Policy-Assisted Planning and Deployment of Virtual Networks

Steven Davy*†, Joan Serrat†, Antonio Astorga† , Brendan Jennings*, Javier Rubio-Loyola‡
* NMG, Universitat Politènica de Catalunya,
Barcelona, Spain
† TSSG, Waterford Institute of Technology,
Waterford, Ireland
‡ CINVESTAV Tamaulipas, Mexico

Abstract—**We present an approach for deploying and subsequently managing a virtual network overlay, which is tailored to an end-user's request. Our approach combines a binary integer optimisation process to decide on the number and placement of virtual routers, and an autonomic network management system that subsequently manages the configuration of the running virtual network. High-level optimisation policies are used to guide the optimisation process to identify a virtual network that favours lower hosting costs or higher network quality (we use mean delays as a quality metric). Low-level deployment policies are generated and used to govern the deployment and management of the virtual networks. Our results indicate that the binary integer optimisation process produces a virtual network that has lower cost as compared to creating a network based on combined shortest paths.**

I. Introduction

Future end user services in the Internet will require a varied set of supporting network functionality. Network topology and functions will need to change in a few seconds, requiring the capacity to bring up or down functional entities to adapt the network to service requirements. To this end virtualisation appears as a technique to configure multiple network entities (network nodes) in a single physical machine that behave differently, but independently of each other. These entities are termed "Virtual Machines (VMs)," where a virtual machine is defined as a software implementation of a computer that executes programs like a real computer. The computational and communications resources provided by a physical machine are partitioned or shared between the VMs, which are fully isolated runtime environments. When the VMs are performing networking functions (store, route and forward functions) we term them virtual nodes and virtual links and hence we talk about "Virtual Networks (VNs)."

As in a conventional physical network, a virtual network has to be pre-provisioned or provisioned on demand to satisfy service user needs. To describe the process we present a generic scenario where a service customer is issuing a service request to a "Virtual Network Provider (VNP)." This request will specify end user service access points to be connected, for instance, to one or more servers for IPTV content delivery. In addition, the customer's request may also specify service quality parameters like maximum available bandwidths or delays. The role of the VNP is to allocate network resources to that user to fulfil his/her needs. This entails identifying the topology and the capacity of the virtual network that is needed for the new service, but without compromising other services currently offered to other users. An intrinsic aspect of this problem is to decide which physical nodes and physical links will be selected to support the virtual network. Note that the options the VNP has to fulfil the customer request are typically uncountable. Therefore, it is necessary to make use of algorithms that help find the solution that best fits not only the customer demands, but also the VNP's own business objectives. In addition, the VNP shall interact with an "Infrastructure Provider (InP)," or many of them if necessary, each owning a physical network. Finally, depending on the agreements between the VNP and the InPs, the former or the latter will be entrusted to deploy and activate the VN.

The problem of creating VNs and assigning virtual to physical resources in particular has attracted considerable attention in recent years. The problem is particularly challenging because it usually yields a NP-Hard problem where the solution can not be obtained by deterministic means and so heuristics must usually be invoked. Hence, different efforts have been undertaken to adopt different heuristics keeping into consideration different problem constraints; we discuss some of these efforts in §II below.

Our approach assumes that requests arrive to the VNP that specify one source and many sinks to be connected by bidirectional links, together with a set of user-related constraints. The VNP considers that request to be fulfilled according its high level policies, also called "Optimisation Policies (OPs)." These policies will favour a solution more or less biased to different VNP business objectives like cost minimisation, loyalty retention or others. Weighting factors of a cost function are derived from these OPs so that its minimisation comes out with the least cost spanning tree that constitutes an initial draft of the VN topology. A refinement process of that topology is then run to determine which of the initially identified physical nodes should host a virtual router. The criteria adopted here is simple because a virtual router will be initiated only in case a routing decision has to be made. Otherwise a simple tunnel is created to bypass the node. Last but not least, the final VPN topology is properly converted into a set of deployment policies (DPs) that will be directly enforced to create the virtual nodes and the virtual links; this step makes use of the AutoI management framework, as described in [1].

The structure of the paper is as follows. After this introduction, §II presents the related work. §III provides an overview of the virtual network management framework proposed by the AutoI project, upon which our solution is built. In §IV we present a detailed description of our approach. §V summarises the main results obtained so far, whereas we present a discussion or our approach and future challenges in §VI. Finally, section §VII contains our conclusions.

II. Related Work

Current efforts researching into the use of virtual infrastructures are doing so de to their ability to abstract the complexity and heterogeneity of physical communication networks. Also value added features such as expandable resources, dynamic deployment and resource migration have seen virtual networks seen as the natural solution to many current ICT problems. The FP7 project GEYSERS are investigating the use of network virtualisation as a means to interconnect virtualised infrastructures that are delivering end users services, where a virtual network is used to abstract the location of the service [2]. The FP7 project SAIL are investigating the concept of Cloud Networking (CloNe) [3] where there vision is to

"To provide cloud network services compliant with application requirements in a dynamic and automated way connecting customers to an operator cloud".

