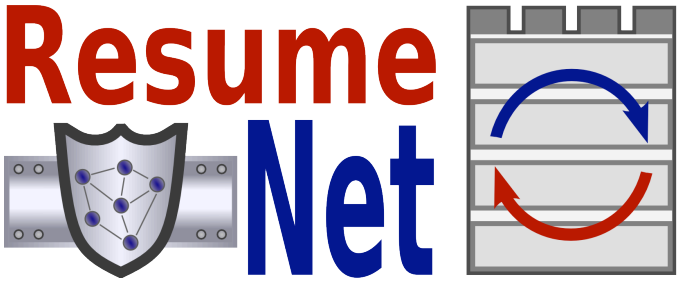
Resilience and Survivability for future networking: framework, mechanisms, and experimental evaluation



Deliverable number	3.1a
Deliverable name	Taxonomy of P2P, Overlays and Virtualization Techniques with respect to service resilience
WP number	3
Delivery date	31/08/2009
Date of Preparation	2/10/2009
Editor	Andreas Fischer (UP)
Contributor(s)	Andreas Fischer (UP), Alexandru Paler (UP), Nafeesa Bohra (UP)
Internal reviewer	Ali Fessi (TUM), Paul Smith (ULANC)

Summary

The Future Internet will be built using recent and future technological advancements, but the ever increasing speed of network development makes it difficult to gain a complete overview of all technologies. The scope of this document is to present the technologies that are being considered in ResumeNet to build resilient services. In particular, system virtualization and Peer-to-Peer overlay networks are analyzed in order to offer a starting point into Future Internet research.

Contents

1 Introduction **4**

2 Overlay networks **4**

2.1 Overview 4

2.1.1 Virtual Private Networks (VPN) 5

2.1.2 Virtual LAN (VLAN) 5

2.1.3 Programmable and Active Networks 6

2.1.4 PlanetLab / GENI 7

2.2 P2P networks 8

2.2.1 Unstructured vs Structured P2P 10

2.2.2 Pure vs. Hybrid P2P 11

3 System virtualization **12**

3.1 Overview 12

3.2 Building overlays using system virtualization 13

4 System Virtualization and P2P Overlays as Service Resilience Enablers **15**

4.1 Increasing resilience through abstraction 15

4.2 Assuring Virtual Resources 16

4.2.1 Definition of Physical and Virtual Resources 16

4.2.2 Security and Assurability of Virtual Resources 17

4.3 Relation to the $\boldsymbol{D^2R^2 + DR}$ Strategy 17

5 Conclusions **18**

1 Introduction

Future Internet research has embraced a broad field of subjects, some of them being ubiquitous also in every day life. System virtualization and Peer-to-Peer (P2P) networks are subjects having appeal to the research community, and containing a relevant dose of material for building resilient and survivable networks. Incorporating resilience into networks means keeping the network operational in the face of adverse conditions with as little impact on the network's service quality as possible. This is a difficult task that needs a vast and deep understanding of networks and network-associated technologies.

The scope of this document is to create a taxonomy of system virtualization and P2P technologies that can build the basis of a new Internet paradigm. The panoramic overview is still an incipient step towards integrating them. Therefore the technologies are presented considering both their advantages and disadvantages in order to underline how they can complement each other.

System virtualization creates a logical abstraction from the hardware layer, and increases the mobility of services. That is why security has also to be considered with respect to this technology. Striving towards a clear concept of the role of system virtualization in a resilient network is an important goal. Together with the description of virtual network elements (section 3), this is going to support future discussions.

Various virtualization techniques are presented in section 2.1. The same section contains also a short description of platforms enabling the testing of new technologies, e.g., PlanetLab. These platforms are built at the intersection between virtualization and networking. P2P networks stem from the idea of equality between participants, offering an additional building block for resilience and survivability. A brief description of overlay P2P networks under consideration of their classification is presented in section 2. Resilience and survivability are discussed in section 4 where the influence of the previously described technologies is depicted. Finally, section 5 contains the conclusions and final remarks.

2 Overlay networks

2.1 Overview

An overlay network is a virtual network topology built on top of the physical network that can directly interface with the users. Nodes in the overlay can be thought of as being connected by virtual or logical links, each of which corresponds to a path, perhaps through many physical links, in the underlying network.

Building overlays is achieved based on a multitude of technologies. Following the standard layered network architecture, these technologies are targeted towards the different layers of the network. Over time, virtualization of networks has shifted the focus towards creating a holistic and generalized network virtualization environment that features a "completely virtualized, highly customizable, and technology-agnostic networking facility for the future Internet" [CB09]. The following sections contain an analysis of existing virtualization technologies, and how these technologies form the basis of evaluation platforms like, for example, PlanetLab or Geni. Using such platforms, it is possible to assess the resilience and the survivability of future networks.

2.1.1 Virtual Private Networks (VPN)

A Virtual Private Network is a way to emulate a private network over a public network, like the Internet. A VPN is a concept composed of two parts: a *virtual network* overlaid on top of the ubiquitous interconnection of the Internet and a *private network* for confidential communications and exclusive usage – the respective architectures can be classified as follows [YS01]:

- *site-to-site intranet VPN*, in which multiple geographically dispersed network sites within the same organization are connected;
- *remote access VPN*, in which a single remote network device is connected to a corporate intranet;
- *extranet VPN*, in which network resources within one corporation are opened for access to other organizations.

