MAS) can be defined as a group of autonomous, interacting entities sharing a common environment, which they perceive with sensors and upon which they act with actuators [5]. Depending on the application, the interaction between

the agents in a MAS can either be cooperative or competitive. Multi-agent systems have – for some time now – been a subject of research for applications in fields which require autonomous and intelligent behaviour such as Robotics, distributed control and telecommunications [4], [6].

Since solving the resource allocation problem of virtual networks is an NP hard problem, we propose to decentralize the solution by the use of a multi-agent system. This is also practical since the various actors in network virtualization are usually different organizations, with different objectives. We are also mindful of the fact that some optimization decisions may require negotiation and/or cooperation between the involved actors, and that each situation-action scenario may be unique; which is why we seek an autonomous solution, with learning capabilities. As stated by [7], multi-agent systems are ideal for problems that require autonomous decision making capabilities. Introducing Reinforcement Learning allows the multi-agent system to adapt to changes, such as the changing resource capacities, resource failures, introduction of new resources and new customer requests. The agents learn how to respond to such occurrences by remembering how well or badly they have dealt with related situations in the past.

To summarize, 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.

3.2. Objective and Contribution

The aim of this work is to propose a solution for an efficient, dynamic and context aware allocation of resources in virtual networks, which is based on Intelligent and Learning Agents. In our proposal, the agents that represent the different virtual networks and the substrate network form a multi-agent system, and these agents cooperate with each other so as to achieve not only their individual goals, but also the overall network goal. Specifically, we propose a reinforcement learning algorithm on the basis of which the agents can perform the dynamic allocation of resources. The aim is to achieve an autonomous, distributed and dynamic resource management solution that is also based on user context, so as to support the requirements of next generation virtual networks. We believe that future virtual networks – in addition to efficiency – should also make decisions based on the perceived user context.

The work of this proposal differs from previous and related works in the following aspects; the current solutions to dynamic resource allocation make simplifying assumptions of unbounded resources so as to overcome the virtual network embedding problem. We propose to use distributed reinforcement learning (RL) for this assignation problem. Distributing the solution by use of intelligent agents makes each of the agents to be concerned only by the

optimization of its virtual network. However, cooperation capabilities allow the agents to achieve overall network objectives. The use of RL introduces learning capabilities which we exploit in order to make the resultant system self adaptive and context aware. To the best of our knowledge, no solution to this problem has been proposed based on reinforcement learning and multi-agent systems. This approach combines both advantages of centralized as well as distributed management, as the agents make independent decisions, but also share information which makes them always aware of the best decisions on a global scale.

We however, are mindful to note that as [8] states; absolute optimization is difficult and is usually unachievable. Specifically, optimization of NP hard problems similar to the one we are faced with could lead to computationally intractable solutions [9]. We therefore seek to only make improvements through introducing learning capabilities, as well as context awareness, with an aim of achieving better resource utilization efficiency as well as better customer experiences.

3.3. Motivation of using a Multi-Agent System approach

An intelligent agent can be considered to have the characteristics of; Autonomy - ability to have control over its own actions without any human or other forms of intervention [10], Pro-activeness – take actions directed to achieving a specific goal without waiting for inputs [11], Reactivity – ability to perceive changes in its environment as take appropriate actions [12], and Social ability – ability to cooperate with other agents so as to achieve specific goals [13]. [14] proposes the agent paradigm as a methodology appropriate to design autonomous computer systems.

While we choose to propose a solution for the dynamic allocation of resources in virtual networks, we also know that this problem is NP hard, the very reason that many approaches to this solution have always made many simplifying assumptions. Our proposal is to distribute the solution over the involved actors, and as [15] and [16] argue, we believe that a “divide and conquer” approach to virtual network resource management would be suited for the nature of the problem at hand. For this reason, the subject of this thesis proposal requires a distributed, cooperative and autonomic solution to network resource management. [17] argues that the technology of intelligent agents and multi-agent systems will alter radically the way in which complex, distributed, open systems are conceptualized and implemented. Even more, [7] proposes that a multi-agent systems (MAS) approach is much better suited for autonomic computing systems which must self-configure, self-protect, self-heal, and self-optimize on both local and system levels.

We also require that our distributed systems elements are able to coordinate among each other so as to achieve the overall system objectives, but not giving up on their individual objectives. As shown by [18], a coordinated approach to virtual network resource allocation outperforms uncoordinated approaches. Our solution also requires a proactive as well as a

reactive approach to resource management. [19], [20], [21] and other works apply the coordination and negotiation capabilities of agents in their respective solutions, while [22] exploits the proactivity property of agents.