Both of these projects are defining the terms of Virtual Networks and the relationships between virtual network operators and physical network operators. One of the aspects of virtual network deployment that attracted most attention in the last decade has been the assignment of physical resources to given virtual network topologies subject to given constraints. The problem statement and the challenges of its solution are clearly elucidated in works such as [4, 5, 6, 7, 8]. In summary, given a number of VN requests (the request specifies the VN's topology) that can be known in advance or appearing randomly in time, the target is to assign physical resources (nodes and links) of a given substrate net- work topology to satisfy the VNs requirements and at the same time fulfil some goals or constraints. Such goals refer either to the virtual or to the physical network and in earlier works consisted of, for instance, the number of virtual nodes assigned to a physical node, the number of virtual links assigned to a physical link [4], node CPU usage[6], or virtual links capacity [5]. In these works the problem is treated analytically transforming the goals to be achieved into a minimisation of a cost function subject to constraints. But this generally leads to a NP-Hard problem that can only be addressed by applying heuristics. More recently, other papers have appeared tackling the same type of problem with complementary points of view and approaches. Mentioning a few of them: Chowdhury and Boutaba [9] proposes algorithms for VN embedding that differ from other algorithms by introducing coordination between node and link mapping phases. The work of Cai et al. [10]

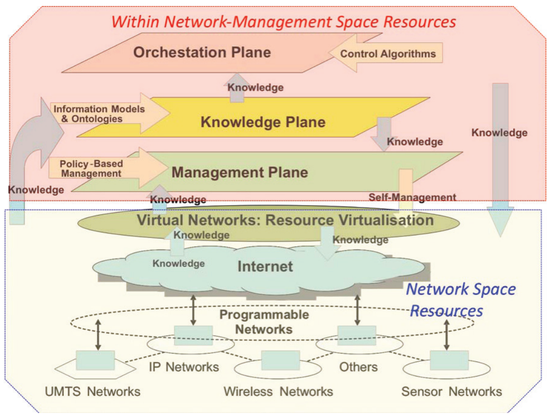


Figure 1. Autonomic Internet Framework.

considers the evolution of the physical network topology (non-static physical network topology). In Razzaq and Rathore [11] virtual nodes are mapped as close as possible thus ensuring that the paths found for the edge mapping are the shortest in length. Finally, service resilience is also used as one of the constraints by Chen et al. [12].

Our paper deals with a different but highly complementary problem. We do not start from a pre-existing VN topology, but from a one-to-many connectivity request that has to be solved by means of a virtual network topology. We address how to plan such a virtual network, taking into consideration the physical network topology and other constraints. Hence, we also output with a VN topology mapped to a physical network, but derived from a different perspective that makes it closer to the real problem faced by VNPs. In our approach we also formulate an optimization problem, related to the Steiner Tree Problem [13], which is known to be NP-Hard. This problem is mostly studied for its application to designing the layout of communication channels on large computer circuits. Beasley et al. [13] presented a simplified linear programming model of the Steiner Tree problem which they solve using the branch and bound algorithm. The model presented in this paper is loosely based on their model, but it is simplified for connecting a single source to multiple targets. The more complex version of Steiner Tree is a Steiner Network problem [14] which aims to solve the problem where there are multiple source and multiple target nodes.

III. Framework

Our solution is built upon the framework developed in the FP7 Autonomic Internet (AutoI) project as depicted in Figure 1. We now present an overview of this framework and focus specifically on the aspects that are of interest to this paper. The AutoI project specified an autonomic management architectural model that incorporates a number of distributed management systems in the network. The model can be described with the help of the abstractions and distributed systems organised as OSKMV (Orchestration, Service Enablers, Knowledge, Management and Virtualisation) planes.

The Orchestration Plane encapsulates the instruments that controls the behaviour of the management systems in the network. Its primary role is to govern, negotiate and federate multiple Autonomic Management Systems. The Service Enablers plane encapsulates functions for the automatic (re)deployment of new services, protocols, resource-facing and end-user facing services. This includes the enablers to allow code to be executed on the network entities and also the service could be activated on demand. The Knowledge Plane encapsulates models and ontologies to provide analysis and inference capabilities; its purpose is to provide knowledge and expertise to enable the network to be self-monitoring, self-analysing, self-diagnosing, and self-maintaining or self-improving. The Management Plane encapsulates the Autonomic Management Systems (AMSs), which are designed to realise autonomic control loops governed via the Orchestration Plane. AMSs are designed to be network embedded, network and service aware, self-adaptive and extensible. These properties make the AMS a highly flexible solution for the management of both physical and virtual networks.