Olifer proposes a distinction of 3 different types of VPNs [Oli]:

- *encrypted VPNs*, where data is encrypted so that potential eavesdroppers cannot understand the content even if it is intercepted
- *tunnel-based VPNs* are based on different technologies, but share a common characteristic. The traffic is transmitted between sites using logical channels (tunnels).
- *optical private networks* are generally not viewed as true VPNs because they do not use a shared packet-switched infrastructure.

Tunnels serve three major purposes in VPNs [YS01]. The encapsulation of one protocol within another so that the data can be transported over an IP infrastructure is the first purpose. The second is to allow secure routing as a result of apparent traffic separation; and the third is to provide data integrity and confidentiality. Tunneling can be operated at different layers of the network. Layer 2 tunneling protocols, like PPTP, L2F and L2TP are designed for the tunneling of Point-to-Point Protocol frames through an IP network. Layer 3 tunneling is useful for site-to-site VPNs and is the result of using IPSec, for example. IPSec is a suite of protocols that defines the architecture and specifications for providing security services within the IP protocol. It was initially conceived for IPv6, but it can be also used for IPv4. There is another type of tunneling that is between layer 2 and layer 3 because a label is placed between these layer headers. The method is referred to as *label switching* and can be realized by using the Multiprotocol Label Switching (MPLS) protocol [YS01]. Encrypted VPNs can use for example SSL, which is a protocol operating in the fourth network layer.

2.1.2 Virtual LAN (VLAN)

VLANs can be seen as analogous to a group of end-stations, perhaps on multiple physical LAN segments, that are not constrained by their physical location and can communicate as if they were on a common LAN [PF96]. In order to use *tagging* to mark the membership of the data transmitted, the Ethernet header has been changed in the IEEE 802.1Q protocol by introducing a VLAN identifier. That is why end-stations need not to be able to support this protocol, whereas networking elements are obliged to. There are several ways how VLAN membership can be defined:

- *Membership by port group*, where the switch ports are considered, which does not allow multiple VLANs to use the same port. Another drawback is that, after physically moving a computer, the switch ports have to be manually reconfigured
- *Membership by MAC address*, which has the drawback that initially all computers must be manually configured
- *Membership by layer 3 information (protocol type, network address)*, which eliminates the tagging overhead of previous methods. Nevertheless, inspecting layer 3 addresses in packets is more time-consuming than looking at MAC addresses in frames, as it requires to extract and decode the IP header information.

Building VPNs and VLANs is a way of abstracting logical networks from physical ones. The logical separation of networks helps to incorporate resilience into them. There is a plethora of existing mechanisms regarding how this can be achieved, for example. There is, however, no standard technology being used, because of the different requirements and expectations of such a network, like: security, private IP address space, independent transport protocols for end-user sites, improved performance, multi-domain capability, multicast support, scalability and configuration complexity [Oli].

2.1.3 Programmable and Active Networks

The ability to rapidly create and manage novel services in response to user demands, network, and environmental challenges could be related to the resilience and survivability of a network. A programmable network is distinguished from any other network environment by the fact that it can be programmed from a minimal set of APIs from which one can ideally compose an infinite spectrum of higher level services [CDMK+99]. The programmability of network services is achieved by introducing computation inside the network, beyond the extent of the computation performed in existing routers and switches. In order to make the network more programmable, the separation of communication hardware from control software is fundamental.

Programmability of networks has undergone two different paths; one is advocated by the *active networks* community and the other one by the *Opensig* community. The difference between the two concepts lies in the way programmability is considered. The Opensig community argues that by modeling communication hardware using a set of open programmable network interfaces, open access to switches and routers can be provided, thereby enabling third-party software providers to enter the market of telecommunications software. The active network community considers the dynamic deployment of new services at runtime mainly within the confines of existing IP networks. They are proposing the concept of *active packets*. An extreme view of these packets are the *capsules*, where every message is a program [TSS+97]. When a capsule arrives at an active node, its contents are evaluated. There are different options when choosing between the program-encoding technologies and some aspects like mobility, safety, and efficiency have to be considered, because each one is influencing the overall result of new service deployment.

Campbell and De Meer [CDMK+99] survey a multitude of programmable networks and conclude that a common set of characteristics govern the construction of these networks:

- *networking technology*, which implicitly limits the programmability that can be delivered to higher levels;

- *level of programmability*, which indicates the method, granularity and time scale over which new services can be introduced into the network infrastructure;
- *programmable communications abstractions*, which indicate the level of virtualization and programmability of networking infrastructure requiring different middleware and potentially network node support;
- *architectural domain*, which indicates the targeted architectural or application domain.

Concluding on the comparison of programmable networks, one can observe that many projects use virtualization techniques to support the programmability of different types of communication abstractions. For example, the Tempest [VdMRLC98] framework is using virtualization of the network infrastructure. Physical switches are abstracted creating sets of interfaces to switch partitions called "switchlets". These switchlets are a kind of *virtual switches*. That is why the dynamic composition and deployment of new services can be extended to include the composition of complete virtual networks, leading to a new paradigm of *programmable virtual networks*. Virtualization appears as a solution to build, deploy, use and evaluate new network services and network architectures.