There are also many other proposed and implemented applications of agents and multi agent systems already done. One case is where [23] proposes that each layer of the ISO OSI model can be thought of as an agent and therefore each node can be represented as a multi-agent system.

[24], [25] and [26] all propose intelligent agents for autonomic management of resources with constraints in different scenarios. [27] designed and implemented a multi-agent system that provides intelligence to a distributed smart grid. [28] proposes a multi-agent based approach for dynamic reconfiguration of a distribution network.

The above works show applications of multi-agent systems in areas whose objectives are autonomous, distributed and cooperative systems, which are similar to the interests of this proposal. It is based on the capabilities of agents and multi-agent systems as illustrated in the respective works, that we propose a multi-agent based approach as a solution to the resource allocation requirements in next generation virtual networks.

3.4. Motivation of using a Reinforcement Learning approach

We seek a solution that is self adaptive, which means that even when presented with unique, previously not encountered cases, the agents should be able to make autonomic decisions. Introducing learning would give us the opportunity to better characterize the resource allocation problem on two fronts; first, the overall idea is that after some time, the agent does not have to explore the whole space of possible solutions, but would exploit its previous knowledge – hence reducing the space and time complexity that has always made this kind of problem computationally intractable. It will be aim of this thesis to develop agents that learn the best resource allocation policy without having to explore the whole space of possible solutions in the long run. Second, this would also allow for the agents to dynamically adapt to network conditions and customer requirements, which – to the best of our knowledge – has not been considered in any of the solutions to this problem. Even more interesting, Reinforcement learning allows us to carry out utility based learning, and hence the optimisation parameters can always be dynamically changed by changing the reward function, and overall reward policy.

Reinforcement Learning based approaches have been used in solutions to problems that have similar requirements to ours. [29] and [30] propose and simulate dynamic resource allocation in telecommunication networks using reinforcement learning and they show improvements introduced by learning compared to other solutions. [31] proposes an approach for dynamic

resource allocation in Clouds based on maximization of a utility function. This is similar the concept of Reinforcement Learning.

[32] and [33] use reinforcement learning to solve problems that have been categorized as NP hard, and report improvements produced by learning. Specifically, [34] and [35] propose solutions to NP hard problems based on distributed reinforcement learning. All the problems solved in the above four cases are very much related to the problem we intend to solve.

We therefore propose a solution based on reinforcement learning in distributed agents with expectations that adopting the solution to virtual networks would not only give a solution to the NP hard resource allocation problem, but that this solution will be better than those in the currently available approaches.

4. VegaNet: A Virtualized Experimentation Platform for Production Networks with Connectivity Consistency

4.1. Problem Statement

We start with analysing the failure traces collected on CERNET2, a national IPv6 backbone that interconnects more than 200 educational institutions in China. It consists of 25 Points-of-Presences (PoPs) that span the whole country. Based on our observations, we motivate the problem of maintaining consistent connectivity views between the virtual and physical networks. We collected six months of failure events on CERNET2 from August, 2010 to January, 2011. In the CERNET2 backbone, we deployed 8 traceroute servers, each of which generates traceroute probes to all 25 the PoPs periodically at every 50ms. Within the six-month span, we observe a total of 12,715 link failures. Among all the 62 links that are covered by our measurements, around 80% of them experience at least one failure. The distribution is heavy-tailed, and around 20% of the links have at least 1,000 failures. All the failures that we observe are short-term, and will recover to the normal state afterwards. About 13% of the failures last for less than 500 ms, and over 96% of the failures last for less than five seconds. Thus, link failures are not uncommon, and it is crucial to expose such failures for network experiments in virtual networks.

We are interested in building a virtual network for network experiments atop a production network, such that the virtual network serves as an experimental testbed to reflect the realistic behaviour of the underlying network, i.e., capturing most (if not all) network events incurring in the network. We observe that most failures are short-term and it is challenging to capture them. Existing virtual networks (e.g., Emulab [36], PlanetLab [43] and VINI [39]) provide experimental platforms that can address connectivity and resource provisioning. How- ever, to the best of our knowledge, none of the existing virtual networks specifically provides solutions to accurately and timely capturing the current connectivity status of the physical network. Enabling the virtual network to mirror the physical network connectivity is crucial, so that network experiments are conducted under the current data forwarding conditions in the underlying physical network.