Most relevant to our work is the Virtualisation Plane. It encapsulates software artefacts to facilitate treatment of selected physical resources as a programmable pool of virtual resources that can be organised by the Orchestration and Management Planes into appropriate sets of virtual resources to form components (e.g., increased storage or memory), devices (e.g., a switch with more ports), or even networks. The organisation is done in order to realise a certain business goal or service requirement. The Virtualisation Plane is used by the Orchestration plane to govern virtual resources, and to construct virtual services and networks that meet stated business goals and having specified service requirements. The AMSs of the Management Plane manage, through the Virtualisation Plane, the physical resources, and the construction of virtual resources from physical resources. In AutoI the separation between the virtualisation plane and other planes relieves the other planes from dealing directly with physical resources. Only virtualised resources are manageable via the virtualisation interfaces and also monitoring information about the physical and virtual resources can be requested from the virtualisation interfaces. This separation enables a system-wide management of virtual resources by other planes, while the management of physical resources is done by the virtualisation plane.

In support of the AMS, is the Model-Based Translator (MBT) which takes configuration files compliant with the AutoI Information Model XML Schema and translates them to device specific commands with the aim to eliminate the need to update management software when managing heterogeneous networked entities whose data models may change regularly. The MBT can be used to communicate the configuration of a single device, or an entire network based on a single interaction with the AMS. It is effectively a management abstraction layer that is highly configurable and extensible for new target data models and management protocols. A Policy-based system supports context-aware, policy-driven decisions for management and orchestration activities; it is used as the basis for policy management in the AMS. Finally, the Virtual Component Programming Interface (vCPI) is a modular and scalable system for monitoring and configuring virtual resources. It operates locally—for each node of a physical network there is an embedded vCPI.

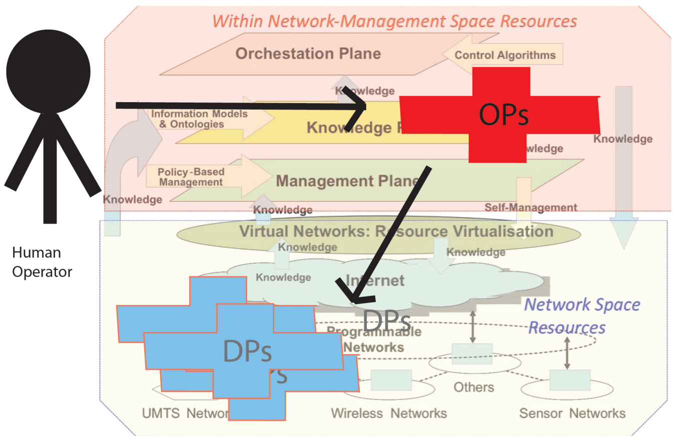


Figure 2. Autonomic Internet Framework Interaction.

IV. Policy Authoring and Optimisation

When a end user requests a virtual network to connect many physically distributed sites together, there may be a variety of potential virtual networks that can be established to meet the end user's demands. The requestor may favour a low virtual router count, or a low hop count between end sites. Other requestors may be interested in low delay across the physical network, or indeed inexpensive hosting of virtual machines. In any case, there is always a trade-off between some of the end-users requirements during the creation of the virtual network. This section describes a policy authoring process that can define the policies that need to be deployed to ensure that end-user requirements are captured and enforced in the virtual network planning process. These policies are defined in light of an optimisation process that also ensures the optimal configuration of virtual resources are deployed and situated correctly across an (AutoI-enabled) physical network. The optimisation process is policy enabled, meaning that the resulting decision can be directed towards favouring virtual networks with different characteristics based on a service request. The interaction with the AutoI framework is depicted in Figure 2

A. Policy Authoring Process

The process for authoring policies is designed to aid an individual or machine in the definition of policies that are conflict free and reflect the requirements of the organisation. We deal with two levels of policies in this scenario. Policies that are defined to guide the formulation of the optimisation problem are henceforth referred to as "Optimisation Policies (OPs)," whilst policies that are defined to enforce the resulting optimal virtual network topology are referred to as "Deployment Policies (DPs)." Deployment policies are particularly difficult to describe by a human operator, even for those with extensive knowledge of the target system. For example, a human operator may not know the current configuration or context of the physical communications network, and may not be able to consider all options for the deployment of virtual routers to support an end user request for a network. For this reason, the authoring of policies are aided by an authoring process that can auto-generate parts of the deployment policies (DPs). The primary steps of the policy authoring process are outlined here and described in more detail below:

1) Define the Optimisation Policies (e.g., “Prefer network quality;” or “Prefer network cost;”);
2) Identify Virtual Topology based on optimisation process;
3) Reduce resulting Virtual Topology in terms of virtual routers and virtual links;
4) Generate appropriate Deployment Policies to realise virtual routers and virtual links.