2.1.4 PlanetLab / GENI

Experimental evaluation of future networking frameworks and mechanisms is needed before their deployment, however, there are multiple means of evaluation. The following presentation of technologies is based on the analysis of Anderson[1] [APST05]. Although simulation and emulation are valuable tools for understanding new designs, they can not substitute experimentation with live traffic. Physical testbeds are traditional platforms for live experimentation. Production test-beds support real traffic from real users, whereas research testbeds do not carry traffic from a wide variety of real users, and instead are typically driven by synthetically generated traffic. One drawback of using physical testbeds is that their use involves substantial costs, which makes them very expensive for large scale experimentations. Overlays are nowadays used both as an experimental platform and a deployment path, but they have not been a source of dramatic architectural advancement, because they have been seen as a way to deploy narrow fixes to specific Internet problems and because their architecture typically assumes IP or an interoverlay node protocol.

Examples of overlay networks (see also 3.1) are PlanetLab[2] and GENI[3]. The difference between these two platforms is the used networking technology. PlanetLab depends on IP, while GENI will be agnostic to any specific technology [CB09]. The novelty of PlanetLab lies in the distributed virtualization: "the acquisition of a distributed set of virtual machines that the system is treating as a single, compound entity" [PACR02]. PlanetLab isolates services and applications from one another, thereby maintaining the illusion that each service runs on a distributed set of private machines. Version 3.0 of the PlanetLab software implemented a virtual machine monitor as a combination of the Linux 2.6 kernel and a set of kernel extensions (vservers 1.9), a Linux patch that provides multiple, independently managed virtual servers running on a single machine and the SILK module that provides CPU scheduling, networking accounting, and safe raw sockets [APST05]. This means PlanetLab is the result of node-level virtualization, connecting virtual nodes on different physical ones.

[1] Chair of the PlanetLab Steering Committee
[2] http://www.planet-lab.org/
[3] http://www.geni.net/

The terminology used by GENI is identical to the one from PlanetLab. Through node virtualization the physical network is being *sliced* resulting in virtual networks *slices*. The nodes of a *slice* are the virtual machines, which are called *slivers*. An experiment is a researcher-defined use of a slice, but experiments are not slices. Many different experiments can be executed in a particular slice concurrently or over time.

As a compromise between these two solutions, physical testbeds and overlays, Anderson is proposing the *virtual testbed*. Virtual testbeds have two basic components: an overlay substrate and a client-proxy mechanism [APST05]. The overlay substrate provides a set of multiplexed dedicated overlay nodes, while a host can use the client-proxy mechanism to opt in to a particular experiment running on a specific substrate overlay.

2.2 P2P networks

The P2P architecture was popularized by file sharing mechanisms in which files can be shared directly between network users without the assistance or the interference of a central server. File sharing is one application area of P2P, but there are many other areas such as: distributed processing, directory service and rendezvous systems. Recent P2P file sharing networks do not require any central server, therefore P2P usually is more scalable and more resilient than the centralized file sharing schemes.

If we look into a brief history of P2P networks, it shows that P2P networking is not a new paradigm in the networking world, but its application to the Internet is a recent development. One of the first P2P networks was developed by the U.S. defense department in a project called ARPANET, in which the computers involved were connected directly together. Another P2P system, called Usenet, was developed in 1979 – it used UUCP (Unix-to-Unix-copy protocol) to connect computers and set up a news message board. A boom of P2P occurred in 1999 with the development of Napster, which is generally considered the first generation of P2P. This first generation of P2P used a centralized file list containing files on a connected user's hard drive. Napster and Napster clones[4] use the Napster model, which is similar to the Internet Domain Name System in its method of operation. While file transfers are performed in a P2P fashion, file searches are performed using the client server model. If a Napster client wishes to search for an audio file, it sends queries to multiple search servers. If the file is found in the server's database, the server returns the IP address of the Napster client that has the file. The first client then connects to the second and attempts to retrieve the file. Napster revolutionized audio file trading and was ultimately responsible for the popularity of Internet P2P file sharing. Because of copyright problems arising through file sharing, Napster was legally convicted and finally had to stop the operation of the Napster server and therefore the service was no longer available. The second generation, which includes Gnutella[5], used a decentralized network. Unlike Napster, Gnutella connects peers directly to a group of other peers and so on. The third generation can be termed as a compromise between 1st and 2nd generations as in 3rd generation the major focus was on improving the speed of file transfer. Included in the third generation is Fasttrack, which is the network that Kazaa[6] and Morpheus use. Fasttrack is the name of the network whereas Kazaa and Morpheus[7] are the names of the different clients that connect to the Fasttrack Network. It means that the users of any of those clients had access to exactly the same files. Currently, most P2P networks are based on

[4] http://www.napster.com/
[5] http://gnutella.wego.com/
[6] http://www.kazaa.com/
[7] http://www.musiccity.com/

DHTs (Distributed Hash Tables, see section 2.2.1) and hence it can be said that DHTs would be the future of the next generation of P2P networks.

A P2P network can be described as a group of entities denoted as peers [TGB06], with a common interest, that build a self-organizing overlay network on top of a mixture of already existing networks. That is, P2P is about the networked cooperation among equals. The main task is the discovery and sharing of pooled and exchangeable resources.

The way Internet and communication technologies are evolving, there has been a great interest in emerging P2P overlay networks since these provide a good substrate for creating large scale data sharing, application-level multicast and content distribution; for example, searching/selection of nearby peers, redundant storage, efficient location/search for data items, trust, authentication, and performance [LCP+05]. Each node or peer acts as router, which forwards the query in the overlay network and acts as a content requester and a content provider at the same time. Typical overlay networks include multicast overlays, peer-to-peer overlays (e.g., Gnutella and Kazaa), parallel file downloading overlays (e.g., Fast-track, BitTorrent and eDonkey), and routing overlays (e.g., Skype for VoIP).