Unfortunately, mirroring the physical network connectivity in a virtual network is a non-trivial task. In particular, if the physical network experiences a link failure, then we argue that this can lead to inconsistent views in both the virtual and physical networks. To motivate, Figure 3 shows a small-scale virtual network, in which we attach a virtual router VR_i to a physical router R_i, where i = 1, 2, 3, 4. Ideally, the virtual network should have a consistent view of the connectivity status as in the physical network. Figure 3(a) illustrates this ideal scenario, in which no link failure occurs. Suppose now that link R_1 -R_3 fails. Then there are three scenarios where inconsistency can occur.

- After failure detection (see Figure 3(b)). Both physical routers R_1 and R_3 detect the failure of link R_1 -R_3 and trigger routing reconvergence. However, virtual

routers VR1 and VR3 may still treat link R1 -R3 is intact and will not immediately recompute new routes.

- After rerouting (see Figure 3(c)). R1 and R3 reroute and form the route R1 -R2 -R4 -R3 , whose number of hops is four. If VR1 and VR3 do not update the routing status immediately, then they will treat the route VR1 -VR3 is directly connected and has only one hop.
- After failure recovery (see Figure 3(d)). New routes, such as VR3 -VR4, may be formed in the virtual network during the failure of R1 -R3 . If the link R1 -R3 is recovered, then R1 and R3 revert to use the original route R1 -R3 as in Figure 3(a), so the physical route between R3 and R4 switches back to R3 -R1 -R2 -R4 . However, VR3 and VR4 may not yet capture this change immediately and still believe VR3 -VR4 is a one-hop route.

From the above examples, there are at least two inconsistent views between the virtual and physical networks: (i) the virtual network cannot timely capture link failures observed by the physical network, and (ii) the virtual routes and the physical routes have different hop information. This inconsistency is further complicated by the fact that network failures are prevalent in production networks and most of the failures are short-term. Thus, the virtual and physical networks can often have inconsistent views.

To maintain the connectivity consistency, a naive approach is to have the physical routers report the up-to-date connectivity status immediately to their attached virtual routers, but this "bottom-up" approach requires re-engineering of the physical routers and is generally infeasible. A more practical approach is to have virtual routers generate frequent probes using traceroute. However, traceroute involves a large volume of probes and may overload the physical network. More importantly, traceroute may be disabled in physical routers, especially in a production network, due to the privacy issue [44]. Therefore, this article aims to address the following question: How can we develop a virtual network that provides connectivity and resource provisioning for network experiments as in existing platforms [36][43][39], while maintaining accurate and timely consistent connectivity views between the virtual and physical networks?

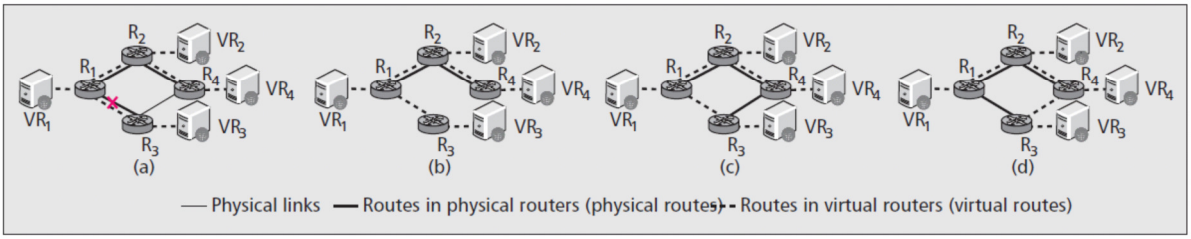

Figure 3. An Example: inconsistency between physical and virtual routes occurs at different stages after the network failure occurs.

4.2. Overview

We propose VegaNet, a virtual network architecture that provides an experimental platform atop a physical production network. Its main goal is to maintain a consistent connectivity view between the virtual and private networks, so that network experiments running atop the virtual network are conducted under the realistic data forwarding conditions of the physical network. In this section, we overview the design of VegaNet, and justify the design features achievable by VegaNet.

VegaNet is a network of nodes that can be deployed on regular PCs or servers. Each VegaNet node can be viewed as a software-based virtual router. It is based on software implementation and supports the basic routing functionalities as seen in a physical router. It is directly attached to a production router and seeks to maintain the same connectivity view with the production router. In this setting, virtual links between two neighbouring VegaNet nodes are used to emulate physical links between neighbouring production routers that they are attached to. To host multiple network experiments, we divide the resources (e.g., CPU, memory) of a VegaNet node into slices, each of which can be independently owned by a network experiment. VegaNet nodes interconnect with each other through tunnelling and exchange control information. Note that VegaNet does not require re-implementation of the production routers and hence it will not interfere in the production network operations.