The OPs are initially described by the human operator to offer guidelines to the optimisation process for building the virtual network. The OPs can be defined to ensure that the optimisation process favours a high quality connected network with low delay, or favour a low cost of running virtual network when network quality of not a major concern. The request from the end-user will clearly state its preferences for network characteristics. The end-user can also specify if they would prefer a balance between the cost of operating the network versus a network with low delay. The end-user indicates this preference by including a SetPreferenceIndicator variable in their policies. The SetPreferenceIndicator value is mapped to a specific variable that is used during the optimisation process known as α. As an example, an OP could be stated as follows:

1) **IF** Prefer_Reduced_Network_Cost **THEN** SetPreferenceIndicator = LOWCOST
2) **IF** Prefer_Low_Network_Delay **THEN** SetPreferenceIndicator = LOWDELAY

The identification of the virtual network topology is the output of the optimisation process. The optimisation problem, described in detail in §IV-B, takes as input the topology of the physically connected network, the link costs, the cost of hosting virtual machines in different physical locations, a single source site and multiple target sites that will make up the virtual network. The links will be bi-directional and so the sites will be able to communicate in both directions. The output of the optimisation process is a least cost spanning tree that originates at the source and only spans to the target sites. The tree also includes edges that connect to intermediate nodes. These will be potential sites for virtual routers. Our algorithm considers the case where there is potentially no feasible solution, in that case, or after a specified amount of time, a negative response can be returned stating that no feasible virtual topology can be generated for the given request.

The reduction of the virtual topology is there to optimise the number of virtual routers in the final network. When a node in the virtual topology is not a site nor a target, then it is a hub node. Hub nodes are used to connect the source node to the target nodes and other hub nodes. However, a virtual router should only be situated at these points if actual routing is required. That is, if a virtual machine at this location must make a decision between edges as to where a packet or flow should be sent. If there is only one edge out and one edge in, then a tunnelling protocol can be used to bypass this node which may originate from an upstream node to the next downstream node. This reduction of nodes is carried out on the virtual topology where ideal placement of virtual routers that connect targets to the source remain.

From the amended list of virtual routers and virtual links, deployment policies (DPs) are generated. These DPs are pushed to the AutoI policy system where that can be used to instruct the vCPI to build the virtual network. The DPs are also instructed to start critical routing services and any other network or customer services that are required before the virtual network can be used.

B. Virtual Topology Optimisation

Consider a graph *G = (V, E)*, where *V* represents the set of physical nodes in the network that can support hosting a virtual router, and *E* is the set of connections between the nodes of *V*. *E* is represented by an ordered pair *(i, j)* ∈*V*. In our model of virtual topology optimisation, there is a single source node *s* that seeks to be connected to a set of target nodes $t \in T \subset V$. We seek to find a least cost spanning tree originating at the source (s) and terminating at all the elements of T. All other nodes are hub nodes that can support routing, or leaf nodes that are not involved in the virtual network. The problem is to find the least cost spanning tree between *s* to all the elements of *T*. The problem can be formulated following the form of a generalised set packing problem. We introduce binary decision variables x_{ij} for every element in *E* indicating whether $(x_{ij} = 1)$ or not $(x_{ij} = 0)$ that edge is included in the solution. Therefore, we can now describe the optimisation problem as follows.

OBJECTIVE:

$$minimise \sum_{ij \in E} k_{ij} x_{ij} \tag{1}$$

CONSTRAINTS:

- Binary Variables:

$$x_{ij} \in \{1, 0\} \quad \forall ij \in E \tag{2}$$

- Source Constraints:

$$-\sum_{j \in E} x_{ji} \leq -1, \;\; i = s \tag{3}$$

- Target Constraints:

$$\sum_{j \in E} x_{ij} - \sum_{j \in E} x_{ji} \leq -1, \;\; \forall i \in T \tag{4}$$

- Hub Constraints:

$$x_{ij} - \sum_{j \in E} x_{ji} \leq 0, \;\; \forall i \in T \tag{5}$$

$$x_{ji} - \sum_{i \in E} x_{ij} \leq 0, \;\; \forall i \in T \tag{6}$$

Note, all constraints are normalised to represent linear optimisation constraints where the RHS is larger then the LHS and where k_{ij} is the weight associated to a particular edge. In (1) we try to minimise the total sum of the weights when decision variables are set. In (2) we insist that the decision variables are 1 or 0. In (3) we describe that the sum of decision variables representing *out* edges (negative) is less then -1, this is to ensure that the decision variable representing the source *out* edge is always selected. In (4), the opposite is the case. We want to ensure that one of the *in* edges is selected, therefore the sum of *in* edges (negative) is less then -1. This constraint is repeated for each target in the network. If a node is designated as a hub, then it can have a higher degree *out* then degree *in*, but only if at least one *in* edge is selected (5). Conversely the opposite is also true, only if and an *in* edge is selected should an *out* edge be selected (6). This will eliminate phantom edges being asserted which originate at hubs. For each *in* edge; the sum of the *in* edge (ij) plus all *out* edges (-ji) has to be less then 0. This will insist that the hubs perform like routers and can split a path out to several destinations towards reaching the optimal topology.