The advantages provided by P2P overlay networks utilize the growing pool of Internet applications/resources. These advantages are listed below:

- For application users and network developers, overlay networks allow to design and implement their own communication environment and protocols on top of the Internet; for example, file sharing and data forwarding. Examples are Gnutella, Fast-Track, and Bit-Torrent like file sharing systems.
- Data forwarding in P2P overlay networks can be flexible since it has the ability to adaptively select a path and hence avoid network congestion.
- Scalability and robustness are the two important features of overlay networks. One end-node is always able to communicate with another end-node as long as the physical network connection exists.
- The increase in the number of joining nodes will also effectively increase the sharing of a huge amount of information and resources that are available over the Internet.
- Unsecured and unsigned codes may allow malicious files to enter into the victims computer or even a network. P2P overlays that use mechanisms like hashing, chunk verification and different encryption techniques receive a gain in security and file verification.

The advantages are counterbalanced by disadvantages that are offering intensive research topics:

- The decentralized nature of P2P overlay networks makes administration difficult.
- Being deployed over public networks (Internet), overlay networks are also open for any kind of users, and hence security and privacy become serious issues.
- Issues like collaboration between end-nodes and fairness of resource sharing in overlay networks have not been well addressed.

After the brief discussion of P2P overlay networks, the approaches which are currently used by P2P are discussed in the next section.

2.2.1 Unstructured vs Structured P2P

In an unstructured P2P system, such as Napster and Gnutella, logical overlay networks are established between the participating peers. Hence, an overlay topology is developed with a high number of redundant paths and loops. Since peers are connected to each other, they form a fully decentralized overlay network. There are no dedicated peers that store information. That is, all the resources are available to the peers in an overlay network. In an unstructured P2P network, queries are broadcasted to neighbors and neighbors broadcast to other neighbors in the overlay network. Hence, searches are performed through a flooding mechanism – i.e., a peer searching for information floods the query to all its neighboring peers in the overlay network, which in turn forwards the query to all its neighbors until the desired information is found. High cost of flooding is the major problem in unstructured P2P networks. In order to prevent the entire network from the high cost of flooding, a TTL (Time to Live) counter is attached with each query. With every hop, there is a decrease in TTL whereas a query packet with zero TTL is simply discarded. The main drawbacks of this approach are:

- No guarantee that a search returns a positive result even if the searched information is stored on a large number of peers [TGB06].
- Unstructured P2P mechanisms do not scale to a large number of information consumers.

Some of the important features of unstructured P2P systems like Napster and Gnutella are listed below:

Napster

- Peers update central server about content.
- Peers get files by querying central server.
- File transfer happens directly between peers.

Gnutella

- Peers form a logical overlay network.
- Queries are broadcasted to neighbors.
- Neighbors broadcast to their neighbors (flooding).
- Query radius is limited (not scalable).

In contrast to unstructured P2P networks, structured P2P networks follow a different approach with respect to network maintenance and lookup. In structured P2P networks, the peers or nodes are organized in such a way that specific services or resources are located in a more direct manner, i.e., a query will take place with fewer number of nodes [SMK+01]. In structured P2P networks, there is a controlled management with respect to the participating nodes and the published data items. For example, the location of data items can be controlled by the overlay structure. In order to search the data items for designated keywords, the queries are forwarded only to the nodes that actually store the requested data. This is different from the search mechanism used in unstructured P2P networks.

Technically, *structured* means that the P2P overlay network topology is tightly controlled. In structured P2P [LCP+05], contents are not placed at random peers; rather contents are placed at specified locations due to which queries are made more efficiently. DHT (distributed hash tables) are by far the most common type of structured P2P networks. DHTs (distributed hash table) are based on structured P2P networks, which use hash keys to store resources [LCP+05]. Chord [SMK+01] is an example for a DHT. Chord arranges the participating peers on a ring topology [SMK+01]. Chord routes queries in a way to locate a key with only a small number of hops, which stays small even if the system contains a large number of nodes. Within the ring topology formed by Chord, the position of a peer on this ring is chosen according to the hash value of a unique attribute of this peer. The idea behind this approach is that each peer has a good knowledge about its overlay neighbors, i.e., its predecessors and successors on the Chord ring, while only maintaining a few connections to more distant peers. In this way the mediation of the information stored in the distributed network can be done using only $O(log_2(n))$.

Structured P2P systems such as Chord [SMK+01], CAN [SRS01], and Pastry [RD01] currently utilize DHT mechanisms. They focus on forming P2P overlays optimized for lookup latency and maintenance cost [LNBK02], determining how nodes leave and join the network rather than using a flooding mechanism as used by unstructured P2P networks such as Gnutella.

From the above discussion, it can be concluded that P2P overlay networks are distributed in nature, i.e, the peers form a self-organizing overlay. In current Internet scenarios, P2P networks are popular and efficient in many applications, such as, file and media sharing systems, distributed systems, wireless systems.

2.2.2 Pure vs. Hybrid P2P

As far as pure P2P systems are concerned, the entire network consists solely of equipotent peers. A pure P2P network does not have the notion of clients or servers but only equal peer nodes that simultaneously function as both "clients" and "servers" to the other nodes on the network. This model of network arrangement differs from the client-server model where communication is usually to and from a central server. In a pure P2P network, peers act as equals, merging the roles of the client as well as the server. A typical example of a file transfer that is not P2P is an FTP server, where the client and server programs are quite distinct, i.e., the clients initiate the upload/downloads, and the servers react to and satisfy these requests. In pure P2P networks, there is no central server managing the network, neither is there a central router. Some examples of pure P2P Application Layer networks designed for file sharing are Gnutella (pre v0.4) and Freenet.