In a high level, a VegaNet node allows different network experiments to have their own routing protocols, network services, and data/control planes. Figure 4 shows an architectural view of VegaNet when it is deployed in a production network. The packets of different new network protocols, such as OpenFlow [45] and DONA [46], can be generated by external users, and be multiplexed into VegaNet nodes and their attached physical routers. Packets with different protocols will be encapsulated with new IP packet headers in VegaNet nodes. The destinations of the new packets are set to their neighbour VegaNet nodes according to the corresponding forwarding information base (FIB). Thus, these packets can be delivered between VegaNet nodes by the production routers. Each VegaNet node works like a physical router, as it forwards packets over the production network. When the packets reach the destination VegaNet node, they will be demultiplexed into the corresponding applications.

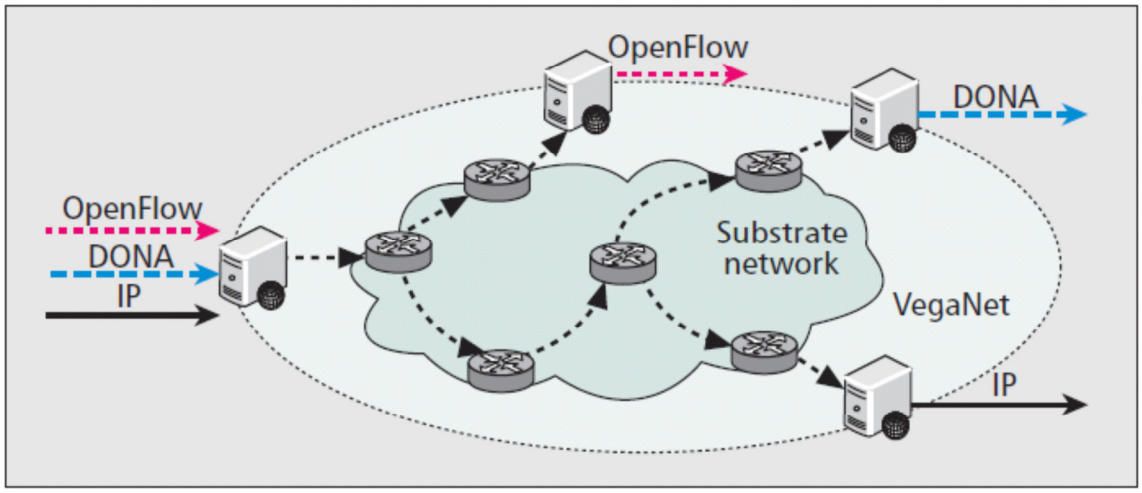

Figure 4 The architectural view of VegaNet atop a physical network.

In a low level, a VegaNet node provides routing and for- warding separation for different network experiments. Figure 5 shows the internal implementation of a VegaNet node. We divide the routing operations of a VegaNet into different slices for network experiments. We implement a router manager (RM), which is responsible for managing all slices. Each slice has its own control object (CO), which determines the routing policy specifically within the slice (or experiment). Each CO computes the forwarding entries based on its routing policy. The forwarding entries of all COs will then be aggregated in a Forwarding Information Base (FIB) (see flow (1) in Figure 5). There can be multiple FIBs, each of which corresponds to a network protocol with its own address format. For example, DONA and OpenFlow may have their own FIBs. We implement a forwarding object (FO), which manages all FIBs. To forward a packet, the FO looks up the FIB with regard to the network protocol and forwards each packet to the correct network interface. By separating the routing and forwarding functions, the slices do not need to coordinate with each other on how to interact with the low-level (i.e., hardware) network interfaces for packet forwarding, which is now centrally handled by the FO. In addition, inside the RM, we implement a forwarding detection object (FDO), which is responsible for detecting connectivity changes as indicated in the FIB (see flow (2) in Figure 5), and notifying the changes to the COs in different slices so that they can recompute new routes (see flow (3) in Figure 5).