The weights of the graph are calculated based on the policies of the virtual network operator and the end user request. To compute the weight along an edge in the network we use the following formula.

$$k_{ij} = (\alpha - 1)\, d_{ij} + \left(\frac{c_i + c_j}{2} \right) \alpha \tag{7}$$

Where d_{ij} is a scaled metric representing the delay on the link between the physical hosts, c_i is the cost of hosting a resource at edge i (j respectively). The variable α is a value between 0 and 1, it is used to indicate a preference between node cost and link delay. This way the ultimate cost of the edge can be based on network performance or virtual router hosting cost. The function c_i can be any function that returns a value between 0 and 1 and represents the pricing model of the virtual router provider. At α = 1 a precedence is given to the cost of the virtual router hosting, and at α = 0 a precedence is given to the network quality.

The result of the optimisation process is a tree structure traced through the topology of the physical graph. The tree will have least cost based on the weights defined by the function k.

C. Virtual Topology Reduction

The virtual topology resulting from the optimisation process represents a single sourced tree connected to the target sites. However, in a virtual network not all the physical nodes need to host a virtual router. A virtual link can be used to connect two virtual routers and can span many physical hosts. This can be implemented by using an IP tunnelling protocol, or a

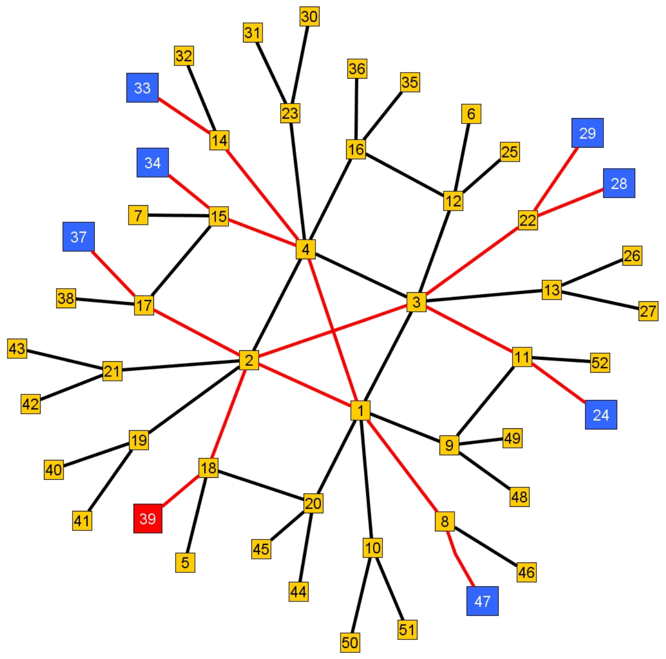
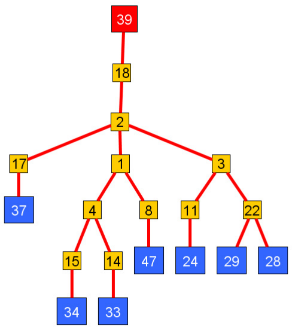
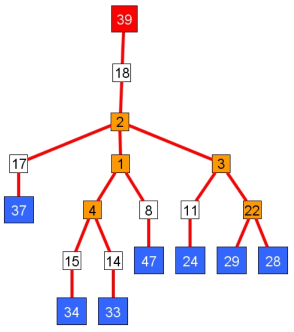


Figure 3. Source (39) tree to 7 Targets



(a) Trimmed Tree



(b) Reduced Tree

Figure 4. Trimmed and Reduced Trees with S = 39 and T={24,28,29,33,34,37,47}

secure VPN connection. The solution reached in the previous sub-section is a tree structure linking nodes in the underlying physical topology. However, it is not reasonable to assume that a virtual router should be placed at each node as indicated in the tree, whether a physical router exists there or not.

Only essential routing nodes should host a virtual router, and these can be discovered by counting the degree of the node in the graph. For example, the tree in Figure 3 is computed from the optimisation process. It shows a tree spanning from node 39 to nodes 24,28,29,33,34,37 and 47.

Figure 4a shows a trimmed tree from the same graph, focusing directly on the resulting solution from Figure 3.