On the other hand, a hybrid P2P architecture is more suitable for an enterprise or an organization. In hybrid P2P systems, peers are distributed into two groups: client nodes and overlay nodes. Typically, each client is able to act according to the momentary need of the network and can become part of the respective overlay network used to coordinate the P2P structure. This division between normal and "better" nodes is done in order to address the scaling problems of early pure P2P networks. An example for such a network is Gnutella (after v0.4). Hybrid P2P architectures can complement existing client-server web deployments such as search engines, portals, intranet servers, document management systems, extranets or network publishing systems. The systems mentioned above hold the most promise for positively impacting the efficiency and effectiveness of a company. In hybrid peer-to-peer systems, a central server is used for indexing functions and to bootstrap the entire system.

This has similarities with a structured architecture in which the connection between peers are not determined by any algorithm. The first prominent and popular peer-to-peer file sharing system, Napster, was an example of the centralized model.

From the discussion given in section 2.2 it can be concluded that P2P networks are typically used for connecting nodes largely via ad hoc connections. Sharing content files containing audio, video, data or anything in digital format is very common, and real time data, such as telephony traffic, can also be forwarded using P2P technology, as demonstrated, for example, by Skype. With the above discussion, questions may arise whether P2P is intended to replace the existing centralized or client-server architecture. The answer is that P2P is not intended to replace the many existing centralized or client-server applications [LCP+05], but is intended to complement them, as well as to create opportunities and take advantage of the computing assets and resources of individuals and organizations.

3 System virtualization

3.1 Overview

System virtualization is currently a very popular technique that is used to virtualize desktops and servers. Virtualized resources are easy to manage and are not bound to specific physical hardware. Especially in data centers, system virtualization is used to provide multiple parallel services, independently from each other on the same hardware, mainly to increase the utilization of resources. Its high popularity, its extensive usage in production environments, and the current rapid development of management solutions for system virtualization are the main reasons to investigate it in the context of network virtualization. The possibilities such a network model offers can be used as basis for the development of an advanced management solution of virtual networks (e.g., to create resilient networks, networks with mobility management, or service-supporting networks). The main elements of a virtualized network infrastructure based on system virtualization are multiple virtual networks running in parallel, consisting of virtual routers and virtual links. Such virtual networks form overlay structures that are not directly related to the underlying physical network. A virtual network has all ordinary properties of a physical network, but it also gains additional features inherited from system virtualization.

In order to discuss the method of system virtualization, first the method of process virtualization has to be explained. Process virtualization has been in use for Operating Systems (OS) for a long time to enable multiprogramming. It allows several different processes to run in parallel while being isolated from each other. Each process experiences full access to all available resources (e.g., CPU, memory, hard disk, etc.) and is not aware of the fact that it is sharing these resources with other processes. In fact, it only owns some time slices of the CPU, a part of the memory (virtual memory), and also shares the peripheral devices with all other processes. The OS is allocating the resources to the processes as needed, following a resource allocation algorithm. Another well known example of process virtualization is the Java Virtual Machine, where processes are executed in sandboxes independent from each other. System virtualization on the other hand takes the concept of resource virtualization one step further. A Virtual Machine (VM) is created in system virtualization, i.e., a full machine is virtualized, consisting of virtual CPUs, virtual memory, virtual hard disk, virtual Network Interface Card (NIC) etc. In the ideal case, a VM is a perfect recreation of a real machine in such a way that an OS can be installed on it without being aware of the resource virtualization. To distinguish virtualized resources from physically available resources, the term Real Machine (RM) is used

to refer to physical hardware [PG74]. The software that provides VMs is usually called Virtual Machine Monitor (VMM) and either located directly on top of the RM (called full virtualization) or on top of an OS (called hosted virtualization). The VMM in the full virtualization approach is also called hypervisor (classical hypervisors are, e.g., XEN [BDF+03] or VMWare ESX Server [vmw]). A VMM can host several VMs that are operating in parallel on a single RM. The VMs are independent from each other and are not necessarily aware of the existence of other VMs, running on the same RM. The number of provided VMs is limited by the available resources on the RM. It is important to see that the virtualization layer causes overhead, thus not all of the available resources can be provided to VMs. An OS can be installed within a VM – it is called Guest OS.

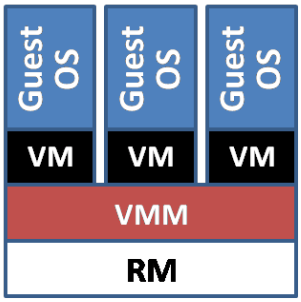


Figure 1: Full system virtualization

Figure 1 illustrates the method of full system virtualization. On top of an RM, a VMM virtualizes the RM's resources and hosts three VMs in this example. In each VM a Guest OS is installed. Full system virtualization is mainly used to virtualize services in data centers today. There are several basic primitives of management functions for VMs available: create VM, destroy VM, start VM, stop VM, move VM, copy VM and pause VM [dmt07]. It is even possible to have a live migration [CFH+05]. This means that a service in a VM can be moved to another RM without being interrupted.

3.2 Building overlays using system virtualization

To be able to present how overlays are built, one has to first define what a network actually is. In this context, communication networks are considered. These networks consist of active network elements, like routers or switches, which contain routing logic, along with passive network elements, like cables or wireless connections, which provide the transport mechanisms. In other words, active elements execute algorithms (in particular routing algorithms) whereas passive elements only transport data.