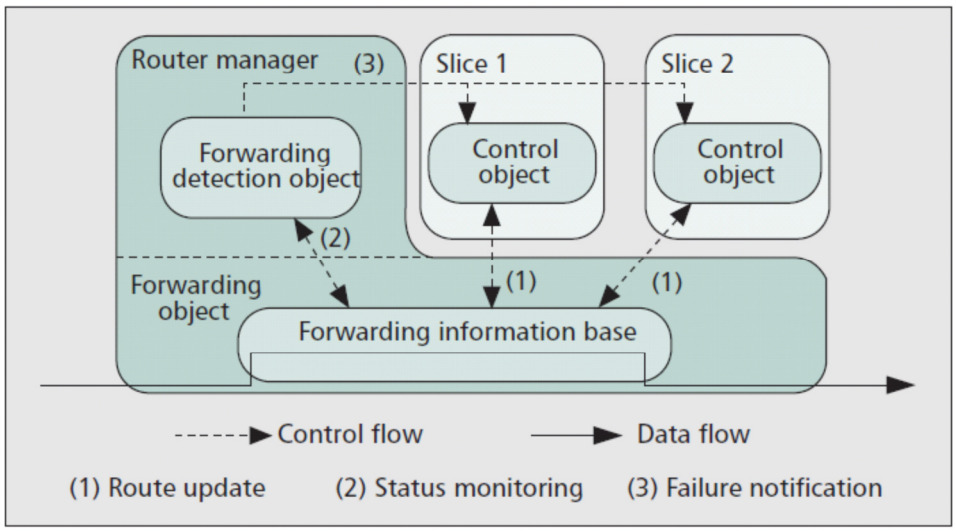

Figure 5. The internal implementation of each VegaNet node.

VegaNet seeks to achieve the following design features:

- Flexible and realistic experimentation. We multiplex traffic with different network protocols into the underlying production network. This multiplexing operation is transparent to the user applications. Aside from the user traffic in VegaNet, the underlying production network also carries the regular traffic within its operation. Such traffic provides a realistic workload pattern for network experiments in VegaNet. The similar design feature has also been addressed in the literature (e.g., VINI [39]).
- Fair resource allocation. Resources within a VegaNet node, such as CPU and memory, are exclusively used by specific slices (or COs) [47]. Also, based on the existing performance isolation mechanism for virtualization, we can provide fair resource allocation for different slices.
- Consistent connectivity views. In each VegaNet node, we unify all forwarding operations of different slices in the FO, which can then easily capture the connectivity status in the physical network by monitoring whether packets can be successfully forwarded. Thus, each experiment hosted in a VegaNet node can obtain a consistent connectivity view with the physical network.

4.3. Achieving Connectivity Consistency

To achieve connectivity consistency in an accurate and timely manner, we leverage active probing, in which each VegaNet node sends probes to their neighbours, and determines

immediately if there exists any connectivity failure or if the failure is recovered. Here, we propose a lightweight adaptive probing algorithm that generates probes based on the current traffic conditions. The probing algorithm is implemented in the forwarding detection object (FDO) of each VegaNet node.

4.3.1. Session Establishment

In VegaNet, each pair of neighbouring VegaNet nodes will have a session, which is used for negotiating control parameters and monitoring connectivity changes in the physical network between the nodes. A session can be viewed as a virtual link between the neighbouring VegaNet nodes. Here, we maintain sessions by extending the state machine design in Bidirectional Forwarding Detection (BFD) [42] in a virtualized network. Although we leverage the basic procedure of BFD for probing, BFD keeps generating probes based on the negotiated probing intervals, without taking into account the current traffic conditions in the data plane. If the data plane is occupied with application traffic, then the probes may further overwhelm the network capacity. We adapt the BFD design to account for the current network traffic conditions, as detailed in Section IV-B.

Each BFD session has three states: (i) DOWN, i.e., the session is torn down; (ii) INIT, i.e., that the session is to be initiated, and (iii) UP, i.e., the session is established. During the session establishment, the pair of neighbouring VegaNet nodes (call them VR_1 and VR_2) will negotiate two sets of parameters: (i) the hop count and (ii) the probing parameters. For the hop count, its main goal is to maintain the connectivity consistency between VR_1 and VR_2. The hop count is defined as the number of physical links between two nodes. For example, if the physical routers R_1 and R_2 are directly connected, then the hop count is given by 3, which includes the links VR_1 -R_1, R_1 -R_2 , and R_2 -VR_2 . We then set the Time To Live (TTL) field in the IP header of the probing packets to be equal to the hop count (e.g., 3 if R_1 and R_2 are directly connected). The intuition is that if the link between R_1 and R_2 is failed and packets are rerouted, then the VegaNet nodes will not receive probing packets from each other, as each probing packet can traverse at most three hops. Thus, the VegaNet nodes can infer that rerouting occurs and can align its connectivity status with that in the physical routers.