Next, Figure 4b highlights the positions of the virtual routers. There are tunnels established between the ingress and egress of each of the virtual routers. In this case, there are 5 virtual routers that are established to connect a network of

8 nodes, 1 source and 7 targets. This final reduced topology is used as a blueprint of the virtual network that should be deployed. The next step is to derive the deployment policies that should be used to actually deploy this virtual network.

D. Generation of Deployment Policies

The generation of the deployment policies aids in the deployment, configuration and management of the virtual network. As presented in §III, the Model-based Translator software can take as input a schematic of a virtual network (in XML) and can generate a set of device specific commands to realise the virtual network. In our case, the commands that must be generated are virtual machine management policies containing the vCPI commands to configure and start specific virtual machines and their virtual links. The behaviour of the virtual network is monitored and controlled by the Autonomic Management Systems or AMS. The generated policies are placed into the policy management services of the AMS. Policy templates are defined *a-priori* to ensure that the state of the virtual network that is finally realised can be maintained to a degree of confidence. This is opposed to just generating commands to bring up the virtual network where no guarantee of the virtual network is offered. In the case of a virtual network augmented with management policies, it can be effectively re-configured should there be a problem in the physical network.

Take for example the following list that is used at a basis for the generated policies:

```
START:
On        ReciptOfNetworkConfig
   THEN   DeriveNetworkConfig
   AND    DeployNetworkConfig
STABLE:
IF        NetworkIsStable
   THEN   DoNothing
LINKLOST:
IF        NetworkLinkLost
   THEN   ReComputeNetworkGraph
   AND    WaitFor(ReciptOfNetworkConfig)
ROUTERLOST:
IF        NetworkRouterLost
   THEN   ReComputeNetworkGraph
   AND    WaitFor(ReciptOfNetworkConfig)
END:
On        ReciptOfNetworkDisolve
   THEN   StopNetwork AND FreeResources
```

The list of policies describe some template behaviour that we wish to deploy to the virtual network, so that it can at least attempt to recover from physical network problem. It is essentially re-configured and re-deployed with a restored state. More policies can be readily added or removed to give the virtual network more elaborate behaviour. In our implementation, the MBT generates the policies and the virtual machine management commands. The resulting policies also interact with the AMS to request for network reconfiguration, in which case the OPs are used to come up with an optimal new network topology that favours the end users specific requirements (e.g. low cost of hosting, vs. low network delay).

V. Experimental Analysis

The traditional approach to virtual networks, or virtual private networks (VPNs) is to connect all sites via a tunnelling protocol such as IPSec, PPP or MPLS. In this case, the sites are all connected to each other via their respective shortest paths. Our approach establishes a virtual network to connect sites based on a least cost spanning tree that encompasses virtual routers inside to route traffic between tunnels (virtual links). At the location of a virtual router a set of tunnels or virtual links are routed to target sites. The basis of our analysis is to show that the generated trees are of a lower cost then the combined shortest paths among the list of sites that need to be connected. This approach is a common approach to approximate the Steiner Tree problem [15].

We compare our approach to that of shortest path union approach, where the links used to connect a source site to each individual target site using a shortest path algorithm are combined together. Each unique link contained in this new tree is assigned the same cost value as used in our binary optimisation approach. We then compare the resulting weights together to capture the difference in cost. We performed a comparison of the cost (be it delay or virtual machine hosting) of our approach per established network versus the cost of establishing multiple shortest path tunnels. Our experiments are all limited to a single source site per iteration; however, the weights in the communication network are directional, therefore the network could have unequal weights in either direction. For simplicity of describing the results, we assume a bi-directional network with equal link weights in both directions. The network we carried out our experiments on is shown in Figure 3 above. It contains, 32 edge nodes, 20 hub nodes and 116 unique edges (counting both directions). We examined typical ISP topologies as presented on the website of Internet Topology Zoo [16] and have developed a topology of reasonable size in comparison to what is located there. The link weights are computed based on the degree of each node for delay and cost of hosting. For nodes with a higher degree delay was high, but virtual machine hosting was lower assuming an economy of scale on larger nodes. Each iteration of the experiment choses a single source node at random, from all leaf nodes, and a set of target nodes at random from the remaining leaf nodes. Iterations of the experiment were carried out with the following parameters: $|S| = 1$; $|T| = 2$ to 16; Alpha = {1 , 0}. Each iteration was run 1000 times to produce mean values as statistically representative results.