Using these definitions, a communication network is then represented by a directed, weighted graph in which the active network elements are represented by the nodes and the passive network elements are represented by the edges connecting the nodes. The graph is directed and weighted, as passive network elements can have different properties and are not necessarily symmetrical with regard to these properties (e.g., ADSL: asymmetric upload/download bandwidth).

Building overlays through system virtualization in this context refers to an abstraction of the available resources, which are the active and passive network elements – the nodes and edges of our graph. The abstraction can take two different routes: network elements can be combined to improve certain characteristics of a virtual network element, or network

elements can be split, to allow several different virtual network elements to coexist on the same structure. A combination of network elements can be realized through parallelization of the task or through redundancy, performing the task in several network elements at once. Parallelization increases performance, whereas redundancy provides a certain level of failure safety. Splitting of resources, on the other hand, can be realized through various kinds of multiplexing.

Therefore, an overlay network, or virtual network, is a network with a mapping to a second network, the substrate network, or underlay network. The mapping of the virtual network onto the substrate network defines the relationship of virtual active network elements (which will be called Virtual Routers here) and virtual passive network elements (which will be called Virtual Links here) to the active and passive network elements of the substrate network.

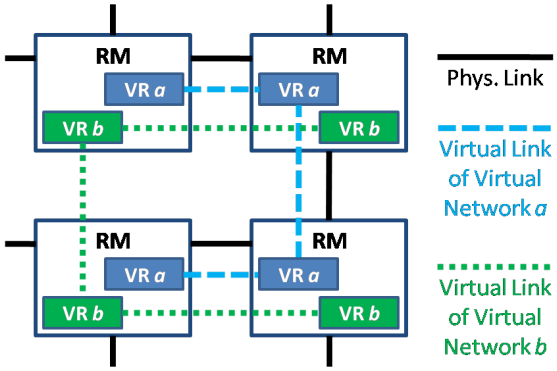


Figure 2: Two virtual networks on top of a physical topology

Combining several active network elements into a single virtual element - either through parallelization or through redundancy – is a difficult task, unless restrained to specific use cases. In particular, parallelization of tasks for active network elements requires intricate knowledge about dependencies within the respective algorithms, whereas true redundancy of tasks (i.e., carrying out the task in two or more different active network elements at the same time) requires a high level of coordination between the respective active network elements. Splitting of resources, on the other hand, is often comparatively easy to perform, requiring only a multiplexing mechanism along with some kind of scheduler. Therefore, most network virtualization approaches only discuss the case of splitting active network elements to host several virtual active network elements. Figure 2 shows an example of such a structure with two different virtual networks on top of a physical topology.

With this restriction in mind, the mapping for active network elements depicts a 1:n relationship between elements of the substrate network and elements of the overlay network. The mapping for passive network elements is more complicated: it can depict an n:m relationship between substrate and overlay elements; it can also map one or several overlay elements to an entire path in the substrate network – a subgraph of the substrate network that is weakly-connected with each node having a maximum in- and/or outdegree of 1.

The application of system virtualization to routers has been investigated in [EGH+07, MCZ06] and is already available in commercial products [cis]. Performance challenges were identified that have to be tackled when virtual routers are based on the popular XEN hypervisor [BDF+03]. In contrast to these related virtualization investigations, this work analyses the network model emerging when the method of system virtualization is applied to the core network infrastructure.

Wang et al. [WKB+08] discussed the prospect of having virtual routers that are movable on physical hardware. They propose an architecture supporting dynamic migration of virtual routers to decouple logical and physical configuration of a router. While they focus on a single network management primitive, this work considers the broader impact of system virtualization on the core network.

There are practical examples of system virtualization being applied to networks. One of those, PlanetLab [CCR+03] (see also 2.1.4) envisions an open distributed platform for distributed end-to-end applications. Resources of end-hosts located all over the world are virtualized. A user is provided a slice that consists of hundreds of virtual machines. Comparable approaches towards end-to-end virtualization are done by other projects (e.g. GENI [gen] or VINI [vin, BFH+06]).

Cloud computing is another practical application. Cloud computing approaches try to offer computing power independent of the actual hardware location. Within the research area of future generation networks virtual home networks are investigated and a Virtual Home Environment (VHE) is proposed, where end-hosts in home networks are virtualized in order to share resources with other end-hosts while reducing the overall energy consumption.

4 System Virtualization and P2P Overlays as Service Resilience Enablers

Sections 3 and 2 introduced several virtualization techniques and presented P2P overlay technologies. This section will discuss how these can be used in order to increase service resilience.

The main goal of service resilience is to keep services functional in the face of adverse conditions with as little impact on service quality as possible. In order to achieve that goal, we can identify two main requirements:

1. The service must remain functional somewhere in the network.
2. Service users must be able to (re-)discover and connect to the service in the network.

System virtualization and P2P overlays can help to achieve this goal by providing appropriate abstraction mechanisms. In the following section, some examples will be given.

4.1 Increasing resilience through abstraction

In the face of connectivity restrictions between two communicating partners – introduced, e.g., through Network Address Translation – Virtual Private Networks can provide connectivity by tunneling traffic through other protocols. Network traffic might for example be limited to HTTP traffic by a firewall – an appropriately set up VPN can tunnel all network traffic through the HTTP protocol then. Thus, the abstraction provided by the VPN allows applications to use the created virtual link without being modified or having to be aware, that the connection is tunneled through another protocol.