For the probing parameters, they are mainly used for specifying the time for detecting a network failure. For each VegaNet node VR_i , the probing parameters include three constants [39]: the Desired Minimal TX Interval (TX_i), Required Minimal RX Interval (RX_i), and Detect Multiple (DM_i). TX_i and RX_i are the minimum sending and receiving intervals supported by the VegaNet node VR_i, respectively, while DM_i denotes the probing timeout period represented as a multiple of the probing interval. Some typical values of such parameters are TX_i = 50ms, RX_i = 50ms, and DM_i = 3 (e.g., see [48]). Then it implies that VR_i sends probes every 50ms and expects to receive probes from each of its neighbours every 50ms. If VR_i does not receive probes from a neighbour for $RX_i \times DM_i$ =

150ms, then it may declare that the link to the neighbour is failed. The probing parameters are included in the control packets during the session establishment process, so that they can be agreed upon by both of the neighbouring VegaNet nodes. It is important to note that each VegaNet node can customize its own set of probing parameters, for example, according to its available link bandwidth.

4.3.2. Lightweight Failure Identification

In VegaNet, each VegaNet node sends probes to each of its neighbours to determine if the virtual link between the node itself and the neighbour is failed, according to the current traffic conditions. VegaNet consists of two types of packet flows: (i) the control packet flow, i.e., session establishment and probing and (ii) the data packet flow, i.e., application traffic generated by the user applications. Both of the control and data flows are interleaved and forwarded by the forwarding object (FO) in each VegaNet node. The intuition here is that if a VegaNet node (say VR_1) can always sends data traffic to its neighbour (say VR_2), then we can infer that the virtual link VR_1 -VR_2 is good and has no failure, without the need of generating additional probes. On the other hand, suppose that the neighbouring VegaNet nodes have no data traffic in between, either because the user applications do not generate any data traffic, or because the virtual link is failed. In this case, they generate probes based on the negotiated probing parameters. If no probes are received between the nodes, then we can infer that the virtual link is failed.

Algorithm 1 shows the pseudo-code of our adaptive probing algorithm, which is called by each VegaNet node (say VR_X) to initiate the sending of probing packets. It extends BFD [42] to account for the data-plane condition. First, VR_X will initialize the packet receiving timer (denoted by ρ_i) and the packet sending timer (denoted by τ_i) associated with its neighbouring VegaNet node VR_i (steps 1 to 5). For ρ_i , it is set to $DM_X \times \max(RX_X , TX_i)$. For τ_i , it is set to $r \times \max(TX_X , RX_i)$, where r is uniformly selected at random between two constants β_{min} and β_{max} such that $0 \leq \beta_{min} < \beta_{max} < 1$. Here, we choose r at random so as to avoid self-synchronization of probing among the neighbouring VegaNet nodes [42]. Also, we generally set β_{max} less than 1 to ensure that the probes reach the other side before the detection timeout (e.g., they may be delayed due to congestion). Here, in our current implementation, we set β_{min} = 0.75 and β_{max} = 0.9.

For each neighbour VR_i in $VR_{[1..n]}$ of VR_X , if there exists traffic sent from VR_X to VR_i , then VR_X can reset the timer τ_i instead of generating additional probes (steps 7-8). Also, if a probe is to be generated, then the TTL of the probe is set to the negotiated hop count to ensure that the probe reaches the neighbour only if the underlying link works (i.e, there is neither failure nor rerouting) (steps 9-10). In the meantime, VR_X checks if it receives any traffic from VR_i before the packet receiving timer ρ_i expires.

If not, then it means that the virtual link VR_x -VR_i is failed (steps 12-14).

Input: n neighboring VegaNet nodes and probing parameters:
$\{VR_1, TX_1, RX_1), \ldots, (VR_n, TX_n, RX_n)\}$;
Output: set of failed links: FailedLinkSet;
1: **for** VR_i in $VR_{[1..n]}$ **do**
2: set timer ρ_i's value = $DM_x \times \max(RX_x, TX_i)$;
3: randomly select $r \in [\beta_{min}, \beta_{max}]$;
4: set timer τ_i's value = $r \times \max(TX_x, RX_i)$;
5: **end for**
6: **for** VR_i in $VR_{[1..n]}$ **do**
7: **if** (exists traffic sent to VR_i before timer τ_i expires) **then**
8: reset timer τ_i before it expires;
9: **else**
10: send a probe with the negotiated hop count;
11: **end if**
12: **if** (no traffic from VR_i when timer ρ_i expires) **then**
13: Add failed link $VR_x - VR_i$ to FailedLinkSet;
14: **end if**
15: **end for**
16: return FailedLinkSet;

Algorithm 1 follows the simplicity of design as in BFD [42] and does not add complicated logic, making it easily implementable. In Section V, we show that the simple design sufficiently achieves our goal of maintaining the connectivity consistency between the virtual and private networks.