From Figure 5, we see that our approach produces a lower cost spanning tree as compared to the union of shortest paths approach. We see that the difference in cost for the virtual network as compared to the cost of the shortest paths network is between 8% to 10% (95% percentile of 1000 data point per no. of targets). The cost saving begin to decline as the virtual network grows because most links are now being used

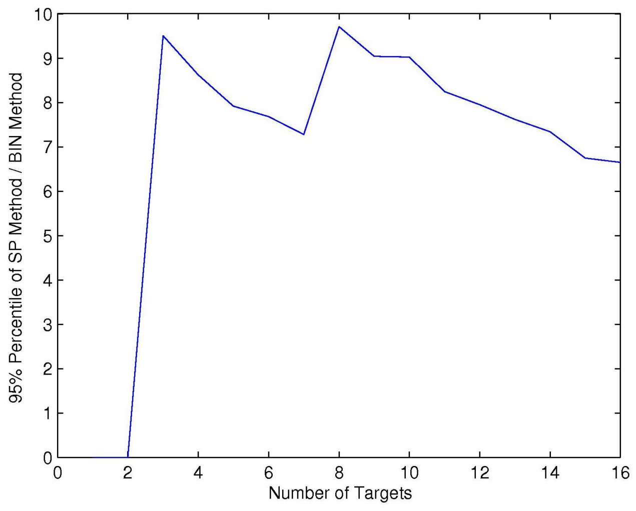


Figure 5. Difference in Link Cost and Alpha = 1

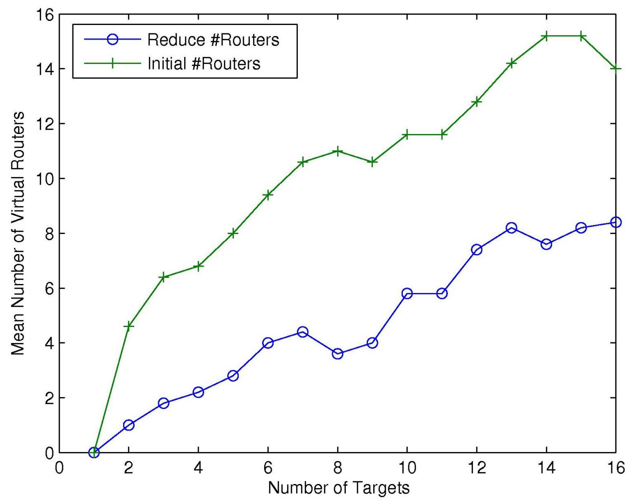


Figure 6. Number of Virtual Routers after Reduction

in both approaches. For smaller virtual networks, with only few edges used, there is less scope for optimisation. Also for larger virtual networks, mostly all edges are being used and so the maximum cost is being reached.

From Figure 6 we see that when the virtual topology is reduce to only essential virtual routers then there is a significant reduction in the number of virtual routers required. These remaining virtual routers are placed at positions of least weight due to the constraints imposed on the optimisation process. We can also see that the number of virtual routers required scales approximately linearly, indicating that there is good consistent performance.

With respect to the runtime complexity of the optimisation process, it is based on the order of constraints and not on the order of how many target nodes must be connected. The number of constraints required for a particular topology

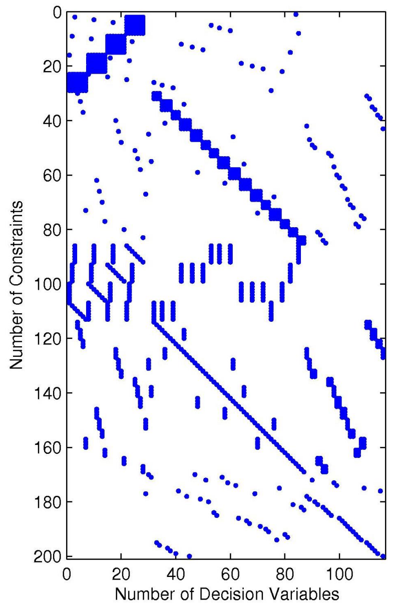


Figure 7. Constraint Matrix

depends on the number of edges and nodes in the physical network. We can compute the number of constraints using the following formula:

$$1+ \mid V \backslash H \mid + \sum_{i \in H} degOut\,(H_i) + \sum_{i \in H} degIn\,(H_i) \qquad (8)$$

where H is the set of hub nodes in the graph. In our experiment we can calculate the number of constraints to be 1 + 31 + 84 + 84 = 200. These constraints are depicted graphically in Figure 7. Each new edge node added will cause at lease 3 new constraints to cater for it as a leaf, and the new edges in and out of its neighbouring hub node. Also, two new decision variables will be required. Therefore, the number constraints and variables grows linearly with the number of edges.

The experiments were conducted in MATLAB 7, where the final topology was described as a connection matrix. This connection matrix is used to build the XML description of the virtual network topology and is then deployed using the AMS software developed in the AutoI project.