System virtualization can provide mobility of services. By encapsulating a service within a virtual machine and abstracting from the underlying hardware, system virtualization enables several forms of service migration, allowing a service to move from one hardware platform to another – ideally with minimal downtime. This is an important asset in the context of resilience,

as it allows to deal with expected, upcoming hardware failures. Examples include: hardware maintenance, power failure (while still running on UPS battery power), or an upcoming large-scale natural disaster like a storm or a flood. Being able to move the service allows to keep it operational. One of the major challenges, however, is to allow users to find the service after migration. Also, service restrictions have to be considered: A service might require existing connections to users to be kept alive. The impact of these restrictions and the solutions to the respective problems are the topic of further research in this area.

P2P networks can help to provide rediscovery of the service in case of service migration. By abstracting the routing process from geographical location, integrating a peer into the logical network based on its identificator, P2P networks can provide the needed mapping between a logical identificator and the physical location of a service. By simply re-registering with the P2P network, a service that has been moved can reintroduce itself to its users, allowing them to use the service even during migration with minimal downtime.

Structured P2P networks are also able to tolerate a relatively high number of nodes joining or leaving the network ("churn rate") [LNBK02]. This allows communications to be kept alive, even if a significant number of hosts are failing and leaving the network.

4.2 Assuring Virtual Resources

Virtualization is offering a new perspective on the way resources are defined, organized, and managed. Considering a stacked architecture of a system, where physical resources are at the bottom and applications are at the top, virtualization can be applied at almost every level. For example, storage can be virtualized through hardware RAID solutions, through software RAID solutions and also by building storage clusters.

Designing systems based on virtual resources needs a firm ground, and hence virtual resource definitions have emerged. The main issue is that the definitions and analysis of a virtual resource's specific characteristics have been, in our opinion, neglected. That is why, after analyzing virtual resources, it becomes clear that existing definitions need subtle changes, and that their characteristics need a deeper inspection.

4.2.1 Definition of Physical and Virtual Resources

Graupner [GKT02] is offering a definition for a physical resource: A physical resource is a device that has some absolute temporal and/or spatial computing parameters. He defines also a virtual resource: A virtual resource is a set of parameters mapped onto physical resources of a device or a collection of devices.

Resources can be virtual or physical, but their main characteristic (of being a resource) is still available. For example, a local file system is a resource – but if it is made available network wide, then it is still a resource, but a virtual one. Graupner's definition is not considering the possibility of a virtual resource to contain/depend on other virtual resources. Resources can be organized as a tree-structure, a resource tree. A hierarchy is built by using one of two possible relations: aggregation (n:1 relation) or partitioning (1:n relation). Using the definition from Graupner [GKT02], a virtual resource is a set of parameters mapped onto a collection containing a collection of physical and/or simple virtual resources.

4.2.2 Security and Assurability of Virtual Resources

The virtualization of resources has been used for securing them, but virtualization is in most cases achieved by hiding the complexity of the base resource under a software virtualization layer. A layered approach cannot make a resource completely secure, but is complementing its existing security [Gru08]. Security threats affecting the virtualization layer are directly affecting the security of the resulting virtual resource. For example, a virtual machine is the result of applying a virtualization layer over the underlying machine. The software solution can be based on system virtualization or process virtualization. There are different standard agents responsible for the security of network or storage resources (firewall, antivirus), but there is no standard agent responsible for the security of the virtualization solution. The subsystem-as-spy problem [Gru08] is another security threat to virtualization, because components of a system (network cards, graphic cards etc.) have the capacity and the possibility to spy all virtual machines built on top of them.

Assuring something means to offer some guarantees about that particular object or subject. We will consider the object being a virtual resource. The guarantees to be made are based on the specific class a resource belongs to. For example, the most wanted guarantees from a CPU are its speed, whereas from a router they are reliability and ease of management. Making guarantees about a virtual resource is related to be able to depend on that particular virtual resource.

Popek and Goldberg [PG74] consider that a virtual machine monitor is system software having the following characteristics: fidelity, performance and safety. Extrapolating this, we can state that a virtual resource must possess these characteristics because the virtualization is directly affecting the performance of the resource.

4.3 Relation to the $D^2R^2 + DR$ Strategy

Several technologies in the area of system virtualization and P2P overlays can be related to the $D^2R^2 + DR$ strategy [HS09] that is central to the ResumeNet project. For example, building a service on a P2P network is clearly related to defense if one expects a high churn rate. Autonomically setting up a VPN tunnel after network traffic restrictions have been detected in order to retain connectivity could be regarded as a remediation strategy. Likewise, in case of an impending service failure, a possible remediation may be to migrate the service. Re-establishing connections lost during migration is connected to service recovery. Alternatively, one could view service migration in combination with connection re-establishment as remediation, in which case the migration (and corresponding connection re-establishment) of the service back to its original position would be the recovery.

Detection of challenges is a component of the $D^2R^2 + DR$ strategy that does not lend itself easily to the techniques described in this deliverable. System virtualization and P2P overlays are foremost abstraction techniques – challenge detection has to be provided by another component. Likewise, these techniques cannot do much about diagnosis. A possible refinement strategy, however, might be to introduce system virtualization or a P2P overlay as a new resilience component. A service could, for example, be virtualized after experiencing service downtime, in order to counter future challenges.