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. Section 4 present Veganet, which is a virtualized experimentation network platform for production networks with connectivity consistency. The main focus is on prompt detection and handling of unexpected failures in the underlying physical network. Detailed solutions will be available in our future work.

References

[1] M. Chowdhuryet al., “PolyViNE: policy-basedvirtual network embedding across multiple domains,” Proc. ACM VISA,2010, pp. 49-56.

[2] I. Houidiet al., “Virtual networkprovisioning across multiple substrate networks,” Comput. Netw., vol. 55,no. 4, pp. 1011-1023, March 2011.

[3] Jacob, IBM , R. Lanyon-Hogg , D. Nadgir and A. Yassin, " Practical Guide to the IBM Autonomic Computing Toolkit", IBM, RedBook, International Technical Support Organization, First Edition, 2004.

[4] P. Stone, "Multi-agent systems: A survey from the machine learning perspective", Autonomous Robots, vol. 8, no. 3, pp. 345–383, 2000.

[5] L. Busoniu , R. Babuska and B. Schutter, " Comprehensive Survey of Multi agent Reinforcement Learning", IEEE Transactions on Systems, Man, and Cybernetics - Applications and Reviews, 2008.

[6] H. Parunak, "Industrial and practical applications of DAI: Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence", G. Weiss, Ed. Cambridge, MA: MIT Press, ch. 9, pp. 377–412, 1999.

[7] G. Tesauro , D.M. Chess , W. Walsh , R. Das , A. Segal "A Multi-Agent Systems Approach to Autonomic Computing", Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-agent Systems, 2004.

[8] H. Tianfield, "Multi-Agent Based Autonomic Architecture for Network Management", Proceedings of IEEE International Conference on Industrial Informatics, INDIN, 2003.

[9] 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.

[10] N. Lidula and A. Rajapakse, "Microgrids research: A review of experimental microgrids and test systems", Renewable and Sustainable Energy Reviews, vol. 15, no. 1, pp. 186-202, 2011.

[11] M. Wissner, "ICT, growth and productivity in the German energy sector -On the way to a smart grid?", Utilities Policy, vol. 19, no. 1, pp. 14-19, 2011.

[12] Y. Shoham and K. Leyton-Brown, "Multi-agent systems: algorithmic, game-theoretic, and logical foundations", Cambridge University Press, 2008.

[13] R. Zamora , A. Srivastava and Syukriyadin, "Microgrids for Reliable, Clean, and Efficient Power Delivery", AIWEST 2009, Banda Aceh, , 2009.

[14] N. Jennings, "On agent-based software engineering", International Joint Conference on Artificial Intelligence, Stockholm Vol. 117, Issue 2, pp. 117:277-296, 2000.

[15] G. Even , J. Naor , S. Rao and B. Scieber, "Divide-and-Conquer Approximation Algorithms via Spreading Metrics", Journal of the ACM, Vol. 47, No. 4, pp. 585-616, 2000.

[16] A. Do-Sung and A. Leon-Garcia, "Virtual network resources management: a divide-and-conquer approach for the control of future networks", IEEE, 1998.

[17] M. Wooldridge, "Agent-based software engineering", IEEE Proceedings on Software Engineering, 1997.

[18] P. Perry , S. Balasubramaniam , J. Mineraud , B. Jennings , L. Murphy "Coordinating Allocation of Resources for Multiple Virtual IPTV Providers to Maximize Revenue", IEEE TRANSACTIONS ON BROADCASTING, VOL. 57, NO. 4, 2011.

[19] M. Rizzo and I. Utting, "A negotiating agents model for the provision of flexible telephony services", Proceedings of Third International Symposium on Autonomous Decentralized Systems, 1997.

[20] M. Cao and Y. Feng, "Negotiating agent architecture and communication model", 3rd International Conference on Intelligent System and Knowledge Engineering, 2008.

[21] J. Debenham and E. Lawrence, "Negotiating Agents for Internet Commerce", International Conference on Computational Intelligence for Modelling, Control and Automation, 2006.

[22] H. Tair , M. Zemerly , M. Al-Qutayri and M. Leida, "Proactive integrated mobile services using multi-agents system", IEEE GCC Conference and Exhibition, 2011.

[23] D. Legge and P. Baxendale, "Agent-Based Network Management System", AISB, 2002.

[24] M. Jacyno , S. Bullock , T. Payne , N. Geard and M. Luck "Autonomic Resource Management through Self-Organising Agent Communities", Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems, 2008.