VI. Discussion and Future Work

The approach presented here has some limitations. These limitations will be discussed and highlighted as avenues for future research challenges. One obvious limitation is that the request for a virtual topology is limited to that of a single source node and multiple target nodes, constraining our solution to produce a tree oriented network topology. Tree based topologies in virtual networks have potentially different characteristics then tree based topologies of physical networks. One of the main differences is that a tree based physical network is typically viewed as having low reliability due to lack of redundant links. However, a tree based virtual network may be supported by a meshed physical network, where there is underlying redundancy for each virtual router and virtual link. Therefore, reliability would be a matter of the physical network provider fulfilling an SLA for each virtual link.

For our optimisation process to receive a request containing multiple sources, we would have to update the optimisation problem defined to be more aligned to that of the Steiner Network problem. This is certainly a potential avenue of future research, where multiple virtual networks may be created simultaneously from a single request. One highly desirable feature that we intend to investigate in the future is that of calculating a minimal change update to the virtual topology given a change in VNP objectives or a change in physical network resources. This would produce a set of migrations or VN updates that would minimally disturb the virtual networks operation. We also intend to carry out our experiments on a larger scale testbed where more realistic scenarios can be replicated, and more experiments can be performed.

VII. Conclusion

The Future Internet requires flexibility from network management systems to provision services and networks on demand with low overhead and cost. We present an policy authoring process and optimisation approach to virtual network deployment. The resulting virtual network places the virtual routers at nodes guided by the policies specified by an end user. This deployed virtual network is then managed through the use of policies that are generated from some desired behaviour. An Autonomic Management System, developed in the FP7 AutoI Project is used to managed and deploy the virtual routers in a physical network. The resulting virtual network is then reduced to eliminate nodes where there is no functionality required and and are not actually making routing decisions. A binary integer optimisation problem is defined and solved in order to build the optimised virtual topology. Experimental analysis shows that a virtual network is deployed that has lower cost with respect to network delay and hosting costs then an equivalent peer-to-peer VPN installation which is based on multiple shortest path links between the networked sites.

Acknowledgments

This work was supported in part by the Irish Research Council for Science, Engineering and Technology and co-funded by Marie Curie Actions in FP7. It was also supported in part by Science Foundation Ireland, via the "FAME" strategic research cluster (grant no. 08/SRC/I1403) and by project TEC2009-14598-C02-02 funded by the Spanish MICINN.

References

[1] Autonomic Internet. (2010, October) http://www.ist-autoi.eu/autoi. [Online]. Available: http://www.ist-autoi.eu/autoi

[2] S. Figuerola, J. A. Garcia-Espin, and N. Ciulli, "An optical network and it infrastructure virtualisation and provisioning framework," in *Optical Internet (COIN), 2010. 9th International Conference on.*, Jeju, South Korea, 11-14 July 2010.

[3] D. Dudkowski, "Sail cloud networking," in *ASMONIA Workshop, Heidelberg, Germany*, March 2011.

[4] Y. Zhu and M. Ammar, "Algorithms for Assigning Substrate Network Resources to Virtual Network Components," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*. IEEE, 2006, pp. 1–12.

[5] J. Lu and J. Turner, "Efficient mapping of virtual networks onto a shared substrate," *Technical Report WUCSE-2006-35, Washington University*, 2006.

[6] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 17–29, 2008.

[7] I. Houidi, W. Louati, and D. Zeghlache, "A distributed virtual network mapping algorithm," in *Communications, 2008. ICC'08. IEEE International Conference on*. IEEE, 2008, pp. 5634–5640.

[8] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, 2005.

[9] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 783–791.

[10] Z. Cai, F. Liu, N. Xiao, Q. Liu, and Z. Wang, "Virtual network embedding for evolving networks," in *GLOBECOM*, 2010, pp. 1–5.

[11] A. Razzaq and M. Rathore, "An approach towards resource efficient virtual network embedding," in *Evolving Internet (INTERNET), 2010 Second International Conference on*, 2010, pp. 68 –73.

[12] Y. Chen, J. Li, T. Wo, C. Hu, and W. Liu, "Resilient Virtual Network Service Provision in Network Virtualization Environments," in *2010 IEEE 16th International Conference on Parallel and Distributed Systems*. IEEE, 2010, pp. 51–58.

[13] J. Beasley, "An algorithm for the Steiner problem in graphs," *Networks*, vol. 14, no. 1, pp. 147–159, 1984.

[14] A. Agrawal, P. Klein, and R. Ravi, "When trees collide: An approximation algorithm for the generalized Steiner problem on networks," in *Proceedings of the twenty-third annual ACM symposium on Theory of computing*. ACM, 1991, pp. 134–144.

[15] S. Voß, "Steiner tree problems in telecommunications," in *Handbook of Optimization in Telecommunications*, M. G. C. Resende and P. M. Pardalos, Eds. Springer US, 2006, pp. 459–492.

[16] Topology Zoo [accessed on 11th April 2011]. [Online]. Available: www.topology-zoo.org