5 Conclusions

Both, system virtualization and P2P overlays, provide a specific kind of abstraction from network hardware. Different implementations exist, each with their respective advantages and disadvantages. The added value of abstraction offered by system virtualization and P2P overlay techniques provides valuable tools to counter specific threats and challenges. While these techniques cannot cover the full $D^2R^2 + DR$ strategy, they fit nicely with the Defend, Remediate, and Recover components and may even be useful in the context of the Refinement component.

References

[APST05] T. Anderson, L. Peterson, S. Shenker, and J. Turner. Overcoming the Internet impasse through virtualization. *Computer*, 38(4):34–41, 2005.

[BDF+03] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review*, 37(5):164–177, 2003.

[BFH+06] Andy Bavier, Nick Feamster, Mark Huang, Larry Peterson, and Jennifer Rexford. In vini veritas: realistic and controlled network experimentation. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 3–14, New York, NY, USA, 2006. ACM.

[CB09] N. M. Mosharaf Kabir Chowdhury and Raouf Boutaba. Network virtualization: State of the art and research challenges. *IEEE Communications Magazine*, 47(7):20 – 26, July 2009. IEEE ComSoc.

[CCR+03] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12, 2003.

[CDMK+99] A.T. Campbell, H.G. De Meer, M.E. Kounavis, K. Miki, J.B. Vicente, and D. Villela. A survey of programmable networks. *ACM SIGCOMM Computer Communication Review*, 29(2):7–23, 1999.

[CFH+05] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2 table of contents*, pages 273–286. USENIX Association Berkeley, CA, USA, 2005.

[cis] Logical Routers Commands on Cisco IOS XR Software. `http://www.cisco.com/en/US/docs/ios_xr_sw/iosxr_r3.2/interfaces/command/reference/hr32lr.html`. Accessed 01. Oct. 2009.

[dmt07] Dmtf standard dsp 1057: Virtual system profile. `http://www.dmtf.org/standards/published_documents/DSP1057.pdf`, May 2007. Accessed 01. Oct. 2009.

[EGH+07] Norbert Egi, Adam Greenhalgh, Mark Handley, Mickael Hoerdt, Laurent Mathy, and Tim Schooley. Evaluating xen for router virtualization. In *16th Int. Conf. on Comp. Commun. and Networks – ICCCN 2007*, pages 1256–1261, Aug. 2007.

[gen] GENI – Global Environment for Networking Innovations. `http://www.geni.net`. Accessed 01. Oct. 2009.

[GKT02] S. Graupner, V. Kotov, and H. Trinks. Resource-sharing and service deployment in virtual data centers. In *IEEE Workshop on Resource Sharing in Massively Distributed Systems (ICDCS-2002)*, 2002.

[Gru08] Galen Gruman. Virtualization's secret security threats. `http://www.infoworld.com/d/security-central/virtualizations-secret-security-threats-159?page=0,0`, March 2008. Accessed 14 Aug. 2009.

[HS09] David Hutchison and James P.G. Sterbenz. Resilinets: Resilient and survivable networks. In *ERCIM News 77*, April 2009.

[LCP+05] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7:72–93, March 2005.

[LNBK02] David Liben-Nowell, Hari Balakrishnan, and David Karger. Analysis of the evolution of peer-to-peer systems. In *PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 233–242, New York, NY, USA, 2002. ACM.

[MCZ06] Aravind Menon, Alan L. Cox, and Willy Zwaenepoel. Optimizing network virtualization in xen. In *USENIX Annual Technical Conference*, pages 15–28, May 2006.

[Oli] V. Olifer. Different Flavours of VPN: Technology and Applications. `http://www.ja.net/documents/development/vpn/different-flavours-of-vpn-web.pdf`. Accessed on 1. Oct. 2009.

[PACR02] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proceedings of the first workshop on Hot Topics in Networking (HotNets-I)*, 2002.

[PF96] D. Passmore and J. Freeman. The Virtual LAN Technology Report. Technical report, 3COM, 1996.

[PG74] Gerald J. Popek and Robert P. Goldberg. Formal Requirements for Virtualizable third Generation Architectures. *Commun. ACM*, 17(7):412–421, 1974.

[SMK+01] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM New York, NY, USA, 2001.

[TGB06] Phuoc Tran-Gia and Andreas Binzenhofer. On the stochastic scalability of information sharing platforms. In *Distributed Cooperative Laboratories: Networking, Instrumentation, and Measurements*, pages 11–27. SpringerUS, June 2006.

[TSS+97] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall, and G.J. Minden. A survey of active network research. *IEEE communications Magazine*, 35(1):80–86, 1997.

[VdMRLC98] JE Van der Merwe, S. Rooney, L. Leslie, and S. Crosby. The Tempest-A Practical Framework for Network Programmability. *IEEE network*, 12(3):20–28, 1998.

[vin] VINI – A virtual network infrastructure. `http://www.vini-veritas.net/`. Accessed 01. Oct. 2009.

[vmw] Vmware esx – bare-metal hypervisor for virtual machines. `http://www.vmware.com/products/vi/esx/`. Accessed 01. Oct. 2009.

[WKB+08] Yi Wang, Eric Keller, Brian Biskeborn, Jacobus van der Merwe, and Jennifer Rexford. Virtual routers on the move: live router migration as a network-management primitive. *SIGCOMM Comput. Commun. Rev.*, 38(4):231–242, 2008.

[YS01] R. Yuan and W.T. Strayer. *Virtual private networks: technologies and solutions*. Addison-Wesley Professional Computing Series, 2001.