[25] H. Chen , W. Zhou and L. Liu, "An Approach of Agent-Based Architecture for Autonomic Network Management", 5th International Conference on Wireless Communications, Networking and Mobile Computing, 2009.

[26] [Liu06] Z. Liu, "Autonomic Agent-based Storage Management for Grid Data Resource", Second International Conference on Semantics, Knowledge and Grid, 2006.

[27] H. Feroze , M. Pipattanasomporn and S. Rahman, "Multi-Agent Systems in a Distributed Smart Grid: Design and Implementation", IEEE/PES Power Systems Conference and Exposition, PSCE, 2009.

[28] Z. Li , X. Chen , K. Yu , B. Zhao and H. Liu "A novel approach for dynamic reconfiguration of the distribution network via multi-agent system", Third International Conference on Electric Utility Deregulation and Restructuring and Power Technologies, 2008.

[29] A. Stefan , M. Ramkumar , R. Nielsen , N. Prasad and R. Prasad "A QoS aware reinforcement learning algorithm for macro-femto interference in dynamic environments", 3rd International Congress on Ultra Modern Telecommunications and Control Systems and Workshops, 2011.

[30] F. Bernardo , R. Agusti , J. Perez-Romero and O. Sallent, "Distributed Spectrum Management based on Reinforcement Learning", PROCEEDINGS OF THE 4th INTERNATIONAL CONFERENCE ON CROWNCOM, 2009.

[31] D. Minarolli and B. Freisleben, "Utility-driven Allocation of Multiple Types of Resources to Virtual Machines in Clouds", IEEE 13th Conference on Commerce and Enterprise Computing (CEC), 2011.

[32] T. Gabel and M. Riedmiller, "Reinforcement Learning for DEC-MDPs with Changing Action Sets and Partially Ordered Dependencies", Proceedings of 7th International Conference on Autonomous Agents and Multi-agent Systems, 2008.

[33] G. Czibula , M. Bocicor and I. Czibula, " Reinforcement Learning Model for Solving the Folding Problem", Int.J.Comp.Tech.Appl, 171-182, 2011.

[34] M. Bocicor , G. Czibula and I. Czibula, "A Distributed Reinforcement Learning Approach for Solving Optimization Problems", Proceedings of the 5th WSEAS, Recent Researches in Communications and IT, 2011.

[35] C. Gabriela , M. Bocicor and I. Czibula, "A Distributed Q-Learning Approach to Fragment Assembly", SIC Journal, Studies in Informatics and Control, Vol. 20, Issue. 3, 2011, pp. 221-232, 2011.

[36] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In Proceeding of OSDI, pages 255–270, 2002.

[37] Deterlab testbed. http://www.isi.edu/deter/.

[38] Planetlab. http://www.planet-lab.org/.

[39] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford. In vini veritas: Realistic and controlled network experimentation. In Proceeding of SIGCOMM, pages 3–14, 2006.

[40] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, Y. Ganjali, and C. Diot. Characterization of failures in an operational ip backbone network. IEEE/ACM Trans. Netw., 16:749–762, 2008.

[41] China education and research network 2 (CERNET2). http://www.edu.cn.

[42] D. Katz and D. Ward. Bidirectional forwarding detection (BFD). RFC5880, July 2010

[43] L. Peterson, A. Bavier, M. E. Fiuczynski, and S. Muir. Experiences building planetlab. In Proceeding of OSDI, 2006.

[44] L. Subramanian, V.RN. Padmanabhan, and R.H. Katz. Geographic properties of internet routing. In Proceedings of USENIX ATC, 2002.

[45] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKe- own, and G. Parulkar. Can the production network be the testbed? In Proceedings of OSDI, 2010.

[46] T. Koponen, M. Chawla, B. Chun, A. Ermolinskiy, K. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. In Proceedings of SIGCOMM, pages 181–192, 2007.

[47] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neuge- bauer, I. Pratt,

and A. Warfield. Xen and the art of virtualization. In Proceedings of SOSP, pages 164–177, 2003.

[48] Cisco Bidirectional forwarding detection. http://www.cisco.com/en/US/docs/ios/120s/feature/guide/fs bfd.html.

[49] M. Xu, Q. Li, P. P. C. Lee, Y. Peng, and J. Wu. VegaNet: A virtualized experimentation platform for production networks with connectivity consistency. Technical report, DCS, Tsinghua University, May 2012. http://routing.netlab.edu.cn/tiki-download file.php?fileld=183.